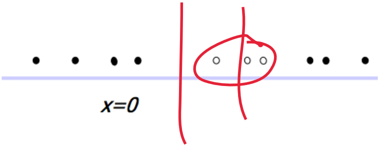
Machine Learning: Chenhao Tan

University of Colorado Boulder

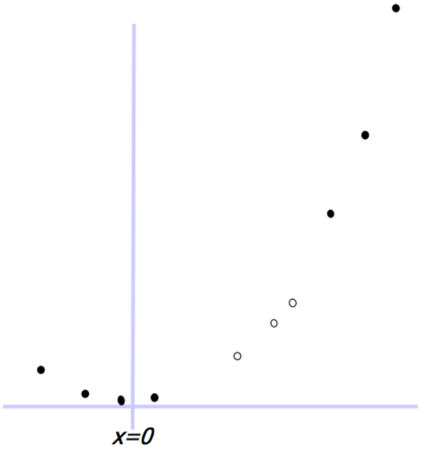
LECTURE 11

Slides adapted from Jordan Boyd-Graber

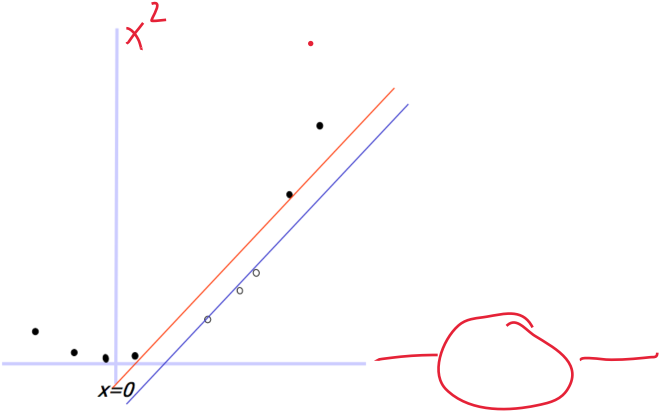
Can you solve this with linear separator?



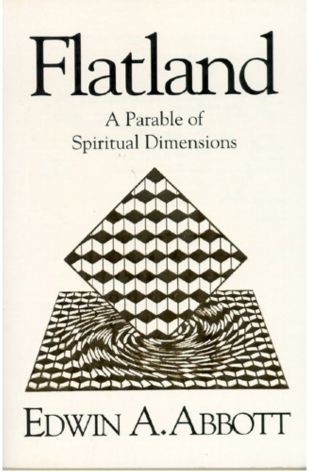
Can you solve this with linear separator?



Can you solve this with linear separator?



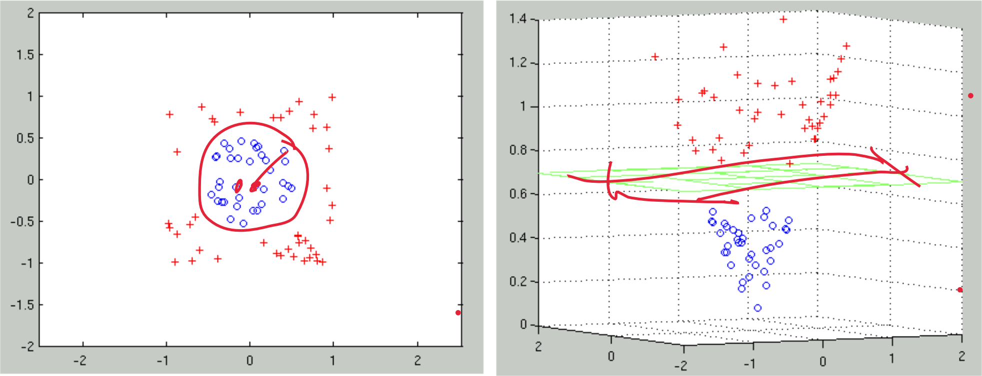
Adding another dimension



Behold yon miserable creature. That Point is a Being like ourselves, but confined to the non-dimensional Gulf. He is himself his own World, his own Universe; of any other than himself he can form no conception; he knows not Length, nor Breadth, nor Height, for he has had no experience of them; he has no cognizance even of the number Two; nor has he a thought of Plurality, for he is himself his One and All, being really Nothing. Yet mark his perfect self-contentment, and hence learn this lesson, that to be self-contented is to be vile and ignorant, and that to aspire is better than to be blindly and impotently happy.

