Machine Learning 10-701

Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

April 7, 2011

Today: Kernel methods, SVM

- Regression: Primal and dual forms
- Kernels for regression
- Support Vector Machines

Readings:

Required:
Kernels: Bishop Ch. 6.1
SVMs: Bishop Ch. 7, through 7.1.2

Optional:
Bishop Ch 6.2, 6.3

Thanks to Aarti Singh, Eric Xing, John Shawe-Taylor for several slides

Kernel Functions

- Kernel functions provide a way to manipulate data as though it were projected into a higher dimensional space, by operating on it in its original space
- This leads to efficient algorithms
- And is a key component of algorithms such as
  - Support Vector Machines
  - kernel PCA
  - kernel CCA
  - kernel regression
  - …

Linear Regression

Wish to learn f: X → Y, where X=<X_1, … X_n>, Y real-valued

Learn $\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} x_i w_i = \langle \mathbf{x}, \mathbf{w} \rangle = \mathbf{x}^{\mathbf{T}}\mathbf{w}$

bold → column vector or matrix

dot product

transpose

$[x_1\ x_2 \cdots x_n]$ $\quad \vec{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$

where $\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{l=1}^{M} (y^l - \mathbf{x}^{\mathbf{T}l}\mathbf{w})^2 + \lambda \sum_{k}^{N} w_k^2$

l th train example

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}}_{Y} - \underbrace{\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ & X & & \\ x_1^M & x_2^M & \cdots & x_N^M \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}}_{w}$$

$$\|Y - Xw\|^2$$

Linear Regression

Wish to learn $f : X \to Y$, where $X = \langle X_1, X_2, \dots X_N \rangle$, $Y \in \Re$

Learn $\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} x_i w_i = \langle \mathbf{x}, \mathbf{w} \rangle = \mathbf{x}^{\mathbf{T}}\mathbf{w}$

where $\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{l=1}^{M} (y^l - \mathbf{x}^{\mathbf{T}l}\mathbf{w})^2 + \lambda \sum_{k=1}^{N} w_k^2$

$$\mathbf{w} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2$$

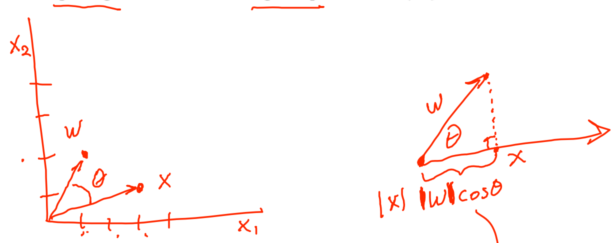
here the l^{th} row of $\mathbf{X}$ is the l^{th} training example $\mathbf{x}^{\mathbf{T}l}$

and $\quad \|\mathbf{w}\|^2 = \sum_{k=1}^{N} w_k^2 = \|\mathbf{w}\|_2^2$

Vectors, Data Points, Inner Products

Consider $\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} x_i w_i = \langle \mathbf{x}, \mathbf{w} \rangle = \mathbf{x}^{\mathbf{T}}\mathbf{w}$

where $\mathbf{x}^T = [3\ 1]$ and $\mathbf{w}^T = [1\ 2]$, so $\hat{f}(\mathbf{x}) =$



for any two vectors, their dot product (aka inner product) is equal to product of their lengths, times the cosine of angle between them

length

$$\langle \mathbf{x}, \mathbf{w} \rangle = \mathbf{x}^{\mathbf{T}}\mathbf{w} \equiv \boxed{\sum_{i=1}^{N} x_i w_i} = \|\mathbf{x}\| \|\mathbf{w}\| \cos(\theta)$$

Linear Regression: Primal Form

Learn $\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} x_i w_i = \langle \mathbf{x}, \mathbf{w} \rangle = \mathbf{x}^{\mathbf{T}}\mathbf{w}$

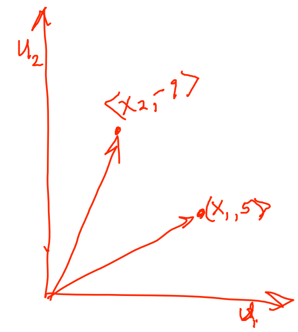
where $\mathbf{w} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2$

solve by taking derivative wrt $\mathbf{w}$, setting to zero…

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I})^{-1}\ \mathbf{X}^T\mathbf{y}$$

so: $\hat{f}(\mathbf{x}_{\mathbf{new}}) = \mathbf{x}_{\mathbf{new}}^{\mathbf{T}}\mathbf{w} = \mathbf{x}_{\mathbf{new}}^{\mathbf{T}}\ (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I})^{-1}\ \mathbf{X}^T\mathbf{y}$

Aha!



