6.036 Introduction to Machine Learning

(meets with 6.862)

Nonlinear Classification - Kernels (Chapter 5 in notes)

Administrivia

Project #1 due tomorrow Friday 3/3 @ 9AM.

Homework #3 posted, due 3/10 @ 9AM.

As always:

- Check LMOD/Piazza for announcements.
- To contact staff, use Piazza (6036-staff@lists.csail.mit.edu for exceptions only)

So far…

- **Tasks:**
  - Binary classification (linear)

$$h\left(\quad\right) = -1 \qquad h : \mathcal{X} \to \{-1, +1\}$$

  - Regression (linear)

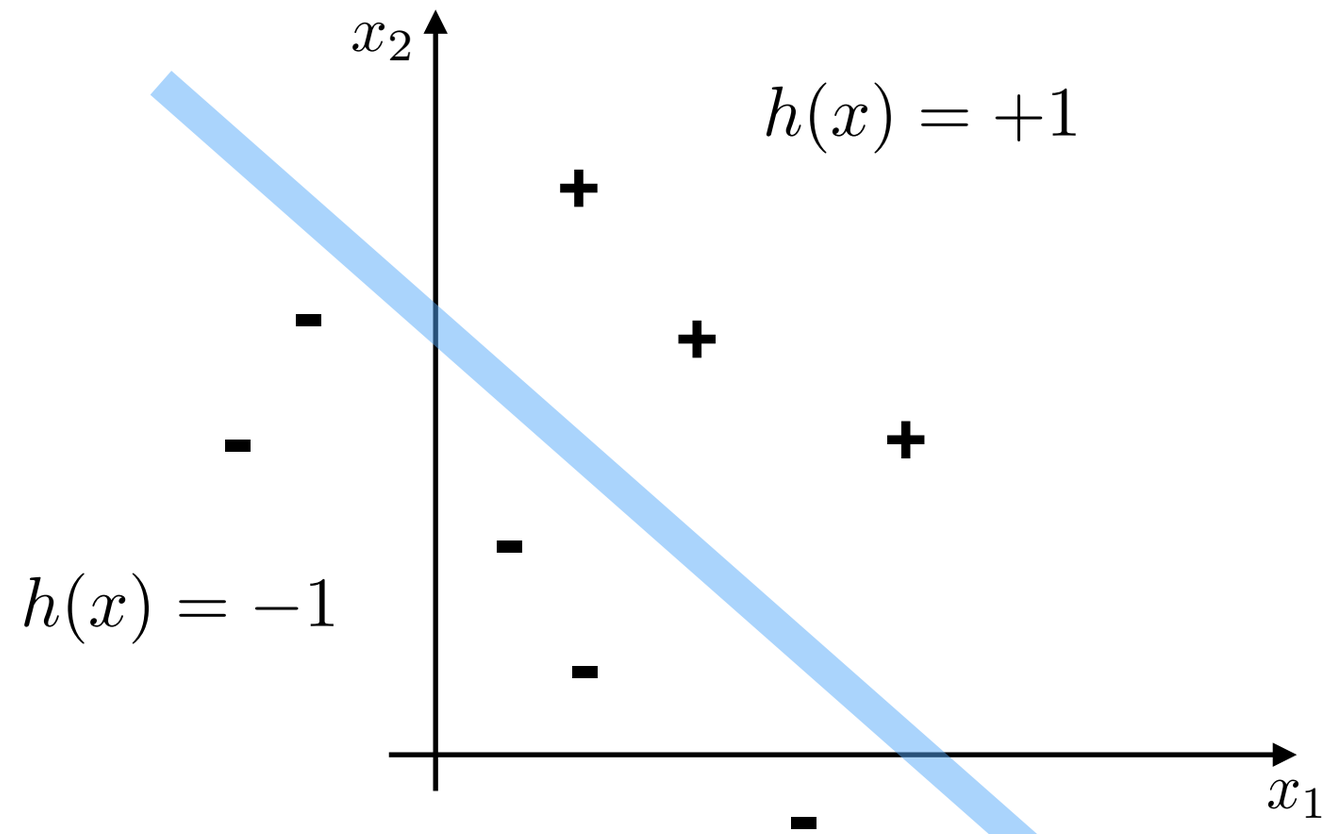
$$h\left(\quad\right) = \$1{,}349{,}000 \qquad h : \mathcal{X} \to \mathbb{R}$$

- **Formulations:** hyperplane separation, empirical risk minimization, regularization, …

- **Algorithms:** perceptron, stochastic gradient, …

Why "linear"?