Problems get easier in higher dimensions

$$(x_1, x_2) \Rightarrow (x_1, x_2, \sqrt{x_1^2 + x_2^2})$$



What's special about SVMs?

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j (\boldsymbol{x_i} \cdot \boldsymbol{x_j}) \tag{1}$$

What's special about SVMs?

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j (\boldsymbol{x_i} \cdot \boldsymbol{x_j}) \tag{1}$$

- This dot product is basically just how much x_i looks like x_j. Can we generalize that?

What's special about SVMs?

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j (\boldsymbol{x_i} \cdot \boldsymbol{x_j}) \tag{1}$$

- This dot product is basically just how much x_i looks like x_j. Can we generalize that?
- Kernels!

What's a kernel?

- A function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a kernel over $\mathcal{X}$.
- This is equivalent to taking the dot product $\langle \phi(x_1), \phi(x_2) \rangle$ for some mapping
- **Mercer's Theorem**: So long as the function is continuous and symmetric, then K admits an expansion of the form

$\phi(x_i) = (x_i, x_i^2)$

$$K(x, x') = \sum_{n=0}^{\infty} a_n \phi_n(x) \phi_n(x') \tag{2}$$

What's a kernel?

- A function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathrm{R}$ is a kernel over $\mathcal{X}$.
- This is equivalent to taking the dot product $\langle \phi(x_1), \phi(x_2) \rangle$ for some mapping
- **Mercer's Theorem**: So long as the function is continuous and symmetric, then K admits an expansion of the form

$$K(x, x') = \sum_{n=0}^{\infty} a_n \phi_n(x) \phi_n(x') \tag{2}$$

- The computational cost is just in computing the kernel

Kernel Matrix

The important property of the kernel matrix $\boldsymbol{K} = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{m \times m}$ is symmetric positive semidefinite.

$$K = \begin{pmatrix} K(x_1, x_1) & \cdots & K(x_1, x_m) \\ \vdots & & \vdots \\ K(x_m, x_1) & \cdots & K(x_m, x_m) \end{pmatrix}$$

Kernel Matrix

The important property of the kernel matrix $\boldsymbol{K} = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{m \times m}$ is symmetric positive semidefinite.

$$\boldsymbol{K}^T = \boldsymbol{K}$$

Kernel Matrix

The important property of the kernel matrix $\boldsymbol{K} = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{m \times m}$ is symmetric positive semidefinite.

$$\boldsymbol{K}^T = \boldsymbol{K}$$

$$\forall \boldsymbol{x}, \boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$$

Kernel Matrix

The important property of the kernel matrix $\boldsymbol{K} = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{m \times m}$ is symmetric positive semidefinite.

$$\boldsymbol{K}^T = \boldsymbol{K}$$

$$\forall \boldsymbol{x}, \boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$$

Also known as Gram matrix.

(x_i, x_i^2)

$c = 0 \quad d = 2$

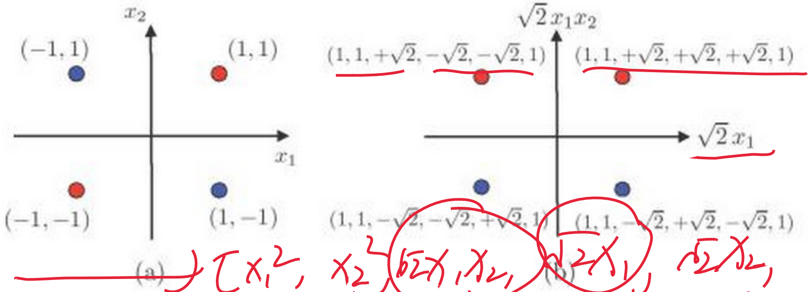
Polynomial Kernel

$$K(x, x') = (x \cdot x' + c)^d \tag{3}$$

Polynomial Kernel

$$K(x, x') = (x \cdot x' + c)^d \quad (3)$$

When $d = 2, c = 1$:



Gaussian Kernel

RBF

$$-\gamma \|x' - x\|^2$$

$$K(x, x') = \exp - \frac{\|x' - x\|^2}{2\sigma^2} \quad (4)$$

Gaussian Kernel

$$K(x, x') = \exp -\frac{\|x' - x\|^2}{2\sigma^2} \tag{4}$$

which can be rewritten as

$$K(x, x') = \sum_n \frac{(x \cdot x')^n}{\sigma^n n!} \tag{5}$$

(All polynomials!)

