CLASSIFICATION
**Input:** As with regression, in a *classification problem* we start with measurements $x_1, \ldots, x_n$ in an input space $\mathcal{X}$. (Again think $\mathcal{X} = \mathbb{R}^d$)

**Output:** The *discrete* output space $\mathcal{Y}$ is composed of $K$ possible *classes*:

- $\mathcal{Y} = \{-1, +1\}$ or $\{0, 1\}$ is called binary classification.
- $\mathcal{Y} = \{1, \ldots, K\}$ is called m-ary classification (“m” for “multiclass”)

Instead of a real-valued response, classification assigns $x$ to a category.

- Regression: For pair $(x, y)$, $y$ is the response of $x$.
- Classification: For pair $(x, y)$, $y$ is the class (or category) of $x$. 

Defining a classifier

Classification uses a function $f$ (called a classifier) to map input $x$ to class $y$.

$$y = f(x) : f \text{ takes in } x \in \mathcal{X} \text{ and declares its class to be } y \in \mathcal{Y}$$

As with regression, the problem is two-fold:

- Define the classifier $f$ and its parameters (if it has any).
- Learn the classification rule using a set of “labeled data” $\{(x_i, y_i)\}$. 
STATISTICAL SETTING

How do we measure the quality of a classifier?

For any classifier we care about two sides of the same coin:

- Prediction accuracy: $P(f(x) = y)$.
- Prediction error: $\text{err}(f) = P(f(x) \neq y)$.

To calculate these values, we assume there is a distribution $\mathcal{P}$ over the space of labeled examples generating the data

$$(x_i, y_i) \overset{iid}{\sim} \mathcal{P}, \quad i = 1, \ldots, n.$$

We don’t know $\mathcal{P}$ (e.g., its functional form), but it represents the underlying generative structure of our data.
When is there any hope for finding an accurate classifier?

**Key assumption:** Data \((x_1, y_1), \ldots, (x_n, y_n)\) are i.i.d. random labeled examples with distribution \(P\).

This assumption allows us to say that the past should look like the future.

Regression makes similar assumptions.
Nearest neighbor classifiers
Given training data \((x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}\), construct the classification function \(\hat{f} : \mathcal{X} \rightarrow \mathcal{Y}\) as follows:

For an input \(x\),

1. Let \(x_i\) be the point among \(x_1, x_2, \ldots, x_n\) that is “closest” to \(x\).
2. Return its label \(y_i\).
Question: How should we measure distance between points in $\mathcal{X}$?

The default distance for data in $\mathbb{R}^d$ is the Euclidean one:

$$
\| u - v \|_2 = \left( \sum_{i=1}^{d} (u_i - v_i)^2 \right)^{\frac{1}{2}} \quad \text{(line-of-sight distance)}
$$

But there are many other options that may sometimes be better than $\ell_2$.

- $\ell_p$ for $p \in [1, \infty]$: $\| u - v \|_p = \left( \sum_{i=1}^{d} |u_i - v_i|^p \right)^{\frac{1}{p}}$.

- Edit distance (for strings): How many add/delete/substitutions are required to transform one string to the other.

- Correlation distance (for signal): Measures how correlated two vectors are for signal detection.
Example: OCR with NN classifier

- **Handwritten digits data**: grayscale $28 \times 28$ images, treated as vectors in $\mathbb{R}^{784}$, with labels indicating the digit they represent.

  0 1 2 3 4 5 6 7 8 9

- Split into training set $S$ (60K points) and testing set $T$ (10K points).
- **Training error**: $\text{err}(\hat{f}, S) = 0 \leftarrow$ declare its class to be its own class!
- **Test error**: $\text{err}(\hat{f}, T) = 0.0309$

- Examples of mistakes: (left) test point, (right) nearest neighbor in $S$:

  2 8 3 5 5 4

- **Observation**: First mistake might have been avoided by looking at three nearest neighbors (whose labels are ‘8’, ‘2’, ‘2’) . . .
Given training data \((x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}\), construct the \(k\)-NN classifier as follows:

For an input \(x\),

1. Let \(x_{i_1}, \ldots, x_{i_k}\) be the \(k\) points among \(x_1, \ldots, x_n\) that are closest to \(x\).

2. Return the majority-vote of \(y_{i_1}, y_{i_2}, \ldots, y_{i_k}\).

(Break ties in both steps arbitrarily.)

