Introduction to Machine Learning
What is Machine Learning?

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
Data mining / machine learning / “big data” are all related to the process of discovering new patterns from large data sets.

There is substantial overlap between all three “fields”
- they all use ideas from statistics and algorithms.

According to Hand, Mannila and Smyth:

“Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.”

But isn’t all this just statistics?!
Differences Between Machine Learning and Statistics?

Traditional statistics:
- First hypothesize, then collect data, then analyze.
- Often model-oriented with an emphasis on parametric models.
- Focus on understanding and hypothesis testing.

Machine learning (ML):
- Few if any a priori hypotheses.
- Typically data has already been collected.
- Analysis is typically data-driven not hypothesis-driven.
- Often algorithm-oriented rather than model-oriented.
- Focus on prediction.
- ML is more associated with artificial intelligence than statistics.

But statistical ideas and algorithms are extremely useful in ML
  - e.g. inference, bootstrapping, over-fitting solutions, regression and classification algorithms.

“Big data” also encompasses data-management technology
  - e.g. cloud computing, NoSQL databases, map-reduce and Hadoop etc.
Robert Tibshirani Compares Machine Learning with Statistics :-)

<table>
<thead>
<tr>
<th>Machine learning</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>network, graphs</td>
<td>model</td>
</tr>
<tr>
<td>weights</td>
<td>parameters</td>
</tr>
<tr>
<td>learning</td>
<td>fitting</td>
</tr>
<tr>
<td>generalization</td>
<td>test set performance</td>
</tr>
<tr>
<td>supervised learning</td>
<td>regression/classification</td>
</tr>
<tr>
<td>unsupervised learning</td>
<td>density estimation, clustering</td>
</tr>
<tr>
<td>large grant = $1,000,000</td>
<td>large grant = $50,000</td>
</tr>
<tr>
<td>nice place to have a meeting: Snowbird, Utah, French Alps</td>
<td>nice place to have a meeting: Las Vegas in August</td>
</tr>
</tbody>
</table>

But “uncool” jargon isn’t unique to statistics; witness “neural networks”.
There are many reasons including:

• The explosion of data from many domains such as biology, the internet, astrophysics, telecommunications, engineering and sensor networks.

• The availability of ever cheaper storage
  - developments in database technology such as NoSQL databases

• Faster and cheaper computation to analyze the data.

• Competitive pressure in business: data has value!

• Software developments:
  - commercial products: SAS, SPSS, Google Analytics, IBM, Oracle, Amazon, S-Plus
  - open source products: R, Python libraries such as scikit learn and pandas, Weka.

Don’t necessarily need to be a ML expert to do ML
- but many pitfalls so some competence is required!
Is “Big Data” Over-Hyped?

But “big data” is perhaps over-hyped in the popular media – and startup scene seems focussed on less important matters:

But There Are Many Data Mining Successes ... 

- Google: the entire company!
- Market Basket (WalMart).
- Recommender Systems: Amazon.com, Netflix.
- Fraud Detection in Telecommunications: AT&T.
- Targeted Marketing: Target.
- DNA Microarray analysis.
- Smart grid: dynamic demand response.
Taxonomy of Machine Learning Problems

Supervised Learning
- Observe \textit{training} samples, and learn a function mapping inputs to outputs.
- Examples include regression, classification, ranking, etc.

Unsupervised Learning
- Observe data and construct a low complexity description of the data.
- Clustering, dimensional reduction techniques such as principal components analysis, non-negative matrix factorization etc.

Reinforcement Learning (RL)
- RL attempts to learn a map from system states to actions with the goal of maximizing total cumulative reward.
- A given action in a given state influences what the next state will be
  - so need to trade off \textit{exploitation} (of current knowledge) with \textit{exploration} (to learn new information).
- Closely related to \textit{dynamic programming} except that we do not know the parameters of the model.

Will focus on supervised and unsupervised problems in this course.
Data Types

Data comes in many forms ... 

- Flat files or vector data.
- Text data; e.g. an article or collection of articles.
- Transactional data: web logs, phone calls.
- Relational data from relational databases.
- Time series data; e.g. financial data, auction data from EBay.
- Image data and video data.
- Spatial temporal data e.g. geophysical data.
- Network data: physical networks, social networks etc.

Regardless of original data-type, typically need to convert it into quantitative data
  – hopefully without losing important information.
Exploratory Data Analysis (EDA)

**Goal:** To obtain a general sense and understanding of the data

– EDA is a vital and necessary first step for any machine learning / statistics task.

EDA tasks:

- What do the marginal distributions look like? Normal, skewed, etc.
- What is the quality of the data? Is some of it missing?
  - If so, why is it missing? Is it missing at random (MAR), missing completely at random (MCAR) or missing not at random (MNAR)?
  - A very important issue!
- Are there outliers? What should we do with them?
- What are the dependencies / correlation between different variables?
- What subsets of the variables are of particular interest?
- What sort of (functional) relationships are we looking for?

General guidelines for EDA:

- Make it data-driven and typically model-free ...
- Think interactive and visual ...
- Many dimensions to play with: x, y, z, space, color, time ...
Some EDA Strategies

Summary statistics:
- mean, median, variance, skewness, kurtosis

Single variable visualization:
- Histograms
- Density estimates
- Box plots

Two variable visualization:
- Scatter plot
- Binning
- Transparent plots
- Contour plots
- Bar charts: categorical variables

Bottom line: it is always well worth looking at your data!
- EDA will not be a focus of this course but it should always be the start of any analysis.
Data Visualization

Data visualization is an increasingly important subfield of “big data”
– if “a picture is worth a thousand words” what is a video worth?

See Hans Rosling’s famous talk on the joy of statistics (and global health dynamics) at http://www.youtube.com/watch?v=jbkSRLYSojo.

Data visualization also becoming increasingly important in journalism
– see http://www.theguardian.com/data if you need convincing.
Domain Knowledge

An expertise in machine learning / statistics generally not enough to apply these methods successfully.

**Domain-specific knowledge** generally also required:

- How can we apply techniques to data from biology, the internet, astrophysics, telecommunications, engineering and sensor networks, etc. without understanding these domains?
- How do we know what the right questions are?
- How can we make sense of the results?
- Or figure out if the results even make sense?

The most successful ML applications almost always based on good domain knowledge

- also helps in the EDA process.

Should avoid the temptation of blindly applying your favorite ML tool

- a hammer looking for a problem is not a good recipe for success!
Introduction to Machine Learning
Supervised Learning: Regression I

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
Linear Regression

- Linear regression assumes the regression function $\mathbb{E}[Y|X]$ is linear in the inputs, $X_1, \ldots, X_p$.

- Developed many years ago but still very useful today
  - simple and easy to understand
  - can sometimes outperform more sophisticated models when there is little data available.

- Linear models can also be applied to transformations of the inputs
  - leads to the basis function approach (and kernel regression)
  - which extends the scope of linear models to non-linear models.

- But linear models also have many weaknesses including a tendency to over-fit the data
  - will return to this later when we discuss the bias-variance decomposition and (in a later set of slides) shrinkage methods.
Linear Regression

In a linear regression model the dependent variable \( Y \) is a random variable that satisfies \( \mathbf{X} = (X_1, \ldots, X_p) \)

\[
Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \epsilon
\]

where \( \epsilon \) is the “error” term.

The linear model therefore implicitly assumes that \( \mathbb{E}[Y \mid \mathbf{X}] \) is approximately linear in \( \mathbf{X} = (X_1, \ldots, X_p) \).

The input or independent variables, \( X_i \), are numerical inputs

- or possibly transformations, e.g. product, log, square root, \( \phi(x) \), of “original” numeric inputs

- the ability to transform provides considerable flexibility.

The \( X_i \)'s can also be used as “dummy” variables that encode the levels of qualitative inputs

- an input with \( K \) levels would require \( K - 1 \) dummy variables, \( X_1, \ldots, X_{K-1} \)
Model Fitting: Minimizing the Residual Sum of Squares

We are given training data: \((y_1, x_1), (y_2, x_2), \ldots, (y_N, x_N)\).

Then obtain \(\hat{\beta}\) by minimizing the residual sum of squares or RSS:

\[
\min_{\beta} \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 = \min_{\beta} \| y - X \beta \|^2
\]

where

\[
y := \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \quad X := \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix}
\]
Model Fitting: Minimizing the Residual Sum of Squares

This is a simple (convex) quadratic optimization problem so \( \hat{\beta} \) satisfies

\[
\nabla \| y - X\beta \|^2 = -2X^T(y - X\beta) = 0 \quad \Rightarrow \quad \hat{\beta} = (X^TX)^{-1}X^Ty
\]

Also have

\[
\hat{y} = \underbrace{X(X^TX)^{-1}X^T}_H y \quad \text{and} \quad \hat{\epsilon} = y - \hat{y} = (I - H)y.
\]

Have (implicitly) assumed that \((X^TX)\) is non-singular.

This is not always the case in which case \( \hat{\beta} \) will not be unique (although \( \hat{y} \) still is)

- can resolve by dropping redundant columns from \( X \)
  - many software packages do this automatically

But in many modern applications \( p \gg N \) in which case at least \( N - p \) columns would need to be dropped – something we may not want to do!

- hence the need for another solution approach e.g. ridge regression.

For now we will assume \( p \leq N \).
Model Fitting: Minimizing the Residual Sum of Squares

The residual-sum-of-squares is defined as

$$
\text{RSS} := \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \hat{e}^\top \hat{e}
$$

whereas the total-sum-of-squares is

$$
\text{TSS} := \sum_{i=1}^{N} (y_i - \bar{y})^2.
$$

The $R^2$ statistic is a measure of the linear relationship between $X$ and $Y$:

$$
R^2 := 1 - \frac{\text{RSS}}{\text{TSS}}.
$$

$R^2$ always lies in the interval $[0, 1]$ with values closer to 1 being “better”

- but whether a given $R^2$ value is good or not depends on the application
- in physical science applications we looks for values close to 1 (if the model is truly linear); in social sciences an $R^2$ of .1 might be deemed good!
Figure 3.1 from HTF: Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of $X$ that minimizes the sum of squared residuals from $Y$. 
The Advertising Data Set from ISLR

Figure 2.1. displays the advertising data set from ISLR. It consists of:

- Sales of a particular product in 200 different markets
- Advertising budgets for the product in each of those markets for three different media:
  1. TV
  2. radio
  3. newspaper

The goal is to answer the following questions:

1. Is there a relationship between advertising budget and sales?
2. How strong is the relationship between advertising budget and sales?
3. Which media contribute to sales?
4. How accurately can we estimate the effect of each medium on sales?
5. How accurately can we predict future sales?
6. Is the relationship linear?
7. Is there synergy among the advertising media?

Section 3.4 of ISLR provides answers to these questions - but need to read earlier sections of chapter 3 first.
Figure 2.1 from ISLR: The Advertising data set. The plot displays sales, in thousands of units, as a function of TV, radio, and newspaper budgets, in thousands of dollars, for 200 different markets. In each plot we show the simple least squares fit of sales to that variable, as described in Chapter 3. In other words, each blue line represents a simple model that can be used to predict sales using TV, radio, and newspaper, respectively.
The Credit Data-Set from ISLR

The credit data-set from ISLR contains quantitative data on following variables for a number of customers. See Fig. 3.6 for corresponding scatter-plot matrix.

- balance (average credit card debt)
- age (in years).
- cards (number of credit cards)
- education (years of education)
- income (in thousands of dollars)
- limit (credit limit)
- rating (credit rating)

There are also four qualitative variables:

- gender
- student (student status)
- status (marital status)
- ethnicity (Caucasian, African American or Asian)

See Section 3.3 of ISLR for analysis and discussion of this data-set and in particular, how to handle qualitative variables using dummy variables.
Figure 3.6 from ISLR: The Credit data set contains information about balance, age, cards, education, income, limit, and rating for a number of potential customers.
Potential Problems with Linear Regression

Many problems can arise when fitting a linear model to a particular data-set:

1. Non-linearity of the response-predictor relationships
   - plotting residuals against fitted values are a useful graphical tool for identifying this problem
   - a simple solution is to use non-linear transformations of the predictors.

2. Correlation of error terms
   - a serious problem since estimation of $\sigma^2$ and statistical tests all depend on assumption of zero-correlation
   - problem can arise with time-series data – can detect it then by plotting residuals against time.

3. Non-constant variance or heteroscedasticity of error terms
   - another important assumption that can be tested by plotting residuals against fitted values
   - if problem exists consider applying a concave function to $Y$.

4. Outliers, i.e. points for which $y_i$ is far from the predicted value $\hat{\beta}^T X_i$
   - could be genuine or a data error
   - may or may not impact fitted model – but regardless will impact $\hat{\sigma}^2$, confidence intervals and p-values, possibly dramatically
   - can identify them by plotting studentized residuals against fitted values – values $> 3$ in absolute value are suspicious.
5. High-leverage points
   - these are points whose presence has a large impact on the fitted model
   - generally correspond to extreme predictor $X$
   - can identify such points via their leverage statistic, $h_i := H_{ii}$; always the case that $h_i \in [1/N, 1]$.

6. Collinearity and multi-collinearity
   - collinearity is the problem when two or more predictor variables are highly correlated
   - difficult then to separate out the individual effects and corresponding coefficients tend to have very high variances
   - can assess multi-collinearity by computing the variance inflation factor (VIF) which is the ratio of $\text{Var} \left( \hat{\beta}_i \right)$ when fitting the full model divided by $\text{Var} \left( \hat{\beta}_i \right)$ if fit on its own
     - smallest possible value is 1; rule of thumb is that values exceeding 5 or 10 indicate collinearity
   - solution is to either drop one of the variables or combine them into a single predictor. e.g. in credit data set could combine limit and rating into a single variable.

See discussion in Section 3.3.3 of ISLR for further discussion.
Linear Regression with Basis Functions

Can also do everything with basis functions

\[ Y = \beta_0 + \sum_{i=1}^{M} \beta_i \psi_i(x) + \epsilon \]

where \( \psi_i : \mathbb{R}^p \rightarrow \mathbb{R} \) is the \( i^{th} \) basis function.

Example: \( \psi_i(x) = \frac{1}{(2\pi\sigma^2)^{p/2}} e^{-\frac{1}{2\sigma^2} \|x-\mu_i\|^2} \).

The \( \psi_i(x) \)'s are often used to encode domain-specific knowledge.

Parameter estimate:

\[ \hat{\beta} = (\Psi^\top \Psi)^{-1} \Psi^\top y \]

where

\[ \Psi = \begin{bmatrix} 1 & \psi_1(x_1) & \psi_2(x_1) & \cdots & \psi_M(x_1) \\ 1 & \psi_1(x_2) & \psi_2(x_2) & \cdots & \psi_M(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \psi_1(x_N) & \psi_2(x_N) & \cdots & \psi_M(x_N) \end{bmatrix} \]
Linear Regression with Basis Functions

Ability to use basis functions extends the scope of linear regression to obtain non-linear relationships between $Y$ and $X$

- this flexibility can be very valuable
- when basis functions are simply powers of the original inputs we call it polynomial regression
- splines can also be implemented via basis functions.

If $M$ gets too large then solving for $\hat{\beta}$ may become intractable

- but kernel methods and so-called “kernel trick” can then come to the rescue
- in which case possible to even take $M = \infty$

Will defer study of kernel methods until we study support vector machines

- but note here they are applicable to many forms of regression including linear and ridge regression.

Can also fit non-linear models using smoothing splines, local regression or GAMs (generalized additive models)

- will not study them in this course but see Chapter 7 of ISLR for details.
Why Minimize the Sum-of-Squares?

Let $X$ be non-random and suppose we want to estimate $\theta := a^\top \beta$.
Then least-squares estimate of $\theta$ is

$$\hat{\theta} = a^\top \hat{\beta} = a^\top (X^\top X)^{-1} X^\top y$$

– a linear function of the response $y$.

If the linear model is correct then easy to check that $E[\hat{\theta}] = \theta$ so $\hat{\theta}$ is unbiased.

**Gauss-Markov Theorem**: Suppose $c^\top y$ is any unbiased estimate of $\theta$. Then

$$\text{Var} \left( a^\top \hat{\beta} \right) \leq \text{Var} \left( c^\top y \right).$$

The Gauss-Markov Theorem says that the least-squares estimator has the smallest variance among all linear unbiased estimators.

**Question**: Great! But is unbiasedness a good thing?
Mean-Squared Error

To answer this question let \( \tilde{\theta} \) be some estimator for \( \theta \).

The mean-squared-error (MSE) then satisfies

\[
\text{MSE}(\tilde{\theta}) = \mathbb{E} \left[ (\tilde{\theta} - \theta)^2 \right] = \text{Var}(\tilde{\theta}) + \left( \mathbb{E} \left[ \tilde{\theta} \right] - \theta \right)^2. 
\]

If the goal is to minimize MSE then unbiasedness not necessarily a good thing

- can often trade a small increase in \( \text{bias}^2 \) for a larger decrease in variance
- will do this later with subset selection methods as well as shrinkage methods
  - an added benefit of some of these methods is improved interpretability.

But first let’s study the bias-variance decomposition.
The Bias-Variance Decomposition

Assume the true model is \( Y = f(X) + \epsilon \) where \( E[\epsilon] = 0 \) and \( \text{Var}(\epsilon) = \sigma^2_{\epsilon} \).

Let \( \hat{f}(X) \) be our estimate at a new fixed point, \( X = x_0 \). Then the error at \( x_0 \) assuming the training inputs are fixed, i.e. non-random, is:

$$
\text{Err}(x_0) = E \left[ (Y - \hat{f}(x_0))^2 \right]
$$

\[= E \left[ (f(x_0) + \epsilon - \hat{f}(x_0))^2 \right]
\]

\[= E[\epsilon^2] + E \left[ (f(x_0) - \hat{f}(x_0))^2 \right]
\]

\[= \sigma_{\epsilon}^2 + \left[ (f(x_0) - E[\hat{f}(x_0)])^2 + E[\hat{f}(x_0)] - \hat{f}(x_0)) \right] \]

\[= \sigma_{\epsilon}^2 + \left[ (f(x_0) - E[\hat{f}(x_0)])^2 + E[\hat{f}(x_0)] - \hat{f}(x_0)) \right] \]

\[= \sigma_{\epsilon}^2 + \text{Bias}^2 \left( \hat{f}(x_0) \right) + \text{Var} \left( \hat{f}(x_0) \right)
\]

\[= \text{Irreducible Error} + \text{Bias}^2(x_0) + \text{Variance}(x_0).
\]
Example: the Bias-Variance Trade-Off

Consider the following example from Bishop:

1. The “true” model to be estimated is

\[ y(x) = \sin(2\pi x) + \epsilon, \quad x \in [0, 1], \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \]

- a very nonlinear function of \( x \).

2. We fit a linear regression model with \( M = 24 \) Gaussian basis functions

\[ \psi_j(x) := e^{-\frac{1}{2\sigma^2}(x-\mu_j)^2} \]

with \( \mu_j = \frac{j}{M-1} \) for \( j = 0, \ldots, M - 1 \) and \( \sigma = \frac{1}{M-1} \).

3. Including the constant term the parameter vector \( \beta \) is \( (M + 1) \times 1 \).

4. We will also include a regularization term so that regression problem solves

\[ \hat{\beta} = \arg\min_{\beta} \sum_{j=1}^{N} \left( Y_j - \beta_0 - \sum_{i=1}^{M} \beta_i \psi_i(x_j) \right)^2 + \frac{\lambda}{2} \beta^\top \beta \]
5. A data-set $D = \{(y_i, x_i) : i = 1, \ldots, N\}$, with $N = 25$
   - the $x_i$’s are sampled randomly from $[0, 1]$
   - the $y_i$’s are then sampled using (3).
     - so noise comes both from measurement and sampling.

6. We generate $L = 100$ of these data-sets.

7. The model is fit by solving (4) for each of the $L$ data-sets and various values of $\lambda$.

Results are displayed in Figure 3.5:

The model bias is clear from graphs in right-hand column.