Learn $\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} x_i w_i = \langle \mathbf{x}, \mathbf{w} \rangle = \mathbf{x}^{\mathbf{T}}\mathbf{w}$

where $\mathbf{w} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2$

solution: $\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\ \mathbf{X}^T\mathbf{y}$

But notice w lies in the space spanned by training examples (why?)

Linear Regression: Dual Form

Primal form:

Learn $\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} x_i w_i = \langle \mathbf{x}, \mathbf{w} \rangle = \mathbf{x}^{\mathbf{T}}\mathbf{w}$

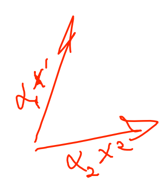
$\mathbf{w} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2$

Solution: $\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\ \mathbf{X}^T\mathbf{y}$

dim of x by # dim of x

Dual form: use fact that $\mathbf{w} = \sum_{l=1}^{M} \alpha_l\ \mathbf{x}^l$

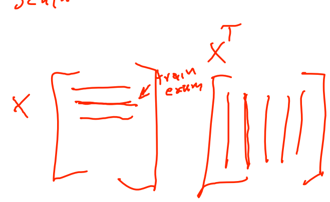
scalar



Learn $\hat{f}(\mathbf{x}) = \sum_{l=1}^{M} \alpha_l \langle \mathbf{x}, \mathbf{x}^l \rangle$

Solution: $\alpha = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\ \mathbf{y}$

train examps x # train examps



A dual solution expresses the weight vector w as a linear combination of the training examples:

$$\mathbf{X}'\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}'\mathbf{y} \quad \text{implies}$$

$$\mathbf{w} = \frac{1}{\lambda}(\mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{X}\mathbf{w}) = \boxed{\mathbf{X}'\frac{1}{\lambda}(\mathbf{y} - \mathbf{X}\mathbf{w})} = \mathbf{X}'\alpha,$$

where

$$\alpha = \frac{1}{\lambda}(\mathbf{y} - \mathbf{X}\mathbf{w}) \qquad (1)$$

or equivalently

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i$$

The vector α is the dual solution.

[slide from John Shawe-Taylor]

Substituting $\mathbf{w} = \mathbf{X}'\alpha$ into equation (1) we obtain:

$$\lambda\alpha = \mathbf{y} - \mathbf{X}\mathbf{X}'\alpha$$

implying

$$(\mathbf{X}\mathbf{X}' + \lambda\mathbf{I}_m)\,\alpha = \mathbf{y}$$

This means the dual solution can be computed as:

$$\alpha = (\mathbf{X}\mathbf{X}' + \lambda\mathbf{I}_m)^{-1}\,\mathbf{y}$$

with the regression function

$$g(\mathbf{x}) = \mathbf{x}'\mathbf{w} = \mathbf{x}'\mathbf{X}'\alpha = \left\langle \mathbf{x}, \sum_{i=1}^{m} \alpha_i \mathbf{x}_i \right\rangle = \sum_{i=1}^{m} \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle$$

[slide from John Shawe-Taylor]

Key ingredients of dual solution

train examps
×
train examps

Step 1: Compute

$$\alpha = (\mathbf{K} + \lambda \mathbf{I}_m)^{-1} \mathbf{y}$$

where $\mathbf{K} = \mathbf{X}\mathbf{X}'$ that is $\mathbf{K}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$

Step 2: Evaluate on new point $\mathbf{x}$ by

$$g(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle$$

Important observation: Both steps only involve inner products between input data points

[slide from John Shawe-Taylor]

Applying the 'kernel trick'

Since the computation only involves inner products, we can substitute for all occurrences of $\langle \cdot, \cdot \rangle$ a kernel function κ that computes:

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

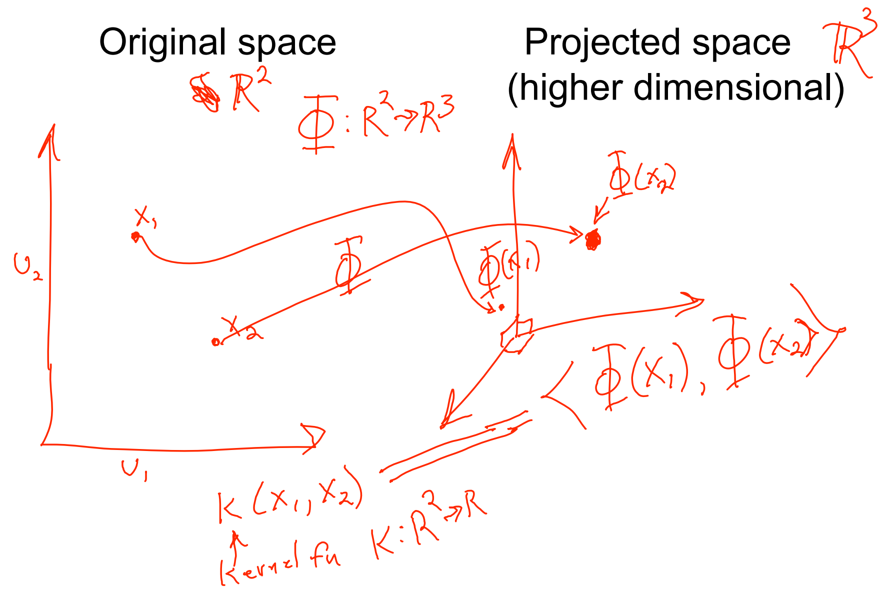
and we obtain an algorithm for ridge regression in the feature space F defined by the mapping

$$\phi : \mathbf{x} \longmapsto \phi(\mathbf{x}) \in F$$

Note if ϕ is the identity this has no effect.

[slide from John Shawe-Taylor]

Kernel functions



Example: Quadratic Kernel

Suppose we have data originally in 2D, but project it into 3D using $\Phi(\mathbf{x})$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \Phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \end{bmatrix} \quad \text{lin regress} \quad \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

this converts our original linear regression into quadratic regression!

But we can use the following kernel function to calculate inner products in the projected 3D space, in terms of operations in the 2D space

$$\langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle = \underline{\langle \mathbf{x_i}, \mathbf{x_j} \rangle}^2 \equiv \underline{\kappa_\Phi}(\mathbf{x_i}, \mathbf{x_j})$$

And use it to train and apply our regression function, never leaving 2D space

$$\hat{f}(\mathbf{x}) = \sum_{l=1}^{M} \alpha_l \; \kappa(\mathbf{x}, \mathbf{x}^l) \qquad \alpha = (\mathbf{K} + \lambda \mathbf{I})^{-1} \; \mathbf{y} \qquad \mathbf{K}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

Implications of the "Kernel Trick"

- Consider for example computing a regression function over 1000 images represented by pixel vectors – say $32 \times 32 = 1024$ pixels.
- By using the quadratic kernel we implement the regression function in a $1,000,000$ dimensional space
- but actually using less computation for the learning phase than we did in the original space – inverting a 1000×1000 matrix instead of a 1024×1024 matrix.

[slide from John Shawe-Taylor]

Some Common Kernels

- Polynomials of degree d

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

Kernel fn.

- Polynomials of degree up to d

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

- Gaussian/Radial kernels (polynomials of all orders – projected space has infinite dimension)

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

- Sigmoid

$$K(\mathbf{u}, \mathbf{v}) = \tanh(\eta \mathbf{u} \cdot \mathbf{v} + \nu)$$

Which Functions Can Be Kernels?

- not all functions
- for some definitions of $k(x_1,x_2)$ there is no corresponding projection $\phi(x)$

- Nice 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, …

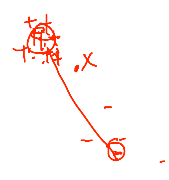
- Some of this covered in 10-702

Kernels : Key Points

- Many learning tasks are framed as optimization problems
- Primal and Dual formulations of optimization problems
- Dual version framed in terms of dot products between x's
- Kernel functions $k(x,y)$ allow calculating dot products $<\Phi(x),\Phi(y)>$ without bothering to project x into $\Phi(x)$
- Leads to major efficiencies, and ability to use very high dimensional (virtual) feature spaces

Kernel Based Classifiers

Simple Kernel Based Classifier



- Consider finding the centres of mass of positive and negative examples and classifying a test point by measuring which is closest

$$h(\mathbf{x}) = \operatorname{sgn}\left(\|\phi(\mathbf{x}) - \phi_{S_-}\|^2 - \|\phi(\mathbf{x}) - \phi_{S_+}\|^2\right)$$