Example: OCR with \(k\)-NN classifier

<table>
<thead>
<tr>
<th>(k)</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>err((\hat{f}_k), (T))</td>
<td>0.0309</td>
<td>0.0295</td>
<td>0.0312</td>
<td>0.0306</td>
<td>0.0341</td>
</tr>
</tbody>
</table>
Effect of $k$

In general:

- Smaller $k$ ⇒ smaller training error. ($k = 1$ ⇒ has zero training error.)
- Larger $k$ ⇒ predictions are more “stable” due to voting.

1-NN

Purple dotted lines: Bayes classifier’s decision boundaries (to come).
Black solid lines: $k$-NN’s decision boundaries.
Bayes Classifiers
Can we talk about what an “optimal” classifier looks like?

Let \((X, Y)^{iid} \sim \mathcal{P}\). (Again, don’t know \(\mathcal{P}\), but assume it exists)

**Some probability equalities with \(\mathcal{P}\):**

1. The expectation of an indicator of an event is the probability of the event, e.g.,

\[
\mathbb{E}[\mathbb{1}(Y = 1)] = P(Y = 1), \quad \leftarrow \mathbb{1}(\cdot) = \{0, 1\} \text{ depending if } \cdot \text{ is true}
\]

2. Conditional expectations can be random variables, and their expectations remove the randomness,

\[
C = \mathbb{E}[A \mid B] : \quad A \text{ and } B \text{ are both random, so } C \text{ is random}
\]

\[
\mathbb{E}[C] = \mathbb{E}[\mathbb{E}[A \mid B]] = \mathbb{E}[A]
\]
For any classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$, its prediction error is

$$P(f(X) \neq Y) = \mathbb{E}[1(f(X) \neq Y)] = \mathbb{E}[\mathbb{E}[1(f(X) \neq Y) | X]]$$ (†)

The above quantity (‡) is minimized for this particular $x \in \mathcal{X}$ when $f(x) = \text{arg max}_{y \in \mathcal{Y}} P(Y = y | X = x)$.

The classifier $f$ with property (⋆) for all $x \in \mathcal{X}$ is called the Bayes classifier, and it has the smallest prediction error (†) among all classifiers.
For any classifier \( f: \mathcal{X} \rightarrow \mathcal{Y} \), its prediction error is

\[
P(f(X) \neq Y) = \mathbb{E}[1(f(X) \neq Y)] = \mathbb{E}[\mathbb{E}[1(f(X) \neq Y) | X]]
\]

(\dagger)

For each \( x \in \mathcal{X} \),

\[
\mathbb{E}[1(f(X) \neq Y) | X = x] = \sum_{y \in \mathcal{Y}} P(Y = y | X = x) \cdot 1(f(x) \neq y)
\]

(\ddagger)
OPTIMAL CLASSIFIERS

For any classifier $f : \mathcal{X} \to \mathcal{Y}$, its prediction error is

$$P(f(X) \neq Y) = \mathbb{E}[\mathbb{1}(f(X) \neq Y)] = \mathbb{E}\left[\mathbb{E}[\mathbb{1}(f(X) \neq Y) | X]\right] \quad (\dagger)$$

For each $x \in \mathcal{X}$,

$$\mathbb{E}[\mathbb{1}(f(X) \neq Y) | X = x] = \sum_{y \in \mathcal{Y}} P(Y = y | X = x) \cdot \mathbb{1}(f(x) \neq y), \quad (\ddagger)$$

The above quantity (\ddagger) is minimized for this particular $x \in \mathcal{X}$ when

$$f(x) = \arg \max_{y \in \mathcal{Y}} P(Y = y | X = x). \quad (\star)$$

The classifier $f$ with property (\star) for all $x \in \mathcal{X}$ is called the Bayes classifier, and it has the smallest prediction error (\dagger) among all classifiers.
**The Bayes Classifier**

This classification rule divides up the input space $\mathcal{X}$ into different regions by how it predicts:

$$f^*(x) := \arg \max_{y \in \mathcal{Y}} P(Y = y | X = x).$$

The boundaries between these regions are called the *decision boundaries*.

Using Bayes rule and ignoring $P(X = x)$ we equivalently have

$$f^*(x) = \arg \max_{y \in \mathcal{Y}} \underbrace{P(Y = y)}_{\text{class prevalence}} \times \underbrace{P(X = x | Y = y)}_{\text{data likelihood | class}}.$$

- $P(Y = y)$ is called the *class prior*.
- $P(X = x | Y = y)$ is called the *class conditional distribution* of $X$.