The variance of individual fits is clear from graphs in left-hand column.

The bias-variance tradeoff is clear and quantified in Figure 3.6.
Figure 3.5 from Bishop: Illustration of the dependence of bias and variance on model complexity, governed by a regularization parameter \( \lambda \), using the sinusoidal data set from Chapter 1. There are \( L = 100 \) data sets, each having \( N = 25 \) data points, and there are 24 Gaussian basis functions in the model so that the total number of parameters is \( M = 25 \) including the bias parameter. The left column shows the result of fitting the model to the data sets for various values of \( \ln \lambda \) (for clarity, only 20 of the 100 fits are shown). The right column shows the corresponding average of the 100 fits (red) along with the sinusoidal function from which the data sets were generated (green).
Figure 3.6 from Bishop: Plot of squared bias and variance, together with their sum, corresponding to the results shown in Figure 3.5. Also shown is the average test set error for a test data set size of 1000 points. The minimum value of $(\text{bias})^2 + \text{variance}$ occurs around $\ln \lambda = -0.31$, which is close to the value that gives the minimum error on the test data.
A Case Study: Overfitting with Polynomials

Overfitting is a very serious issue that needs to be handled in supervised learning problems.

To explore overfitting in further detail we will consider two 1-dimensional polynomial regression problems.

**Problem 1**

- True model is $y = f(x) + \epsilon$ where $\epsilon$ is IID noise and $f(x)$ is a $10^{th}$ order polynomial on $x \in \mathbb{R}$.
- There are $n = 15$ datapoints: $(x_1, y_1), \ldots, (x_n, y_n)$
  - the $x_i$’s were generated $\sim U(-1, 1)$ and then $y_i = f(x_i) + \epsilon_i$ where the $\epsilon_i$’s were generated IID $N(0, 3)$.
- We fit $2^{nd}$ and $10^{th}$ order polynomials to this data via simple linear regression, that is we regress $Y$ on $1, X, \ldots, X^J$ where $J = 2$ or $J = 10$.
- The results are displayed in the figure on the next slide.
Fitting a Low-Order Polynomial With Noisy Data: The target curve is the 10th order polynomial \( y = f(x) \).
Question: Which regression results in a superior fit to the data?

Question: Which regression results in a superior out-of-sample or generalization error?

Note that the set of $10^{th}$ order polynomials contains the true target function, $y = f(x)$, whereas the set of $2^{nd}$ order polynomials does not.

We might therefore expect the $10^{th}$ order fit to be superior to the $2^{nd}$ order fit - but this is not the case!

Question: Why do you think the $2^{nd}$ order fit does a better job here?

Question: Do you think the $2^{nd}$ order fit will always be better irrespective of $N$, the number of data-points?
A Case Study: Overfitting with Polynomials

Problem 2

• True model is \( y = f(x) \) and \( f(x) \) is a \( 50^{th} \) order polynomial on \( x \in \mathbb{R} \).
• There are \( n = 15 \) datapoints: \((x_1, y_1), \ldots, (x_n, y_n)\)
  - the \( x_i \)'s were generated \( \sim U(-1, 1) \) and then \( y_i = f(x_i) \) so the observations are noiseless.
• We fit \( 2^{nd} \) and \( 10^{th} \) order polynomials to this data via simple linear regression, that is we regress \( Y \) on \( 1, X, \ldots, X^J \) where \( J = 2 \) or \( J = 10 \).
• The results are displayed in the figure on the next slide.

Commonly thought that overfitting occurs when the fitted model is too complex relative to the true model
  - but this is not the case here: clearly the \( 10^{th} \) order regression overfits the data but a \( 10^{th} \) order polynomial is considerably less complex than a \( 50^{th} \) order polynomial.

What matters is how the model complexity matches the quantity and quality of the data, not the (unknown) target function.
Fitting a High-Order Polynomial With Noiseless Data: The target curve is the $10^{th}$ order polynomial $y = f(x)$.

Note: This case study is based on the case study in Section 4.1 of “Learning from Data” by Abu-Mostafa, Magdon-Ismail and Lin.
It is **vital** then to control over-fitting when performing supervised learning, i.e. regression or classification. There are many approaches:

- **Subset selection** where we retain only a subset of the independent variables
- **Shrinkage methods** where coefficients are shrunk towards zero.
- **Regularization** where we penalize large-magnitude parameters
  - shrinkage often achieved via regularization

**Cross-validation** often used to select the specific model. Other methods include:

- **Bayesian models**
  - many shrinkage / regularization methods can be interpreted as Bayesian models where the penalty on large-magnitude parameters becomes a *prior distribution* on those parameters.

- Methods that explicitly penalize the number of parameters, $p$, in the model
  - Akaike Information Criterion (AIC) = $-2 \ln(\text{likelihood}) + 2(p + 1)$
  - Bayesian Information Criterion (BIC): $-2 \ln(\text{likelihood}) + (p + 1) \ln(N)$

Choose the model that minimizes the AIC or BIC
- these methods apply to models fit via MLE.

Will return to some of these methods during the course!
Introduction to Machine Learning
Model Selection & Assessment: Cross Validation

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
The goal of model selection is usually in obtaining the best predictions with new data.

Complex models will generally fit a given training set better than less complex models
- follows since more complex models have more flexibility
- but often results in over-fitting in which case the fitted complex model does not generalize well.

This is related to the bias-variance decomposition that we saw earlier when we studied regression.

We use the term training error to refer to the classification error on the data-set that was used to train the classifier.

We use the term test error to refer to the classification error on a new or holdout data-set that was not used to train the classifier
- also sometimes referred to as generalization error.
Training Error Versus Test Error

Figure taken from Ben Taskar’s web-site at U Penn.

- In the above Figure we see that the training error generally declines as model complexity increases. Why?
- However, the true error, i.e. test or generalization error, tends to decrease for a while and then increase as it begins to over-fit the data.
Approaches to Control Over-Fitting

There are many approaches that are used to control over-fitting:

- **The Akaike information criterion (AIC) and the Bayesian information criterion (BIC) penalize the effective # of parameters**
  - used in MLE settings when we can compute effective # of parameters.

- **Bayesian models** control over-fitting naturally by modeling parameters as random variables
  - estimation in these models therefore implicitly accounts for parameter uncertainty
  - Bayesian models are very popular in statistics and ML.

- **Regularization approaches** that explicitly penalize parameter magnitudes along with the misclassification or prediction error in the objective function
  - smaller magnitude parameters are therefore preferred to larger magnitude parameters
  - the degree of regularization is controlled via a regularization parameter, $\lambda$
    
    e.g. ridge regression solves

    $$
    \min_{\beta} \left\{ \frac{1}{2} \| y - X\beta \|^2 + \lambda \cdot \frac{1}{2} \| \beta \|^2 \right\}.
    $$
Approaches to Control Over-Fitting

- regularization methods can also often be viewed as specific examples of Bayesian models.

- A standard approach is to use training, validation and test sets.
- Another very useful technique is cross-validation
  - often the tool of choice in data mining and machine learning
  - and often used to choose regularization parameters; e.g. $\lambda$ in ridge regression.

For now we will restrict ourselves to the training, validation and test set approach.

Later we will when we study resampling methods we will consider cross-validation.
Training, Validation and Test Sets

- In order to assess model performance we often partition the data into a **training set** and a **validation set**.
- The training set is used to the construct the classifier(s) and the validation set is used to assess their performance(s).
- The performance of each classifier on the validation set gives an **unbiased estimator** of the classifier’s performance.
- If we have trained many classifiers then in the model selection stage we can choose the classifier that performed best on the validation set.

**Question:** When we choose a classifier this way, is its performance on the validation set still an unbiased estimator of its performance?

**Answer:** No. Why?

As a result we would like an additional **test set** which is used to evaluate the selected classifier. The test set is **never** used in the training and validation stages.
Problems in the Training, Validation and Test Set Approach

- Data is often scarce in which case we cannot afford to set aside separate validation (and/or test sets) when training a classifier or fitting a regression.

- There are also some drawbacks to using a validation set in the training procedure:
  1. Performance on the validation set is a (often highly variable) random variable depending on the data-split into training and validation sets.
  2. The error on the validation set tends to over-estimate the test-error rate of the model that is fitted to the entire data-set. Why?

- Instead we can perform $K$-fold cross-validation
  - closely related to the validation set approach but it does address the drawbacks highlighted above.
$K$-Fold Cross-Validation

$K$-Fold Cross-Validation proceeds as follows:

1. Partition the entire data-set into $K$ equal-sized components.
2. For $k = 1, \ldots, K$, fit the model to the other $K - 1$ components and calculate the prediction error on the $k^{th}$ component.
3. Combine the $K$ estimates of the prediction error to obtain an average prediction error:

$$ \text{CV}_K (\hat{f}) = \frac{1}{N} \sum_{i=1}^{N} L \left(y_i, \hat{f} - \kappa(i)(x_i) \right) $$

where:

- $\kappa(i) \in \{1, \ldots, K\}$ denotes the partition component to which the $i^{th}$ observation was allocated
- $\hat{f} - \kappa(i)(\cdot)$ denotes the fitted classifier in step (2) above when $k = \kappa(i)$. 
**K-Fold Cross-Validation**

- Typical choices of \( K \) are 5 or 10.
- When \( K = N \) the procedure is called **leave-one-out** cross-validation
  - very little bias then (why?) but also computationally very expensive.
- If we have a set of models indexed by \( \alpha \) then we can use \( K \)-fold cross-validation to estimate \( \text{CV}_K \left( \hat{f}, \alpha \right) \) for each \( \alpha \)
  - can then plot \( \text{CV}_K \left( \hat{f}, \alpha \right) \) against \( \alpha \) to yield a **test-error** curve
  - choose \( \alpha = \hat{\alpha} \) where \( \hat{\alpha} \) minimizes this curve
  - then obtain \( f(x, \hat{\alpha}) \) by fitting to the **entire** data-set with \( \alpha = \hat{\alpha} \).
- Click on the image below for a brief but very nice tutorial by David Weiss on the use of cross-validation in the context of \( k \)-Nearest Neighbors.
Consider a classification problem with a large number of predictors. One possible strategy for analysis is:

1. Screen the predictors to find a subset of “good” predictors that show strong univariate correlation with the class labels.

2. Use just this subset of predictors to build a multivariate classifier.

3. Use cross-validation to estimate the unknown tuning parameters (e.g. \( k \) in \( k \)-NN or a regularization parameter \( \lambda \)) and / or to estimate the prediction error of the final model.

**Question:** Is this a correct application of cross-validation?
The Wrong Way to Do Cross-Validation

**Answer:** No! Consider a scenario with $N = 50$ samples in two equal-size classes and $p = 5,000$ standard Gaussian predictors that are independent of the class labels.

Then the true test-error rate of any classifier will be (why?) $50\%$.

HTF carried out steps (1) to (3):

1. they chose the 100 predictors having the highest correlation with the class labels.
2. they built a 1-NN classifier using the 100 predictors from step (1).
3. they estimated the prediction error using CV.

They repeated this experiment 50 times and obtained an average CV error of $3\%$!

**Question:** What has happened?

**Question:** What is the correct way to do cross-validation here?
Figure 7.10 from HTF: Cross-validation the wrong and right way: histograms shows the correlation of class labels, in 10 randomly chosen samples, with the 100 predictors chosen using the incorrect (upper red) and correct (lower green) versions of cross-validation.
An Extreme Example with Leave-One-Out CV

- Consider a random data-set with $2N$ observations, 2 classes and $M$ predictors.

- $N$ of the observations have been assigned to the first class and the other $N$ have been assigned to the second class.

- Suppose this class assignment has been made independently of the predictors.

- Suppose we use leave-one-out cross-validation to estimate the prediction error of a given model.

- Under leave-one-out CV the best model will predict the majority class.

- The true test error will then be 50%.

- But the predicted error will be 100%!

See Chapter 7 of HTF for further details on cross-validation and controlling over-fitting.
Introduction to Machine Learning

Supervised Learning: Regression II

Martin Haugh

Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
Recall our linear regression model:

\[ Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \epsilon. \]

Have seen how to fit this model via least squares but often preferable to use other solutions techniques as they often result in:

1. Superior prediction accuracy, especially when \( p \) is close to \( N \)
   - in fact if \( p > N \) then least squares does not yield a unique \( \hat{\beta} \)
   - superior prediction will result from controlling overfitting and identifying a good bias-variance trade-off.
2. Better interpretability via the exclusion of irrelevant variables.

These other solution techniques include:

1. **Subset selection** where only a subset of the independent variables are retained.
2. **Shrinkage methods** where coefficients are shrunk towards zero
   - typically achieved via regularization.

**Cross-validation** often used to select the specific model.

We will consider only shrinkage methods here.
Shrinkage Methods

Will focus mainly on two shrinkage methods:

1. **Ridge regression** where we solve:

   \[
   \min_{\beta} \left\{ \frac{1}{2} \| y - X\beta \|^2 + \lambda \cdot \frac{1}{2} \| \beta \|^2 \right\}.
   \]

2. The **Least Absolute Shrinkage and Selection Operator** or **Lasso** solves

   \[
   \min_{\beta} \left\{ \frac{1}{2} \| y - X\beta \|^2 + \lambda \| \beta \|_1 \right\} \quad \| \beta \|_1 = \sum_{j=1}^{n} |\beta_j|
   \]

   As \( \lambda \) increases, coefficients will abruptly drop to zero.

**Question:** How should we choose \( \lambda \)?

**Note:** Shrinkage methods can also be applied to classification problems!
Ridge Regression

Ridge regression solves

\[
\hat{\beta}^R = \arg\min_{\beta} \left\{ \frac{1}{2} \| y - X\beta \|^2 + \frac{\lambda}{2} \sum_{j=1}^{p} \beta_j^2 \right\}
\]

- shrinks regression coefficients towards 0 by imposing a penalty on their size
- \( \lambda \) is a complexity parameter that controls the amount of shrinkage.

An equivalent formulation is

\[
\hat{\beta}^R = \arg\min_{\beta} \left\{ \frac{1}{2} \| y - X\beta \|^2 \right\}
\]

subject to \( \sum_{j=1}^{p} \beta_j^2 \leq s \)

It is standard (why?) to scale and standardize inputs before applying ridge regression.
Ridge Regression

Note $\beta_0$ is generally not shrunk so that procedure does not depend on origin chosen for $Y$.

To handle this and use matrix notation we can split estimation into two steps:

1. Set $\hat{\beta}_0 = \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i$

2. Center the inputs so that $x_{ij} \rightarrow x_{ij} - \bar{x}_j$.

Now estimate $\beta_1, \ldots, \beta_p$ using ridge regression without intercept and using the centered $x_{ij}$’s.

Dropping $\beta_0$ from $\beta$, the ridge regression of step 2 therefore solves

$$\hat{\beta}^R = \arg\min_{\beta} \left\{ \frac{1}{2} \| y - X\beta \|^2 + \frac{\lambda}{2} \beta^\top \beta \right\}$$

which has solution

$$\hat{\beta}^R = (X^\top X + \lambda I)^{-1} X^\top y.$$  \hspace{1cm} (6)
Ridge Regression

Note that $\hat{\beta}^R$ is obtained as the solution of a least squares problem except that a positive term, i.e. $\lambda$, has been added to the diagonal of $X^TX$

- this makes the problem non-singular, even if $X^TX$ does not have full rank
- this was the main motivation for ridge regression when first introduced.

Ridge regression estimates can easily be obtained in a Bayesian setting

- prior distribution on each $\beta_i$ is independent normal $N(0, \tau^2)$
- then with $\lambda := \sigma^2/\tau^2$, obtain $\hat{\beta}^R$ as mean of posterior distribution.

Figure 6.4 from ISLR displays $\hat{\beta}^R$ for various values of $\lambda$ and $||\hat{\beta}^R_\lambda||_2/||\hat{\beta}||_2$

- can interpret $||\hat{\beta}^R_\lambda||_2/||\hat{\beta}||_2$ as a measure of the total shrinkage achieved
- note that we recover the least squares solution as $\lambda \to 0$. 
Figure 6.4 from ISLR: The standardized ridge regression coefficients are displayed for the Credit data set, as a function of $\lambda$ and $||\hat{\beta}^R_\lambda||_2 / ||\hat{\beta}_\lambda||_2$.

Note that as $\lambda$ increases coefficients are shrunk towards zero. Also note that coefficients are generally non-zero for any value of $\lambda$ - so ridge regression does not result in sparse models.
Selecting $\lambda$ Via Cross-Validation

Figure 6.12 from ISLR: Cross-validation errors that result from applying ridge regression to the Credit data set with various value of $\lambda$. Right: The coefficient estimates as a function of $\lambda$. The vertical dashed lines indicate the value of $\lambda$ selected by cross-validation.

Using cross-validation to select $\lambda$ for the Credit data set results in only a modest amount of shrinkage.

And the cv error is relatively insensitive to choice of $\lambda$ here - so little improvement over least squares solution.
Figure 6.5 from ISLR: Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on a simulated data set, as a function of $\lambda$ and $||\hat{\beta}_R^\lambda||_2/||\hat{\beta}_\lambda||_2$. The horizontal dashed lines indicate the minimum possible MSE. The purple crosses indicate the ridge regression models for which the MSE is smallest.

Ridge regression (and Lasso) often (significantly) outperform least-squares because it is capable (through selection of $\lambda$) of trading off a small increase in bias for a potentially much larger decrease in variance.
The Lasso

Recall that the Lasso solves

$$\min_{\beta} \left\{ \frac{1}{2} \|y - X\beta\|^2 + \lambda \|\beta\|_1 \right\}$$

where $\|\beta\|_1 := \sum_{j=1}^n |\beta_j|$.

Penalizing the 1-norm ensures that coefficients will abruptly drop to zero as $\lambda$ increases – results in superior interpretability.

The Lasso can also be formulated by constraining $\|\beta\|_1$:

$$\hat{\beta}^L = \arg\min_{\beta} \left\{ \frac{1}{2} \|y - X\beta\|^2 \right\}$$

subject to

$$\sum_{j=1}^p |\beta_j| \leq s$$

Unlike ridge regression, a closed-form solution is not available for the Lasso - but it can be formulated as a convex quadratic optimization problem and is therefore easy to solve numerically.
Figure 6.6 from ISLR: The standardized lasso coefficients on the Credit data set are shown as a function of $\lambda$ and $\|\hat{\beta}_\lambda^L\|_1/\|\hat{\beta}_\lambda\|_1$.

Note how coefficients abruptly drop to 0 as $\lambda$ increases in Figure 6.6 - contrast this with ridge regression!

Lasso results in sparse models then and can be viewed as a method for subset selection.
Figure 6.9 from ISLR: Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso. The simulated data is similar to that in Figure 6.8, except that now only two predictors are related to the response. Right: Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their $R^2$ on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

Figure 6.9 displays results from a simulated data set with $p = 45$ predictors – but the response $Y$ is a function of only 2 of them!
Selecting $\lambda$ Via Cross-Validation

Figure 6.13 from ISLR: Left: Ten-fold cross-validation MSE for the lasso, applied to the sparse simulated data set from Figure 6.9. Right: The corresponding lasso coefficient estimates are displayed. The vertical dashed lines indicate the lasso fit for which the cross-validation error is smallest.

Note how the optimal $\lambda$ (chosen via cross-validation) correctly identifies the model with the 2 predictors

- contrast with least squares solution at far right of right-hand figure!
Lasso Versus Ridge Regression

Figure 6.7 from ISLR: Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.

Contours of the error and constraint functions of the formulations in (5) and (7) are displayed in Figure 6.7.

This perspective makes it clear why Lasso results in a sparse solution whereas ridge regression does not.
Ridge Regression Versus Lasso

The following example (taken from ISLR) provides further intuition for why Lasso results in sparse solutions and ridge regression does not. We assume:

- $N = p$.
- $X$ is a diagonal matrix with 1’s on the diagonal.
- There is no intercept term.

**Least squares** then solves $\min_{\beta_1, \ldots, \beta_p} \sum_{j=1}^{N} (y_j - \beta_j)^2$

Solution is $\hat{\beta}_j = y_j$.

