Machine Learning 10-601

Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

November 8, 2017

Today: Kernel methods, SVM

- Support Vector Machines
- Margin-based classifiers

Readings:

Recommended:
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, Maria Balcan, John Shawe-Taylor for several slides

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

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

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

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

Key idea: Dual form expresses linear regression f(x) in terms of dot products with training examples.

→ more efficient computation when features per example > # examps

→ turns training and application of learned model into dot products, which allows using kernel functions

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 $\langle \Phi(x), \Phi(y) \rangle$ without bothering to project x into $\Phi(x)$
- Leads to major efficiencies, and ability to use very high dimensional (virtual) feature spaces

Example: Quadratic Kernel

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

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \Phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

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

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

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

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

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

Some Common Mercer Kernels

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

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

- String kernels: **u** and **v** are strings over vocabulary A

$$K(u, v) = \sum_{s \in A^*} w_s \; c(s, u) \; c(s, v)$$

$$c(s, x) = \text{number of times substring } s \text{ occurs in } x$$

Properties of Kernels

Theorem (Mercer)

K is a kernel if and only if:

- K is symmetric
- For any set of training points $x_1, x_2, \ldots, x_m$ and for any $a_1, a_2, \ldots, a_m \in R$, we have:

$$\sum_{i,j} a_i a_j K(x_i, x_j) \geq 0$$

$$a^T K a \geq 0$$

I.e., $K = (K(x_i, x_j))_{i,j=1,\ldots,n}$ is positive semi-definite.

[Maria Balcan]

Kernel, Closure Properties

Easily create new kernels using basic ones!

Fact: If $K_1(\cdot,\cdot)$ and $K_2(\cdot,\cdot)$ are kernels $c_1 \geq 0, c_2 \geq 0$,

then $K(x,z) = c_1 K_1(x,z) + c_2 K_2(x,z)$ is a kernel.

Key idea: concatenate the ϕ spaces.

$$\phi(x) = (\sqrt{c_1}\,\phi_1(x), \sqrt{c_2}\,\phi_2(x))$$

$$\phi(x)\cdot\phi(z) = c_1\,\phi_1(x)\cdot\phi_1(z) + c_2\,\phi_2(x)\cdot\phi_2(z)$$

$K_1(x,z)$ $K_2(x,z)$

[Maria Balcan]

Kernel, Closure Properties

Easily create new kernels using basic ones!

Fact: If $K_1(\cdot,\cdot)$ and $K_2(\cdot,\cdot)$ are kernels,

then $K(x,z) = K_1(x,z)K_2(x,z)$ is a kernel.

Key idea: $\phi(x) = \left(\phi_{1,i}(x)\,\phi_{2,j}(x)\right)_{i\in\{1,\ldots,n\},j\in\{1,\ldots,m\}}$

$$\phi(x)\cdot\phi(z) = \sum_{i,j}\phi_{1,i}(x)\,\phi_{2,j}(x)\phi_{1,i}(z)\,\phi_{2,j}(z)$$

$$= \sum_{i}\phi_{1,i}(x)\,\phi_{1,i}(z)\left(\sum_{j}\phi_{2,j}(x)\,\phi_{2,j}(z)\right)$$

$$= \sum_i \phi_{1,i}(x)\,\phi_{1,i}(z)K_2(x,z) = K_1(x,z)\,K_2(x,z)$$

[Maria Balcan]

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

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

- we can express as a function of kernel evaluations

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

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]

