01.112/50.007 Machine Learning

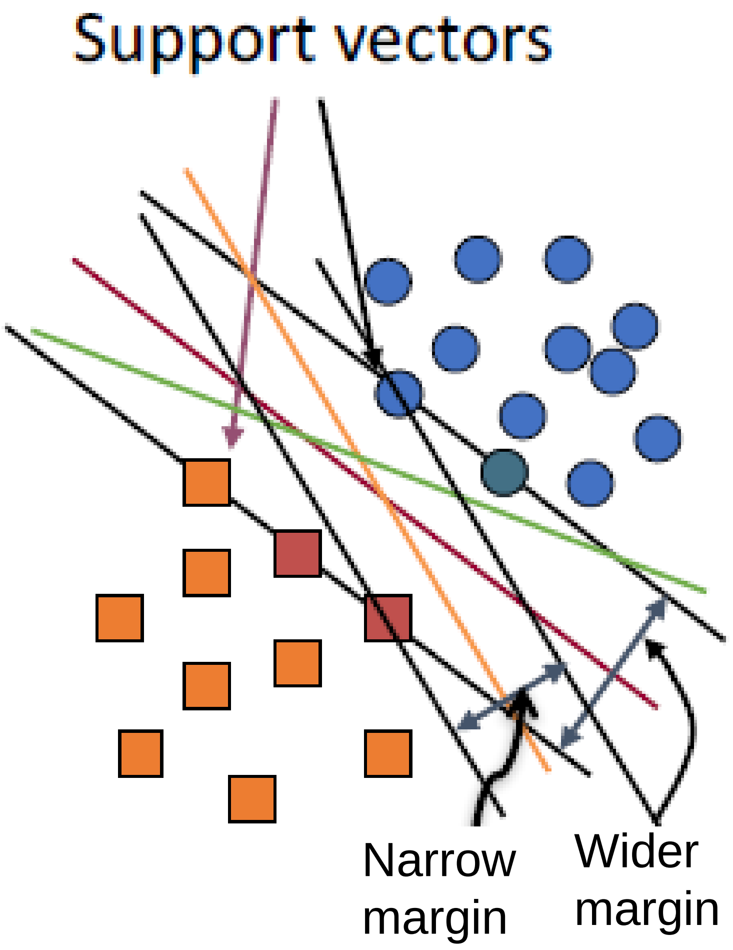
Lecture 9

Kernel Methods

Recap

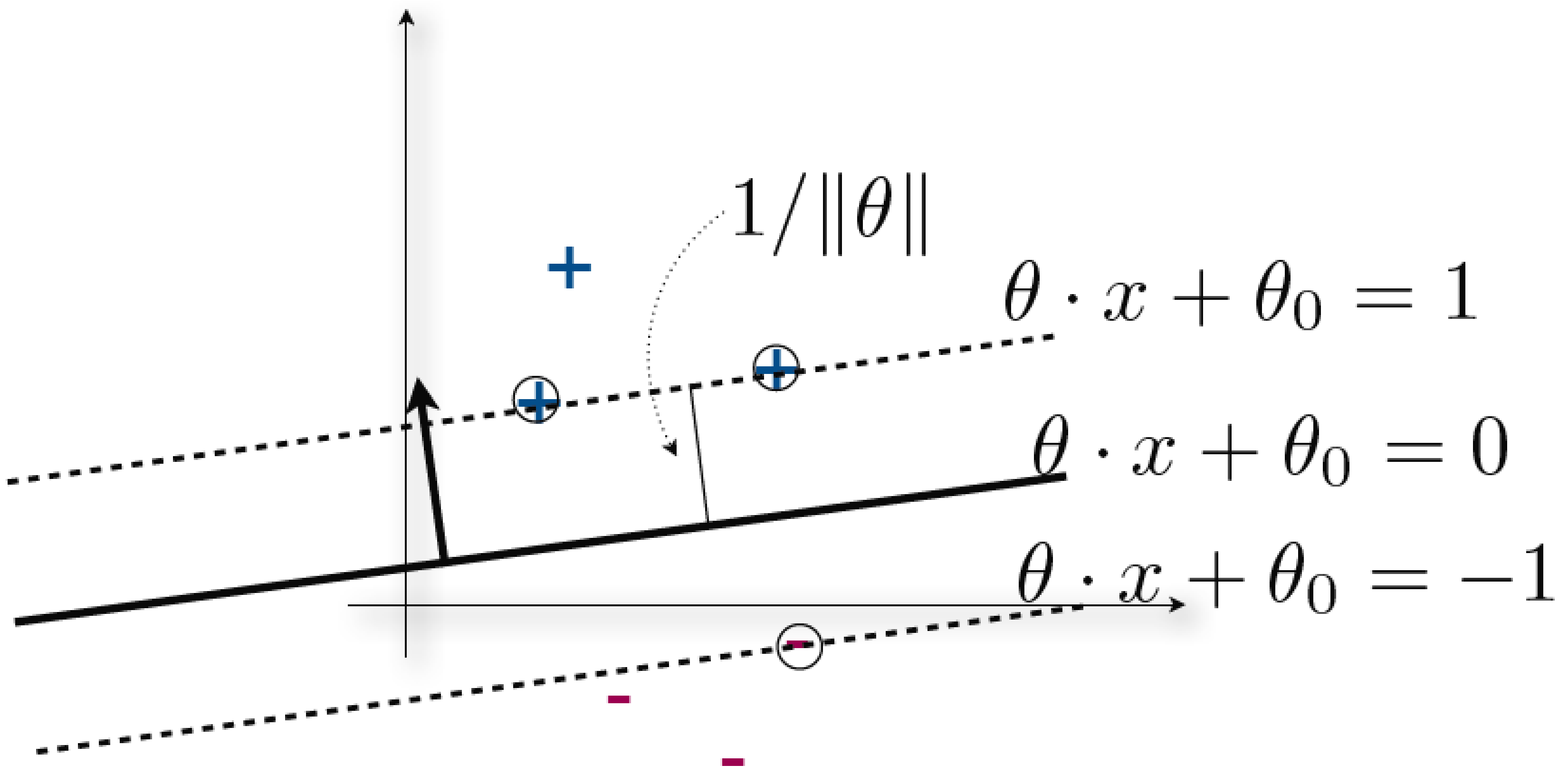
Support Vector Machine (SVM)