**Ridge regression** solves $\min_{\beta_1, \ldots, \beta_p} \sum_{j=1}^{N} (y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

Can check solution is $\hat{\beta}_j^R = y_j / (1 + \lambda)$.

**Lasso** solves $\min_{\beta_1, \ldots, \beta_p} \sum_{j=1}^{N} (y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$

Can check solution is

$$\hat{\beta}_j^L = \begin{cases} 
  y_j - \lambda/2, & \text{if } y_j > \lambda/2; \\
  y_j + \lambda/2, & \text{if } y_j < -\lambda/2; \\
  0, & \text{if } |y_j| \leq \lambda/2.
\end{cases}$$
High Dimensional Problems

Traditionally problems in statistics were low-dimensional with \( p < N \) and often \( p << N \).

But many modern settings have \( p > N \). For example:

1. Classical statistics might attempt to predict blood pressure as a function of age, gender, and body-mass-index (BMI). Modern methods might also use measurements for approx 500k single nucleotide polymorphisms (SNPs).

2. Online advertisers may want to predict the purchasing behavior of someone using a search engine. Dummy variables for each of \( p \) search terms might be included as predictors with \( p_i = 1 \) if the \( i^{th} \) term was previously searched by the user and \( p_i = 0 \) otherwise.

3. Speech recognition problems where we have speech samples for \( N \) speakers. To represent a speech sample as a numeric vector we require very large \( p \).

Need to be very careful in these high-dimensional settings where (unique) least squares solutions do not even exist.

Even if \( p \) is smaller than but still close to \( N \) then similar problems still arise. Similar observations hold true for classification problems that use classical approaches such as LDA, QDA, logistic regression etc.
Figure 6.22 from ISLR: Left: Least squares regression in the low-dimensional setting. Right: Least squares regression with \( n = 2 \) observations and two parameters to be estimated (an intercept and a coefficient).

Problem in Fig. 6.22 is low dimensional but demonstrates what can go wrong when we have too little data relative to problem dimension
- this certainly occurs when \( p \approx N \)
- saw similar issues with the case-study in Regression I slides.

When \( p \geq N \) least squares can fit the data perfectly and so \( R^2 \) will equal 1
- but likely that massive over-fitting is taking place.
**Issues in High Dimensions**

**Figure 6.23 from ISLR**: On a simulated example with \( n = 20 \) training observations, features that are completely unrelated to the outcome are added to the model. Left: The \( R^2 \) increases to 1 as more features are included. Center: The training set MSE decreases to 0 as more features are included. Right: The test set MSE increases as more features are included.

Note that in Figure 6.23 the features are completely unrelated to the response! Estimating test error is therefore particularly vital in these settings – but \( C_p \), AIC and BIC are not suitable due to difficulty in estimating \( \sigma^2 \).

The solution is to restrict the choice of models which is exactly what subset selection, ridge regression, lasso etc. do.
Figure 6.24 from ISLR: The lasso was performed with $n = 100$ observations and three values of $p$, the number of features. Of the $p$ features, 20 were associated with the response. The boxplots show the test MSEs that result using three different values of the tuning parameter $\lambda$ in (6.7). For ease of interpretation, rather than reporting $\lambda$, the degrees of freedom are reported; for the lasso this turns out to be simply the number of estimated non-zero coefficients. When $p = 20$, the lowest test MSE was obtained with the smallest amount of regularization. When $p = 50$, the lowest test MSE was achieved when there is a substantial amount of regularization. When $p = 2,000$ the lasso performed poorly regardless of the amount of regularization, due to the fact that only 20 of the 2,000 features truly are associated with the outcome.
Issues in High Dimensions

Note results in Figure 6.24 where only 20 features were relevant.

Degrees-of-freedom, df(λ), is reported instead of λ

- df(λ) = number of non-zero coefficient estimates in the lasso solution
- much easier to interpret!

When $p = 20$ or $p = 50$ we see the importance of choosing a good value of $λ$.
But we also see that lasso performed poorly when $p = 2000$

- because test error tends to increase with $p$ unless the new features are actually informative

- note the implications of this observation – there is a cost to be paid for blindly adding new features to a model even when regularization is employed!

Multi-collinearity is clearly present in high-dimensional problems – therefore cannot hope to identify the very best predictors

- instead hope to identify good predictors.

Note that linear models – which we have been considering – are generally popular for high dimensional problems. Why?
Introduction to Machine Learning
Supervised Learning: Classification

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
What is Classification?

- Goal is to predict a categorical outcome from a vector of inputs
  - the inputs can be quantitative, ordinal, or categorical
  - inputs could also be images, speech, text, networks, temporal data, spatial data etc.
- Classification algorithms generally require inputs to be encoded in quantitative form
  - can result in very high dimensional problems!
- The simplest and most common type of classification is binary classification
  - email classification: spam or not spam?
  - sentiment analysis
    - is the movie review good or bad?
    - is this good news for the stock or bad news?
  - fraud detection
  - revenue management: will a person buy or not?
  - medical diagnosis: does a patient have a disease or not?
  - will somebody vote for Obama or not?
  - is somebody a terrorist or not?
- But also have classification problems with multiple categories.
What is Classification?

• Classification often used as part of a decision support system.
• There are many(!) different classification algorithms
  - will cover some of the best known algorithms
  - but will not have time to cover all of them such as neural networks, ensemble methods etc.
  - will also cover support vector machines.
• Most classification algorithms can be categorized as generative or discriminative.
• Classification algorithms learn a classifier using training data
  - then used to predict category or class for new inputs or test data.
• Will use $X$ or $x$ to denote vector of inputs and $G$ or $Y$ to denote category or class.
**Generative Classification Algorithms**

- **Generative** methods focus on modeling $P(X, G)$ and then use $\hat{G}(X)$ as a classifier where

$$
\hat{G}(X) := \arg\max_G \hat{P}(G|X) = \arg\max_G \frac{\hat{P}(X|G)\hat{P}(G)}{\sum_i \hat{P}(X|G_i)\hat{P}(G_i)}
$$

(8)

- Examples include **linear discriminant analysis (LDA)**, **quadratic discriminant analysis (QDA)** and **naive Bayes**.

- **LDA and QDA** assume Gaussian densities for $\hat{P}(X|G)$.

- **Naive Bayes** assumes

$$
P(X|G) = \prod_j P(X_j|G)
$$

so the features are independent **conditional** on $G$.

- a strong and generally unrealistic assumption but naive Bayes still often works very well in practice.
**Discriminative Classification Algorithms**

- Discriminative methods focus on modeling $P(G|X)$ directly
  - examples include least squares or regression-based classifiers, logistic regression and Bayesian logistic regression.

- Discriminative methods may also focus on minimizing the expected classification error directly without making any assumptions regarding $P(X, G)$ or $P(G|X)$.

Examples include:
- decision or classification trees
- $k$-nearest neighbors
- support vector machines (SVMs)
- neural networks

and many others.
Optimal Bayes Classifier

- Consider again the generative framework where we have the joint distribution function, \( P(X, G) \), for the feature vector, \( X \), and the associated category, \( G \).
- For a given classifier, \( \hat{G}(\cdot) \), and a given loss function, \( L(G, \hat{G}(X)) \), the expected prediction error (EPE) is given by

\[
\text{EPE} = \mathbb{E}_{X,G} \left[ L(G, \hat{G}(X)) \right] \\
= \mathbb{E}_X \left[ \mathbb{E}_G \left[ L(G, \hat{G}(X)) | X \right] \right] \\
= \mathbb{E}_X \left[ \sum_{k=1}^{K} L(G_k, \hat{G}(X)) P(G_k|X) \right]
\]

- We wish to minimize the EPE as a function of \( X \) and so we can do it pointwise to obtain

\[
\hat{G}(x) = \operatorname*{argmin}_{G \in \mathcal{G}} \sum_{k=1}^{K} L(G_k, G) P(G_k|X = x) \quad (9)
\]
**Optimal Bayes Classifier**

- A commonly used loss function for classification problems is the $0 - 1$ loss function which assigns a loss of 0 to correctly classified data-points and 1 to those that are incorrectly classified.
- If we now assume this loss function then (9) reduces to

$$
\hat{G}(x) = \arg\min_{G \in G} [1 - P(G|X = x)]
= \arg\max_{G \in G} P(G|X = x)
$$

so that we classify to the most probable class.
- This is known as the Bayes classifier and the error rate of this classifier is called the Bayes rate.
- The Bayes rate is the best possible error rate for the $0 - 1$ loss function - but generally not achievable in practice because we do not know $P(X, G)$.
- But it’s often available in simulated problems where we wish to evaluate the performance of other classification algorithms.
Naive Bayes

• Recall that naive Bayes is a generative classifier that estimates $P(\mathbf{X}, G)$.

• It assumes

$$P(\mathbf{X}|G) = \prod_j P(X_j|G)$$

(10)

so the features are independent conditional on $G$.

• Since $P(\mathbf{X}, G) = P(\mathbf{X}|G)P(G)$ naive Bayes estimates $P(G)$ and the $P(X_j|G)$’s separately via MLE and then classifies according to

$$\hat{G}(\mathbf{x}) = \arg\max_{G \in \mathcal{G}} P(G|\mathbf{X} = \mathbf{x})$$

$$= \arg\max_{G \in \mathcal{G}} \hat{P}(G)\hat{P}(\mathbf{x}|G)$$

$$= \arg\max_{G \in \mathcal{G}} \hat{P}(G)\prod_i \hat{P}(x_i|G)$$

• Assumption (10) is strong and generally unrealistic

  - but when $\mathbf{X}$ is high-dimensional and categorical estimating $P(\mathbf{X}, G)$ is generally (why?) not possible

  - so assumption (10) makes estimation much easier

  - and naive Bayes still often works very well in practice!
Naive Bayes: There are 2 equiprobable classes and the class-conditional densities are bivariate Normal. The assumptions of naive Bayes do not apply. Why?
Naive Bayes: The contours of the fitted class-conditional densities. These densities are also assumed to be bivariate Normal. The naive Bayes classifier is given by the red curve.
Naive Bayes and Text Classification

- One of the reasons naive Bayes often works well is because the data cannot support a more complex classifier.
  - this is the bias-variance decomposition again.
- Has been very successful in text classification or sentiment analysis.
  - e.g. is an email spam or not?
- But how would you encode text like an email into a numerical input vector?
- A simple and common way to do this is via the bag-of-words model.
  - completely ignores the ordering of the words.
  - stop-words such as “the”, “and”, “of”, “about” etc. are thrown out.
  - words such as “walk”, “walking”, “walks” etc. all identified as “walk”.
- Email classification is then done by assuming that a given email comes from either a “spam” bag or a “non-spam” bag.
  - naive Bayes assumes \( P(\text{spam}|X) \propto P(\text{spam}) \prod_{\text{word} \in \text{email}} P(\text{word}|\text{spam}) \).
- Bag-of-words also often used in document retrieval and document classification.
  - leads to the term-document matrix.
  - also then need a measure of similarity between documents, e.g. cosine distance possibly in TF-IDF version of term-document matrix.
Linear Discriminant Analysis (LDA)

- LDA is a generative model that assumes the class-conditional densities, \( f_k(x) \), are Gaussian with a common covariance matrix so that

\[
 f_k(x) = \frac{1}{(2\pi)^{M/2} |\Sigma|^{1/2}} e^{-\frac{1}{2} (x - \mu_k)^\top \Sigma^{-1} (x - \mu_k)}
\]

- Writing \( \pi_k \) for \( P(G = k) \) we see from (8) that the LDA classifier satisfies

\[
 \hat{G}(x) = \arg\max_k \frac{\hat{f}_k(x) \hat{\pi}_k}{\sum_{j=1}^K \hat{f}_j(x) \hat{\pi}_j} \tag{11}
\]

where we have used \( \hat{f}_k(\cdot) \) and \( \hat{\pi}_k \) to denote MLE estimates of \( f_k(\cdot) \) and \( \pi_k \) with

\[
 \hat{\pi}_k = N_k / N \\
 \hat{\mu}_k = \sum_{g_i = k} x_i / N_k \\
 \hat{\Sigma} = \frac{\sum_{k=1}^K \sum_{g_i = k} (x_i - \hat{\mu}_k) (x_i - \hat{\mu}_k)^\top}{N - K} \tag{12}
\]
Linear Discriminant Analysis (LDA)

- We can also calculate the log-ratio between two classes, $k$ and $l$, to get

$$
\log \frac{P(G = k | X = x)}{P(G = l | X = x)} = \log \frac{\hat{\pi}_k}{\hat{\pi}_l} - \frac{1}{2} (\hat{\mu}_k + \hat{\mu}_l)^\top \hat{\Sigma}^{-1} (\hat{\mu}_k - \hat{\mu}_l)
+ x^\top \hat{\Sigma}^{-1} (\hat{\mu}_k - \hat{\mu}_l)
$$

(13)

- Based on (13) we can define the linear discriminant functions,

$$
\delta_k(x) := x^\top \hat{\Sigma}^{-1} \hat{\mu}_k + \log \hat{\pi}_k - \frac{1}{2} \hat{\mu}_k^\top \hat{\Sigma}^{-1} \hat{\mu}_k
$$

(14)

for $k = 1, \ldots, K$.

- The LDA classifier of (11) reduces to

$$
\hat{G}(x) = \arg\max_k \delta_k(x).
$$
Figure 4.6 from ISLR: An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with $p = 2$, with a class-specific mean vector and a common covariance matrix. Left: Ellipses that contain 95% of the probability for each of the three classes are shown. The dashed lines are the Bayes decision boundaries. Right: 20 observations were generated from each class, and the corresponding LDA decision boundaries are indicated using solid black lines. The Bayes decision boundaries are once again shown as dashed lines.
The Default Data from ISLR

ISLR use LDA to obtain a classification rule for default / non-default on the basis of: (i) credit card balance and (ii) student status.

There were 10,000 training samples and the overall default rate was 3.33%.

The training error rate of LDA was 2.75%. But how good is this?

- note training error rate will usually be biased low and therefore lower than test / generalization error
- for comparison, how well does the useless classifier that always predicts non-default do?

We are often interested in breaking out the (training) error rate into the two possible types of error:

1. False positives
2. False negatives.

This leads to the so-called confusion matrix.
The Confusion Matrix

<table>
<thead>
<tr>
<th>Predicted default status</th>
<th>True default status</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>9,644</td>
<td>252</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>23</td>
<td>81</td>
</tr>
<tr>
<td>Total</td>
<td>9,667</td>
<td>333</td>
<td>10,000</td>
</tr>
</tbody>
</table>

Table 4.4 from ISLR: A confusion matrix compares the LDA predictions to the true default statuses for the 10,000 training observations in the Default data set. Elements on the diagonal of the matrix represent individuals whose default statuses were correctly predicted, while off-diagonal elements represent individuals that were misclassified. LDA made incorrect predictions for 23 individuals who did not default and for 252 individuals who did default.

Note that the LDA classifier only predicts $\frac{81}{81 + 252} = 24.3\%$ of the true defaults.

Do you think this would be acceptable? If not, do we need to abandon LDA or can we somehow “rescue” it?
The Confusion Matrix for a Different Threshold

<table>
<thead>
<tr>
<th>Predicted default status</th>
<th>True default status</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>9,432</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>138</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td>9,570</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>235</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>195</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td>430</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td>9,667</td>
</tr>
<tr>
<td></td>
<td></td>
<td>333</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td>10,000</td>
</tr>
</tbody>
</table>

Table 4.5 from ISLR: A confusion matrix compares the LDA predictions to the true default statuses for the 10,000 training observations in the Default data set, using a modified threshold value that predicts default for any individuals whose posterior default probability exceeds 20%.

We can rescue LDA by simply adjusting the threshold to emphasize one type of error over another.

In Table 4.5 we “predict“ default if the LDA model has

\[
\text{Prob}(\text{default} = \text{Yes} | X = x) > 0.2.
\]

What happens the overall training error rate with this new rule? Do we care?
The Tradeoff from Modifying the Threshold

**Figure 4.7 from ISLR:** For the Default data set, error rates are shown as a function of the threshold value for the posterior probability that is used to perform the assignment. The black solid line displays the overall error rate. The blue dashed line represents the fraction of defaulting customers that are incorrectly classified, and the orange dotted line indicates the fraction of errors among the non-defaulting customers.

Domain specific knowledge required to decide on appropriate threshold - the **ROC curve** often used for this task.
Previous slide: Figure 4.8 from ISLR: A ROC curve for the LDA classifier on the Default data. It traces out two types of error as we vary the threshold value for the posterior probability of default. The actual thresholds are not shown. The true positive rate is the sensitivity: the fraction of defaulters that are correctly identified, using a given threshold value. The false positive rate is 1-specificity: the fraction of non-defaulters that we classify incorrectly as defaulters, using that same threshold value. The ideal ROC curve hugs the top left corner, indicating a high true positive rate and a low false positive rate. The dotted line represents the â€œno informationâ€ classifier; this is what we would expect if student status and credit card balance are not associated with probability of default.

Overall performance of the classifier – summarized over all possible thresholds – is given by the AUC or “area under the curve”.

The ideal classifier will have an AUC of 1 and will hug top left corner.

ROC curves are useful for comparing classifiers as they factor in all possible thresholds.

Clearly we can alter the threshold for many classifiers and so confusion matrix / ROC curve can be constructed for most binary classifiers.
Quadratic Discriminant Analysis (QDA)

- If we drop the assumption of equal covariances, then we obtain quadratic discriminant functions

\[
d_k(x) := -\frac{1}{2} \log |\hat{\Sigma}_k| - \frac{1}{2} (x - \hat{\mu}_k)^\top \hat{\Sigma}_k^{-1} (x - \hat{\mu}_k) + \log \hat{\pi}_k
\]

- The QDA classifier is then

\[
\hat{G}(x) = \arg\max_k \delta_k(x)
\]

- the decision boundaries between each pair of classes then given by quadratic functions of \(x\).

- LDA using linear and quadratic features generally gives similar results to QDA
  - QDA generally preferred due to greater flexibility at the cost of more parameters to estimate.

- LDA and QDA are very popular and successful classifiers
  - probably because of the bias-variance decomposition and because the data can often "only support simple decision boundaries".
Figure 4.6 from HTF: Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $X_1$, $X_2$, $X_1 X_2$, $X_1^2$, $X_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.
**Figure 4.1 from HTF:** The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space $X_1, X_2, X_1 X_2, X_1^2, X_2^2$. Linear inequalities in this space are quadratic inequalities in the original space.
LDA v QDA

Figure 4.9 from ISLR: Left: The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with $\Sigma_1 = \Sigma_2$. The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA. Right: Details are as given in the left-hand panel, except that $\Sigma_1 \neq \Sigma_2$. Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.

Note that LDA is superior when true boundary is (close to) linear. Why do you think this happens?
Logistic Regression

- Suppose we have 2 possible classes encoded as \( y \in \{0, 1\} \).
- We assume
  \[
  P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{x})}{1 + \exp(\mathbf{w}^\top \mathbf{x})}
  \]
  where \( \mathbf{w} \) is an \( m + 1 \)-parameter vector and we assume the first element of \( \mathbf{x} \) is the constant 1.
- It follows that
  \[
  P(y = 0|\mathbf{x}, \mathbf{w}) = 1 - P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(\mathbf{w}^\top \mathbf{x})}
  \]
- Given \( N \) IID data-points we can write the likelihood as
  \[
  L(\mathbf{w}) = \prod_{i=1}^{N} p_i(\mathbf{w})^{y_i} (1 - p_i(\mathbf{w}))^{1-y_i}
  \]
  where \( p_i(\mathbf{w}) := P(y_i = 1|\mathbf{x}_i, \mathbf{w}) \).
- Then obtain \( \mathbf{w} \) by maximizing the log-likelihood
  \[
  l(\mathbf{w}) = \sum_{i=1}^{N} \left( y_i \mathbf{w}^\top \mathbf{x}_i - \log (1 + \exp(\mathbf{w}^\top \mathbf{x}_i)) \right)
  \]
MLE Estimation

• In order to maximize $l(w)$ we set its derivatives to 0 and obtain

$$\frac{\partial l(w)}{\partial w} = \sum_{i=1}^{N} x_i (y_i - p_i(w)) = 0$$

– so have $m + 1$ non-linear equations in $w$.

