FEATURE EXPANSIONS
Feature expansions, also called basis expansions, are names given to a technique we’ve already discussed and made use of.

Problem: A linear model on the original feature space $x \in \mathbb{R}^d$ doesn’t work.

Solution: Map the features to a higher dimensional space $\phi(x) \in \mathbb{R}^D$, where $D > d$, and do linear modeling there.

Examples

- For polynomial regression on $\mathbb{R}$, we let $\phi(x) = (x, x^2, \ldots, x^p)$.
- For jump discontinuities, $\phi(x) = (x, 1\{x < a\})$. 
Mapping example for regression

(a) Data for linear regression

(b) Same data mapped to higher dimension

High-dimensional maps can transform the data so output is linear in inputs.

**Left:** Original $x \in \mathbb{R}$ and response $y$.

**Right:** $x$ mapped to $\mathbb{R}^2$ using $\phi(x) = (x, \cos x)^T$. 
Mapping Example for Regression

Using the mapping $\phi(x) = (x, \cos x)^T$, learn the linear regression model

$$y = w_0 + \phi(x)^T w + \epsilon$$

$$= w_0 + w_1x + w_2 \cos x + \epsilon.$$

Left: Learn $(w_0, w_1, w_2)$ to approximate data on the left with a plane.

Right: For each point $x$, map to $\phi(x)$ and predict $y$. Plot as a function of $x$. 
High-dimensional maps can transform the data so it is linearly separable.

**Left:** Original data in $\mathbb{R}^2$.

**Right:** Data mapped to $\mathbb{R}^3$ using $\phi(x) = (x_1^2, x_1x_2, x_2^2)^T$. 
MAPPING EXAMPLE FOR CLASSIFICATION

Using the mapping $\phi(x) = (x_1^2, x_1x_2, x_2^2)^T$, learn a linear classifier

\[
y = \text{sign}(w_0 + \phi(x)^T w) = \text{sign}(w_0 + w_1 x_1^2 + w_2 x_1 x_2 + w_3 x_2^2).
\]

Left: Learn $(w_0, w_1, w_2, w_3)$ to linearly separate classes with hyperplane.

Right: For each point $x$, map to $\phi(x)$ and classify. Color decision regions in $\mathbb{R}^2$. 
What expansion should I use?

This is not clear-cut. The illustrations required knowledge about the data that we likely wouldn’t have (especially if it’s in high dimensions).

One solution is to use the “kitchen sink”: If you can think of it, then use it. Then do $\ell_1$ minimization

$$w_{\ell_1} = \arg\min_w \sum_{i=1}^{n} f(y_i, \phi(x_i), w) + \lambda \|w\|_1.$$  

We know that this will find a sparse subset of the dimensions of $\phi(x)$ to use.

Often however, we only really need the dot products $\phi(x_i)^T \phi(x_j)$. This is called a *kernel* and can produce some interesting results.
Kernels
Predicting new data

Let \( x_i \in \mathbb{R}^d \) and \( y_i \in \{-1, +1\} \) for \( i = 1, \ldots, n \) observations. We saw with the Perceptron that the separating hyperplane is constructed from the data,

\[
w = \sum_{i \in \mathcal{M}} y_i x_i,
\]

where \( \mathcal{M} \) is the sequentially constructed set of misclassified examples. Predictions for new data \( x_0 \) are

\[
y_0 = \text{sign}(x_0^T w) = \text{sign} \left( \sum_{i \in \mathcal{M}} y_i x_0^T x_i \right)
\]

We’ve taken feature expansion for granted up until now, but we can write this version explicitly as

\[
y_0 = \text{sign}(\phi(x_0)^T w) = \text{sign} \left( \sum_{i \in \mathcal{M}} y_i \phi(x_0)^T \phi(x_i) \right)
\]

The dot product \( \phi(x_0)^T \phi(x_i) \) is referred to as a \textit{kernel}. 
Kernel definition

A kernel function $K(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a symmetric function with the following property:

**Definition:** For any set of data $x_1, \ldots, x_n \in \mathbb{R}^d$, the $n \times n$ matrix $K$, with $K_{ij} = K(x_i, x_j)$ is **positive semidefinite**.

Intuitively, this means $K$ satisfies the properties of a covariance matrix.

Mercer’s theorem

If the function $K(\cdot, \cdot)$ satisfies the above properties of a kernel, then there exists a mapping $\phi : \mathbb{R}^d \to \mathbb{R}^D$ such that

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j).$$

If we directly define $\phi(\cdot)$, then this is straightforward. However, sometimes we only need $K(\cdot, \cdot)$, in which case we can avoid ever defining $\phi(\cdot)$. 
Gaussian kernel (radial basis function)

By far the most popular kernel is the Gaussian kernel, also called the *radial basis function* (RBF),

\[ K(x, x') = a \exp \left\{ -\frac{1}{b} \|x - x'\|^2 \right\}. \]

- This is a good, general-purpose kernel that usually works well.
- It takes into account proximity of \( x \) – things close together in space, as defined by \( b \) (the *kernel width*), should be similar in class or response.