Support Vector Machines

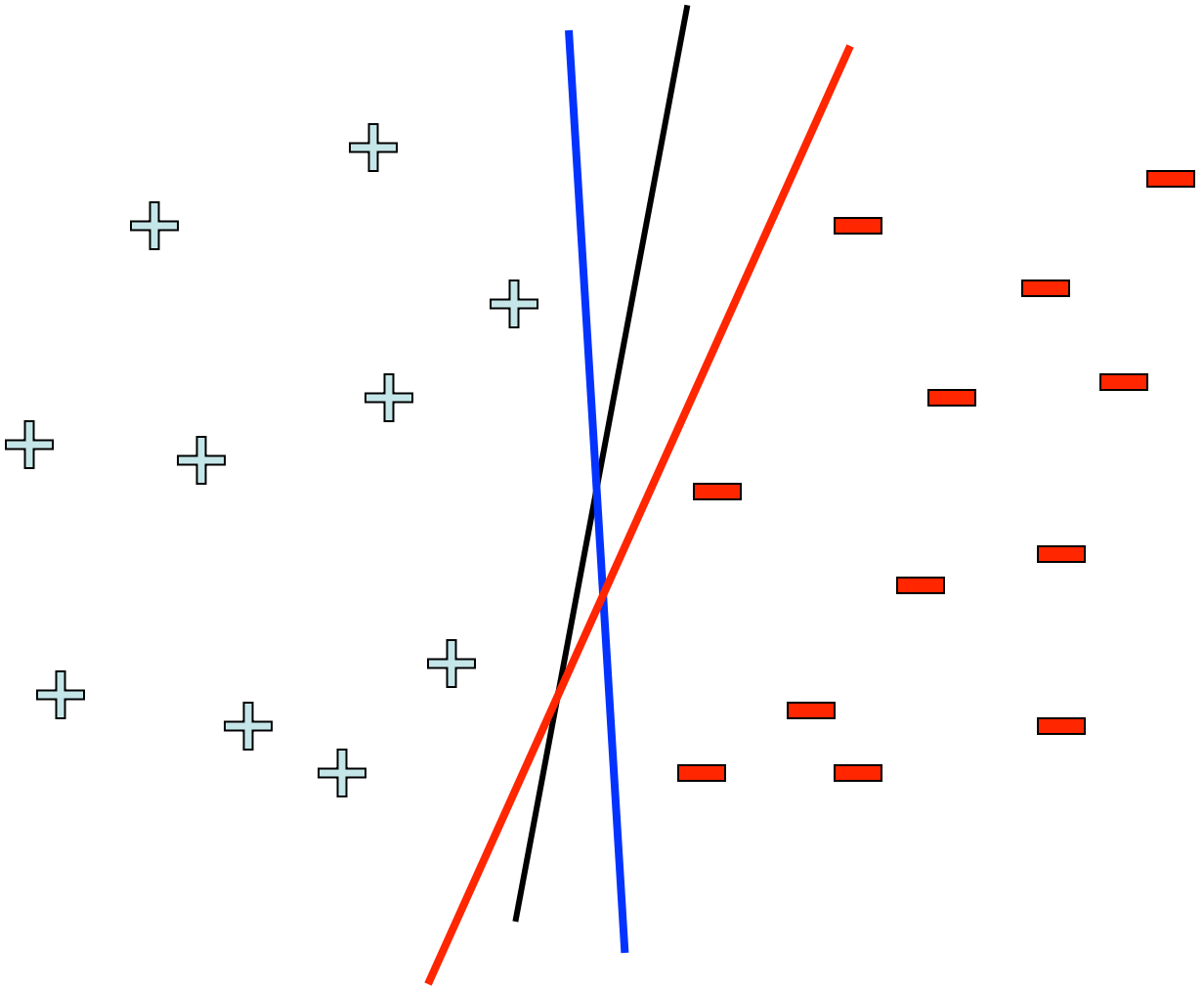
Support Vector Machines

Two key ideas:

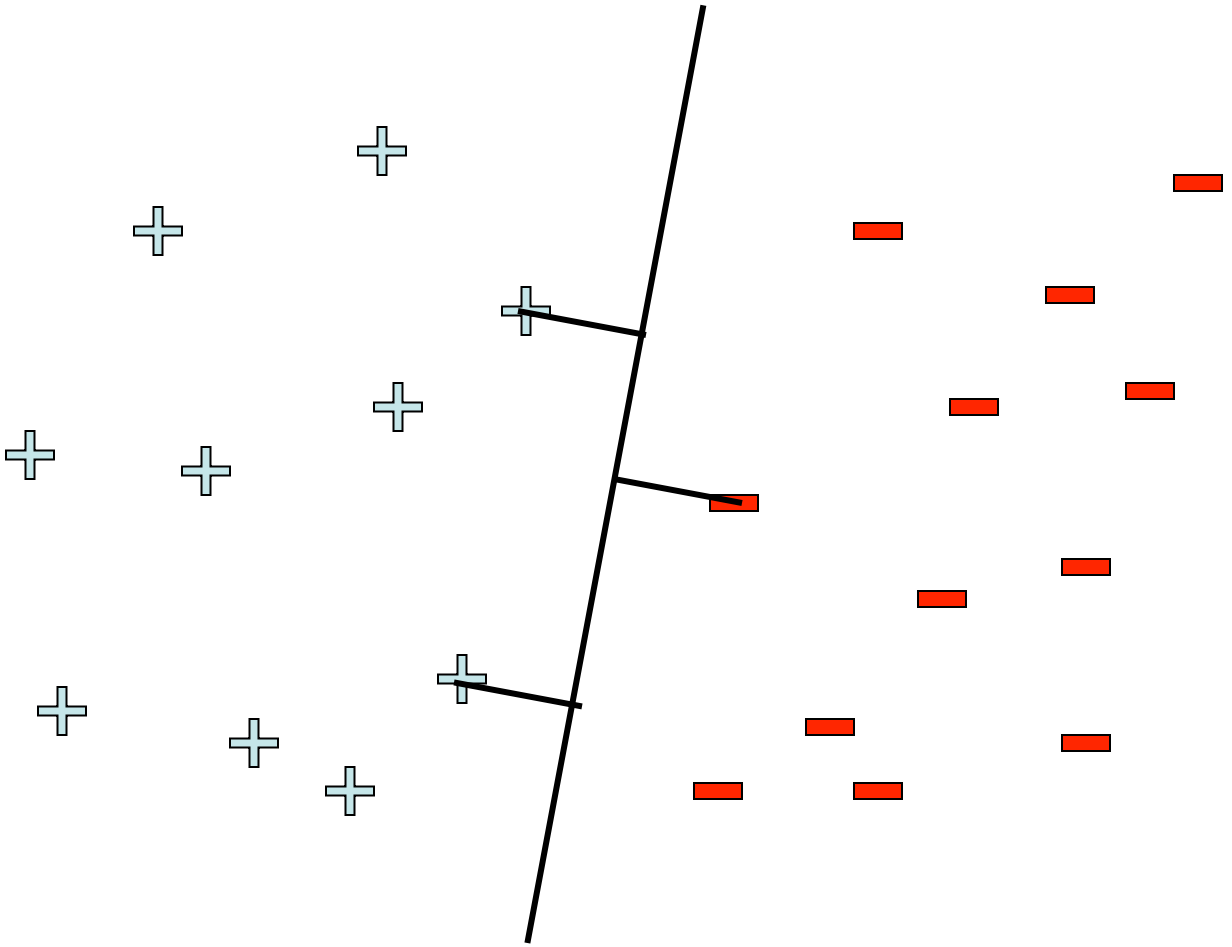
1. new loss function (maximize margin)
2. use kernels to achieve non-linear boundaries