• Since first component of each $x_i$ is 1 we see that at the MLE solution we have $\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p_i(w)$.

• We can solve (16) iteratively using Newton-Raphson steps

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \left( \frac{\partial^2 l(w)}{\partial \mathbf{w} \partial \mathbf{w}^\top} \right)^{-1} \frac{\partial l(w)}{\partial \mathbf{w}}$$

where the partial derivatives are evaluated at $\mathbf{w}_{old}$.

• Let $\mathbf{V}$ be the $N \times N$ diagonal matrix with $V_{i,i} = p_i(w)(1 - p_i(w))$.

• Let $\mathbf{X}$ be the $N \times (m + 1)$ matrix of $x_i$'s.
MLE Via Iteratively Reweighted Least Squares

We can then write

\[
\frac{\partial l(w)}{\partial w} = X^\top (y - p)
\]

\[
\frac{\partial^2 l(w)}{\partial w \partial w^\top} = -X^\top VX
\]

so that (17) becomes

\[
w_{new} = w_{old} + \left( X^\top V_{old} X \right)^{-1} X^\top (y - p_{old})
\]

\[
= \left( X^\top V_{old} X \right)^{-1} X^\top V_{old} \left( Xw_{old} + V_{old}^{-1} (y - p_{old}) \right)
\]

\[
= \left( X^\top V_{old} X \right)^{-1} X^\top V_{old} z_{old}
\]

where \( z_{old} := Xw_{old} + V_{old}^{-1} (y - p_{old}) \) and where \( V_{old} \) and \( p_{old} \) are \( V \) and \( p \), respectively, evaluated at \( w_{old} \).
We now iterate (18) until convergence
- which typically does occur.

If convergence does not take place then a slight variant can be used which will guarantee convergence
- possibly to infinity if the two classes are \textit{linearly separable}

If classes are linearly separable, then can handle this via regularization or equivalently by placing a prior on $w$.

Note that $w_{new}$ as given by (18) also satisfies

$$w_{new} = \arg\min_w (z_{old} - Xw)^\top V_{old} (z_{old} - Xw)$$

– a weighted least-squares problem and hence iterating (18) is often called \textit{iteratively reweighted least squares}.

Even in the 2-class case, logistic regression has many advantages over classification by least squares
- not sensitive to extreme or outlying points.
Figure 4.4 from Bishop: The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve). The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.
Multinomial Logistic Regression

- When there are $K > 2$ classes we can use multinomial or multi-class logistic regression.
- Let $G_1, \ldots, G_K$ denote the $K$ categories.
- We then assume

$$P(G_k | \mathbf{x}, \mathbf{w}) = \frac{\exp (\mathbf{w}_k^\top \mathbf{x})}{\sum_j \exp (\mathbf{w}_j^\top \mathbf{x})}$$

and as before we can use maximum likelihood to estimate the $\mathbf{w}_k$’s.
- As with 2-class case, this can be done via an iterative numerical scheme such as Newton-Raphson.
Figure 4.5 from Bishop: Example of a synthetic data set comprising three classes, with training data points denoted in red (×), green (+), and blue (○). Lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. On the left is the result of using a least-squares discriminant. We see that the region of input space assigned to the green class is too small and so most of the points from this class are misclassified. On the right is the result of using logistic regressions showing correct classification of the training data.
• Have already seen how expanding the feature space can provide much greater flexibility
  - e.g. using LDA with quadratic basis functions on Slide 96
• Non-separable data in original feature space may become separable when features are projected into a higher-dimensional space.
• In fact the kernel "trick" allows us to project into infinite-dimensional spaces
  - will discuss the kernel trick later in the course in the context of support vector machines (SVMs) and principal component analysis (PCA).
Figure 4.12 from Bishop: Illustration of the role of nonlinear basis functions in linear classification models. The left plot shows the original input space \((x_1, x_2)\) together with data points from two classes labeled red and blue. Two ‘Gaussian’ basis functions \(\phi_1(x)\) and \(\phi_2(x)\) are defined in this space with centers shown by the green crosses and with contours shown by the green circles. The right-hand plot shows the corresponding feature space \((\phi_1, \phi_2)\) together with the linear decision boundary obtained given by a logistic regression model of the form discussed in Section 4.3.2. This corresponds to a nonlinear decision boundary in the original input space, shown by the black curve in the left-hand plot.
A Comparison of Classification Methods

Figure 4.10 from ISLR: Boxplots of the test error rates for each of the linear scenarios described in the main text.

See Section 4.5 of ISLR for description of scenarios.

Here we simply note that in Fig 4.10 the true boundary in each scenario is linear - and the linear classifiers perform best.
A Comparison of Classification Methods

Figure 4.11 from ISLR: Boxplots of the test error rates for each of the non-linear scenarios described in the main text.

Here we note that in Fig 4.11 the true boundary in each scenario is non-linear - and the linear classifiers are no longer the best.

Aside: KNN-CV refers to KNN with K chosen via cross-validation - to be studied soon!
Introduction to Machine Learning
Supervised Learning: Support Vector Machines

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
Introduction to Support Vector Machines

• Support vector machines are non-probabilistic binary linear classifiers.
• The use of basis functions and the kernel trick mitigates the constraints of being a linear classifier
  – in fact SVMs are particularly associated with the kernel trick.
• The use of kernels is the main reason why SVM classifiers are trained via the dual optimization problem.
• Only a small subset of data-points are required to define the SVM classifier
  - these points are called support vectors.
• SVMs are very popular classifiers and applications include
  - text classification
  - outlier detection
  - face detection
  - database marketing
  - and many others.
• SVMs are also used for multi-class classification.
• Also have support vector regression.
The Separable Case

- There are two classes and assume (for now) that they are linearly separable.
- We have training data $x_1, \ldots, x_n$ with corresponding targets, $t_1, \ldots, t_n$ with each $t_i \in \{-1, 1\}$.
- We consider a classification rule of the form of the form

\[
h(x) = \text{sign} \left( w^\top x + b \right)
\]

where $y(x) := w^\top x + b$.

- Note we can re-scale $(w, b)$ without changing the decision boundary.
- Therefore choose $(w, b)$ so that training points closest to boundary satisfy $y(x) = \pm 1$

  - see left-hand component of Figure 7.1 from Bishop.
- Let $x_1$ be closest point from class with $t_1 = -1$ so that $w^\top x_1 + b = -1$.
- And let $x_2$ be closest point from class with $t_2 = 1$ so that $w^\top x_2 + b = 1$. 
Figure 7.1 from Bishop: The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.
Geometry of Maximizing the Margin

- Recall the perpendicular distance of a point $x$ from the hyperplane, $\mathbf{w}^\top x + b = 0$, is given by $|\mathbf{w}^\top x + b|/||\mathbf{w}||$.

- Therefore distance of closest points in each class to the classifier, i.e. hyperplane, is $1/||\mathbf{w}||$.

- A support vector machine seeks the maximum margin classifier that separates all the data
  - seems like a good idea
  - but can also be justified by statistical learning theory.

- Maximizing the margin, $1/||\mathbf{w}||$, is equivalent to minimizing $f(\mathbf{w}) := \frac{1}{2} \mathbf{w}^\top \mathbf{w}$.

- Therefore obtain the following primal problem for the separable case:

\[
\begin{align*}
\min_{\mathbf{w}, b} & \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w} \\
\text{subject to} & \quad t_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n
\end{align*}
\]  
(19)

(20)

- Note that (20) ensures that all the training points are correctly classified.
The Primal Problem

• The primal problem is a **quadratic program** with linear inequality constraints
  - moreover it is **convex** and therefore has a unique minimum.
• From the problem’s geometry should be clear that only the points closest to the boundary are required to define the optimal hyperplane
  - see right-hand component of Figure 7.1 from Bishop.
  - these are called the **support vectors**
  - and will see that the solution can be expressed using only these points.
• Could stop here but need to go to the corresponding **dual problem** to fully understand SVMs.
The Dual Problem in Separable Case

- We use a Lagrange multiplier $\alpha_i \geq 0$ for each constraint in (20).
- The Lagrangian is then given by

$$L(w, b; \alpha) = \frac{1}{2} w^T w + \sum_{i=1}^{n} \alpha_i \left(1 - t_i \left(w^T x_i + b\right)\right)$$  \hspace{1cm} (21)

- We now wish to solve for $g(\alpha) := \min_{w,b} L(w, b; \alpha)$.
- Note that (why?) $g(\alpha) \leq f(w^*)$ where $(w^*, b^*)$ is the optimal solution to the primal problem.
- Can therefore formulate the dual problem:

$$\max_{\alpha \geq 0} g(\alpha)$$  \hspace{1cm} (22)

- Since primal problem is convex it follows that minimum of primal equals maximum of dual problem
  - i.e. $g(\alpha^*) = f(w^*)$ where $\alpha^*$ is the optimal solution to the dual problem
  - this is strong duality.
The Dual Problem in Separable Case

• To solve (22) first need to solve for $g(\alpha)$: the first order conditions (FOC) are

\[
\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i t_i = 0
\]

\[
\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i t_i \mathbf{x}_i
\] (23)

• The FOC are necessary and sufficient (why?) for optimality. Can substitute them into (21) to obtain

\[
L(w, b; \alpha) = \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \left( 1 - t_i \left( \sum_{j=1}^{n} \alpha_j t_j \mathbf{x}_j^T \mathbf{x}_i + b \right) \right)
\]

\[
= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j
\]
The Dual Problem in Separable Case

- Then dual problem in the separable case reduces to

\[
\begin{align*}
\max_{\alpha \geq 0} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j x_i^\top x_j \\
\text{subject to} & \sum_{i=1}^{n} \alpha_i t_i = 0
\end{align*}
\]  (24)

- also a convex quadratic program, but now with a single linear constraint.

- The complementary slackness conditions imply that only the support vectors will have non-zero \( \alpha \)'s in the optimum solution.

- Let \( \alpha^* \) be the optimal solution to the dual problem. Then (23) yields

\[
\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* t_i \mathbf{x}_i
\]

and we obtain \( b^* \) by noting that for any \( i \) with \( \alpha_i^* > 0 \)

\[
t_i \left( \mathbf{w}^* \top \mathbf{x}_i + b^* \right) = 1.
\]
Introduction to Machine Learning

The Kernel Trick

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
The Kernel Trick

- The kernel-trick is a very commonly used technique in regression, classification, PCA etc. that allows the problem to be easily embedded in much higher dimensional spaces and often infinite dimensional spaces - but without having to do an infinite amount of work.

- Suppose instead of using $x \in \mathbb{R}^m$ to describe the inputs we instead use a feature map, $\phi(x)^\top \in \mathbb{R}^M$, often with $M >> m$.

- Then if the data is linearly separable in $\mathbb{R}^M$ can solve the same dual problem as (24) and (25) except we replace $x_i^\top x_j$ with $\phi(x_i)^\top \phi(x_j)$.

- Can obtain corresponding optimal $b^*$ via

$$b^* = t_j - \sum_{i=1}^{n} \alpha_i^* t_i \phi(x_i)^\top \phi(x_j) \text{ for any } \alpha_j^* > 0 \tag{26}$$

and, for a new data-point $x$, the prediction

$$\text{sign} \left( w^*^\top \phi(x) + b^* \right) = \text{sign} \left( \sum_{i=1}^{n} \alpha_i^* t_i \phi(x_i)^\top \phi(x) + b^* \right). \tag{27}$$
A Detour on Kernels

- Define the Gram matrix, $K = \phi \phi^\top$ to be the $n \times n$ matrix with
  \[ K_{ij} := \phi(x_i)^\top \phi(x_j) =: k(x_i, x_j) \]

- For any set of points, $x_1, \ldots, x_n$, the kernel matrix $K$ is positive semi-definite so that $z^\top K z \geq 0$ for all $z \in \mathbb{R}^n$.

**Definition.** We say a function, $k(x, x')$, is a kernel if it corresponds to a scalar product, $\phi(x)^\top \phi(x')$ in some feature space, $\mathbb{R}^M$, possibly with $M = \infty$.

**Mercer’s Theorem.** A necessary and sufficient condition for a function, $k(x, x')$, to be a kernel is that the corresponding Gram matrix, $K$, be positive semi-definite for all possible choices of $x_1, \ldots, x_n$. 
A Detour on Kernels

- **Key implication** of theorem is possibility of implicitly defining a (possibly infinite-dimensional) feature map, $\phi(\cdot)$, using a kernel function $k(\cdot, \cdot)$.

- Note that $\phi(\cdot)$ is not explicitly required to state the dual problem, nor is it required in (26) and (27)
  - only $k(\cdot, \cdot)$ is required!
  - a potentially big advantage since far less work may be required to compute $k(\cdot, \cdot)$.

**Example.** Let $m = 2$ and define $k(x, x') := (x^\top x')^2$.

Easy to check that $k(x, x') = \phi(x)^\top \phi(x')$ where $\phi(x) := (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$.

But calculating $k(x, x')$ requires $O(m)$ work whereas calculating $\phi(x)^\top \phi(x')$ requires $O(M) = O(m^2)$ work.

More generally, we could define $k(x, x') := (x^\top x' + c)^p$. Computing it will still be $O(m)$ but working with corresponding feature mapping will be $O(m^p)$. 
Constructing New Kernels (Bishop)

We assume:

- $k_1(x, x')$ and $k_2(x, x')$ are valid kernels.
- $c > 0$ is a constant.
- $f(\cdot)$ is any function.
- $q(\cdot)$ is a polynomial with nonnegative coefficients.
- $\phi(x)$ is a function from $x$ to $\mathbb{R}^M$.
- $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$.
- $A$ is a symmetric positive semi-definite matrix.
- $x_a$ and $x_b$ are variables (not necessarily disjoint) with $x = (x_a, x_b)$.
- $k_a$ and $k_b$ are valid kernel functions over their respective spaces.
Then the following are all valid kernels:

\[
\begin{align*}
    k(x, x') &= ck_1(x, x') \\
    k(x, x') &= f(x)k_1(x, x')f(x') \\
    k(x, x') &= q(k_1(x, x')) \\
    k(x, x') &= \exp(k_1(x, x')) \\
    k(x, x') &= k_1(x, x') + k_2(x, x') \\
    k(x, x') &= k_1(x, x')k_2(x, x') \\
    k(x, x') &= k_3(\phi(x), \phi(x')) \\
    k(x, x') &= x^\top A x' \\
    k(x, x') &= k_a(x_a, x'_a) + k_b(x_b, x'_b) \\
    k(x, x') &= k_a(x_a, x'_a)k_b(x_b, x'_b)
\end{align*}
\]
The Gaussian kernel

The Gaussian kernel is given by:

\[ k(x, x') = \exp \left( -\frac{||x - x'||^2}{2\sigma^2} \right) \]  

(30)

It is a valid kernel because

\[ \exp \left( -\frac{||x - x'||^2}{2\sigma^2} \right) = \exp \left( -\frac{x^\top x}{2\sigma^2} \right) \exp \left( \frac{x^\top x'}{\sigma^2} \right) \exp \left( -\frac{x'^\top x'}{2\sigma^2} \right) \]

\[ = f(x) \exp \left( \frac{x^\top x'}{\sigma^2} \right) f(x') \]

and now we can apply (28) and (29).
Introduction to Machine Learning

Back to Support Vector Machines

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
The Kernel-Separated Dual

Returning to SVMs, when the data is kernel-separated our dual problem becomes:

\[
\max_{\alpha \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j k(x_i, x_j)
\]

subject to \( \sum_{i=1}^{n} \alpha_i t_i = 0. \)

Given a solution \( \alpha^* \) to the dual, can obtain corresponding optimal \( b^* \) via

\[
b^* = t_j - \sum_{i=1}^{n} \alpha_i^* t_i k(x_i, x_j) \quad \text{for any } \alpha_j^* > 0
\]

and, for a new data-point \( x \), the prediction

\[
\text{sign} \left( w^* \phi(x) + b^* \right) = \text{sign} \left( \sum_{i=1}^{n} \alpha_i^* t_i k(x_i, x) + b^* \right).
\]
Figure 7.2 from Bishop: Example of synthetic data from two classes in two dimensions showing contours of constant $y(x)$ obtained from a support vector machine having a Gaussian kernel function. Also shown are the decision boundary, the margin boundaries, and the support vectors.

- Note that the data is linearly separable in the Gaussian-kernel space but not in the original space.
A Demo of SVM Classification with Polynomial Kernel by Udi Aharoni
The Non-Separable Case

- In general the data will be non-separable so the primal problem of (19) and (20) will be infeasible.
- Several ways to proceed: e.g. minimize the number of misclassified points, but this is NP-hard.
- Instead we allow points to violate the margin constraints and penalize accordingly in the objective function.
- This yields the more general non-separable primal problem:

\[
\begin{align*}
\min_{\mathbf{w}, \xi, b} & \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^{n} \xi_i \\
\text{subject to} & \quad t_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\
& \quad \xi_i \geq 0, \quad i = 1, \ldots, n
\end{align*}
\] (31)

- again a convex quadratic programming problem with linear constraints – the penalty \( C \) usually chosen by cross-validation.
Figure 7.3 from Bishop: Illustration of the slack variables in $\xi_n \geq 0$. Data points with circles around them are support vectors.

- Note that the slack variables allow points to be misclassified.
The Non-Separable Dual Problem

- As with the separable case, it’s more convenient to work with the dual.
- Because the primal problem is convex the dual and primal have equal optimal objective functions.
- The non-separable dual problem reduces to

$$\max_{\alpha \geq 0, \lambda \geq 0} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j x_i^\top x_j \right\}$$

subject to

$$\sum_{i=1}^{n} \alpha_i t_i = 0$$

$$C - \alpha_i - \lambda_i = 0, \quad i = 1, \ldots, n$$

(33)

where $\lambda = (\lambda_1, \ldots, \lambda_n)$ are Lagrange multipliers for the constraints (32) – again a convex quadratic program with linear constraints – the original dual plus the additional linear constraints of (33).

- Note we can remove $\lambda$ from the dual by replacing (33) with $\alpha_i \leq C$ for $i = 1, \ldots, n$. 
Kernelizing the Dual

- As with the separable case, we can easily apply the kernel trick to obtain the following general non-separable dual problem:

\[
\max_{\alpha \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j k(x_i, x_j)
\]

subject to

\[
\sum_{i=1}^{n} \alpha_i t_i = 0
\]

\[
\alpha_i \leq C, \quad i = 1, \ldots, n
\]

- The problem of (34), (35) and (36) is the problem solved by SVM software.
- Given an optimal solution, \(\alpha^*\), we can recover the SVM classifier as:

\[
b^* = t_j - \sum_{i=1}^{n} \alpha_i^* t_i k(x_i, x_j)
\]

for any \(C > \alpha_j^* > 0\)

and, for a new data-point \(x\), the prediction

\[
\text{sign} \left( w^* \phi(x) + b^* \right) = \text{sign} \left( \sum_{i=1}^{n} \alpha_i^* t_i k(x_i, x) + b^* \right).
\]
Numerical Optimization of the SVM Problems

- Since both primal and dual are convex quadratic problems we could solve either one using convex optimization techniques.
- But we prefer to solve the dual because then we can apply the dual trick
  - and because constraints in the dual are easier to handle.
- But standard convex optimization techniques not suitable for the dual problem because the “$Q$” matrix in (34) is often too large
  - training sets on the order of 20,000 points not uncommon
  - standard gradient methods would require storing a $20k \times 20k$ matrix!
- So special purpose solvers are used instead
  e.g. the sequential minimization optimization (SMO) algorithm
  – avoids the need for storing the entire $Q$ matrix.
Figure 7.4 from Bishop: Illustration of the $\nu$-SVM applied to a nonseparable data set in two dimensions. The support vectors are indicated by circles.
Introduction to Machine Learning

Unsupervised Learning: PCA

Martin Haugh

Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
Supervised Versus Unsupervised Learning

The goal of **supervised learning** is to predict an output \( y \) as a function of inputs \( x = (x_1, \ldots, x_p) \)

- We are given training data: \((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\)
  - \((x_i, y_i)\) is known for all \( i = 1, \ldots, N \).
- The goal is to compute \( E[y \mid x] \) or otherwise predict \( y \) given \( x \).