- SVMs **maximize the *margin*** around the separating hyperplane. A.k.a. **large margin classifiers.**
- The decision function is fully specified by a subset of training samples, ***the support vectors***.
- Solving SVMs is a ***quadratic programming* problem**.
- Seen by many as the most successful current text classification method*



*but other discriminative methods often perform very similarly

Computing the margin



Maximum Margin

Our goal is to

maximize $1/\|\theta\|$

subject to $y(\theta^\top x + \theta_0) \geq 1$ for all data (x, y)

Or equivalently,

minimize $\frac{1}{2}\|\theta\|^2$

subject to $y(\theta^\top x + \theta_0) \geq 1$ for all data (x, y)

Primal-Dual

Primal.

minimize $\frac{1}{2}\|\theta\|^2$

subject to $y(\theta^\top x + \theta_0) \geq 1$ for all data (x, y)

Dual.

maximize $\sum_{(x,y)} \alpha_{x,y} - \frac{1}{2}\sum_{(x,y)}\sum_{(x',y')} \alpha_{x,y}\,\alpha_{x',y'}\,yy'(x^\top x')$

subject to $\alpha_{x,y} \geq 0$ for all (x, y)

$\sum_{(x,y)} \alpha_{x,y}\, y = 0$

Support Vectors

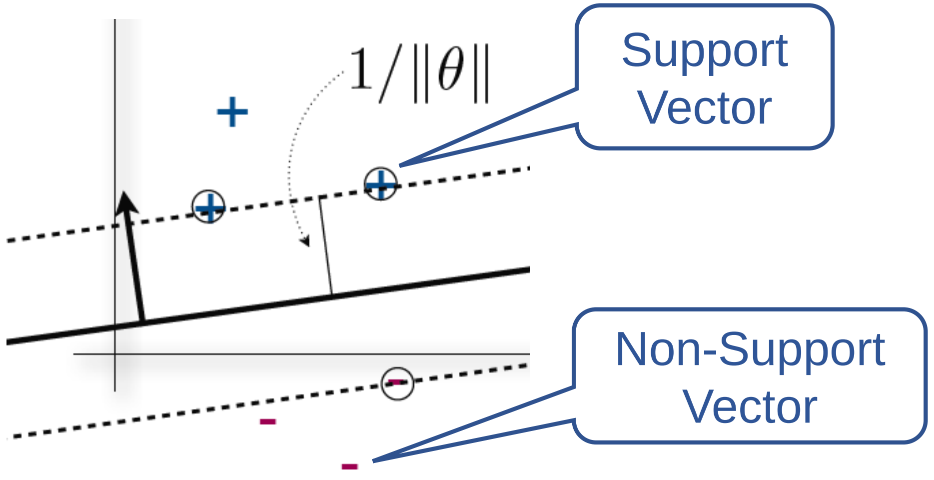
Complementary Slackness.