- Binary Classification: **hyperplane** separation



What if data is *not* linearly separable?

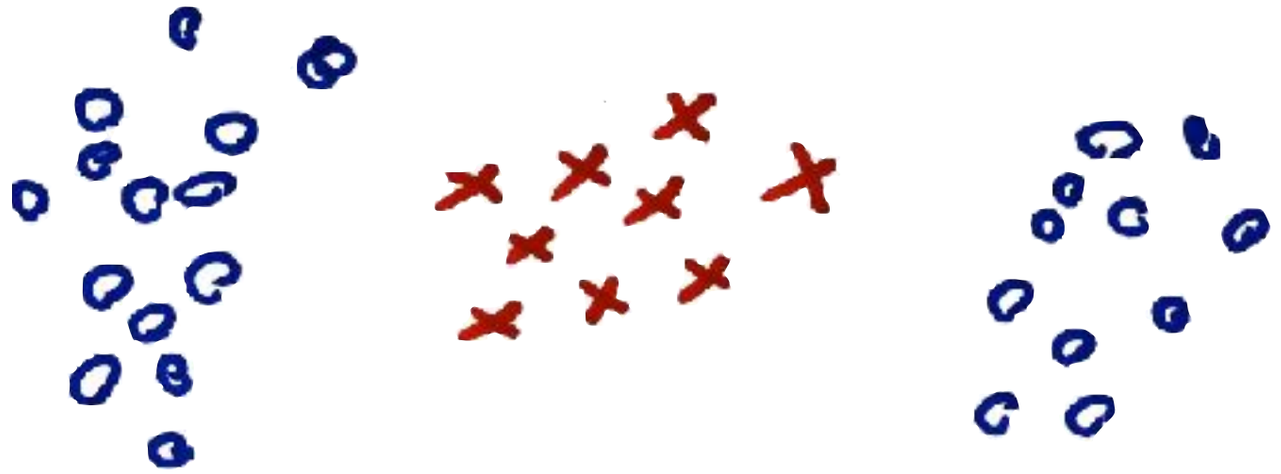
- Regression: predictor is a **linear function** of **feature vectors**

$$f(x;\theta,\theta_0) = \theta \cdot x + \theta_0 = \sum_{i=1}^{d} \theta_i x_i + \theta_0$$

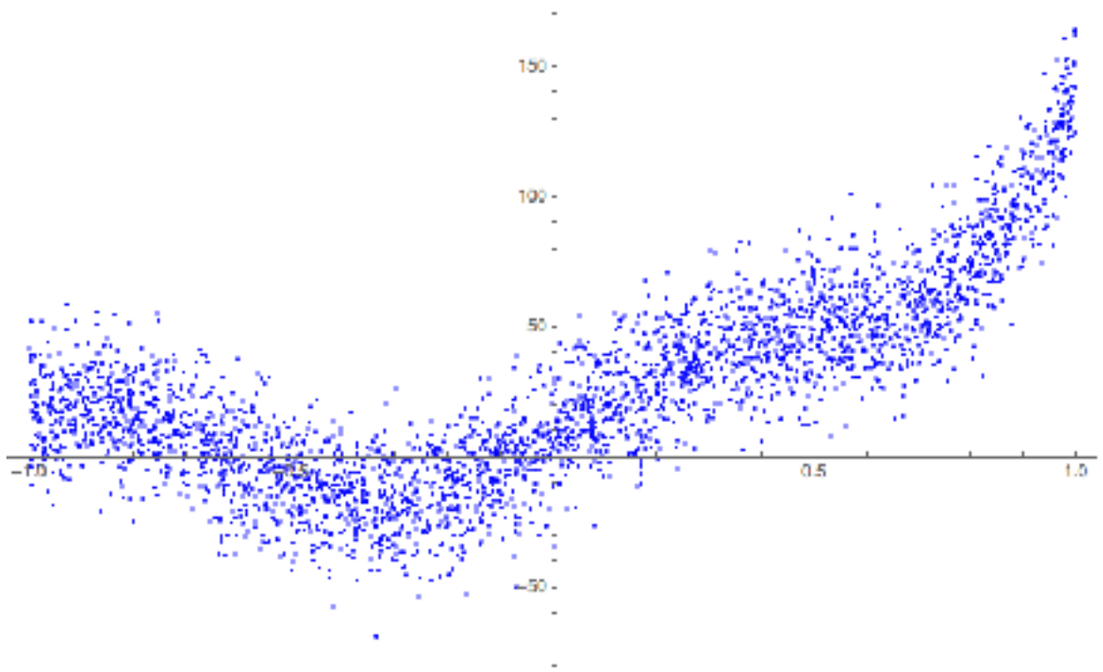
What if the *right* predictor is not linear?