SUPPORT
VECTOR
MACHINES

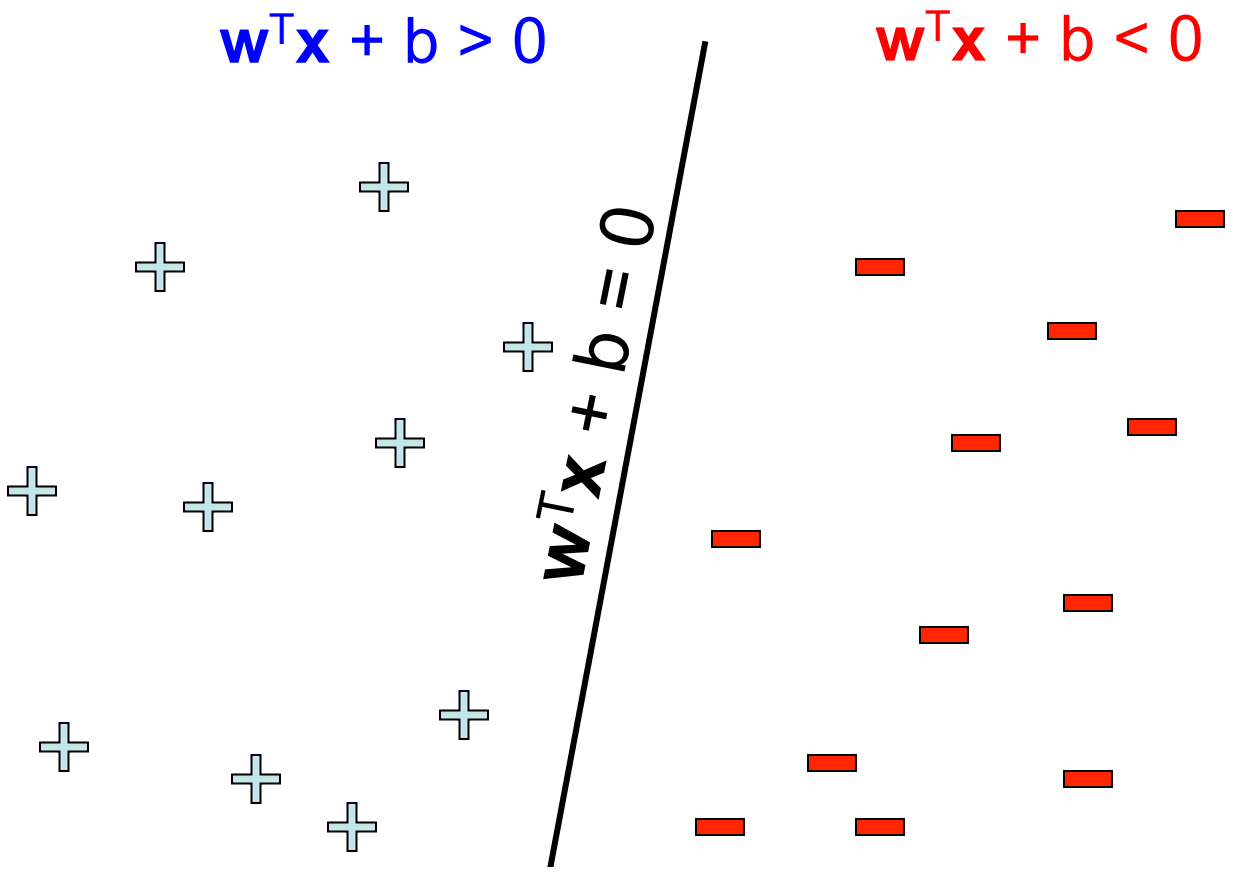
Linear classifiers – which line is better?