Regression and classification are examples of supervised learning.

The goal of **unsupervised learning** is to characterize the distribution \( \mathbb{P}(x) \) of the inputs \( x \). There is no target output \( y \).

Standard examples of unsupervised learning methods include:

- **Clustering methods**: Identify multiple regions of the \( x \)-space that contain modes of \( \mathbb{P}(x) \). One can then represent \( \mathbb{P}(x) \) as a mixture of simpler densities representing distinct types or classes of observations.

- **Dimension-reduction methods**: Identify low-dimensional manifolds in \( x \)-space that represent high data density. Examples include principal component analysis (PCA), independent component analysis (ICA), spectral clustering, etc.
Let $\mathbf{x} = (x_1, \ldots, x_d)^\top$ denote an $d$-dimensional random vector with variance-covariance matrix, $\Sigma$.

The goal of PCA is to construct linear combinations

$$ p_i = \sum_{j=1}^{d} w_{ij} x_j, \quad \text{for } i = 1, \ldots, d $$

in such a way that:

(1) The $p_i$’s are orthogonal so that $E[p_i p_j] = 0$ for $i \neq j$

(2) The $p_i$’s are ordered in such a way that:

(i) $p_1$ explains the largest percentage of the total variance

(ii) each $p_i$ explains the largest percentage of the total variance that has not already been explained by $p_1, \ldots, p_{i-1}$. 
The Eigen Decomposition of a Matrix

In practice it is common to apply PCA to the normalized random variables so that $E[x_i] = 0$ and $\text{Var}(x_i) = 1$

- achieved by subtracting the means from the original random variables and dividing by their standard deviations.
- done to ensure that no one component of $x$ can influence the analysis by virtue of that component’s measurement units.

Key tool of PCA is the eigen decomposition of a square matrix.
The eigen decomposition implies that any symmetric matrix, $A \in \mathbb{R}^{d \times d}$ can be written as

$$A = \Gamma \Delta \Gamma^\top$$ (38)

where:

(i) $\Delta$ is a diagonal matrix, $\text{diag}(\lambda_1, \ldots, \lambda_d)$, of the eigen values of $A$
- without loss of generality ordered so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$.

(ii) $\Gamma$ is an orthogonal matrix with $i^{th}$ column of $\Gamma$ containing $i^{th}$ standardized eigen-vector, $\gamma_i$, of $A$.
- “standardized” means $\gamma_i^\top \gamma_i = 1$
- orthogonality of $\Gamma$ implies $\Gamma \Gamma^\top = \Gamma^\top \Gamma = I_d$. 
PCA and the Factor Loadings

Since $\Sigma$ is symmetric can take $A = \Sigma$ in (38). The positive semi-definiteness of $\Sigma$ implies $\lambda_i \geq 0$ for all $i = 1, \ldots, d$.

The principal components of $x$ then given by $p = (p_1, \ldots, p_d)$ satisfying

$$ p = \Gamma^\top x. \quad (39) $$

Note that:

(a) $E[p] = 0$ since $E[x] = 0$

(b) $\text{Cov}(p) = \Gamma^\top \Sigma \Gamma = \Gamma^\top (\Gamma \Delta \Gamma^\top) \Gamma = \Delta$

- so components of $p$ are uncorrelated as desired
- and $\text{Var}(p_i) = \lambda_i$.

The matrix $\Gamma^\top$ is called the matrix of factor loadings. We can invert (39) to obtain

$$ x = \Gamma p \quad (40) $$

- so easy to go back and forth between $x$ and $p$. 

Explaining The Total Variance

We can measure the ability of the first few principal components to explain the total variance:

\[
\sum_{i=1}^{d} \text{Var}(p_i) = \sum_{i=1}^{d} \lambda_i = \text{trace}(\Sigma) = \sum_{i=1}^{d} \text{Var}(x_i). \tag{41}
\]

If we take \(\sum_{i=1}^{d} \text{Var}(p_i) = \sum_{i=1}^{d} \text{Var}(x_i)\) to measure the total variability then by (41) can interpret

\[
\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}
\]

as the percentage of total variability explained by first \(k\) principal components.
The Minimum Variance Formulation

Let $p_1 = \gamma_1^\top x$ be the $1^{st}$ principal component. Can easily show that $\gamma_1$ solves

$$\max_a \ Var(a^\top x)$$

subject to $a^\top a = 1$. 

More generally, let $p_i = \gamma_i^\top x$ be the $i^{th}$ principal component for $i = 1, \ldots, d$.

Then $\gamma_i$ solves

$$\max_a \ Var(a^\top x)$$

subject to $a^\top a = 1$  

$$a^\top \gamma_j = 0, \quad j = 1, \ldots, i - 1.$$ 

So each successive principal component finds the linear combination of the components of $x$ that has maximal variance subject to the normalization constraint (42) and the orthogonal constraints (43).
The Minimum Reconstruction Formulation

We can also obtain the principal components of $\mathbf{x}$ by posing the problem as one of minimizing the sum of squared differences between each sample vector, $\mathbf{x}_i$ and it’s representation, $\tilde{\mathbf{x}}_i$, in some lower $M$-dimensional sub-space

– here we let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ denote (centered) sample observations of $\mathbf{x}$.

We have the following problem:

$$
\min_{\mathbf{B}, \mathbf{Z}} \sum_{i=1}^{n} \sum_{j=1}^{d} \left[ \mathbf{x}_{j,i} - \sum_{k=1}^{M} b_{j,k} z_{k,i} \right]^2
$$

(44)

where $\mathbf{B}_{j,k} := b_{j,k}$ is the basis in the $M$-dimensional subspace and $\mathbf{Z}_{k,i} := z_{k,i}$ is the weighting of the $i^{th}$ sample on the $k^{th}$ element in the basis.

Problem (44) is sufficient to identify a unique reconstruction point for each $\mathbf{x}_i$.

But it is not sufficient to identify a unique optimal $\mathbf{B}$ and $\mathbf{Z}$. Why?

– hence we impose the constraint that $\mathbf{B}^\top \mathbf{B} = \mathbf{I}_M$

– but rotations still a problem

– choosing directions of maximal variance resolves this issue.
**Figure 12.2 from Bishop**: Principal component analysis seeks a space of lower dimensionality, known as the principal subspace and denoted by the magenta line, such that the orthogonal projection of the data points (red dots) onto this subspace maximizes the variance of the projected points (green dots). An alternative definition of PCA is based on minimizing the sum-of-squares of the projection errors, indicated by the blue lines.
High-Dimensional Data

In practice it is often the case that $n << d$

- e.g. have $n = 500$ images of $d = 1000 \times 1000 = 10^6$ pixels
- can cause computational difficulties since computing eigen-decomposition takes $O(d^3)$ operations.

But number of non-zero eigen values cannot exceed $n$. Why?

Can exploit this fact as follows:

1. Let $x_1, \ldots, x_n$ be the $n$ samples of the $d$-dimensional $x$

2. Recall we can write $\Sigma = \frac{1}{n-1}XX^\top$ where $X = [x_1, \ldots, x_n]$.

3. Let $E$ be the $d \times n$ matrix of eigen-vectors of $XX^\top$ with corresponding eigen-values in the $n \times n$ diagonal matrix $\Delta$
Ignoring the scalar $1/(n-1)$ (which can be included later) we obtain

$$XX^\top E = E\Delta$$

so that

$$X^\top XX^\top E = X^\top E \Delta.$$ (45)

If we set $\tilde{E} := X^\top E$ then (45) can be written as

$$X^\top X\tilde{E} = \tilde{E}\Delta$$

so that $\tilde{E}$ is the matrix of eigen vectors for the $n \times n$ matrix $X^\top X$

- can be calculated in $O(n^3)$ operations
- so much easier to calculate then eigen-vectors of the $d \times d$ matrix $XX^\top$.

So we simply solve for $E$ by finding $\tilde{E}$ and then setting (why?)

$$E = X\tilde{E}\Delta^{-1}$$

- eigen values are unchanged.
We will consider several application domains for PCA:

- Data compression
  - hand-writing
  - eigen faces
- Reduced-rank LDA
- Nearest neighbor classification
- Financial modeling
- Latent semantic analysis
- Missing data and collaborative filtering / recommender systems.

There are many, many more application domains.
Data Compression: Digit Recognition Part I

- Figure 14.22 from HTF displays 130 handwritten 3’s from a total of 658 digitized as $16 \times 16$ grayscale images.
- Can view these images as points in $\mathbb{R}^{256}$.
- Figure 14.23 from HTF plots the first 2 principal components along with images that are close to the vertices of the grid - vertices are placed at 5%, 25%, 50%, 75% and 95% quantile points.
- Can see that first component accounts for “lengthening of the lower tail of the three”.
- The second component accounts for “character thickness”.
- Equation (14.55) from HTF yields the 2-component model

$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$$= \begin{pmatrix} 3 \\ 0 \end{pmatrix} + \lambda_1 \cdot \begin{pmatrix} 3 \\ 0 \end{pmatrix} + \lambda_2 \cdot \begin{pmatrix} 3 \\ 0 \end{pmatrix}.$$ 

- Figure 14.24 from HTF displays the singular values (from the SVD) compared with a randomized version of the data.
Figure 14.22 from HTF: A sample of 130 handwritten 3â£™s shows a variety of writing styles.
Figure 14.23 from HTF: Left panel: the first two principal components of the hand-written threes. The circled points are the closest projected images to the vertices of a grid, defined by the marginal quantiles of the principal components. Right panel: The images corresponding to the circled points. These show the nature of the first two principal components.
Figure 14.24 from HTF: The 256 singular values for the digitized threes, compared to those for a randomized version of the data (each column of $X$ was scrambled).
Figure 15.5 from Barber: 100 training images. Each image consists of $92 \times 112 = 10304$ greyscale pixels. The train data is scaled so that, represented as an image, the components of each image sum to 1. The average value of each pixel across all images is $9.70 \times 10^{-5}$. This is a subset of the 400 images in the full Olivetti Research Face Database.
Figure 15.6 from Barber: (a): SVD reconstruction of the images in fig(15.5) using a combination of the 49 eigen-images. (b): The eigen-images are found using SVD of the images in fig(15.5) and taking the mean and 48 eigenvectors with largest corresponding eigenvalue. The images corresponding to the largest eigenvalues are contained in the first row, and the next 7 in the row below, etc. The root mean square reconstruction error is $1.121 \times 10^{-5}$, a small improvement over PLSA (see fig(15.16)).
PCA and Financial Modeling

There are many applications of PCA in finance:

1. Representing movements in term-structure of interest-rates
   – useful for interpretation
   – building factor models
   – and hedging.

2. Representing movements in futures strips
   – useful for interpretation
   – building factor models
   – and hedging.


4. Scenario generation in risk-management.

5. Estimation of risk measures such as VaR and CVaR.

6. And others ...
Nearest-Neighbor Classification

In nearest-neighbor classification it can be expensive to compute the distance between data-points when dimensionality is high. Can overcome this problem by using PCA to approximate distances.

Let $\Gamma_1$ contain the first $k$ principal components with $k$ large enough so that

$$x = \Gamma p \approx \Gamma_1 p_1$$

provides a good approximation. If $x^a$ and $x^b$ are any two points in $\mathbb{R}^d$ then

$$\begin{align*}
(x^a - x^b)^\top (x^a - x^b) &= (\Gamma p^a - \Gamma p^b)^\top (\Gamma p^a - \Gamma p^b) \\
&\approx (\Gamma_1 p_1^a - \Gamma_1 p_1^b)^\top (\Gamma_1 p_1^a - \Gamma_1 p_1^b) \\
&= (p_1^a - p_1^b)^\top \Gamma_1^\top \Gamma_1 (p_1^a - p_1^b) \\
&= (p_1^a - p_1^b)^\top (p_1^a - p_1^b) .
\end{align*}$$

– can be much cheaper to compute (46).

And data is also often noisy so projecting onto a lower dimensional sub-space can produce superior classification: see Figure 15.7 from Barber.
Figure 15.7 from Barber: Finding the optimal PCA dimension to use for classifying hand-written digits using nearest neighbours. 400 training examples are used, and the validation error plotted on 200 further examples. Based on the validation error, we see that a dimension of 19 is reasonable.
Latent Semantic Analysis (LSA)

In the document analysis literature, PCA is called latent semantic analysis (LSA). We assume:

1. There are \( d \) words in our dictionary, \( D \)
2. And \( n \) documents in the corpus.

The \( j^{th} \) document could then be represented by \( \mathbf{x}^j = (x^j_1, \ldots, x^j_d) ^\top \)

- the so-called bag-of-words representation
- where \( x^j_i \) refers to the \( \# \) occurrences of the \( i^{th} \) word in the \( j^{th} \) document.

But more common to normalize this number:

**Definition.** The term-frequency, \( \text{tf}^j_i \), is the number of times that the word \( i \) appears in document \( j \) divided by the number of words in document \( j \), i.e.

\[
\text{tf}^j_i := \frac{\#_{i,j}}{\sum_i \#_{i,j}}
\]

where \( \#_{i,j} \) is the number of times that word \( i \) appears in document \( j \).
The TF-IDF Representation

Definition. The inverse-document-frequency, $idf_i$, is defined as

$$idf_i := \log \left( \frac{n}{\text{# of documents that contain word } i} \right).$$

It is possible to use different definitions of $idf_i$

– as long as rarely occurring words are given more weight when they do occur.

Can now define the TF-IDF representation:

$$x_{ji} := tf_{ji} \times idf_i$$

A corpus of documents is then represented by

$$X = [x^1, \ldots, x^n]$$

– a $d \times n$ matrix which is typically very large!
Latent Semantic Analysis (LSA)

LSA is simply a matter of applying PCA to (the variance-covariance matrix of) $X$
– with the interpretation that the principal directions, i.e. eigen vectors, now define topics.

One weakness of LSA is that eigen vectors can have negative components
– but how can a “topic” contribute negatively to a document?

Probabilistic latent semantic analysis (PLSA) overcomes this problem
– based on non-negative matrix factorization.

Topic modeling now a “hot topic” in the machine learning community.
Examples 15.4 from Barber

The dictionary, $\mathcal{D}$, contains 10 words:

- influenza, flu, headache, nose, temperature,
- bed, cat, dog, rabbit, pet

The document corpus contains 2000 documents:

1. some articles discuss ailments, and some of these discuss influenza
2. some articles related to pets
3. some articles are background articles unrelated to ailments

Some of the ailment articles are informal and use the word “flu” whereas others use the more formal “influenza”.

Each document is therefore represented by a 10-dimensional vector

- $x_i^j = 1$ if $i^{th}$ word occurs in $j^{th}$ document
- $x_i^j = 0$ otherwise.

See Figures 15.8 and 15.9 from Barber.
Figure 15.8 from Barber: (Top) Document data for a dictionary containing 10 words and 2000 documents. Black indicates that a word was present in a document. The data consists of two distinct topics and a random background topic. The first topic contains two sub-topics which differ only in their usage of the first two words, ‘influenza’ and ‘flu’. (Bottom) The projections of each datapoint onto the three principal components.
Figure 15.9 from Barber: Hinton diagram of the eigenvector matrix $E$ where each eigenvector column is scaled by the corresponding eigenvalue. The dark shaded squares indicates positive and light shaded squares negative values (the area of each square corresponds to the magnitude), showing that there are only a few large eigenvalues. Note that the overall sign of any eigenvector is irrelevant. The first eigenvector corresponds to a topic in which the words influenza, flu, headache, nose, temperature, bed are prevalent. The second eigenvector denotes the ‘pet’ topic words. The third eigenvector shows that there is negative correlation between the occurrence of influenza and flu. The interpretation of the eigenvectors as ‘topics’ can be awkward since the basis eigenvectors are by definition orthogonal. Contrast this basis with that found using PLSA with two components, fig(15.14).
Information Retrieval and Example 15.5 from Barber

Consider a large collection of documents from which a dictionary, \( D \), is created.

Given a document, \( x^f \), how do we find the “closest” document to it in the collection?

First need a measure of dissimilarity between documents
– could use \( d(x^f, x^i) := (x^f - x^i)^\top (x^f - x^i) \).

And then select the document that solves

\[
\min_i d(x^f, x^i).
\]

In this case a good idea is to scale each vector so that it has unit length
– why?
– leads to the equivalent cosine similarity

\[
s(x^f, x^i) := \cos(\theta)
\]

where \( \theta \) is the angle between \( x^f \) and \( x^i \).
Figure 15.10 from Barber: (a): Two bag-of-word vectors. The Euclidean distance between the two is large. (b): Normalised vectors. The Euclidean distance is now related directly to the angle between the vectors. In this case two documents which have the same relative frequency of words will both have the same dissimilarity, even though the number of occurrences of the words is different.

Problem with bag-of-words representation is that term-document matrix will have mainly zeros

– so differences may be due to noise.

LSA can help solve this problem. Why?

– consider Example 15.4 from Barber.
Missing Data and Collaborative Filtering

Sometimes find that elements of the matrix $\mathbf{X}$ are missing
– in fact this is often the case and to be expected.

**e.g.** Consider a $d \times n$ movies-ratings matrix with $d$ movies and $n$ users
– then $x^j_i$ represents the rating (on some scale) of user $j$ for movie $i$
– $\mathbf{X}$ will then be very **sparse** in that very few of the $x^j_i$’s will be known.

The goal then is to somehow “fill in” or **impute** the missing values
– this has obvious applications in **collaborative filtering** and **recommender systems**, e.g. the Netflix prize.

Matrix factorization methods including PCA can be used to do this
– a very active area of research
– computational efficiency is vital since $\mathbf{X}$ is typically very large.

We will consider our earlier **minimum reconstruction formulation** of PCA
– see Section 15.5 of Barber.
Our Earlier Minimum Reconstruction Formulation

We previously posed the PCA problem as one of minimizing the sum of squared differences between each sample vector, \( x_i \) and it’s representation, \( \tilde{x}_i \), in some lower \( M \)-dimensional sub-space

– here we let \( x_1, \ldots, x_n \) denote (centered) sample observations of \( x \).

We have the following problem:

\[
\min_{B, Z} \sum_{i=1}^{n} \sum_{j=1}^{d} \left[ x_{j,i} - \sum_{k=1}^{M} b_{j,k} z_{k,i} \right]^2 \tag{47}
\]

where \( B_{j,k} := b_{j,k} \) is the basis in the \( M \)-dimensional subspace and \( Z_{k,i} := z_{k,i} \) is the weighting of the \( i^{th} \) sample on the \( k^{th} \) element in the basis.

When some elements of \( X \) are missing we can instead solve

\[
\min_{B, Z} \sum_{i=1}^{n} \sum_{j=1}^{d} \gamma_{j,i} \left[ x_{j,i} - \sum_{k=1}^{M} b_{j,k} z_{k,i} \right]^2 \tag{48}
\]

where

\[
\gamma_{j,i} := \begin{cases} 
1, & \text{if } x_{j,i} \text{ is available} \\
0, & \text{otherwise.}
\end{cases}
\]
A New Minimum Reconstruction Formulation

There are at least two approaches to solving (48):

1. Find the optimal $Z$ in terms of $B$ and then try to minimize over $B$.

2. Iterative algorithm where we select an initial $\hat{B}$ and then iterate the following two steps until convergence:

   (i) Optimize over $Z$ for given $\hat{B}$. Let $\hat{Z}$ be the optimal solution.

   (ii) Optimize over $B$ for given $\hat{Z}$. Let $\hat{B}$ be the optimal solution.