The world is complicated…

Sometimes (often), even if data points are well-separated, a hyperplane may not be enough…

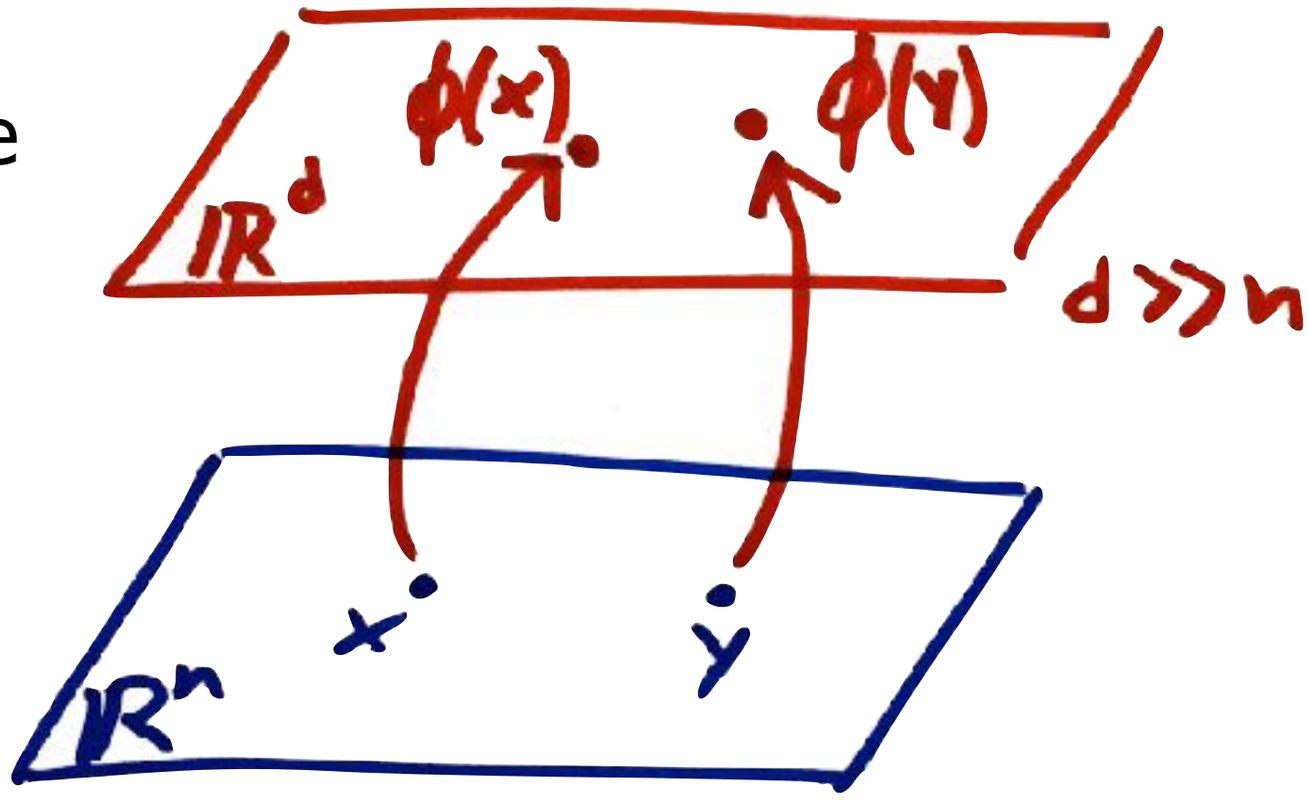


Or, the “true” relationship may be nonlinear…



Key idea: *create* new features

- Nonlinear map φ(x) to a higher-dimensional space



- Example:

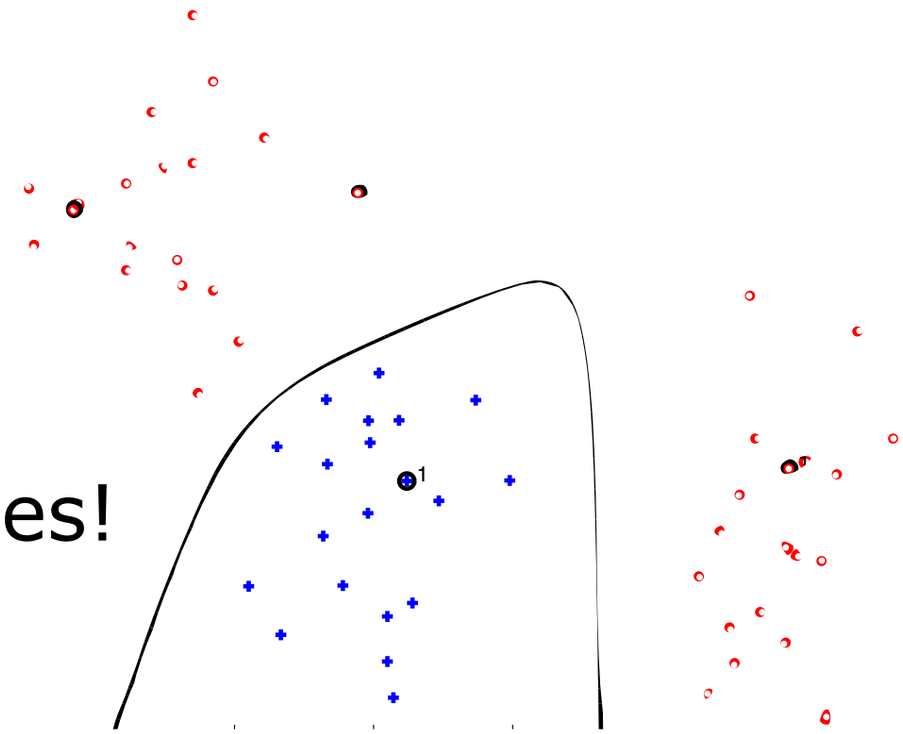
$$\phi(\,[x_1, x_2]^T\,) = [x_1,\ x_2,\ \sqrt{2}x_1x_2,\ x_1^2,\ x_2^2]^T$$

- Here the map is φ:R^2 -> R^5
- Why this may be a good idea?

Linear becomes nonlinear!

- Upstairs, we separate with hyperplanes

- However, in the base space, decision boundaries are *not* hyperplanes!



Q: Why?

- *Much* more expressive power! :)

- But, dimension may be *much* higher: in our example, $O(n^2)$
- Typically, computationally too expensive :(

However…

Notice that many algorithms *only rely on inner products*

- Binary classification (e.g., **perceptron**)

$$\text{If } y^{(t)}(\theta \cdot x^{(t)}) \leq 0 \ \ (\text{mistake})$$
$$\text{then } \theta \leftarrow \theta + y^{(t)} x^{(t)}$$

- Linear regression (e.g, **stochastic gradient**)

$$\text{set } \theta^{(0)} = 0$$
$$\text{randomly select } t \in \{1, \ldots, n\}$$
$$\theta^{(k+1)} = \theta^{(k)} + \eta_k (y^{(t)} - \theta \cdot x^{(t)}) x^{(t)}$$

Kernel "magic"

In some cases, we can very efficiently compute *inner products between feature vectors*.