In this case, the mapping \( \phi(x) \) that produces the RBF kernel is *infinite dimensional* (i.e., a function instead of a vector). Therefore

\[ K(x, x') = \int \phi_t(x) \phi_t(x') \, dt. \]

- \( K(x, x') \) is like a Gaussian on \( x \) with \( x' \) as the mean (or vice versa).
- Therefore, the above integral indicates it is the marginal distribution using a distribution \( \phi_t(x) \) with \( x \) a parameter and \( t \) a random variable.
Another kernel

$$\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{1 + \frac{d}{2} + \frac{d(d-1)}{2}}$$

Map: $$\phi(x) = (1, \sqrt{2}x_1, \ldots, \sqrt{2}x_d, x_1^2, \ldots, x_d^2, \ldots, \sqrt{2}x_ix_j, \ldots)$$

Kernel: $$\phi(x)^T \phi(x') = K(x, x') = (1 + x^T x')^2$$

In fact, we can show: $$K(x, x') = (1 + x^T x')^b$$, for $$b > 0$$ is a kernel as well.

Kernel arithmetic

Certain functions of kernels can produce new kernels.

Let $$K_1$$ and $$K_2$$ be any two kernels, then constructing $$K$$ in the following ways produces a new kernel (among many other ways):

$$K(x, x') = K_1(x, x')K_2(x, x')$$
$$K(x, x') = K_1(x, x') + K_2(x, x')$$
$$K(x, x') = \exp\{K_1(x, x')\}$$
KERNELIZED PERCEPTRON

Returning to the Perceptron

We write the feature-expanded decision as

\[ y_0 = \text{sign} \left( \sum_{i \in \mathcal{M}} y_i \phi(x_0)^T \phi(x_i) \right) \]
\[ = \text{sign} \left( \sum_{i \in \mathcal{M}} y_i K(x_0, x_i) \right) \]

We can pick the kernel we want to use. Let’s pick the RBF (set \( a = 1 \)). Then

\[ y_0 = \text{sign} \left( \sum_{i \in \mathcal{M}} y_i \exp\left\{-\frac{1}{b} \|x_0 - x_i\|^2\right\} \right) \]

The dot product reduces integrals of functions \( \phi_t(x) \) to a simple equation.

What is this doing?

- Notice \( 0 < K(x_0, x_i) \leq 1 \), with bigger values when \( x_0 \) is closer to \( x_i \).
- This is like a “soft voting” among the data picked by Perceptron.
- We can generalize this to something very similar to a “soft” \( k \)-NN.
**Kernel k-NN**

**An extension**

Instead of summing over misclassified data $\mathcal{M}$, sum over all the data:

$$y_0 = \text{sign} \left( \sum_{i=1}^{n} y_i \exp \left\{ -\frac{1}{b} \| x_0 - x_i \|^2 \right\} \right).$$

Notice that the **decision** doesn’t change if we divide by a positive constant.

Let: $Z = \sum_{j=1}^{n} \exp \left\{ -\frac{1}{b} \| x_0 - x_j \|^2 \right\}$

Construct: Probability vector $p(x_0)$, $p_i(x_0) = \frac{1}{Z} \exp \left\{ -\frac{1}{b} \| x_0 - x_i \|^2 \right\}$

Declare: $y_0 = \text{sign} \left( \sum_{i=1}^{n} y_i p_i(x_0) \right)$

- Consider the case where we set the $k$ largest values in $p(x_0)$ to 1, and the rest to 0. This is exactly $k$-NN! (for the binary case)
- Now we have added a confidence to the vote, and incorporate all data.
- Set $b$ so that most $p_i(x_0) \approx 0$. (i.e., only focus on neighborhood)
Kernel regression

Nadaraya-Watson model

The developments are almost limitless (see, e.g., ESL Ch. 5-6).