This algorithm is guaranteed to converge to a local minimum (why?) although in practice it may be slow to do so

   – Figure 15.11 from Barber uses this method

   – can find principal directions or eigen vectors by performing SVD on final $X \approx \hat{B}\hat{Z}$. 
Figure 15.11 from Barber: Top: original data matrix $\mathbf{X}$. Black is missing, white present. The data is constructed from a set of only 5 basis vectors. Middle: $\mathbf{X}$ with missing data (80% sparsity). Bottom: reconstruction found using svdm.m, SVD for missing data. This problem is essentially easy since, despite there being many missing elements, the data is indeed constructed from a model for which SVD is appropriate. Such techniques have application in collaborative filtering and recommender systems where one wishes to ‘fill in’ missing values in a matrix.
Introduction to Machine Learning

Unsupervised Learning: Reputation Systems & PageRank

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
Reputation Systems

The ever increasing amount of online activity and data has been accompanied by the decentralization of control. For example:

1. IMDB and Rotten Tomatoes help us choose what movies to watch.
2. How do we decide what sellers to trust on Ebay?
3. Who should we follow on Twitter?
4. How does Google figure out the importance of web-pages?
5. Why is Yelp so popular?

An important problem is in figuring out who or what pages are the “best”

- clearly this is also a dimension reduction problem!

Here we will consider Google’s PageRank system:

- this is clearly of enormous importance
- has applications to networks beyond evaluating the importance of web-pages
- can use it to highlight the ongoing challenges with reputation systems.

How do we define reputation? How about this definition?

“You’re great if other great people think you’re great.”

Any problems with this? Yes … but how do reputations in the real world work?
Google’s Naive PageRank Algorithm

Consider the web and suppose there are a total of $d$ pages. We say $i \rightarrow j$ is there is a link from page $i$ to page $j$.

Let $X_t \in \{1, \ldots, d\}$ denote the page that a web-surfer visits at time $t$. Our initial model assumes a transition matrix $Q$ where

$$Q_{i,j} := \frac{1}{c(i)}, \quad \text{if } i \rightarrow j$$

$$= 0, \quad \text{otherwise}$$

where $c(i) := \#\text{ of out-links from page } i$

- $X_t$ is then a Markov chain.

Would like to find a stationary distribution, $\mu$, of the Markov chain so that

$$\mu = \mu Q.$$ 

Could then use $\mu(i)$ to measure the importance of the $i^{th}$ page. Why?

But $\mu$ will not be unique unless $X$ is irreducible

- and $X$ will not be irreducible in general because (for example) many web-pages will not have out-links.
Another (less important) issue with defining web-page reputations in this way is that it is easily “gamed” by just a couple of web-pages that collude - see for example “Making Eigenvector-Based Reputation Systems Robust to Collusion” (2004) by Zhang, Goel, Govindan, Mason and Van Roy.
Google’s PageRank Algorithm

We resolve the uniqueness problem (and make collusion harder) by instead assuming a transition matrix $\bar{Q}$ where

$$\bar{Q}_{i,j} := (1 - \epsilon) Q_{i,j} + \frac{\epsilon}{d} > 0$$

for $0 < \epsilon < 1$. We can write (49) equivalently as

$$\bar{Q} := (1 - \epsilon) Q + \frac{\epsilon}{d} 11^\top$$

where $1$ is a $d \times 1$ vector of 1’s.

Note that (50) implies that $\bar{Q}$ is irreducible and therefore has a unique stationary distribution, $\mu$, satisfying

$$\mu = \mu \bar{Q}.$$  (51)

Can interpret the resulting Markov chain as one where with probability $\epsilon$ we choose a new web-page randomly from the entire universe of web-pages and with probability $1 - \epsilon$ we click randomly on a link on the current page - a random walk interpretation.
Google’s PageRank Algorithm

The stationary distribution $\mu$ gives the page-ranks of all the web-pages

- if we run the Markov chain for a sufficiently long time then the proportion of time that we will spend on page $i$ is $\mu(i)$
- note that the page-rank or “importance” of a page is then captured by the importance of the pages that link to it
- note that (51) resolves our circular definition of reputation.

PageRank has been generalized, manipulated, defended, etc. and reputation systems in general have been the cause of much litigation

- should give you an idea of just how important they are!

Computing PageRank is computationally intensive. Rather than solving (51) directly there are two popular approaches:

1. **Power iteration** – the most commonly used approach.
2. Monte-Carlo which exploits our earlier *random walk* interpretation
   - an approach that has been proposed for the fast *updating* of PageRank. (See “Fast Incremental and Personalized PageRank” by Bahmani, Chowdhury and Goel (2010).)
Power iteration is very simple:

1. Choose an initial vector, $\mu_0$. (Easy choice is $\mu_0 = 1/d$.)
2. Then iterate $\mu_{t+1} = \mu_t \bar{Q}$ for $t = 1, \ldots$ until convergence.

**Question:** Is convergence guaranteed?

**Answer:** Yes!

**Proof:** Let $\Delta_t := \sum_{i=1}^{d} |\mu_t(i) - \mu(i)|$ where $\mu$ is the solution to (51). Using the triangle inequality it is easy to see that

$$|\mu_{t+1}(i) - \mu(i)| \leq (1 - \epsilon) \sum_{j: j \text{ links to } i} \frac{|\mu_t(j) - \mu(j)|}{c(j)}$$

(52)

Summing (52) over $i$ yields $\Delta_{t+1} \leq (1 - \epsilon)\Delta_t$. Therefore (why?) $\Delta_t \to 0$ as $t \to \infty$ and so $\mu_t(i)$ converges to $\mu(i)$. □

**Question:** Is the convergence fast?
PageRank

Google originally used PageRank to index the web and to provide an ordering of web-pages according to their reputation / importance.

This ordering was vital to search and advertising
  - but also provided an obvious motive for companies to try and increase the PageRank of their web-pages.

Over the years the PageRank system has adapted and heuristics added
  - in part to counter attempts at collusion and manipulation.

While the specific details are private it is very likely that PageRank still plays the key role in Google’s search algorithms.
Question: How might you use PageRank as part of a search system?

Here’s how …

1. First compute the PageRank of all web-pages and store it in descending order of importance
   - just like a book with the 1\textsuperscript{st} page having the highest PageRank, the 55\textsuperscript{th} page having the 55\textsuperscript{th} highest PageRank etc.

2. A reverse index for all commonly searched terms is also computed and stored
   - this is exactly like an index at the back of a book telling you what pages important terms appear on.

Steps 1 and 2 are updated periodically.

3. When a user searches for “dog” say, the reverse index tells you exactly what web-pages the term “dog” appears on and returns the first $n$ of them.
   - This is easy and “correct”. Why?

Question: How might you handle searching for multiple terms like “dog” \textit{and} “pony”?
Other Applications / Extensions of PageRank

The ordering returned by PageRank can easily be adapted to reflect a preference for a specific subset, $C$ say, of web-pages.

Can do this by changing the algorithm so that with probability $\epsilon$ we choose a new web-page randomly from $C$ and with probability $1 - \epsilon$ we click randomly on a link on the current page

- this is known as personalized PageRank.

PageRank can be applied in most network settings where links between nodes indicate some sort of preference. Other examples include:

1. Recommendation systems: we form a bipartite graph with, for example, movies on one side and users on the other. A link is added between a movie and a user if the user liked that movie. In this case PageRank gives an ordering of movies and users.

   **Question:** How might you choose what movies to recommend to a specific user?

2. Networks can also be created out of users and tweets on Twitter with links added to identify creators if tweets and followers. Can use such a system to identify influential people and make recommendations to users.
Introduction to Machine Learning

Unsupervised Learning: Clustering

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
Clustering Methods

The goal with clustering methods is to partition the data into clusters with low intra-cluster dissimilarity and large inter-cluster dissimilarity.

There are many approaches:

- **Non-probabilistic combinatorial methods**: $K$-means, $K$-medoids etc.
- **Mixture modeling**: Data is an IID draw from a mixture but the mixture indicator is latent or hidden.
- **Mode-seeker**: Seek modes of the joint PDF of $\mathbf{x}$ in a non-parametric manner

We will focus on the first method here.
A key component of any clustering algorithm is how the distance, $d(x, y)$, between points $x$ and $y$ are measured.

- Will always assume $d(x, x) = 0$
- Typically assume $d(\cdot, \cdot)$ is symmetric: $d(x, y) = d(y, x)$
- In general do not assume the triangle inequality holds
  - so there may exist $x, y, z$ such that $d(x, z) > d(x, y) + d(y, z)$.

Examples:

- Weighted difference between attributes: $d(x, y) = \sum_{j=1}^p w_j d_j(x_j, y_j)$
- Quantitative variables: $d_j(x_j, y_j) = f_j(|x_j - y_j|)$, for $f_j(\cdot)$ non-negative and non-decreasing.
- Rank variables: the rank $i \mapsto \frac{i-1/2}{N}$, $i = 1, \ldots, N$
  - now treat as quantitative variable.
- Categorical variables: often choose $d(x, y) := c 1\{x \neq y\}$.

**Dissimilarity matrix:** $\mathbf{D} \in \mathbb{R}^{N \times N}$ with $D_{ij} := d(x_i, x_j)$

  - if clustering algorithm requires symmetric $\mathbf{D}$ then can use $(\mathbf{D} + \mathbf{D}')/2$. 

How should we handle missing data?
How Do We Choose the $w_j$’s?

We have $N$ points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ so that $d_k(x_{ik}, x_{jk}) = \text{dissimilarity measure on the } k^{th} \text{ attribute of } \mathbf{x}_i \text{ and } \mathbf{x}_j$.

Combined dissimilarity measure: $d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p} w_k d_k(x_{ik}, x_{jk})$

How do we set the weights $w_k$? Total average dissimilarity is:

$$
\bar{d} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p} w_k \cdot \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d_k(x_{ik}, x_{jk})
$$

In order to treat all attributes equally we should choose: $w_k \sim \frac{1}{\bar{d}_k}$

– and also typically have $\sum_{k=1}^{p} w_k = 1$.

For $d_k(x_{ik}, x_{jk}) = (x_{ik} - x_{jk})^2$, the normalization is $\bar{d}_k = 2\text{Var}(x_k)$

But equal weighting is not always appropriate

– see Figure 14.5 from HTF.
FIGURE 14.5. Simulated data: on the left, $K$-means clustering (with $K = 2$) has been applied to the raw data. The two colors indicate the cluster memberships. On the right, the features were first standardized before clustering. This is equivalent to using feature weights $1/[2 \cdot \text{var}(X_j)]$. The standardization has obscured the two well-separated groups. Note that each plot uses the same units in the horizontal and vertical axes.

Figure 14.5 taken from HTF
A clustering algorithm with $K$ clusters and $N \ll K$ points should produce:

- A cluster map $i \in \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$.
  The $k^{th}$ cluster is then $C_k := \{i : i \mapsto k\}$
- Cluster “centers” $m_k$ for $k = 1, \ldots, K$
  $m_k$ approximates all points in the $k^{th}$ cluster.

Within and between segment dissimilarity can be computed as

$$
\frac{1}{2} \sum_{i=1}^{N} \sum_{\ell=1}^{N} d(x_i, x_\ell) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{\ell \in C_k} d(x_i, x_\ell) + \frac{1}{2} \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{\ell \notin C_k} d(x_i, x_\ell)
$$

$$
= \frac{1}{2} \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{\ell \in C_k} d(x_i, x_\ell) + \frac{1}{2} \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{\ell \notin C_k} d(x_i, x_\ell)
$$

$$
= W(C) + B(C)
$$

$W(C) =$ within-cluster dissimilarity and $B(C) =$ between-cluster dissimilarity.
Clustering Algorithms

**Goal:** Find clusters $C := \{C_k : k = 1, \ldots K\}$ so that within-cluster dissimilarity is minimized

– or equivalently (why?) so that between cluster dissimilarity is maximized.

But this is an **NP-hard** problem!

It can be shown that the number of possible cluster assignments is

$$S(N, K) = \frac{1}{K!} \sum_{k=1}^{K} (-1)^{K-k} \binom{K}{k} k^N$$

According to HTF, $S(10, 4) = 34,105$ and $S(19, 4) \approx 10^{10}$

– so $S(N, K)$ grows rapidly and there is no chance of using enumeration to find global optimal for reasonable values of $(N, K)$.

Instead we seek a good **local optimum**

– typically using algorithms based on **iterative greedy descent**.
**K-Means Clustering**

*K*-means is one of the most popular clustering algorithms. The dissimilarity measure is squared Euclidean distance: \( d(x, y) = \| x - y \|_2^2 \) – so all variables required to be of the quantitative type.

For each \( C_k \) the minimum distortion center, \( m_k \), is easily found ...

\[
m_k := \underset{m}{\text{argmin}} \sum_{x_i \in C_k} \| x_i - m \|_2^2 = \frac{1}{N_k} \sum_{i \in C_k} x_i
\]

Moreover can easily check

\[
W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i,j \in C_k} \| x_i - x_j \|_2^2 = \sum_{k=1}^{K} N_k \sum_{i \in C_k} \| x_i - m_k \|_2^2.
\]
The $K$-means clustering problem is to solve:

$$\min_{C} \min \{m_k\}_{k=1}^{K} \sum_{k=1}^{K} N_k \sum_{i \in C_k} \|x_i - m_k\|_2^2$$

As explained earlier, we cannot hope to find the global optimal to this problem.

So we use the following iterative algorithm:

Step 1. Fix clusters $\{C_k\}$ and set $m_k = \frac{1}{N_k} \sum_{i \in C_k} x_i$

Step 2. Fix $\{m_k\}$ and choose clusters: $C(i) = \text{argmin}_k \{\|x_i - m_k\|_2\}$

– so clusters $\{C_k\}$ are identified in this step.

Some approximation guarantees are available for this algorithm.

And it is guaranteed (why?) to converge to a local minimum

– so always a good idea to run algorithm from many different starting points.

Next slide shows snapshots of the $K$-means iterations

– clusters are given by the Voronoi regions determined by the blue lines.
Application: Vector Quantization

Note that a gray-scale image can be represented as a vector $X \in \mathbb{R}^n$

- e.g. the left image of statistician Ronald Fisher consists of $1,024 \times 1,024$ pixels and can therefore be represented by $X \in \mathbb{R}^{1024 \times 1024}$
- each pixel is a grayscale value ranging from 0 to 255
- so each pixel requires 1 byte of storage
- so entire image requires approx 1 megabyte of storage.

For an image $X$, can define a collection of vectors $\mathcal{X} := \{x_{ij}\} \subset \mathbb{R}^4$ with

$$x_{ij} := \text{vec} \left( \begin{bmatrix} X_{2i-1,2j-1} & X_{2i,2j-1} \\ X_{2i-1,2j} & X_{2i,2j} \end{bmatrix} \right)$$

Can now define each $x_{ij}$ as a data-point and use $K$-means clustering to compress the image.
FIGURE 14.9. Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024 × 1024 grayscale image at 8 bits per pixel. The center image is the result of 2 × 2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel.

Figure 14.9 taken from HTF
**Application: Vector Quantization**

So run $K$-means on $\mathcal{X}$ to obtain cluster “centers” $\{m_k\}_{k=1}^K \subseteq \mathbb{R}^4$

Each group of 4 pixels is replaced by its assigned cluster center
- so if $C(x_{ij}) = k$ we then set

$$
\begin{bmatrix}
\hat{X}_{2i-1,2j-1} & \hat{X}_{2i,2j-1} \\
\hat{X}_{2i-1,2j} & \hat{X}_{2i,2j}
\end{bmatrix} = \text{mat}(m_k)
$$

where $\text{mat}(m_k)$ is the $2 \times 2$ matrix constructed from the $4 \times 1$ vector, $m_k$.

This compression algorithm is known as **lossy compression**.

**Question:** How do we use the output of the $K$-means algorithm to compress the image?

**Question:** How would we vary the amount of compression?
Example: Clustering of Senators

Roll call data is a great resource for measuring “dissimilarity” between legislators.

Keith Poole (University of Georgia) and Howard Rosenthal (NYU) maintain an interesting web-site on this topic at http://www.voteview.com.

Question: How could we cluster senators?
Answer: We need a measure of dissimilarity between senators.

Let $V = [v_{ik}]$ denote the votes of legislator $i$ on bill $k$

$$v_{ik} = \begin{cases} 
+1 & \text{vote of legislator } i \text{ on bill } k \text{ is “aye”} \\
-1 & \text{vote of legislator } i \text{ on bill } k \text{ is “nay”} \\
0 & \text{legislator } k \text{ abstained on bill } k
\end{cases}$$

Dissimilarity between legislators $d(x_i, x_j) = \sqrt{\sum_k (v_{ik} - v_{jk})^2}$.

Is this a reasonable dissimilarity measure?
– if so, then we can just go ahead and run $K$-means.
**Kernel $K$-Means Clustering**

Can extend $K$-means to general kernels $K(x, y) \neq x^T y$ via the “kernel trick”.

Mercer’s Theorem implies $K(x, y) = \phi(x)^T \phi(y)$ for some (possibly unknown) $\phi : \mathbb{R}^p \mapsto \mathbb{R}^d$ where typically $d \gg p$.

Recall that the cluster centers satisfy

$$m_k := \arg\min \sum_{i \in C_k} \|\phi(x_i) - m_k\|_2^2 = \frac{1}{N_k} \sum_{i \in C_k} \phi(x_i)$$

Cluster assignment in iteration $t + 1$

$$C^{(t+1)}(i) = \arg\min_k \|\phi(x_i) - m_k\|_2^2$$

$$= \arg\min_k \left\{ K(x_i, x_i) - \frac{2}{N_k} \sum_{\ell \in C_k^{(t)}} K(x_i, x_\ell) + \frac{1}{N_k^2} \sum_{\ell, \ell' \in C_k^{(t)}} K(x_\ell, x_{\ell'}) \right\}$$

So the cluster assignment can be computed without reference to $\phi$

– only the kernel function $K(\cdot, \cdot)$ is required!

**Question:** But what about computing the $m_k$’s?
**K-Medoids Clustering**

*K*-means is very sensitive to outliers as produce large distances

– so maybe worth exploring other distance measures.

**Fact**: Given \( x = \{x_1, \ldots, x_N\} \), median(\( x \)) \( \in \arg\min_z \sum_{i=1}^{N} |x_i - z| \)

– so could cluster by using the median to construct distance measure.

More generally, we can assign cluster “center” to be the cluster medoid so we set \( m_k := x_{i_k} \) where

\[
i_k = \arg\min_{\ell \in C_k} \sum_{i \in C_k} d(x_\ell, x_i).
\]

Form new clusters by assigning points to the nearest cluster center

\[
C(i) = \arg\min_{1 \leq k \leq K} d(x_i, m_k)
\]

This yields the *K*-medoids clustering algorithm ...
**K-Medoids Clustering**

Algorithm 14.2 *K*-medoids Clustering.

1. For a given cluster assignment \( C \) find the observation in the cluster minimizing total distance to other points in that cluster:

\[
i_k^* = \text{argmin}_{\{i: C(i) = k\}} \sum_{C(i') = k} D(x_i, x_{i'}). \tag{14.35}
\]

Then \( m_k = x_{i_k^*} \), \( k = 1, 2, \ldots, K \) are the current estimates of the cluster centers.

2. Given a current set of cluster centers \( \{m_1, \ldots, m_K\} \), minimize the total error by assigning each observation to the closest (current) cluster center:

\[
C(i) = \text{argmin}_{1 \leq k \leq K} D(x_i, m_k). \tag{14.36}
\]

3. Iterate steps 1 and 2 until the assignments do not change.

*Algorithm 14.2 in HTF.*

Note that we do not need to explicitly compute cluster centers — so only dissimilarity matrices rather than actual points are needed. But \( K \)-medoids is more computationally demanding than \( K \)-means.
Normal Mixture Models

Clustering via normal mixture models is an example of probabilistic clustering – we assume the data are IID draws.

e.g. (Scalar Case) Suppose $\mathcal{X} = (X_1, \ldots, X_n)$ are IID random variables each with PDF

$$f_x(x) = \sum_{j=1}^{m} p_j \frac{e^{-(x-\mu_j)^2/2\sigma_j^2}}{\sqrt{2\pi\sigma_j^2}}$$

where $p_j \geq 0$ for all $j$ and where $\sum p_j = 1$

– parameters are the $p_j$’s, the $\mu_j$’s and the $\sigma_j$’s
– typically estimated via MLE
– which we can do via the EM algorithm
  – an extremely important algorithm in statistics / machine learning.
Introduction to Machine Learning

An Introduction to Bayesian Modeling

Martin Haugh

Department of IE & OR
Columbia University

Email: martin.b.haugh@gmail.com
Bayes Theorem

Not surprisingly, Bayes’s Theorem is the key result that drives Bayesian modeling and statistics.