E.g, for

$$\phi(\,[x_1, x_2]^T\,) = [x_1,\ x_2,\ \sqrt{2}x_1x_2,\ x_1^2,\ x_2^2]^T$$

we have (magic!)

$$\phi(x) \cdot \phi(y) = x \cdot y + (x \cdot y)^2$$

More generally, we may have

$$\phi(x) \cdot \phi(y) = K(x, y)$$

For some "easy" function K(.,.).

Kernels

Associated to a feature map φ(.), define the *kernel:*

$$K(x, y) := \phi(x) \cdot \phi(y)$$

If kernel can be computed efficiently, then we can compute inner products between feature vectors!

Can extend ("*kernelize*") all the algorithms we have learned so they can use nonlinear features!

Kernel perceptron

$$\text{If } y^{(t)}(\theta \cdot \phi(x^{(t)})) \leq 0 \ \ (\text{mistake})$$

$$\text{then } \theta \leftarrow \theta + y^{(t)}\phi(x^{(t)})$$

By construction, at any stage of the algorithm we have

$$\theta = \sum_{i=1}^{n} \alpha_i y^{(i)} \phi(x^{(i)})$$

where α_i is the number of mistakes made in data point *i*

But then

$$\theta \cdot \phi(x) = \sum_{i=1}^{n} \alpha_i y^{(i)} \phi(x^{(i)}) \cdot \phi(x) = \sum_{i=1}^{n} \alpha_i y^{(i)} K(x^{(i)}, x)$$

To run the algorithm, *only* need the kernel, not features!

Standard vs kernel perceptron

$$\text{If } y^{(t)}(\theta \cdot \phi(x^{(t)})) \leq 0 \ \ \text{(mistake)}$$

$$\text{then } \theta \leftarrow \theta + y^{(t)}\phi(x^{(t)})$$

Keep parameter in "feature" representation, then:

$$\text{If } y^{(t)} \sum_{i=1}^{n} \alpha_i y^{(i)} K(x^{(i)}, x^{(t)}) \leq 0 \ \ \text{(mistake)}$$

$$\text{then } \alpha_t \leftarrow \alpha_t + 1$$

Often, solution quite sparse (many α_i are zero)

Kernel linear regression

- Recall regularized empirical risk minimization:

$$J_{n,\lambda}(\theta) = R_n(\theta) + \frac{\lambda}{2}\|\theta\|^2$$

$$= \frac{1}{n}\sum_{t=1}^{n}(y^{(t)} - \theta \cdot x^{(t)})^2/2 + \frac{\lambda}{2}\|\theta\|^2$$

- Solving optimality conditions (derivative equal to zero):

$$n\lambda\alpha_t + \sum_{i=1}^{n}\alpha_i K(x^{(i)}, x^{(t)}) = y^{(t)} \quad \text{for all } t = 1, \ldots, n$$

- Predictor is now

$$\hat{\theta} \cdot \phi(x) = \sum_{i=1}^{n}\hat{\alpha}_i \phi(x^{(i)}) \cdot \phi(x) = \sum_{i=1}^{n}\hat{\alpha}_i K(x^{(i)}, x)$$

Examples of kernels

- *Polynomial* kernel

$$K(x, y) = (1 + x \cdot y)^d$$

- *Gaussian* (or "radial basis") kernel

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Q: What are the associated feature maps?

Kernel rules

How to *combine* known kernels to produce new ones?

- $K(x,y) = 1$ is a kernel
- If $K(x,y)$ is a kernel , then so is $f(x)*K(x,y)*f(y)$.
- If K_1 and K_2 are kernels, so are K_1+K_2 and K_1*K_2

Q: Can you use this to show the Gaussian kernel is actually valid?

Summary - Kernels

- Nonlinear map φ(x) to high-dim feature space

- Kernel efficiently computes inner products in feature space (e.g., polynomial, Gaussian)

$$K(x, y) := \phi(x) \cdot \phi(y)$$

- Algorithms (e.g., perceptron and linear regression) can be "kernelized", by replacing inner products by kernel

- Yields nonlinear decision boundaries for classification, or nonlinear functions for regression
  - much more powerful!