$\hat{\alpha}_{x,y} > 0$: $\quad y(\hat{\theta}^\top x + \theta_0) = 1 \quad$ Support Vectors

$\hat{\alpha}_{x,y} = 0$: $\quad y(\hat{\theta}^\top x + \theta_0) > 1 \quad$ Non-Support Vectors

Sparsity

Since very few data points are support vectors, most of the $\hat{\alpha}_{x,y}$ will be zero.



Inner Product

- Consider the dual objective function

$$\ell(\alpha) = \sum_{(x,y)} \alpha_{x,y} - \frac{1}{2} \sum_{(x,y)} \sum_{(x',y')} \alpha_{x,y}\, \alpha_{x',y'} y y' (x^\top x')$$

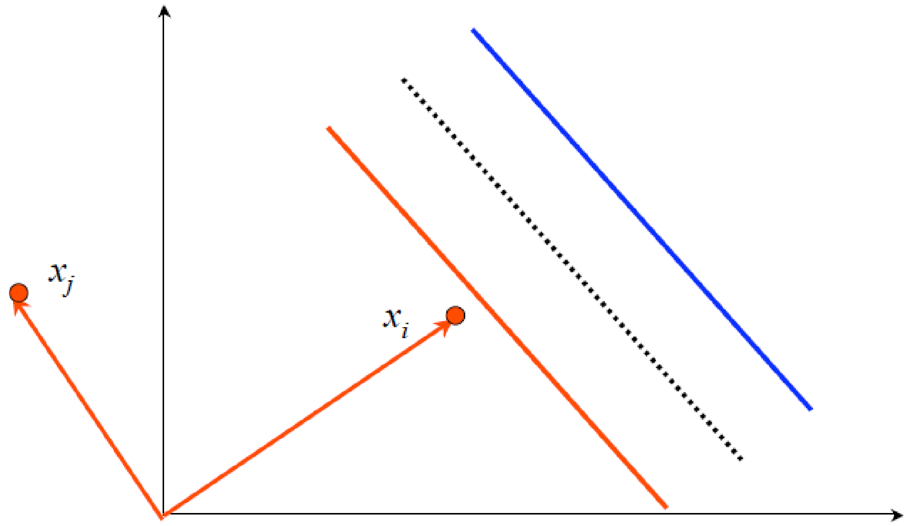
- **Claim:** This function is maximized only for support vectors from opposite sides of the margin.
  - **Case 1:** Inner product is 0 (dissimilar)
  - **Case 2:** Inner product is 1 (similar)
    - **Sub-case 1:** same class
    - **Sub-case 2:** different classes

Inner Product

- Consider the dual objective function

$$\ell(\alpha) = \sum_{(x,y)} \alpha_{x,y} - \frac{1}{2} \sum_{(x,y)} \sum_{(x',y')} \alpha_{x,y}\, \alpha_{x',y'} y y' (x^\top x')$$

- Inner product is 0 (dissimilar vectors)

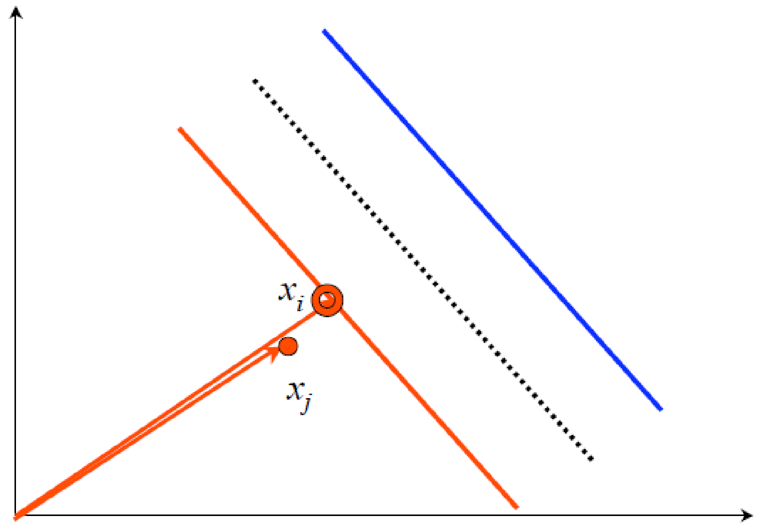


Inner Product

- Consider the dual objective function

$$\ell(\alpha) = \sum_{(x,y)} \alpha_{x,y} - \frac{1}{2} \sum_{(x,y)} \sum_{(x',y')} \alpha_{x,y}\, \alpha_{x',y'} y y' (x^\top x')$$