Technical comment: If $X$ is a continuous-valued random variable, replace $P(X = x | Y = y)$ with *class conditional density* $p(x | Y = y)$. 
Example: Gaussian Class Conditional Densities

Suppose $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{0, 1\}$, and the distribution $\mathcal{P}$ of $(X, Y)$ is as follows.

- **Class prior**: $P(Y = y) = \pi_y$, $y \in \{0, 1\}$.
- **Class conditional density** for class $y \in \{0, 1\}$: $p_y(x) = N(x|\mu_y, \sigma^2_y)$.
- **Bayes classifier**:

$$f^*(x) = \arg\max_{y \in \{0, 1\}} p(X = x|Y = y)P(Y = y)$$

$$= \begin{cases} 1 & \text{if} \quad \frac{\pi_1}{\sigma_1} \exp \left[ -\frac{(x - \mu_1)^2}{2\sigma^2_1} \right] > \frac{\pi_0}{\sigma_0} \exp \left[ -\frac{(x - \mu_0)^2}{2\sigma^2_0} \right] \\ 0 & \text{otherwise} \end{cases}$$

Notice that we now have a *generative* model.

- For regression we never modeled $x$ with a distribution.
- Now the distribution on $x$ factors into the classification of $y$. 
Example: Gaussian class conditional densities

\[ \pi_0 = \frac{1}{2} \]
\[ \mu_0 = 0 \]
\[ \sigma_0 = 1 \]

\[ \pi_1 = \frac{1}{2} \]
\[ \mu_1 = 1 \]
\[ \sigma_1 = 1 \]

1/2 of x’s from \( N(0, 1) \) (w/ y = 0)
1/2 of x’s from \( N(1, 1) \) (w/ y = 1)
**Example: Gaussian class conditional densities**

\[
\begin{align*}
\pi_0 &= \frac{1}{2} \\
\mu_0 &= 0 \\
\sigma_0 &= 1
\end{align*}
\]

\[
\begin{align*}
\pi_1 &= \frac{1}{2} \\
\mu_1 &= 1 \\
\sigma_1 &= 1
\end{align*}
\]

1/2 of \( x \)'s from \( N(0, 1) \) (w/ \( y = 0 \))

1/2 of \( x \)'s from \( N(1, 1) \) (w/ \( y = 1 \))

**Bayes classifier:**

\[
f^*(x) = \begin{cases} 
1 & \text{if } x > 1/2; \\
0 & \text{otherwise.}
\end{cases}
\]
Example: Gaussian class conditional densities

\[
\pi_0 = 1/2, \quad \mu_0 = 0, \quad \sigma_0 = 1 \\
\pi_1 = 1/2, \quad \mu_1 = 1, \quad \sigma_1 = 1/2
\]

1/2 of x’s from \(\mathcal{N}(0, 1)\) (w/ \(y = 0\))
1/2 of x’s from \(\mathcal{N}(1, 1/2^2)\) (w/ \(y = 1\))
**Example: Gaussian class conditional densities**

\[ \pi_0 = \frac{1}{2}, \quad \mu_0 = 0, \quad \sigma_0 = 1 \]

\[ \pi_1 = \frac{1}{2}, \quad \mu_1 = 1, \quad \sigma_1 = \frac{1}{2} \]

1/2 of x’s from \( \mathcal{N}(0, 1) \) (w/ \( y = 0 \))

1/2 of x’s from \( \mathcal{N}(1, 1/2^2) \) (w/ \( y = 1 \))

**Bayes classifier:**

\[
f^*(x) = \begin{cases} 
1 & \text{if } x \in [0.38, 2.29]; \\
0 & \text{otherwise.}
\end{cases}
\]
Example: Multivariate Gaussians

Data: $\mathcal{X} = \mathbb{R}^2$, Label: $\mathcal{Y} = \{0, 1\}$

Class conditional densities are Gaussians in $\mathbb{R}^2$ with covariance $\Sigma_0$ and $\Sigma_1$.

$\Sigma_0 = \Sigma_1$

Bayes classifier: linear separator

$\Sigma_0 \neq \Sigma_1$

Bayes classifier: quadratic separator
In general, the Bayes classifier may be rather complicated!
**Plug-in classifiers**

**Bayes classifier**

- The Bayes classifier has the smallest prediction error among all possible classifiers.
- Problem: We can’t construct the Bayes classifier without knowing \( \mathcal{P} \)!
  - What is \( P(Y = y|X = x) \), or equiv., \( P(X = x|Y = y) \) and \( P(Y = y) \)?
  - All we have are labeled examples drawn from the distribution \( \mathcal{P} \).

