Kernel methods II
The kernel trick

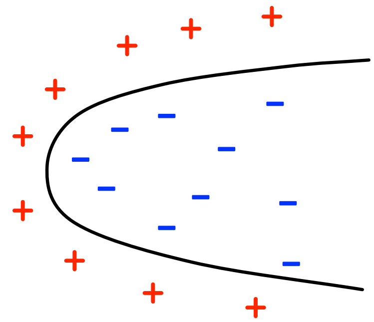
Sanjoy Dasgupta

University of California, San Diego

Topics we'll cover

1. The kernel trick for quadratic boundaries
2. The kernel Perceptron

Adding new features



Actual boundary is something like $x_1 = x_2^2 + 5$.

- This is quadratic in $x = (x_1, x_2)$
- But it is linear in $\Phi(x) = (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$

Basis expansion: embed data in a higher-dimensional feature space.
Then we can use a linear classifier!

Perceptron with basis expansion

Learning in the higher-dimensional feature space:

- $w = 0$ and $b = 0$
- while some $y(w \cdot \Phi(x) + b) \leq 0$:
  - $w = w + y\,\Phi(x)$
  - $b = b + y$

Problem: number of features has now increased dramatically.
For MNIST, with quadratic boundary: from 784 to 308504.

The kernel trick: implement this without ever writing down a vector in the higher-dimensional space!

The kernel trick

- $w = 0$ and $b = 0$
- while some $y^{(i)}(w \cdot \Phi(x^{(i)}) + b) \leq 0$:
  - $w = w + y^{(i)}\,\Phi(x^{(i)})$
  - $b = b + y^{(i)}$

1. w is always a linear combination of the $\Phi(x^{(i)})$.

$$w = \sum_{j=1}^{n} \alpha_j y^{(j)} \Phi(x^{(j)})$$

   Represent w in **dual** form: $\alpha = (\alpha_1, \ldots, \alpha_n)$.

2. Compute $w \cdot \Phi(x)$ using the dual representation.

$$w \cdot \Phi(x) = \sum_{j=1}^{n} \alpha_j y^{(j)} (\Phi(x^{(j)}) \cdot \Phi(x))$$

3. Compute $\Phi(x) \cdot \Phi(z)$ without ever writing out $\Phi(x)$ or $\Phi(z)$.

Computing dot products

First, in 2-d.
Suppose $x = (x_1, x_2)$ and $\Phi(x) = (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$.

Actually, tweak a little: $\Phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$

What is $\Phi(x) \cdot \Phi(z)$?

Computing dot products

Suppose $x = (x_1, x_2, \ldots, x_d)$ and

$$\Phi(x) = (1, \sqrt{2}x_1, \ldots, \sqrt{2}x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{d-1}x_d)$$

$$\begin{aligned}\Phi(x) \cdot \Phi(z) &= (1, \sqrt{2}x_1, \ldots, \sqrt{2}x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{d-1}x_d)\cdot \\ &\quad (1, \sqrt{2}z_1, \ldots, \sqrt{2}z_d, z_1^2, \ldots, z_d^2, \sqrt{2}z_1z_2, \ldots, \sqrt{2}z_{d-1}z_d) \\ &= 1 + 2\sum_i x_i z_i + \sum_i x_i^2 z_i^2 + 2\sum_{i \neq j} x_i x_j z_i z_j \\ &= (1 + x_1 z_1 + \cdots + x_d z_d)^2 \quad = \quad (1 + x \cdot z)^2\end{aligned}$$

For MNIST:
We are computing dot products in 308504-dimensional space.
But it takes time proportional to 784, the original dimension!

Kernel Perceptron

Learning from data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathcal{X} \times \{-1, 1\}$

Primal form:

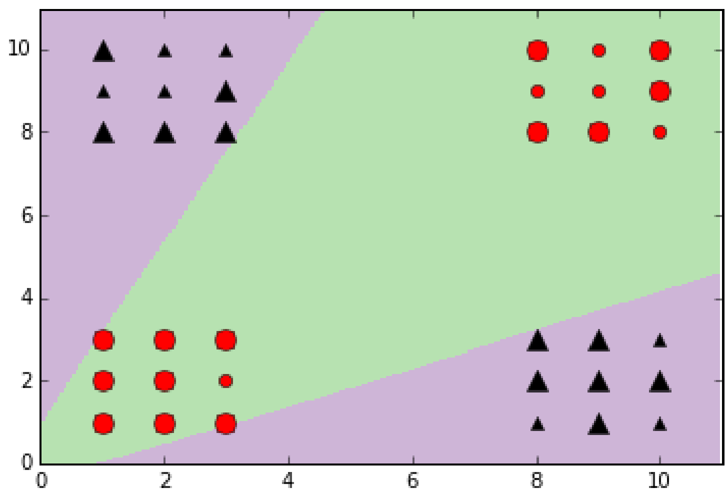
- $w = 0$ and $b = 0$
- while there is some i with $y^{(i)}(w \cdot \Phi(x^{(i)}) + b) \leq 0$:
  - $w = w + y^{(i)}\,\Phi(x^{(i)})$
  - $b = b + y^{(i)}$

Dual form: $w = \sum_j \alpha_j y^{(j)} \Phi(x^{(j)})$, where $\alpha \in \mathbb{R}^n$

- $\alpha = 0$ and $b = 0$
- while some i has $y^{(i)} \left(\sum_j \alpha_j y^{(j)} \Phi(x^{(j)}) \cdot \Phi(x^{(i)}) + b \right) \leq 0$:
  - $\alpha_i = \alpha_i + 1$
  - $b = b + y^{(i)}$

To classify a new point x: sign $\left(\sum_j \alpha_j y^{(j)} \Phi(x^{(j)}) \cdot \Phi(x) + b \right)$.

Kernel Perceptron: examples



Kernel Perceptron: examples