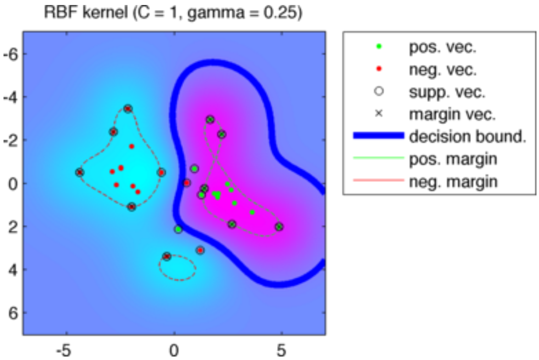
Gaussian Kernel

$$K(x,x') = \exp - \frac{\|x' - x\|^2}{2\sigma^2} \tag{4}$$

which can be rewritten as

$$K(x,x') = \sum_n \frac{(x \cdot x')^n}{\sigma^n n!} \tag{5}$$

(All polynomials!)

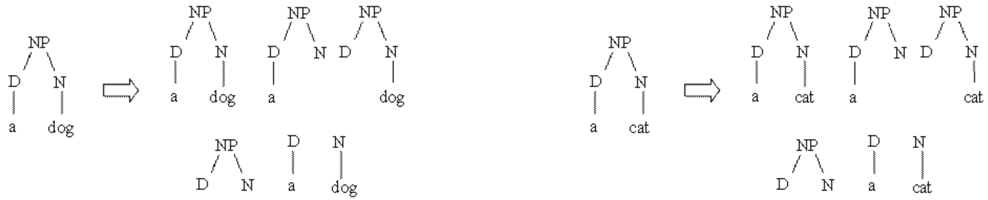


Tree Kernels

- Sometimes we have example x that are hard to express as vectors
- For example sentences “a dog” and “a cat”: internal syntax structure

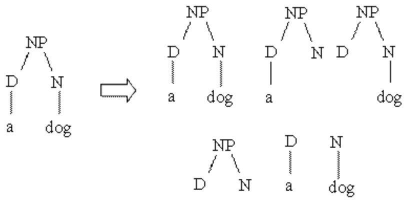
Tree Kernels

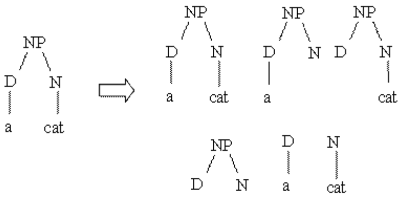
- Sometimes we have example x that are hard to express as vectors
- For example sentences “a dog” and “a cat”: internal syntax structure



Tree Kernels

- Sometimes we have example x that are hard to express as vectors
- For example sentences "a dog" and "a cat": internal syntax structure





$3/5$ structures match, so tree kernel returns $.6$

What does this do to learnability?

- Kernelized hypothesis spaces are obviously more complicated
- What does this do to complexity?

How does it affect optimization

- Replace all dot product with kernel evaluations $K(x_1, x_2)$
- Makes computation more expensive, overall structure is the same
- Try linear first!

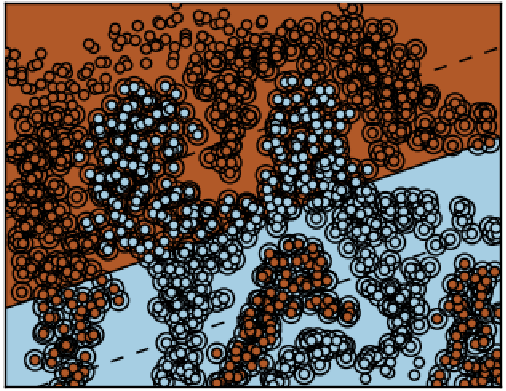
Outline

Examples

Kernelized SVM

```
X, Y = read_data("ex8a.txt")
clf = svm.SVC(kernel=kk, degree=dd, gamma=gg)
clf.fit(X, Y)
```