- we can express as a function of kernel evaluations

\# + example

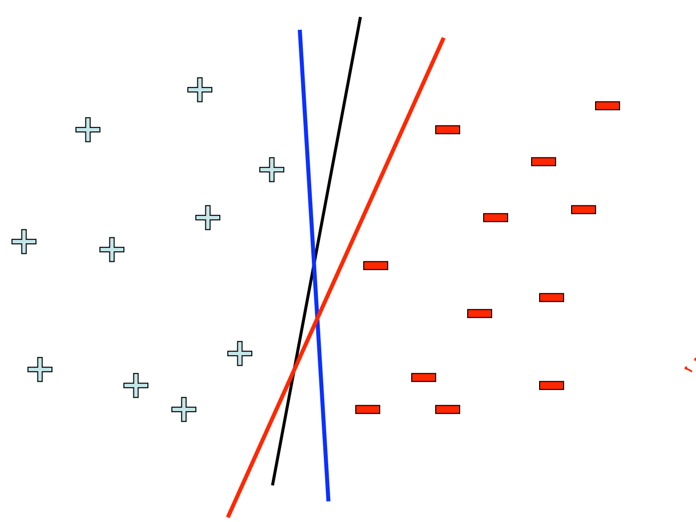
$$h(\mathbf{x}) = \operatorname{sgn}\left(\frac{1}{m_+}\sum_{i=1}^{m_+}\kappa(\mathbf{x},\mathbf{x}_i) - \frac{1}{m_-}\sum_{i=m_++1}^{m}\kappa(\mathbf{x},\mathbf{x}_i) - b\right),$$

where

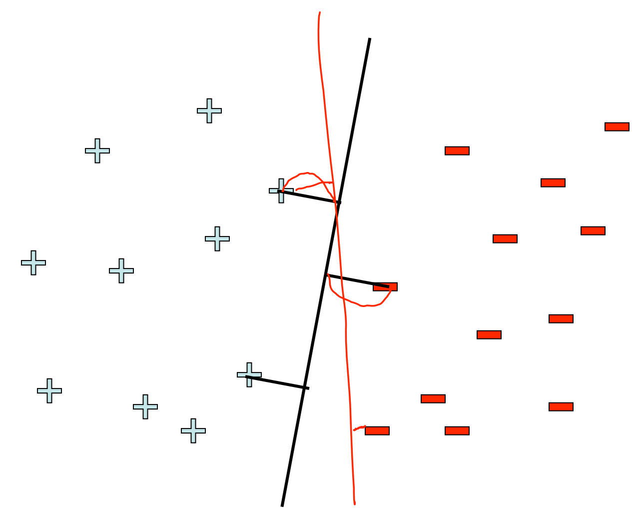
$$b = \frac{1}{2m_+^2}\sum_{i,j=1}^{m_+}\kappa(\mathbf{x}_i,\mathbf{x}_j) - \frac{1}{2m_-^2}\sum_{i,j=m_++1}^{m}\kappa(\mathbf{x}_i,\mathbf{x}_j)$$

[slide from John Shawe-Taylor]

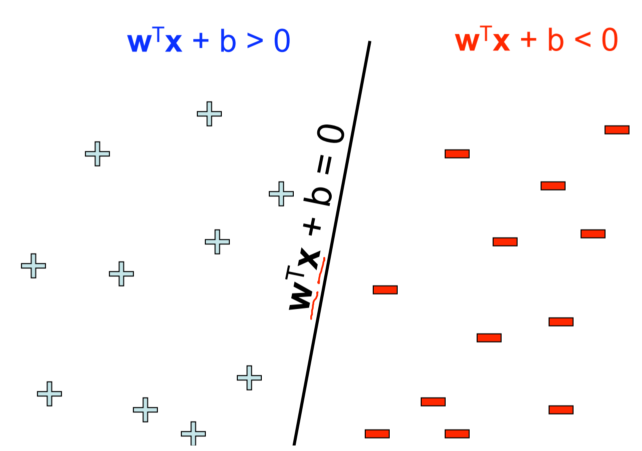
Linear classifiers – which line is better?



Pick the one with the largest margin!