Let $S$ be a sample space and let $B_1, \ldots, B_K$ be a partition of $S$ so that (i) $\bigcup_k B_k = S$ and (ii) $B_i \cap B_j = \emptyset$ for all $i \neq j$.

**Bayes’s Theorem:** Let $A$ be any event. Then for any $1 \leq k \leq K$ we have

$$P(B_k \mid A) = \frac{P(A \mid B_k)P(B_k)}{P(A)} = \frac{P(A \mid B_k)P(B_k)}{\sum_{j=1}^K P(A \mid B_j)P(B_j)}.$$

Of course there is also a continuous version of Bayes’s Theorem with sums replaced by integrals.

Bayes’s Theorem provides us with a simple rule for updating probabilities when new information appears

- in Bayesian modeling and statistics this new information is the observed data
- and it allows us to update our prior beliefs about parameters of interest which are themselves assumed to be random variables.
The Prior and Posterior Distributions

Let \( \theta \) be some unknown parameter vector of interest. We assume \( \theta \) is random with some prior distribution, \( \pi(\theta) \).

There is also a random vector, \( X \), with PDF (or PMF) \( p(x \mid \theta) \)
- this is the likelihood.

The joint distribution of \( \theta \) and \( X \) is then given by \( p(\theta, x) = \pi(\theta)p(x \mid \theta) \).

We can compute the posterior distribution via Bayes’s Theorem:

\[
\pi(\theta \mid x) = \frac{\pi(\theta)p(x \mid \theta)}{p(x)} = \frac{\pi(\theta)p(x \mid \theta)}{\int_\theta \pi(\theta)p(x \mid \theta) \, d\theta} \quad (53)
\]

\[\propto \pi(\theta)p(x \mid \theta).\]

Much of Bayesian analysis is concerned with “understanding” the posterior:

1. Sometimes can recognize the form of the posterior by simply inspecting \( \pi(\theta)p(x \mid \theta) \)
2. But often cannot recognize the posterior and can’t compute the denominator in (53) either
   - so approximate inference techniques such as MCMC must be used.
Let $\theta \in (0, 1)$ represent some unknown probability. We assume a \(\text{Beta}(\alpha, \beta)\) prior so that

$$
\pi(\theta) = \frac{\theta^{\alpha-1}(1 - \theta)^{\beta-1}}{B(\alpha, \beta)}, \quad 0 < \theta < 1.
$$

We also assume that $X \mid \theta \sim \text{Bin}(n, \theta)$ so that

$$
p(x \mid \theta) = \binom{n}{x} \theta^x (1 - \theta)^{n-x}, \quad x = 0, \ldots, n.
$$

The posterior then satisfies

$$
p(\theta \mid x) \propto \pi(\theta) p(x \mid \theta)
= \frac{\theta^{\alpha-1}(1 - \theta)^{\beta-1}}{B(\alpha, \beta)} \binom{n}{x} \theta^x (1 - \theta)^{n-x}
\propto \theta^{\alpha+x-1}(1 - \theta)^{n-x+\beta-1}
$$

which we recognize as the \(\text{Beta}(\alpha + x, \beta + n - x)\) distribution!

**Question:** How can we interpret the prior distribution in this example?
E.G: A Beta Prior and Binomial Likelihood

Figure 20.1 from Ruppert’s Statistics and Data Analysis for FE: Prior and posterior densities for $\alpha = \beta = 2$ and $n = x = 5$. The dashed vertical lines are at the lower and upper 0.05-quantiles of the posterior, so they mark off a 90% equal-tailed posterior interval. The dotted vertical line shows the location of the posterior mode at $\theta = 6/7 = 0.857$. 

Conjugate Priors

Consider the following probabilistic model:

- parameter $\theta \sim \pi(\cdot; \alpha_0)$
- data $X = (X_1, \ldots, X_N) \sim p(x \mid \theta)$

As we saw before posterior distribution satisfies

$$p(\theta \mid x) \propto p(\theta, x) = p(x \mid \theta)\pi(\theta; \alpha_0)$$

Say the prior $\pi(\theta \mid \alpha)$ is a conjugate prior for the likelihood $p(x \mid \theta)$ if the posterior satisfies

$$p(\theta \mid x) = \pi(\theta; \alpha(x))$$

so observations influence the posterior only via a parameter change $\alpha_0 \rightarrow \alpha(x)$ – the form or type of the distribution is unchanged.

e.g. In the previous example we saw the beta distribution is conjugate for the binomial likelihood.
Suppose $\theta \sim \mathcal{N}(\mu_0, \gamma_0^2)$ and $p(X_i \mid \theta) = \mathcal{N}(\theta, \sigma^2)$ for $i = 1, \ldots, N$
- so $\alpha_0 = (\mu_0, \gamma_0^2)$ and $\sigma^2$ is assumed known.

If $\mathbf{X} = (X_1, \ldots, X_N)$ we then have

$$p(\theta \mid \mathbf{x}) \propto p(\mathbf{x} \mid \theta)p(\theta)$$

$$\propto e^{-\frac{(\theta - \mu_0)^2}{2\gamma_0^2}} \prod_{i=1}^{N} e^{-\frac{(x_i - \theta)^2}{2\sigma^2}}$$

$$\propto \exp\left(-\frac{(\theta - \mu_1)^2}{2\gamma_1^2}\right)$$

where

$$\gamma_1^{-2} := \gamma_0^{-2} + N \sigma^{-2}$$

and

$$\mu_1 := \gamma_1^2 (\mu_0 \gamma_0^{-2} + \sum_{i=1}^{n} x_i \sigma^{-2}).$$

Of course we recognize $p(\theta \mid \mathbf{x})$ as the $\mathcal{N}(\mu_1, \gamma_1^2)$ distribution.
Conjugate Prior for Mean and Variance of a Normal Dist.

Suppose that \( p(X_i \mid \theta) = N(\mu, \sigma^2) \) for \( i = 1, \ldots, N \) and let \( \mathbf{X} := (X_1, \ldots, X_N) \).

Now assume \( \mu \) and \( \sigma^2 \) are unknown so \( \theta = (\mu, \sigma^2) \).

We assume a joint prior of the form

\[
\pi(\mu, \sigma^2) = \pi(\mu \mid \sigma^2)\pi(\sigma^2) = N(\mu_0, \sigma^2/\kappa_0) \times \text{Inv-}\chi^2(\nu_0, \sigma_0^2)
\]

\[
\propto \sigma^{-1}(\sigma^2)^{-(\nu_0/2+1)} \exp \left( -\frac{1}{2\sigma^2} [\nu_0\sigma_0^2 + \kappa_0(\mu_0 - \mu)^2] \right)
\]

– the \( \text{N-Inv-}\chi^2(\mu_0, \sigma_0^2/\kappa_0, \nu_0, \sigma_0^2) \) density.

Note that \( \mu \) and \( \sigma^2 \) are not independent under this joint prior.

**Exercise:** Show that multiplying this prior by the normal likelihood yields a \( \text{N-Inv-}\chi^2 \) distribution.
Selecting a Prior

Selecting an appropriate prior is a key component of Bayesian modeling.

With only a finite amount of data, the prior can have a very large influence on the posterior
- important to be aware of this and understand sensitivity of posterior inference to the choice of prior
- often try to use non-informative priors to limit this influence
- when possible conjugate priors often chosen for tractability reasons

A common misconception that the only advantage of the Bayesian approach over the frequentist approach is that the choice of prior allows us to express our prior beliefs on quantities of interest
- in fact there are many other more important advantages including modeling flexibility via MCMC, exact inference rather than asymptotic inference, ability to estimate functions of any parameters without “plugging” in MLE estimates, more accurate estimates of parameter uncertainty, etc.
- of course there are disadvantages as well including subjectivity induced by choice of prior and high computational costs.
Inference in Bayesian Modeling

Despite differences in Bayesian and frequentist approaches we do have:

**Bernstein-von Mises Theorem:** Under suitable assumptions and for sufficiently large sample sizes, the posterior distribution of $\theta$ is approximately normal with mean equal to the true value of $\theta$ and variance equal to the inverse of the Fisher information matrix.

This theorem implies that Bayesian and MLE estimators have the same large sample properties

- not really surprising since influence of the prior should diminish with increasing sample sizes.

But this is a theoretical result and often don’t have “large” sample sizes so quite possible for posterior to be (very) non-normal and even multi-modal.

Most of Bayesian inference is concerned with (which often means simulating from) the posterior

$$\pi(\theta \mid x) \propto \pi(\theta)p(x \mid \theta)$$

without knowing the constant of proportionality in (54).

This leads to the following sampling problem:
The Basic Sampling Problem

Given a distribution function

\[ p(z) = \frac{1}{Z_p} \tilde{p}(z) \]

where \( \tilde{p}(z) \geq 0 \) is easy to compute but \( Z_p \) is (too) hard to compute.

This very important problem arises in several contexts:

1. In Bayesian models where \( \tilde{p}(\theta) := p(x | \theta) \pi(\theta) \) is easy to compute but \( Z_p := p(x) = \int_{\theta} \pi(\theta) p(x | \theta) d\theta \) can be very difficult or impossible to compute.

2. In models from statistical physics, e.g. the Ising model, we only know \( \tilde{p}(z) = e^{-E(z)} \), where \( E(z) \) is an “energy” function
   - the Ising model is an example of a Markov network or an undirected graphical model.

3. Dealing with evidence in directed graphical models such as belief networks aka directed acyclic graphs
   - see last section of these slides.
Markov Chain Monte-Carlo (MCMC)

- MCMC algorithms were originally developed in the 1940’s by physicists at Los Alamos
  - Ulam (playing solitaire!), Von Neumann (acceptance-rejection!) and others.
- They were interested in modeling the probabilistic behavior of collections of atomic particles
  - could not be done analytically but maybe they could use simulation?
- Simulation was difficult – the normalization constant $Z_p$ was not known
  - and simulation hadn’t (why?) been “discovered” yet
  - although simulation ideas had been around for some time
    - e.g. Buffon’s needle (1700’s), Lord Kelvin (1901), Fermi (1930’s).
    - in fact the term “Monte-Carlo” was coined at Los Alamos.
- Ulam and Metropolis overcame this problem by constructing a Markov chain for which the desired distribution was the stationary distribution
  - then only needed to simulate the Markov chain until stationarity achieved
  - they introduced the Metropolis algorithm and its impact was enormous.
Suppose we want to sample from a distribution \( p(x) := \frac{\tilde{p}(x)}{Z_p} \).

We can construct a (reversible) Markov chain as follows. Let \( X_t = x \) be the current state:

- Generate \( Y \sim Q(\cdot \mid x) \) for some Markov transition matrix \( Q \).
  Let \( y \) be the generated value.
- Set \( X_{t+1} = y \) with probability \( \alpha(y \mid x) := \min \left\{ \frac{\tilde{p}(y)}{\tilde{p}(x)} \cdot \frac{Q(x \mid y)}{Q(y \mid x)}, 1 \right\} \).
  Otherwise set \( X_{t+1} = x \).

**Claim:** The resulting Markov chain is reversible with stationary distribution \( p(x) = \frac{\tilde{p}(x)}{Z_p} \).

Note that \( Z_p \) is not required for the algorithm!

Note also that if \( Y = y \) is rejected then the current state \( x \) becomes the next state so that \( X_t = X_{t+1} = x \).

Can therefore sample from \( p(x) \) by running the algorithm until stationarity is achieved and then using generated points as our samples.
Figure 11.9 from Bishop: A simple illustration using Metropolis algorithm to sample from a Gaussian distribution whose one standard-deviation contour is shown by the ellipse. The proposal distribution is an isotropic Gaussian distribution whose standard deviation is 0.2. Steps that are accepted are shown as green lines, and rejected steps are shown in red. A total of 150 candidate samples are generated, of which 43 are rejected.
Example: Sampling from a Multi-Modal Distribution

**Figure 27.8 from Barber:** Metropolis-Hastings samples from a bi-variate distribution $p(x_1, x_2)$ using a proposal $\tilde{q}(x'|x) = N(x'|x, I)$. We also plot the iso-probability contours of $p$. Although $p(x)$ is multi-modal, the dimensionality is low enough and the modes sufficiently close such that a simple Gaussian proposal distribution is able to bridge the two modes. In higher dimensions, such multi-modality is more problematic.

**Question:** Why do you think it might sometimes be difficult to sample from a multi-modal distribution?
Gibbs Sampling

Gibbs sampling is an MCMC sampler introduced by Geman and Geman in 1984 - named after the physicist J. W. Gibbs who died 80 years earlier.

Let $x^{(t)} \in \mathbb{R}^m$ denote the current sample. Then Gibbs sampling proceeds as follows:

1. Pick an index $k \in \{1, \ldots, m\}$ either via round-robin or uniformly at random
2. Set $x^{(t+1)}_j = x^{(t)}_j$, for $j \neq k$, i.e. $x^{(t+1)}_k = x^{(t)}_k$
3. Generate $x^{(t+1)}_k \sim p(x_j \mid x^{(t)}_{-k})$
   
   - so only one component of $x$ is updated at a time.

Common to simply order the $m$ components and update them sequentially. Can then let $x^{(t+1)}_k$ be the value of the chain after all $m$ updates rather than each individual update.

A very popular method when the (true) conditional distributions, $p(x_j \mid x^{(t)}_{-k})$, are easy to simulate from

- which is the case for conditionally conjugate models and others.
A Simple Example

Consider the distribution

\[ p(x, y) = \frac{n!}{(n-x)!x!} y^{(x+\alpha-1)} (1 - y)^{(n-x+\beta-1)}, \quad x \in \{0, \ldots, n\}, y \in [0, 1]. \]

Hard to simulate directly from \( p(x, y) \) but the conditional distributions are easy to work with. We see that

- \( p(x \mid y) \equiv \text{Bin}(n, y) \)
- \( p(y \mid x) \equiv \text{Beta}(x + \alpha, n - x + \beta) \)

Since it’s easy to simulate from each conditional, it is easy to run a Gibbs sampler to simulate from the joint distribution.

**Question:** Given one of our earlier examples, can you identify a situation where this distribution might arise?

The marginal distribution of \( x \) is the **beta-binomial** distribution.
Hierarchical Models

Table 11.2 Coagulation time in seconds for blood drawn from 24 animals randomly allocated to four different diets. Different treatments have different numbers of observations because the randomization was unrestricted. From Box, Hunter, and Hunter (1978), who adjusted the data so that the averages are integers, a complication we ignore in our analysis.

<table>
<thead>
<tr>
<th>Diet</th>
<th>Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>62, 60, 63, 59</td>
</tr>
<tr>
<td>B</td>
<td>63, 67, 71, 64, 65, 66</td>
</tr>
<tr>
<td>C</td>
<td>68, 66, 71, 67, 68, 68</td>
</tr>
<tr>
<td>D</td>
<td>56, 62, 60, 61, 63, 64, 63, 59</td>
</tr>
</tbody>
</table>

Gibbs sampling is particularly suited for hierarchical modeling
- we will consider an example from Bayesian Data Analysis by Gelman et al.
- the data is in Table 11-2 above.
The Hierarchical Normal Model

Data $y_{ij}$, for $i = 1, \ldots, n_j$ and $j = 1, \ldots, J$ are assumed to be independently normally distributed within each of $J$ groups with means $\theta_j$ and common variance $\sigma^2$. That is, $y_{ij} \mid \theta_j \sim N(\theta_j, \sigma^2)$.

Total number of observations is $n = \sum_{j=1}^J n_j$.

Group means are assumed to follow a normal distribution with unknown mean $\mu$ and variance $\tau^2$. That is $\theta_j \sim N(\mu, \tau^2)$.

A uniform prior is assumed for $(\mu, \log \sigma, \tau)$ so $p(\mu, \log \sigma, \log \tau) \propto \tau$  
- if a uniform prior was assigned to $\log \tau$ then posterior would be improper as discussed in Gelman et al  
  - this emphasizes the importance of understanding the issues associated with choosing priors.

The posterior then given by

$$p(\theta, \mu, \log \sigma, \log \tau \mid y) \propto \tau \prod_{j=1}^J N(\theta_j \mid \mu, \tau^2) \prod_{j=1}^J \prod_{i=1}^{n_j} N(y_{ij} \mid \theta_j, \sigma^2).$$
The Gibbs Sampler for the Hierarchical Normal Model

Will see that all conditional distributions required for Gibbs sampler have simple conjugate forms:

1. **Conditional Posterior Distribution of Each $\theta_j$**

   Just gather terms from posterior that only involve $\theta_j$ and then simplify to obtain
   
   $$
   \theta_j \mid (\theta_{-j}, \mu, \sigma, \tau, y) \sim N\left(\hat{\theta}_j, V_{\theta_j}\right)
   $$

   where

   $$
   \hat{\theta}_j := \frac{1}{\tau^2} \mu + \frac{n_j}{\sigma^2} \bar{y}_j
   $$

   $$
   V_{\theta_j} := \frac{1}{\tau^2 + n_j/\sigma^2}.
   $$

   These conditional distributions are independent so generating the $\theta_j$’s one at a time is equivalent to drawing $\Theta$ all at once.
2. **Conditional Posterior Distribution of** $\mu$

Again, just gather terms from posterior that only involve $\mu$ and then simplify to obtain

$$
\mu \mid (\theta, \sigma, \tau, y) \sim \mathcal{N} \left( \hat{\mu}, \frac{\tau^2}{J} \right)
$$

where

$$
\hat{\mu} := \frac{1}{J} \sum_{j=1}^{J} \theta_j.
$$

3. **Conditional Posterior Distribution of** $\sigma^2$

Again, just gather terms from posterior that only involve $\sigma$ and then simplify to obtain

$$
\sigma^2 \mid (\theta, \mu, \tau, y) \sim \text{Inv-}\chi^2 \left( n, \hat{\sigma}^2 \right)
$$

where

$$
\hat{\sigma}^2 := \frac{1}{n} \sum_{j=1}^{J} \sum_{i=1}^{n_j} (y_{ij} - \theta_j)^2.
$$
4. *Conditional Posterior Distribution of $\tau^2$*

Again, gather terms from posterior that only involve $\tau$ and then simplify to obtain

$$\tau^2 \mid (\theta, \mu, \sigma, y) \sim \text{Inv-}\chi^2(J - 1, \hat{\tau}^2)$$

where

$$\hat{\tau}^2 := \frac{1}{J - 1} \sum_{j=1}^{J} (\theta_j - \mu)^2.$$ 

To start the Gibbs sampler we need starting points for $\theta$ and $\mu$ – but not (why?) for $\tau$ or $\sigma$. 
Difficulties With Gibbs Sampling

Gibbs sampling is a very popular MCMC technique that is widely used.

It does have some potential drawbacks, however:

1. Need to be able to show that the Gibbs sampler Markov chain is ergodic
   - obvious in many circumstances but sometimes an issue
   - for example Figure 27.5 from Barber shows a 2-dimensional example where
     the chain is not irreducible.

2. If the variables are strongly correlated (negatively or positively) then it may
take too long to reach the stationary distribution
   - see Figure 27.7 from Barber and Figure 11.11 from Bishop.