- Inner product is 1 (same class)

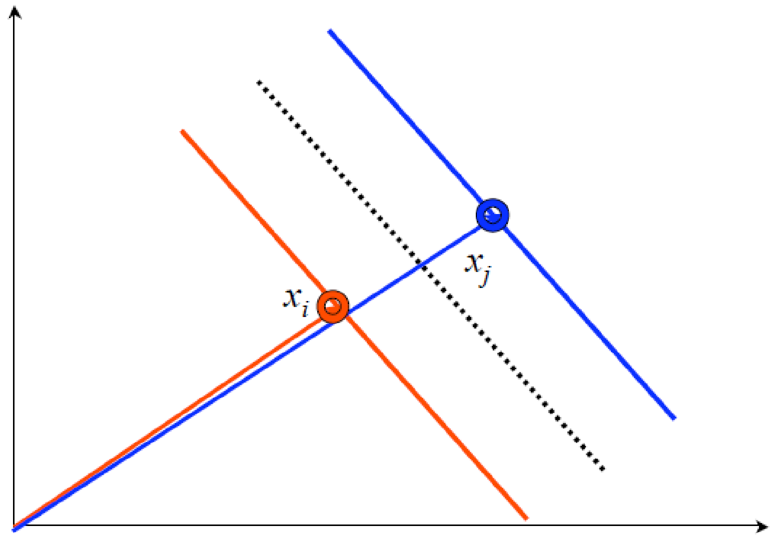


Inner Product

- Consider the dual objective function

$$\ell(\alpha) = \sum_{(x,y)} \alpha_{x,y} - \frac{1}{2} \sum_{(x,y)} \sum_{(x',y')} \alpha_{x,y}\, \alpha_{x',y'} y y' (x^\top x')$$

- Inner product is 1 (different class)



SVM

Learning.

$$\ell(\alpha) = \sum_{(x,y)} \alpha_{x,y} - \frac{1}{2} \sum_{(x,y)} \sum_{(x',y')} \alpha_{x,y}\, \alpha_{x',y'} y y' (x^\top x')$$

Inner product between training examples

Prediction.

$$h(x;\theta) = \text{sign}(\theta^\top x + \theta_0) = \text{sign}\left(\sum_{(x',y')} \alpha_{x',y'} y' (x^\top x') + \theta_0\right)$$

Inner product between training example and new test sample

For the dual, we don't need the feature vectors x, x'.
Knowing just the dot products $(x^\top x')$ is enough.

Recall that $(x^\top x')$ is a measure of similarity between x and x'.
This similarity function is also called a *kernel*.

Kernels

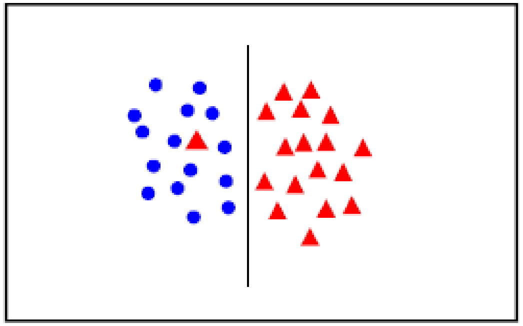
Non-Linear SVM

Primal.

minimize $\frac{\lambda}{2}\|\theta\|^2 + \frac{1}{n}\sum_{(x,y)} \xi_{x,y}$

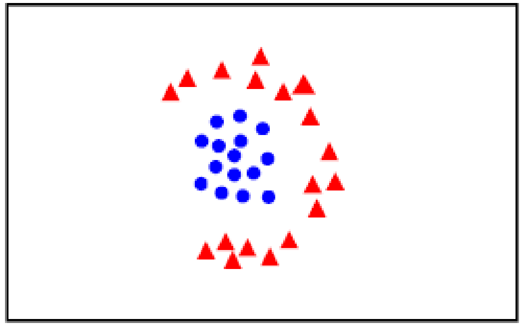
subject to $y(\theta^\top x + \theta_0) \geq 1 - \xi_{x,y}$ for all data (x, y)

$\xi_{x,y} \geq 0$ for all data (x, y)



Dual.

maximize $\sum_{(x,y)} \alpha_{x,y} - \frac{1}{2}\sum_{(x,y)} \sum_{(x',y')} \alpha_{x,y}\, \alpha_{x',y'} yy'(x^\top x')$