Here’s a regression example almost identical to the kernelized $k$-NN:

**Before:** $y \in \{-1, +1\}$

**Now:** $y \in \mathbb{R}$

Using the RBF kernel, for a new $(x_0, y_0)$ predict

$$y_0 = \sum_{i=1}^{n} y_i \frac{K(x_0, x_i)}{\sum_{j=1}^{n} K(x_0, x_j)}.$$

**What is this doing?**

We’re taking a locally weighted average of all $y_i$ for which $x_i$ is close to $x_0$ (as decided by the kernel width). *Gaussian processes* are another option. . .
GAUSSIAN PROCESSES
**Kernelized Bayesian Linear Regression**

**Regression setup:** For \( n \) observations, with responses in \( y \in \mathbb{R}^n \) and their features in the \( n \) rows of matrix \( X \), we define the prior and likelihood

\[
w \sim N(0, \lambda^{-1}I), \quad y \sim N(Xw, \sigma^2 I).
\]

**Marginalizing:** What if we integrate out \( w \)? Then we can find that

\[
p(y|X) = \int p(y|X, w)p(w)dw = N(0, \sigma^2 I + \lambda^{-1}XX^T).
\]

**Kernelization:** Replace the \( i \)th row of \( X \) with mapping \( \phi(x_i)^T: XX^T \rightarrow \Phi\Phi^T \). Then \( (\Phi\Phi^T)_{ij} = \phi(x_i)^T\phi(x_j) = K(x_i, x_j) \). We can define \( K \) directly and say

\[
p(y|X) = \int p(y|X, w)p(w)dw = N(0, \sigma^2 I + \lambda^{-1}K).
\]

This is called a *Gaussian process*. 
**Gaussian processes**

**Definition**

- Let $f(x) \in \mathbb{R}$ and $x \in \mathbb{R}^d$.
- Define the kernel $K(x, x')$ between two points $x$ and $x'$.
- Then $f(x)$ is a **Gaussian process** and $y(x)$ is the noisy observed process if, for all finite sets $\{(x_i, y_i)\}_{i=1}^n$,

$$
y | f \sim N(f, \sigma^2 I), \quad f \sim N(0, K) \quad \iff \quad y \sim N(0, \sigma^2 I + K)$$

where $y = (y_1, \ldots, y_n)^T$ and $K$ is $n \times n$ with $K_{ij} = K(x_i, x_j)$.

**Comments:**

- Combined the previous $\lambda^{-1}$ with $K$ (for notation only).
- Typical breakdown: $f(x)$ is the GP and $y(x)$ equals $f(x)$ plus i.i.d. noise.
- The full definition also has a mean function $m(x)$, which we’ll ignore (just center $y$ so $\bar{y} = 0$ and it’s not needed).
- The kernel is what keeps this from being “just a Gaussian.”
**Gaussian Processes**

Above: A Gaussian process $f(x)$ generated using $m(x) = 0$ and the Gaussian kernel,

$$K(x_i, x_j) = \exp \left\{ -\frac{\|x_i - x_j\|^2}{b} \right\}.$$

Right: The covariance of $f(x)$ defined by $K$. 

COMS W4721: Machine Learning for Data Science, Spring 2015
GAUSSIAN PROCESSES

\[ f(x) \]

\[ y(x) \]

↓: data are noisy samples from this underlying function
**Predictions with Gaussian processes**

**Posterior of $y(x)$**

Given measured data $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$, the posterior distribution of $y(x)$ can be evaluated at any *new* $x$ to make predictions.

Let $K(x, \mathcal{D}_n) = [K(x, x_1), \ldots, K(x, x_n)]$ and $K_n$ be the kernel matrix restricted to $\mathcal{D}_n$. Following Bayes rule

$$y(x)|X, Y \sim N(mean(x), var(x)),$$

$$mean(x) = K(x, \mathcal{D}_n)K_n^{-1}y,$$

$$var(x) = \sigma^2 + K(x, x) - K(x, \mathcal{D}_n)K_n^{-1}K(x, \mathcal{D}_n)^T$$

As before, if we want the posterior of $f(x)$ instead, we have the same mean and the variance doesn’t have $\sigma^2$. 


What does the posterior distribution of \( y(x) \) look like?

- We have data marked by an \( \times \).
- These values pin down the function \( y(x) \) nearby (\( \sigma^2 \) set very small)
- From three slides back, we get a mean and variance for every possible \( x \).
As previously discussed, in many situations we can choose the location \( x \) to measure \( y(x) \). Active learning is a way to do this efficiently.

- Let \( \mathcal{D}_n = \{x_i\} \) contain \( n \) locations with measurements \( y(x_i) \). Let \( \mathcal{D} = \{x\} \) be a set of unmeasured locations we can choose from.

- We saw how an active learning approach is to measure at the location \( x \in \mathcal{D} \) with the greatest posterior uncertainty, in this case

\[
x_{n+1} = \arg \max_{x \in \mathcal{D}} k(x, x) - k(x, \mathcal{D}_n)K^{-1}_n k(x, \mathcal{D}_n)^T.
\]

- Two observations:
  1. Selection doesn’t depend on measurements \( y(x_i) \).
  2. Points will be selected far away from \( x \in \mathcal{D}_n \).
**Figure:** The first 15 points selected for active learning using Gaussian process.
A Gaussian process $f(x)$ passed to sigmoid $\sigma(f(x))$.

$\sigma(f(x))$ decision boundary in $\mathbb{R}^2$.