**Question:** Suppose the random variables $x_1, \ldots, x_d$ are independent. How long
do you think it will take the Gibbs sampler to reach stationarity in that case?
An Example Where Gibbs Fails

Figure 27.5 from Barber: A two dimensional distribution for which Gibbs sampling fails. The distribution has mass only in the shaded quadrants. Gibbs sampling proceeds from the $l^{th}$ sample state $(x_1^l, x_2^l)$ and then sampling from $p(x_2|x_1^l)$, which we write $(x_1^{l+1}, x_2^{l+1})$ where $x_1^{l+1} = x_1^l$. One then continues with a sample from $p(x_1|x_2 = x_2^{l+1})$, etc. If we start in the lower left quadrant and proceed this way, the upper right region is never explored.
MCMC Output Analysis

We are usually interested in scalar-valued functions of the parameter vector $\theta$. Let $\psi(\theta)$ be one such function.

If we have $n$ MCMC samples from the stationary distribution then we have $n$ samples of $\psi(\theta)$:

$$\{\psi_1 := \psi(\theta_1), \ldots, \psi_n := \psi(\theta_n)\}.$$ 

The sample mean is then given by $\bar{\psi} = n^{-1} \sum_{i=1}^{n} \psi_i$.

Posterior intervals for $\psi(\theta)$ can also be calculated:

1. Let $L(\alpha_1) := \alpha_1$ lower sample quantile and $U(\alpha_2) := \alpha_2$ upper sample quantile of $\psi_1, \ldots, \psi_n$. Then $(L(\alpha_1), U(\alpha_2))$ is a $1 - (\alpha_1 + \alpha_2)$ posterior interval.

2. If $\alpha_1 = \alpha_2 = \alpha/2$ then we obtain an equi-tailed $1 - \alpha$ posterior interval.

3. For a highest posterior density interval we solve (numerically) for $\alpha_1$ and $\alpha_2$ such that $\alpha = \alpha_1 + \alpha_2$ and $U(\alpha_2) - L(\alpha_1)$ is minimized
   - could be a union of intervals if posterior of $\psi(\theta)$ is not unimodal
   - kernel density estimates of the posterior density can be plotted to help determine number of modes.
Convergence Diagnostics

In order to use the MCMC samples for inference we must:

1. Ensure the Markov chains have reached stationarity
2. Only use those samples that have been generated after stationarity has been reached.

But it’s impossible to ensure when these two conditions are satisfied since the Markov chain does not begin with the stationary distribution. Instead we can use various methods to assess whether or not stationarity appears to have been reached:

1. **Visual inspection** where we plot variables (of interest) vs iteration #, plot running means of variables (of interest) etc.
   - can be very informative but they also require “manual” work.

2. **Statistical summaries** of MCMC output which are designed to diagnose convergence / non-convergence
   - they can be programmed and so “manual” labor not required
Figure 11.2 Five independent sequences of a Markov chain simulation for the bivariate unit normal distribution, with overdispersed starting points indicated by solid squares. (a) After 50 iterations, the sequences are still far from convergence. (b) After 1000 iterations, the sequences are nearer to convergence. Figure (c) shows the iterates from the second halves of the sequences. The points in Figure (c) have been jittered so that steps in which the random walks stood still are not hidden. The simulation is a Metropolis algorithm described in the example on page 290.
Introduction to Machine Learning
An Extremely Brief Introduction to Graphical Models

Martin Haugh
Department of IE & OR
Columbia University
Email: martin.b.haugh@gmail.com
Overview of Graphical Models

In many applications we often need to model very complex stochastic systems. Such systems often impossible to analyze without imposing some structure: - graphical models provide a very convenient framework for doing this.

E.G. Consider a distribution, \( p(x_1, \ldots, x_N) \), over \( N \) binary variables. To compute we \( p(x_1) \) we need to compute \( \sum_{x_2, \ldots, x_N} p(x_1, \ldots, x_N) \)
- an intractable summation for large \( N \).

But suppose \( p(x_1, \ldots, x_N) = \prod_{i=1}^{N-1} \phi(x_i, x_{i+1}) \). Then

\[
p(x_1) = \sum_{x_2, \ldots, x_N} \prod_{i=1}^{N-1} \phi(x_i, x_{i+1}) = \sum_{x_2} \phi(x_1, x_2) \sum_{x_3, \ldots, x_N} \prod_{i=2}^{N-1} \phi(x_i, x_{i+1})
\]

Now continue to push summations as far to the right as possible and obtain

\[
p(x_1) = \sum_{x_2} \phi(x_1, x_2) \sum_{x_3} \phi(x_2, x_3) \sum_{x_4} \phi(x_3, x_4) \ldots \sum_{x_N} \phi(x_{N-1}, x_N)
\]
- very easy to deal with! Why?
Overview of Graphical Models

Key to identifying structure is to recognize (conditional) dependence / independence relationships between variables.

Graphical models provide a very convenient framework for describing these dependence / independence relationships.

Two main types of graphical models:

1. Undirected graphical models which are also known as Markov networks.
2. Directed graphical models which are also known as Bayesian networks
   - belief networks, i.e. directed acyclic graphs (DAG’s), an important subclass.

Each node in graph corresponds to a random variable.

The edge structure of the graph (and edge direction in case of directed graphs) help determine the conditional independence / dependence relationships between random variables

- these relationships often enable inference, e.g. computation of conditional distributions, to be performed very efficiently.

Graphical models now very popular in statistics and machine learning.
Some History

Bayesian networks are also commonly referred to as directed graphical models or causal models
- developed partly from so-called expert systems literature in 80’s / 90’s
- common apps of expert systems: medical diagnosis, system control etc.

Bayesian networks are generally used when there is understood to be a causal relationship between variables
- although this can be misleading.

Bayesian networks are not Bayesian models in general (although they can be)
- the term “Bayesian” simply reflects the use of Bayes’ Theorem when performing inference in these models.

Markov networks first developed in the physics and vision communities.
They are generally used when the relationship between variables is not causal.

The most general-purpose inference algorithm – the junction tree algorithm – can be applied to both Bayesian networks and Markov networks
- requires conversion of Bayes net to “corresponding” Markov networks
- we will not study the junction tree algorithm in this course.
A Simple Bayesian Network

Figure taken from “An Introduction to Graphical Models”, by Kevin P. Murphy (2001).
A Simple Bayesian Network

Consider the example in the previous figure where each node represents a binary random variable.

There are 4 such random variables:

- “it is cloudy” ($C = \text{true}$)
- “water sprinkler is on” ($S = \text{true}$)
- “it is raining” ($R = \text{true}$)
- “grass is wet” ($W = \text{true}$)

The conditional independence relationships are encoded so that:

A node is independent of its ancestors given its parents.

The model is encoded using a conditional probability table (CPTs) for each node

- the CPT at a node contains that node’s probabilities conditional on the values of the node’s parents
- the CPT for a node without parents ($C$ in our example) just contains the prior probabilities for that node.

**Note:** CPT’s are used for discrete variables but can also use continuous r. vars – Gaussian is a common choice.
A Simple Bayesian Network

Consider now the joint distribution of \((C, S, R, W)\). It must (why?) satisfy

\[
p(c, s, r, w) = p(c) p(s | c) p(r | c, s) p(w | c, s, r).
\] (55)

But now we can use the conditional independence relationships in the graph to simplify (55) and obtain

\[
p(c, s, r, w) = p(c) p(s | c) p(r | c) p(w | s, r).
\] (56)

We used: (i) \(R \perp S | C\) and (ii) \(W \perp C | S, R\).

Note that the joint distribution can be represented more compactly using (56) instead of (55)

- the savings are small in this example but in larger more complex models the savings can be very significant.

**Question:** Can any distribution on \((X_1, \ldots, X_N)\) be represented as a Bayesian network?
Directed Acyclic Graphs (DAGs)

A directed-acyclic graph (DAG) (or belief network) is a special class of Bayesian network where there are no directed cycles

- implies there is a node numbering such that any link from any node always goes to a higher numbered node.

Note that every distribution on \( (X_1, \ldots, X_N) \) can be represented as a DAG since

\[
p(x_1, \ldots, x_n) = p(x_1) p(x_2 | x_1) \ldots p(x_n | x_1, \ldots, x_{n-1})
\]

- ordering of r.vars important for computational reasons as we want to identify and avail of conditional independence relationships

- sometimes a natural ordering will present itself, especially when modeling causal relationships as with the “wet grass” example.

Many efficient algorithms exist for performing inference in DAGs

- inference is the problem of “understanding” the conditional distribution of the graph when some nodes are observed.
Directed Acyclic Graphs (DAGs)

Note ordering of nodes in the DAG of Figure 8.2. This ordering can be used to write

\[
p(x_1, x_2, \ldots, x_7) = p(x_7 \mid x_4, x_5) \cdot p(x_6 \mid x_4) \cdot p(x_5 \mid x_1, x_3) \cdot p(x_4 \mid x_1, x_2, x_3) \cdot p(x_3) \cdot p(x_2) \cdot p(x_1).
\]

More generally for any DAG we have

\[
p(x) = \prod_{k=1}^{K} p(x_k \mid \text{pa}(x_k)) \quad (57)
\]

where \( \text{pa}(x) \) denotes the “parents” of node \( x_k \).

It’s easy (why?) to simulate from a belief network using (57)

- simulating using representation in (57) is called \textit{ancestral sampling}.

Not easy to simulate from conditional distribution when some nodes are observed

- but will see that \textit{Gibbs sampling} easy to implement in that case.
The graphical model of Figure 13.5 represents a dynamic model where time is indexed by \( n \).

Hidden Markov Models (HMMs) and linear dynamical systems are classic examples of these models
- the \( z_i \)'s represent latent variables
- and the \( x_i \)'s are observed variables.

These models have many important applications with the general goal being to infer the values of the \( z_i \)'s from the observed \( x_i \)
- leads to filtering and smoothing problems.
Markov Networks

Markov networks are also known as Markov random fields and undirected graphical models.

These models are also intended to capture certain types of conditional independence relationships. Consider again 3 sets of nodes $A$, $B$ and $C$ as in Figure 8.27.

![Figure 8.27 from Bishop](image)

Would like to define these models in such a way that graph separation holds:

$$A \perp B \mid C \text{ if and only if (in general) any path from any node in } A \text{ to any node in } B \text{ passes through at least one node in } C.$$
Markov Networks

So how should we define probability distributions on these networks so that the above conditional independence test holds?

To answer this we need the following definitions:

**Definition:** A **clique** is a completely connected subset of a graph. A **maximal clique** is a clique that is not contained in a larger clique.

**Definition:** A **potential** $\phi(x)$ is a non-negative function of its arguments.

For a Markov network over random variables $x$ we will assume that

$$p(x) = \frac{1}{Z} \prod_{c \in C} \phi(x_c)$$

(58)

where $C$ denotes the set of maximal cliques in the network and $Z$ is a normalization constant that is sometimes called the **partition function**.
The Hammersley-Clifford Theorem

Let $UI$ be the set of distributions defined over the nodes of an undirected network that are consistent with graph separation holding.

Let $UF$ be the set of distributions defined over the same nodes that can be represented using a factorisation of the form (58).

**Theorem (Hammersley-Clifford):** $UI$ and $UF$ are identical.

So as long as we define distributions in Markov networks according to (58) then graph separation will hold.
We therefore know how to define distributions on Markov networks so that our
conditional independence test holds.

\[ p(x) \propto \phi(x_1, x_2, x_3)\phi(x_2, x_3, x_4)\phi(x_4, x_5, x_6)\phi(x_5, x_6, x_7). \]

Conditional independence statements can now be inferred from structure of network:

- \( p(x_1, x_7|x_4) = p(x_1|x_4)p(x_7|x_4) \)
- \( p(x_k|X_{-k}) = p(x_k|N(x_k)) \) provided \( p(x) > 0, \forall x \)
- \( N(x_k) \) denotes the neighbors, i.e. Markov blanket, of \( x_k \).
The potentials in Markov networks do not have a specific probabilistic interpretation.

This provides us with greater flexibility in choosing potentials
- but ... also raises the question as to how to motivate a particular choice of potential functions.

Quoting from Section 8.3.2 of Bishop:

“This can be done by viewing the potential function as expressing which configurations of the local variables are preferred to others. Global configurations that have a relatively high probability are those that find a good balance in satisfying the (possibly conflicting) influences of the clique potentials.”

Will soon see a (very common) application of Markov networks to image de-noising that adopts this philosophy.
Inference In Graphical Models

Inference is the problem of “understanding” the conditional distribution of the graph when some nodes are observed.

Belief propagation is an efficient “message-passing” algorithm for performing inference in DAGs
- will see examples of message-passing algorithms when we study HMMs
- “efficient” refers to simply doing multiplications and summations efficiently!

Belief propagation can be viewed as a special case of a more general “sum-product” / message-passing algorithm that is performed on so-called factor graphs
- a bipartite graph that is easily constructed from DAGs or Markov networks that are trees.

The junction-tree algorithm is an exact and efficient inference algorithm that applies to general networks
- but for large networks it may be too computationally demanding
- hence the need for approximate inference algorithms.

MCMC and deterministic algorithms, e.g. variational Bayes or loopy belief propagation, are commonly used for large-scale problems.
Example: Oil Exploration

Consider the oil exploration example on the next slide.

A DAG is used to model the geology of a particular area below the seabed of the North Sea

- geology is complex and locating oil requires both exploration and inference.

A decision has been made to drill at node A and the figures display the changes in probabilities of oil being present at other nodes conditional on:

(i) oil being found at A (left-hand heat-map).
(ii) only partial oil being found at A (right-hand heat-map).
Oil Exploration Using a DAG

Figure 4: Difference between conditional probabilities given evidence and prior probabilities. Evidence is observed in prospect A, and follows the explanations in section 3.4. Figure shows the effect of Evidence 1 (left) and 2 (right).

Figure taken from “Strategies for Petroleum Exploration Based on Bayesian Networks: a Case Study”, by Martinelli et al. (2012).
Recall Figure 8.2 which implies

\[ p(x_1, x_2, \ldots, x_7) = p(x_7 \mid x_4, x_5) \cdot p(x_6 \mid x_4) \cdot p(x_5 \mid x_1, x_3) \cdot p(x_4 \mid x_1, x_2, x_3) \cdot p(x_3) \cdot p(x_2) \cdot p(x_1). \]

More generally for any DAG we recall that

\[ p(x) = \prod_{k=1}^{K} p(x_k \mid \text{pa}(x_k)) \tag{59} \]

where \( \text{pa}(x) \) denotes the “parents” of node \( x_k \).

It’s easy (why?) to simulate from a belief network using (59)
- simulating using representation in (59) is called **ancestral sampling**.

Not easy to simulate from conditional distribution when some nodes are observed
- but will see that **Gibbs sampling** easy to implement in that case.
Suppose now that $x_3$, $x_5$ and $x_6$ have been observed and we want to compute the conditional distribution of the unobserved variables.

Using (59) this conditional distribution satisfies

$$p(x_1, x_2, x_4, x_7 \mid x_3, x_5, x_6) = \frac{p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)}{p(x_3, x_5, x_6)}$$

$$= \frac{p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)}{\sum_{x_1, x_2, x_4, x_7} p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)}$$

$$= \frac{\prod_{k=1}^{7} p(x_k \mid \text{pa}(x_k))}{\sum_{x_1, x_2, x_4, x_7} \prod_{k=1}^{7} p(x_k \mid \text{pa}(x_k))}$$

(60)

where $x_3$, $x_5$ and $x_6$ are “clamped” at their observed values in (60).

Computing the normalizing factor, i.e. the denominator, in (60) can be computationally demanding — especially for very large DAGs.

Note that the ordering of the original DAG (with no observed variables) is no longer helpful
- e.g. $x_1$ and $x_3$ are no longer independent once $x_5$ has been observed. Why?
Inference for DAGs via Gibbs Sampling

Questions: Can we still use ancestral sampling to simulate from \( p(x_1, x_2, x_4, x_7 \mid x_3, x_5, x_6) \)? If so, is it efficient?

Question: Can we simulate efficiently from \( p(x_1, x_2, x_4, x_7 \mid x_3, x_5, x_6) \)?

Solution: Yes, using Gibbs sampling!

At each step of the Gibbs sampler we need to simulate from \( p(x_i \mid x_{-i}) \) where any observed values in \( x_{-i} \) are clamped at these values throughout the simulation.

But it’s easy to see (why?) that

\[
p(x_i \mid x_{-i}) = \frac{1}{Z} \ p(x_i \mid pa(x_i)) \prod_{j \in ch(i)} p(x_j \mid pa(x_j))
\]

where \( pa(x_i) \) and \( ch(i) \) are the parent and children nodes, respectively, of \( x_i \), and \( Z \) is the (usually easy to compute) normalization constant

\[
Z = \sum_{x_i} p(x_i \mid pa(x_i)) \prod_{j \in ch(i)} p(x_j \mid pa(x_j)).
\]

The parents of \( x_i \), the children of \( x_i \) and the parents of the children of \( x_i \) are known collectively as the Markov blanket of \( x_i \).
A Markov Network Example: the Ising Model

Figure 8.31 displays an example of an Ising model
- commonly used in statistical physics
- and our application here, computer vision.

$x_i$'s $\in \{-1, 1\}$ represent pixel values in a binary image
- the $x_i$'s are unobserved.

Each $y_i \in \{-1, 1\}$ is a possibly corrupted value of $x_i$.

Goal is to reconstruct original image (as determined by the $x_i$'s) from the corrupted image determined by the $y_i$'s.

Note that all maximal cliques in Fig. 8.31 contain just two random variables
- hence the joint distribution can take the form

$$p(x, y) = \frac{1}{Z} e^{-E(x, y)} \quad (61)$$

where $E(x, y) := h \sum_i x_i - \beta \sum_{i \sim j} x_i x_j - \eta \sum_i x_i y_i$
**Question:** What signs should $\beta$ and $\eta$ have?

**Question:** Does this model capture the “physics” of images?

Given the corrupted image an obvious way to de-noise or restore the original image is to simply look for an image, $x$, that has high conditional probability, i.e. high $p(x \mid y) \propto p(x, y)$.

The method of **iterative conditional modes** (ICM) is an easy-to-implement technique that applies **coordinate ascent** to obtain a local max:

- Initialize by setting $x = y$.
- Pick an $x_i$ and maximize $p(x, y)$ over $x_i$ keeping $x_{-i}$ fixed.
- Cycle through all $x_i$’s and repeat the previous step until convergence.

Easy to improve efficiency of this algorithm.

There are more effective algorithms based on message-passing techniques. And for certain types of model (including this one) there are **graph-cut** algorithms that are guaranteed to find the global max

- see Section 28.9 of Barber and lower-right image in Figure 8.30.
Figure 8.30 from Bishop: Illustration of image de-noising using a Markov random field. The top row shows the original binary image on the left and the corrupted image after randomly changing 10% of the pixels on the right. The bottom row shows the restored images obtained using iterated conditional models (ICM) on the left and using the graph-cut algorithm on the right. ICM produces an image where 96% of the pixels agree with the original image, whereas the corresponding number for graph-cut is 99%.
Inference for Markov Networks via Sampling

MCMC methods are commonly used for inference in Markov networks.

Gibbs sampling, for example, is easy to apply in the Ising model
- can simply divide $\mathbf{x}$ into just two blocks – can you see what they are?
- but when dependence is strong in the Ising model, i.e. when $\beta$ is large, Gibbs can become very slow.
- alternative MCMC methods such as slice sampling can then be employed to overcome this problem.

For general Markov networks it is possible to perform perfect sampling using ancestral sampling on the corresponding junction tree.
Graphical models is now a very large topic in its own right and there are many other interesting questions and issues including:

1. Parameter estimation – this is usually done via MLE, sometimes via the EM algorithm.

2. **Learning the network structure**
   - in many applications (including these slides) the structure was given
   - but in other applications, e.g. computational biology, the central goal is to learn this structure
   - this is a very hard problem (why?) and so greedy-type algorithms are typically employed with penalties imposed for including too many links.

3. **Causality** is an important topic in many domains, e.g. econometrics, public policy. It can be studied using the formalism of Bayesian networks as developed by Judea Pearl
   - see Section 3.4 of Barber, for example.