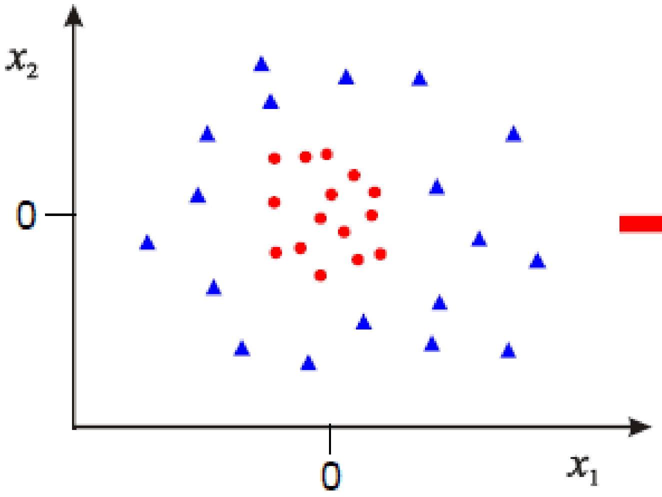
subject to $1/\lambda \geq \alpha_{x,y} \geq 0$ for all (x, y)

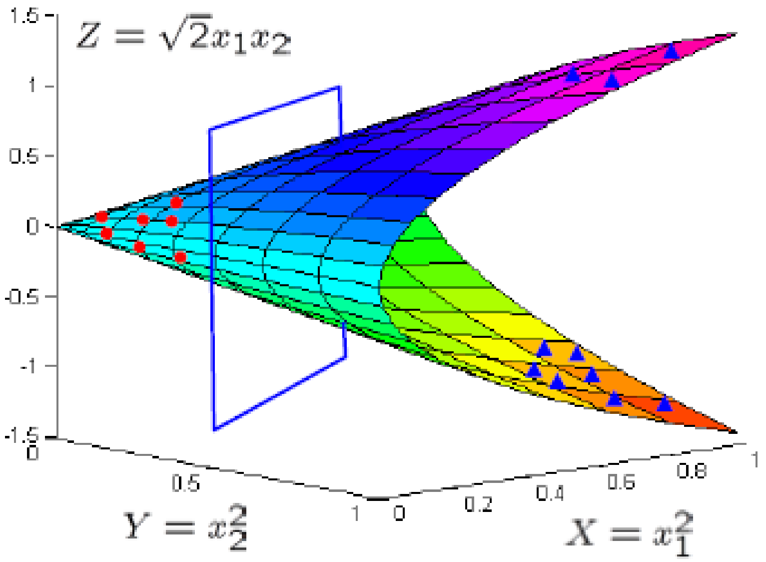
$\sum_{(x,y)} \alpha_{x,y}\, y = 0$

Non-Linear SVM

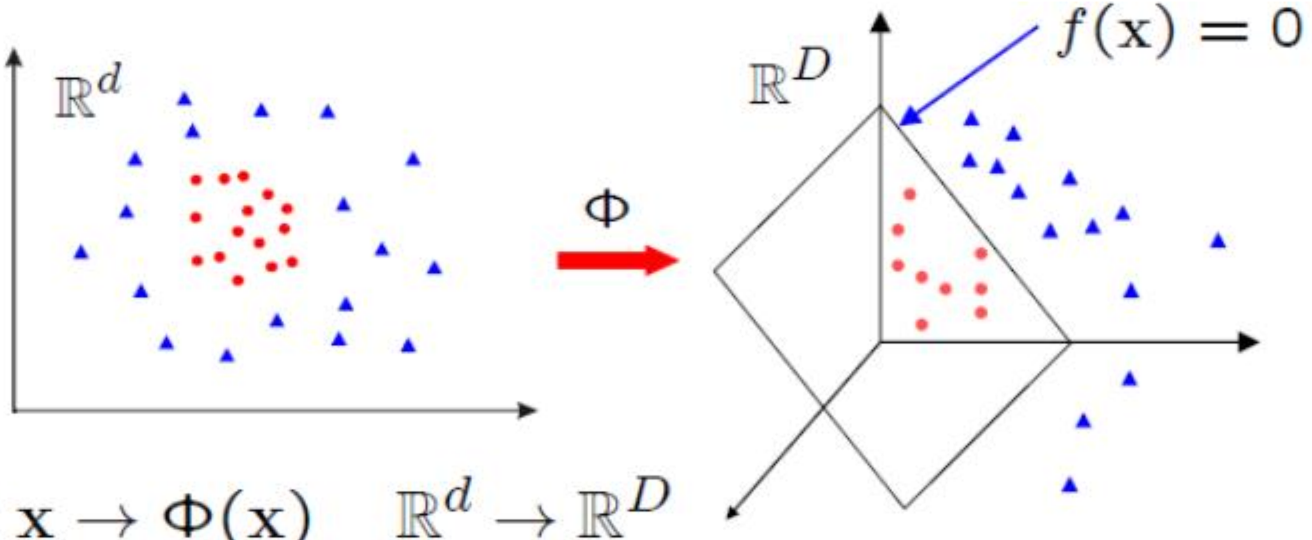
- Transform the input feature vectors to higher dimensions
- Data is now linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \to \mathbb{R}^3$$