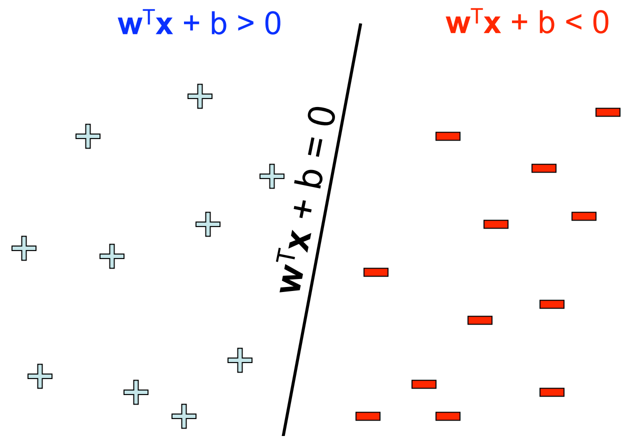


Parameterizing the decision boundary



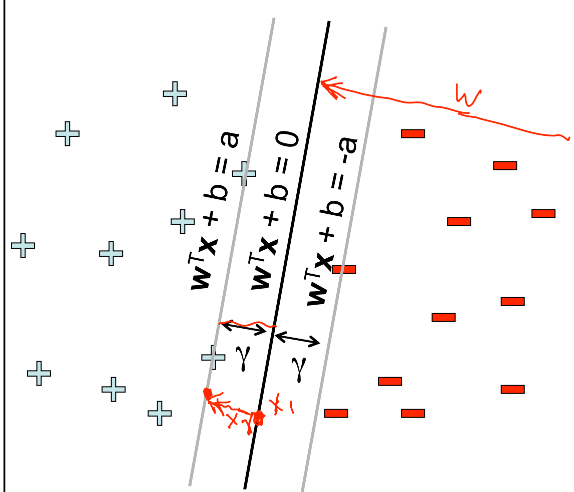
Labels $y_i \in \{-1, +1\}$ — class

Parameterizing the decision boundary



Labels $y_i \in \{-1, +1\}$ — class

Maximizing the margin



Margin = Distance of closest examples from the decision line/ hyperplane

$\boxed{\text{margin} = \gamma = a/\|w\|}$

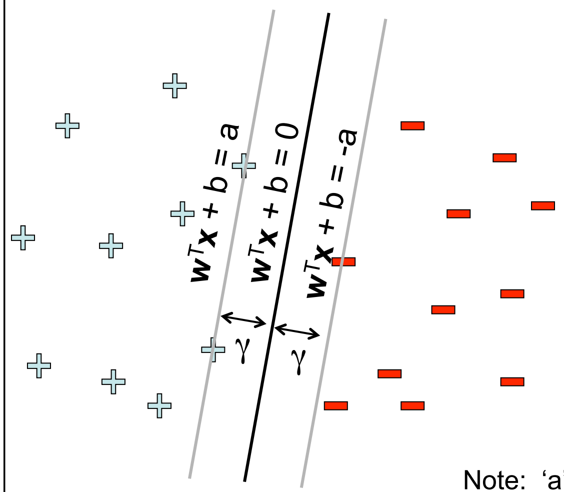
$$w^T(x_1 + x_\gamma) + b = a$$

$$\underbrace{w^T x_1 + b}_{0} + \underbrace{w^T x_\gamma = a}$$

$$\|w\| \, \underbrace{\|x_\gamma\|} = a$$

$$\|w\| \, \gamma = a$$

Maximizing the margin



Margin = Distance of closest examples from the decision line/ hyperplane

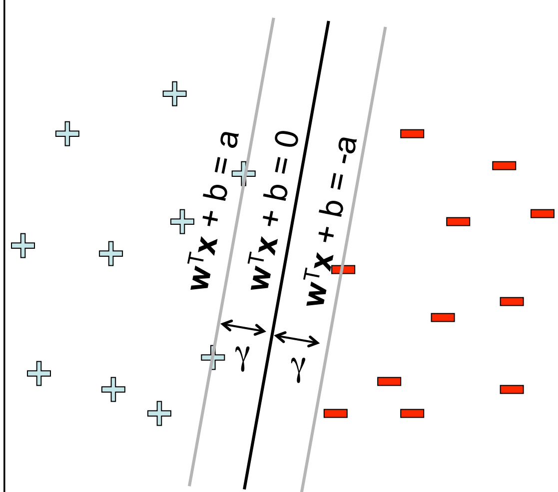
$\boxed{\text{margin} = \gamma = a/\|w\|}$

$$\max_{\mathbf{w},b} \ \gamma = a/\|w\|$$

$$\text{s.t. } (\mathbf{w}^T\mathbf{x}_j + b)\, y_j \geq a \ \ \forall j$$

Note: 'a' is arbitrary (can normalize equations by a)

Support Vector Machine



$$\min_{\mathbf{w},b} \ \mathbf{w}^T\mathbf{w}$$

$$\text{s.t. } (\mathbf{w}^T\mathbf{x}_j + b)\, y_j \geq 1 \ \ \forall j$$

Solve efficiently by quadratic programming (QP)

- Well-studied solution algorithms

Linear hyperplane defined by "support vectors"