Acknowledgement:
Many of these slides are derived from Tom Mitchell, Pascal Poupart, Pieter Abbeel, Eric Eaton, Carlos Guestrin, William Cohen, and Andrew Moore.
Kernel Methods

- Goal: keep advantages of linear models, but make them capture non-linear patterns in data!

- **How?**
  - By mapping data to higher dimensions where it exhibits linear patterns
  - By rewriting linear models so that the mapping never needs to be explicitly computed
The Kernel Trick

- Rewrite learning algorithms so they only depend on dot products between two examples

- Replace dot product $\phi(x)^T \phi(z)$ by kernel function $k(x, z)$ which computes the dot product implicitly

How?
Recall perceptron

- Let $y \in \{-1, 1\}$ for all $y$
- Initialize weights $\mathbf{w} = 0$, $b = 0$
- Run through the training data $i = 1, \ldots, n$
  - if $y^i(\mathbf{w} \cdot \mathbf{x}^i + b) < 0$
    - $\mathbf{w} \leftarrow \mathbf{w} + y^i \mathbf{x}^i$
    - $b \leftarrow b + y^i$
"Kernelizing" the perceptron

- **Naïve approach**: let’s explicitly train a perceptron in the new feature space

- Let \( y \in \{-1, 1\} \forall y \)

- Initialize weights \( w = 0, b = 0 \)

- Run through the training data \( i = 1, ..., n \)
  - if \( y^i (w \cdot \phi(x^i) + b) < 0 \)
    - \( w \leftarrow w + y^i \phi(x^i) \)
    - \( b \leftarrow b + y^i \)

- And then summarize

- For making prediction on a new example \( x^i \)
“Kernelizing” the perceptron

- **Naïve approach**: let’s explicitly train a perceptron in the new feature space

- Let \( y \in \{-1, 1\} \ \forall y \)

- Initialize weights \( w = 0, b = 0 \)

- Run through the training data \( i = 1, \ldots, n \)
  - if \( y^i(w \cdot \phi(x^i) + b) < 0 \)
    - \( w \leftarrow w + y^i\phi(x^i) \)
    - \( b \leftarrow b + y^i \)

- And then summarize
  - \( w = \sum_{j=1}^{n} \alpha_j y^i \phi(x^j) \), where \( \alpha_j \) is the number of misclassifications
  - For making prediction on a new example \( x^i \)
    - \( w \cdot \phi(x^i) + b = \sum_{j=1}^{n} \alpha_j y^i \phi(x^j) \cdot \phi(x^i) + b = \sum_{j=1}^{n} \alpha_j y^i k(x^j, x^i) + b \)
Kernel perceptron

- Let $y \in \{-1, 1\} \forall y$
- Initialize
- Run through the training data $i = 1, \ldots, n$
  - if

- So we can plug in kernel functions and implicitly operate in the higher-dimensional space
Kernel perceptron

- Let $y \in \{-1, 1\} \forall y$

- Initialize $\alpha_1 = \ldots = \alpha_n = 0, b = 0$

- Run through the training data $i = 1, \ldots, n$
  - if $y^i (\sum_{j=1}^{n} \alpha_j y^j k(x^j, x^i) + b) < 0$
    - $\alpha_i \leftarrow \alpha_i + 1$
    - $b \leftarrow b + y^i$

- So we can plug in kernel functions and implicitly operate in the higher-dimensional space
Kernel perceptron: primal and dual forms

**Primal form**

- Let $y \in \{-1, 1\} \forall y$
- Initialize weights $w = 0, b = 0$
- Run through the training data $i = 1,\ldots,n$
  - if $y^i(w \cdot \phi(x^i) + b) < 0$
    - $w \leftarrow w + y^i \phi(x^i)$
    - $b \leftarrow b + y^i$

**Dual form**

- Let $y \in \{-1, 1\} \forall y$
- Initialize $\alpha_1 = \ldots = \alpha_n = 0, b = 0$
- Run through the training data $i = 1,\ldots,n$
  - if $y^i(\sum_{j=1}^{n} \alpha_j y^j k(x^j, x^i) + b) < 0$
    - $\alpha_i \leftarrow \alpha_i + 1$
    - $b \leftarrow b + y^i$
Commonly used Kernels

- Polynomial Kernel of degree exactly $d$
  \[ K(u, v) = (u \cdot v)^d \]

- Polynomial Kernel of degree up to $d$
  \[ K(u, v) = (u \cdot v + 1)^d \]

- Gaussian Kernel
  \[ K(u, v) = \exp \left( -\frac{||u - v||^2}{2\sigma^2} \right) \]

- ...and many others
Which functions can be Kernels?

- not all functions
- for some definitions of $k(x,z)$ there is no corresponding projection $\varphi(x)$

- Well developed theory on this, including how to construct new kernels from existing ones

- Initially kernels were defined over data points in Euclidean space, but more recently over strings, over trees, over graphs, ...
Kernel Methods

- Goal: keep advantages of linear models, but make them capture non-linear patterns in data!

- How?
  - By mapping data to higher dimensions where it exhibits linear patterns
  - By rewriting linear models so that the mapping never needs to be explicitly computed
Discussion

- Other algorithms can be “Kernelized”:
  - Logistic regression
  - We’ll talk about Support Vector Machines next…

- Do Kernels address all the downsides of “feature explosion”?
  - Helps reduce computation cost during training
  - But overfitting remains an issue