SVM in transformed feature space



$\Phi : \mathbf{x} \rightarrow \Phi(\mathbf{x}) \quad \mathbb{R}^d \rightarrow \mathbb{R}^D$

Learn classifier linear in $\boldsymbol{\theta}$ for $\mathbb{R}^D$:

$$y = \text{sign}(\theta \cdot \phi(x) + \theta_0)$$

$\Phi(\mathbf{x})$ is a feature map

SVM in transformed feature space

Learning.

$$\ell(\alpha) = \sum_{(x,y)} \alpha_{x,y} - \frac{1}{2} \sum_{(x,y)} \sum_{(x',y')} \alpha_{x,y}\, \alpha_{x',y'} y y' (x^\top x')$$

Prediction.

$$h(x;\theta) = \mathrm{sign}(\theta^\top x + \theta_0) = \mathrm{sign}\Big(\sum_{(x',y')} \alpha_{x',y'} y' (x^\top x') + \theta_0\Big)$$

$$\Big\Downarrow\ \emptyset$$

Learning.

$$\ell(\alpha) = \sum_{(x,y)} \alpha_{x,y} - \frac{1}{2} \sum_{(x,y)} \sum_{(x',y')} \alpha_{x,y}\, \alpha_{x',y'} y y' (\emptyset(x).\emptyset(x'))$$

Prediction.

$$h(x;\theta) = \mathrm{sign}(\theta^\top x + \theta_0) = \mathrm{sign}\Big(\sum_{(x',y')} \alpha_{x',y'} y' (\emptyset(x).\emptyset(x')) + \theta_0\Big)$$

Kernel Methods

- Inner product is easier to obtain, than transforming each feature to higher dimension space.

$$\mathrm{K}(\mathrm{x}, \mathrm{x}') = \emptyset(x).\emptyset(x')$$

- Kernel methods can be used for any linear prediction methods to deal with non-linear data.

$$\begin{aligned} \text{Perceptron :} \quad & y = \operatorname{sign}(\theta \cdot \phi(x) + \theta_0) \\ \text{SVM :} \quad & y = \operatorname{sign}(\theta \cdot \phi(x) + \theta_0) \\ \text{Linear Regression :} \quad & y = \theta \cdot \phi(x) + \theta_0 \end{aligned}$$

Inner product of polynomials

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad \Phi(\mathbf{x}) = \text{polynomials of degree exactly } m$$

m=1 $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = x_1 z_1 + x_2 z_2 = \mathbf{x} \cdot \mathbf{z}$

m=2 $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = \begin{bmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} z_1^2 \\ \sqrt{2} z_1 z_2 \\ z_2^2 \end{bmatrix} = x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2$

$$= (x_1 z_1 + x_2 z_2)^2$$

$$= (\mathbf{x} \cdot \mathbf{z})^2$$

m $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^m = K(\mathbf{x}, \mathbf{z})$

Don’t store high-dimensional features – Only evaluate inner product with kernels

Kernel Methods

- Properties of kernel functions

1. $K(x, x') = 1$ is a kernel function.
2. Let $f : \mathcal{R}^d \to \mathcal{R}$ be any real valued function of x. Then, if $K(x, x')$ is a kernel function, then so is $\tilde{K}(x, x') = f(x)K(x, x')f(x')$
3. If $K_1(x, x')$ and $K_2(x, x')$ are kernels, then so is their sum. In other words, $K(x, x') = K_1(x, x') + K_2(x, x')$ is a kernel.
4. If $K_1(x, x')$ and $K_2(x, x')$ are kernels, then so is their product $K(x, x') = K_1(x, x')K_2(x, x')$