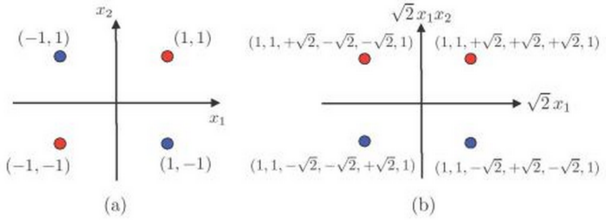
Linear Kernel Doesn't Work



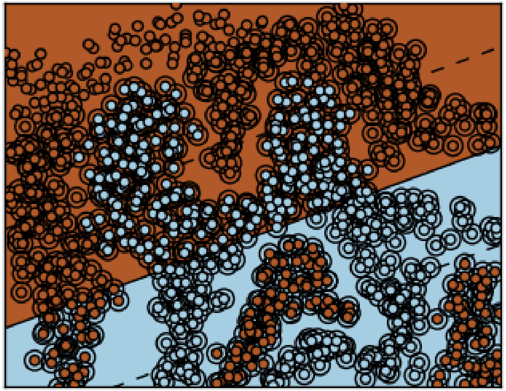
Polynomial Kernel

$$K(x, x') = (x \cdot x' + c)^d \tag{6}$$

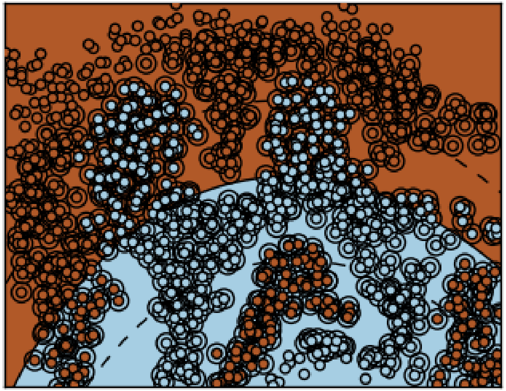
When $d = 2$:



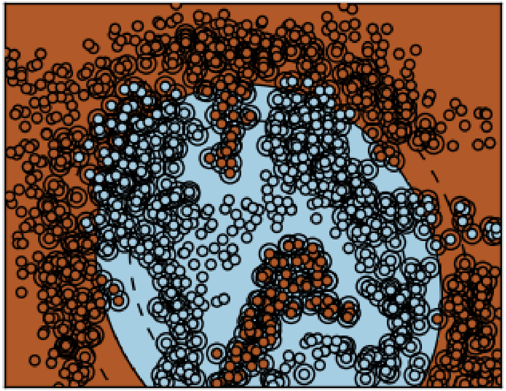
Polynomial Kernel $d = 1, c = 5$



Polynomial Kernel $d = 2, c = 5$

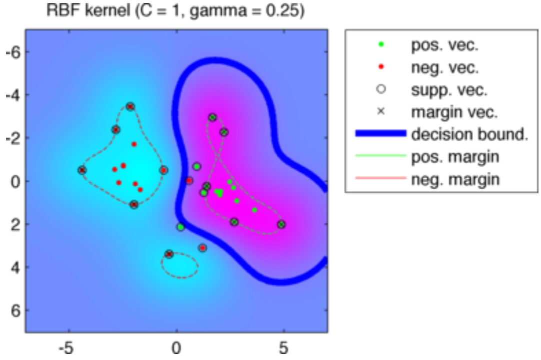


Polynomial Kernel $d = 3, c = 5$

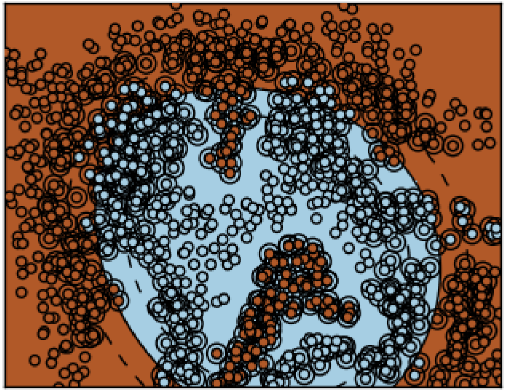


Gaussian Kernel

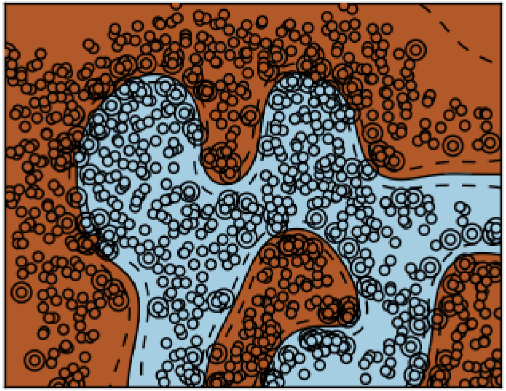
$$K(x, x') = \exp\left(\gamma \left\|x' - x\right\|^2\right) \tag{7}$$



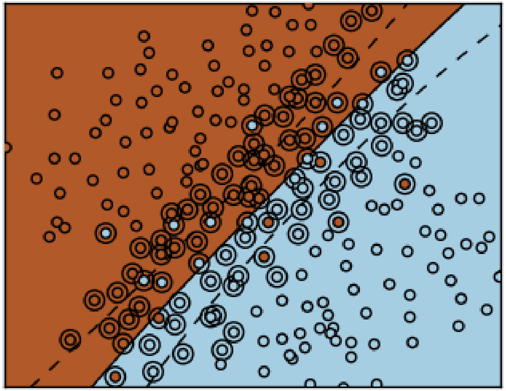
RBF Kernel $\gamma = 2$



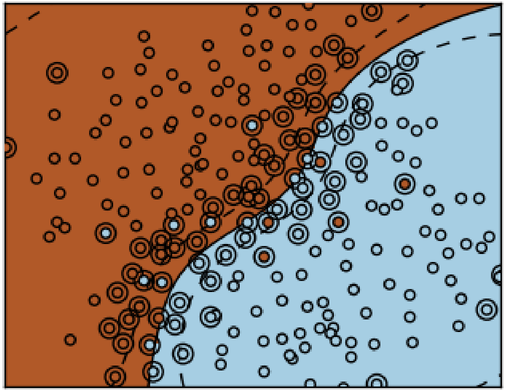
RBF Kernel $\gamma = 100$



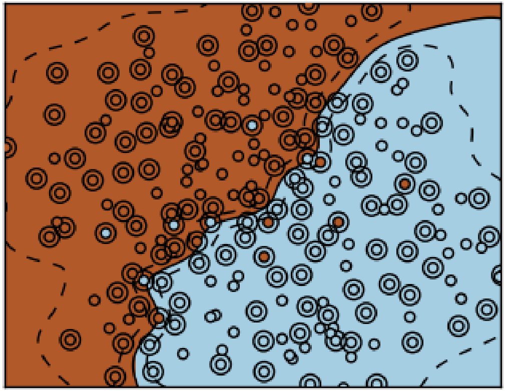
RBF Kernel $\gamma = 1$



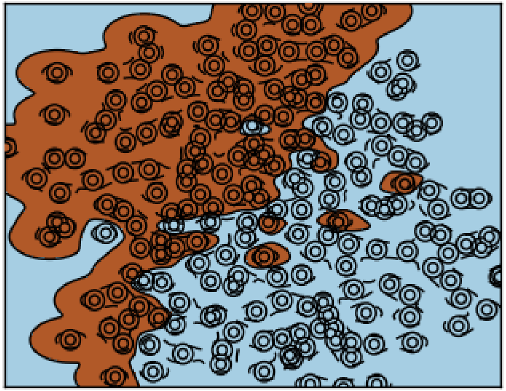
RBF Kernel $\gamma = 10$



RBF Kernel $\gamma = 100$



RBF Kernel $\gamma = 1000$



Be careful!

- Which has the lowest training error?
- Which one would generalize best?

Recap

- This completes our discussion of SVMs
- Workhorse method of machine learning
- Flexible, fast, effective

Recap

- This completes our discussion of SVMs
- Workhorse method of machine learning
- Flexible, fast, effective
- Kernels: applicable to wide range of data, inner product trick keeps method simple