**Plug-in classifiers**

- Using the available data, form an approximation to \( P(Y = y) \) and \( P(X = x|Y = y) \), then “plug-in” to the formula for the Bayes classifier.
A **plug-in classifier** using generative models is:

1. Use *training data* (labeled examples) to obtain approximations for each component in the Bayes classifier formula:

   \[
   f^*(x) = \arg \max_{y \in \mathcal{Y}} P(Y = y)P(X = x | Y = y)
   \]

   So need to approximate class priors and class conditional distributions.

2. **Plug-in** the approximations to this formula to form classifier \( \hat{f} \).

   - Estimating class priors (#1) is easy:
     - Want \( P(Y = y) = \pi_y \), the probability of class \( y \).
     - MLE for \( \pi_y \) is straightforward (was a homework problem for binary case).

   - Estimating class conditional distributions (#2) is hard in general.
     - Usually just use a *simple* parametric models.
**Example: Gaussian class conditional densities**

Here, $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, \ldots, K\}$. Estimate Bayes classifier via MLE:

- **Class priors:** The MLE estimate of $\pi_y$ is $\hat{\pi}_y = \frac{1}{n} \sum_{i=1}^{n} 1(y_i = y)$.

- **Class conditional density:** Choose $p(x|Y = y) = N(x|\mu_y, \Sigma_y)$. The MLE estimate of $(\mu_y, \Sigma_y)$ is

  \[
  \hat{\mu}_y = \frac{1}{n_y} \sum_{i=1}^{n} 1(y_i = y) x_i,
  \]

  \[
  \hat{\Sigma}_y = \frac{1}{n_y} \sum_{i=1}^{n} 1(y_i = y) (x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T.
  \]

  As per MLE, this is just the empirical mean and covariance of class $y$.

- **Plug-in classifier:**

  \[
  \hat{f}(x) = \arg \max_{y \in \mathcal{Y}} \hat{\pi}_y |\hat{\Sigma}_y|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (x - \hat{\mu}_y)^T \hat{\Sigma}_y^{-1} (x - \hat{\mu}_y) \right\}.
  \]
Representing emails

- **Input**: \( x = \mathbb{N}_+^d \) (\( d \) dimensions). \( x_j = \# \) times word \( j \) appears in email
- **Output**: \( \mathcal{Y} = \{-1, +1\} \). Map \{email \to -1, spam \to +1\}

For example, if axis \( j \) represents the term “the,” \( x_j = 3 \) means that “the” occurs three times in an email \( x \).

**Example dimensions**

<table>
<thead>
<tr>
<th></th>
<th>george</th>
<th>you</th>
<th>your</th>
<th>hp</th>
<th>free</th>
<th>work</th>
<th>!</th>
<th>our</th>
<th>re</th>
<th>click</th>
<th>remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>spam</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>email</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Using Bayes classifier**

\[
f(x) = \arg\max_{y \in \{-1,+1\}} P(Y = y|X = x) = \arg\max_{y \in \{-1,+1\}} p(x|Y = y)P(Y = y)
\]
Simplifying assumption

**Naive Bayes** is a Bayes classifier that makes the assumption

\[
p(X = x | Y = y) = \prod_{j=1}^{d} p_j(X_j = x_j | Y = y),
\]

i.e., it treats the dimensions of \( X \) as *conditionally independent* given \( y \).

In spam example

- Number of occurrences of a word carries information about \( y \).
- Co-occurrences (combinations of words occurring together) is ignored.
- These assumptions are made purely for computational reason, i.e., the distributions are easier to define and learn.
Estimation

Class prior

The distribution $P(Y = y)$ is again easy to estimate from the training data:

$$P(Y = y) = \frac{\text{#observations in class } y}{\text{#observations}}$$

Class-conditional distributions

The class conditionals $P(X = x|Y = y) = \prod_j p_j(X_j = x_j|Y = y)$ again requires a modeling assumption for each $j$. For the spam model we could set:

- $P(x_j|Y = y) = \text{Poisson}(\lambda_y^{(j)})$ and learn $\lambda_y^{(j)}$.
- Binarize $x$ (word is/isn’t in email) and use $P(x_j = 1|Y = y) = \pi_y^{(j)}$.
- In either case, we can learn $\lambda_y^{(j)}$ and $\pi_y^{(j)}$ using maximum likelihood.