Common Kernels

- Polynomials of degree exactly *d*

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$$

- Polynomials of degree up to *d*

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^d$$

- Gaussian kernels

$$K(\vec{u},\vec{v}) = \exp\left(-\frac{||\vec{u}-\vec{v}||_2^2}{2\sigma^2}\right)$$

Euclidean distance, squared

- And many others: very active area of research!
(e.g., structured kernels that use dynamic programming
to evaluate, string kernels, …)

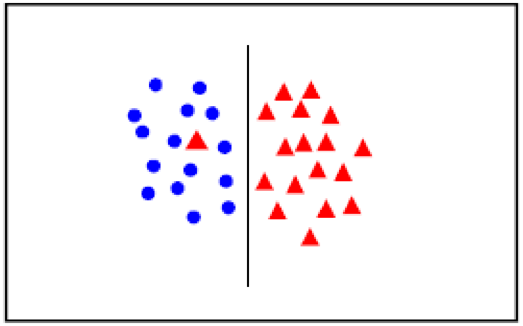
Non-Linear SVM

Primal.

minimize $\frac{\lambda}{2}||\theta||^2 + \frac{1}{n}\sum_{(x,y)} \xi_{x,y}$

subject to $y(\theta^\top x + \theta_0) \geq 1 - \xi_{x,y}$ for all data (x, y)

$\xi_{x,y} \geq 0$ for all data (x, y)

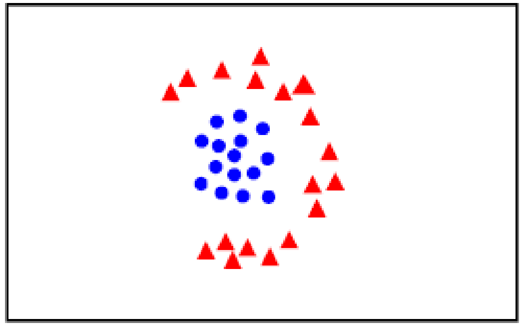


Dual.

maximize $\sum_{(x,y)} \alpha_{x,y} - \frac{1}{2}\sum_{(x,y)} \sum_{(x',y')} \alpha_{x,y}\, \alpha_{x',y'} yy'(\emptyset(x).\emptyset(x'))$

subject to $1/\lambda \geq \alpha_{x,y} \geq 0$ for all (x, y)

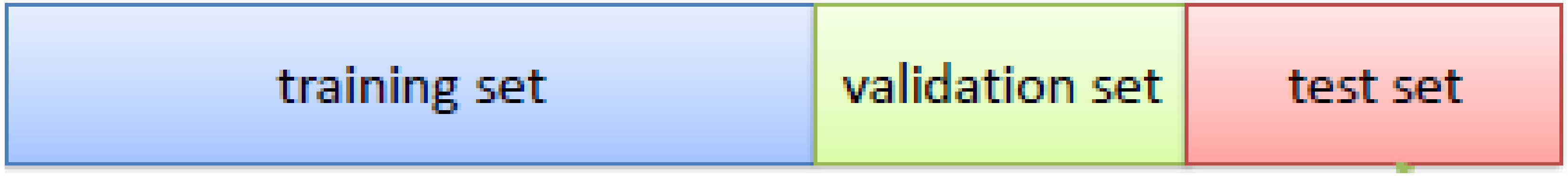
$\sum_{(x,y)} \alpha_{x,y}\, y = 0$



Cross Validation

General Strategy

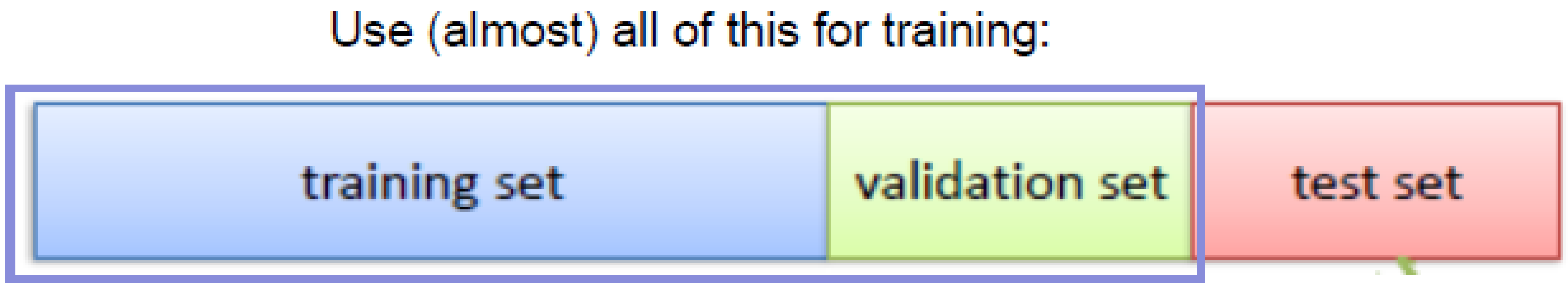
- Split data up into three parts



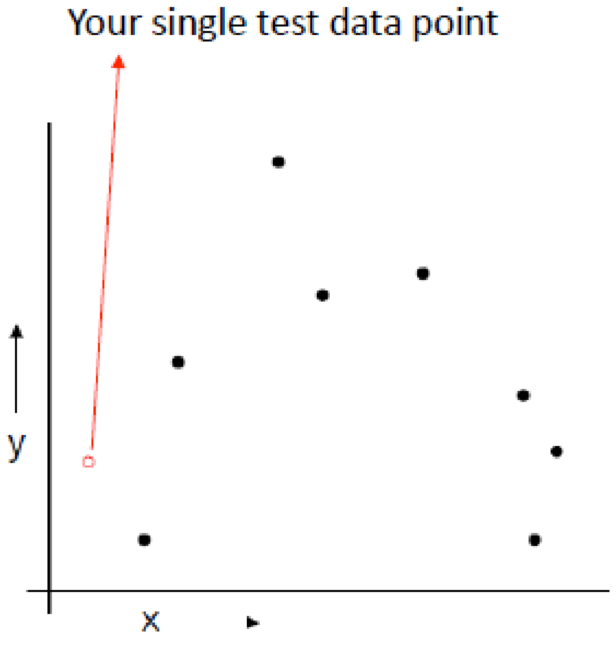
- Assume that the data is randomly allocated to these three, e.g. 60/20/20

Cross Validation

- Recycle the data



Leave one out Cross Validation



Lets say we have N data points
k indices the data points, i.e. k=1...N

Let (x_k, y_k) be the k^{th} example

Temporarily remove (x_k, y_k) from the dataset

Train on the remaining N-1 data points

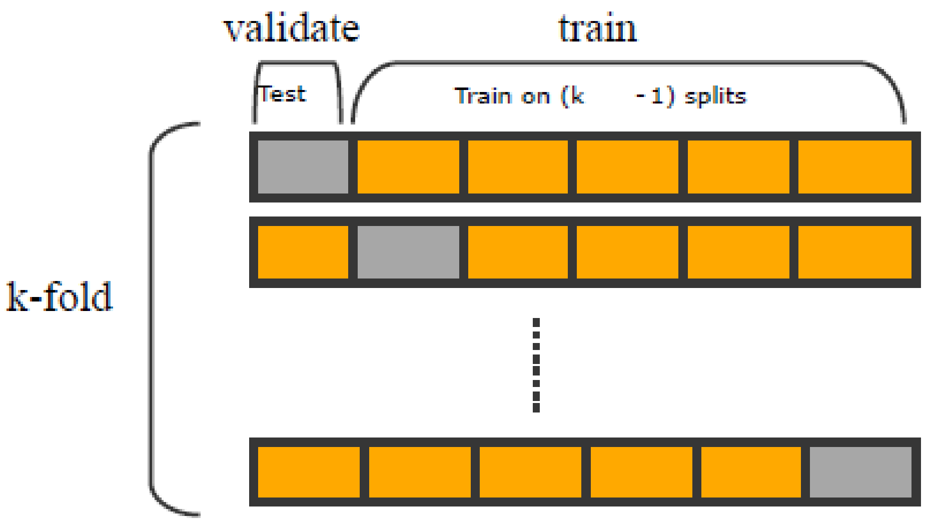
Test your error on (x_k, y_k)

Do this for each k=1..N and report the average error

Once the best parameters (e.g., choice of C for the SVM) are found, re-train using all of the training data

K-fold cross validation

- E.g. In 3 fold cv, there are 3 runs
- The error is averaged over all runs



Summary

- Classifiers can be learnt for high dimensional features spaces, without actually having to map the points into the high dimensional space.

- Data may be linearly separable in the high dimensional space, but not linearly separable in the original feature space.

- Kernels can be used for an SVM because of the scalar product in the dual form, but can also be used elsewhere – they are not tied to the SVM formalism.

Intended Learning Outcomes

- What is a kernel function and how it is used in the context of SVM.
- What are the essential properties associated with a kernel function.
- How to identify whether a function is a valid (or an invalid) kernel function.
- What is the definition of a RBF kernel.