Pick the one with the largest margin!

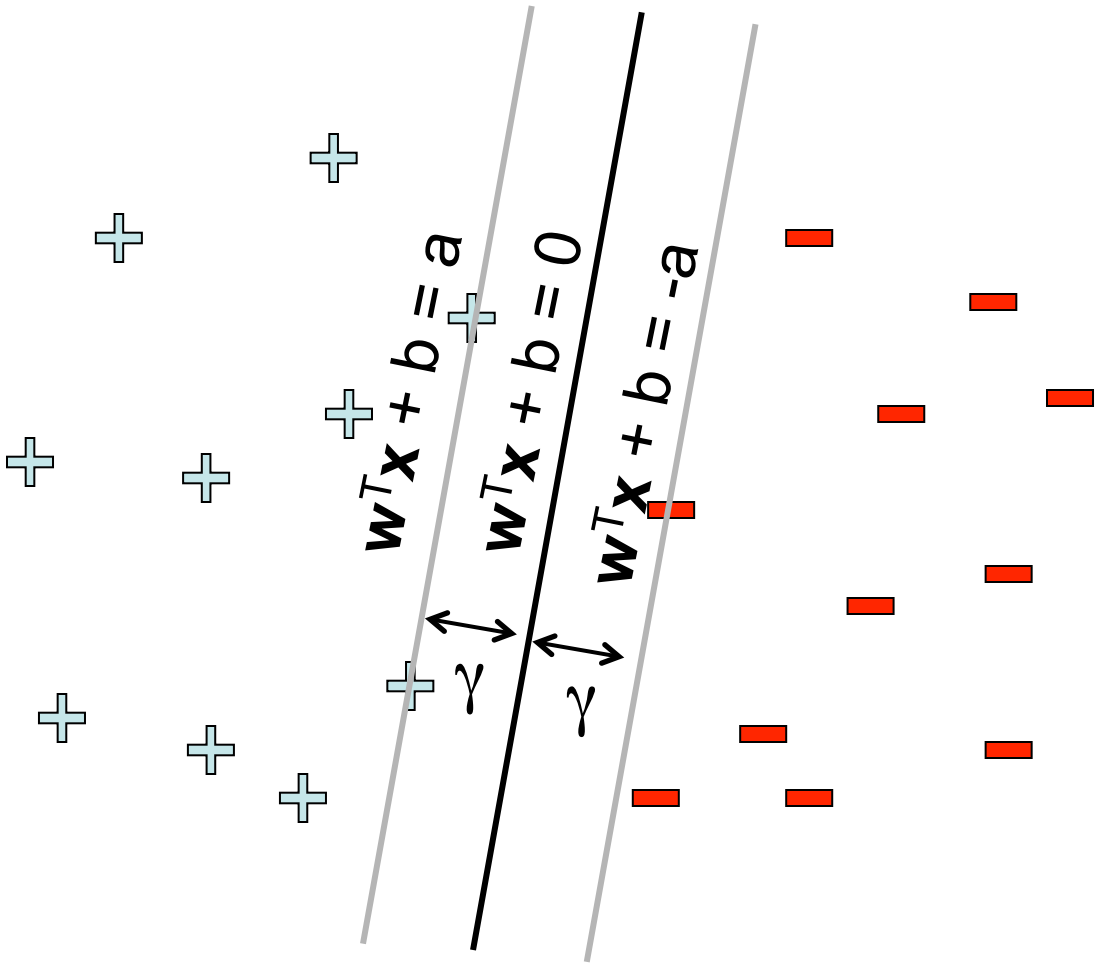


Parameterizing the decision boundary



Labels $y_i \in \{-1, +1\}$ — class

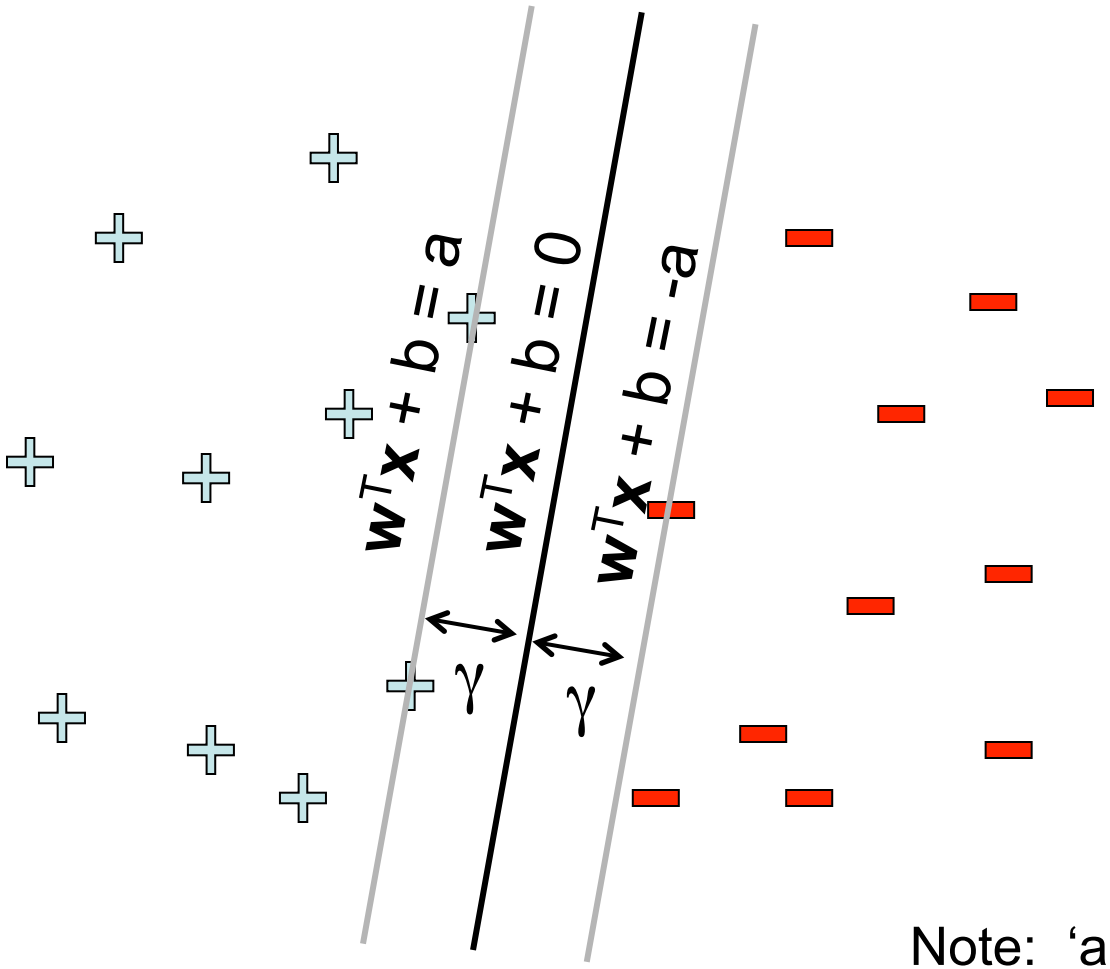
SVM: Maximize the margin



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

margin = $\gamma = a/\|w\|$

Maximizing the margin



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

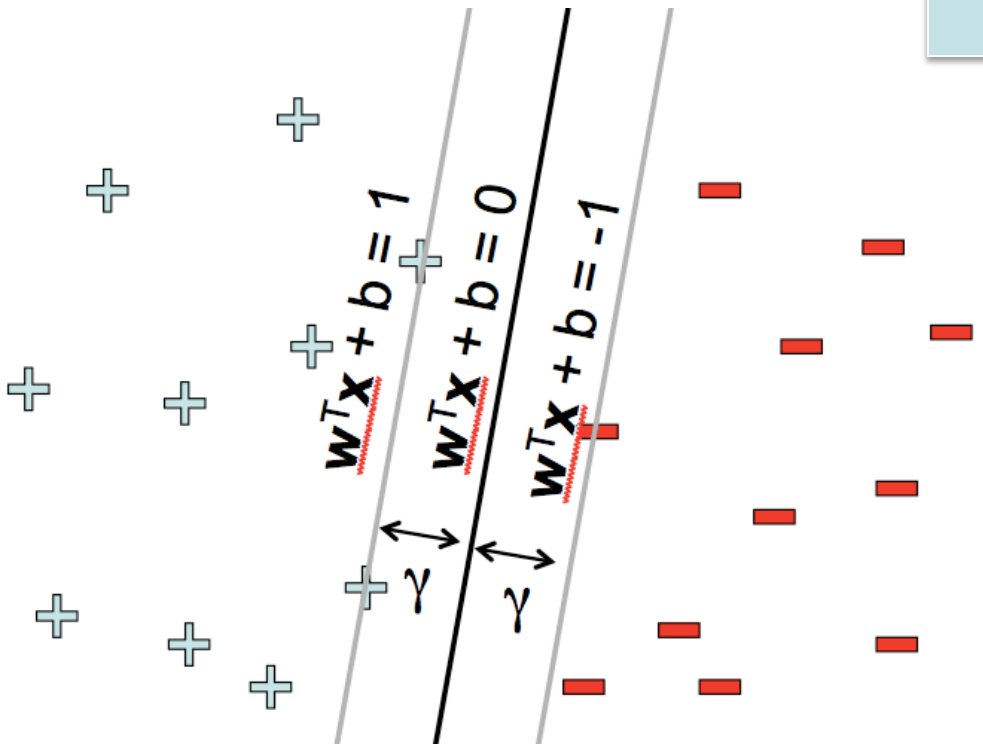
$$\text{margin} = \gamma = a/\|w\|$$

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

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

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

Support Vector Machine (primal form)



$$\max_{\mathbf{w},b} \ \gamma = 1/\|\mathbf{w}\|$$

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

Primal form:

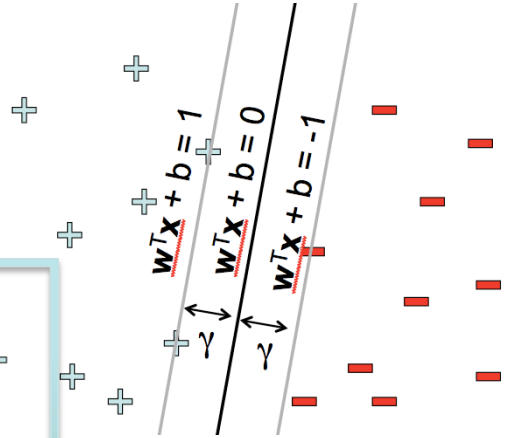
$$\min_{\mathbf{w},b} \ \mathbf{w}^T\mathbf{w}$$

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

Solve efficiently by quadratic programming (QP)

- Well-studied solution algorithms

We can solve either primal or dual forms



Primal form: solve for $\mathbf{w}$, b

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

$$\text{s.t.} \quad y_l(\mathbf{w}^T\mathbf{x}_l + b) \geq 1 \quad \forall l \in \text{ training examples}$$

Classification test for new $\mathbf{x}$: $\mathbf{w}^T\mathbf{x} + b > 0$

Dual form: solve for $\alpha_1 \ldots \alpha_M$

$$\max_{\alpha_1 \ldots \alpha_M} \quad \sum_{l=1}^{M} \alpha_l - \frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{M} \alpha_j\alpha_k y_j y_k \langle \mathbf{x_j}, \mathbf{x_k} \rangle$$

$$\text{s.t.} \quad \alpha_l \geq 0 \quad \forall l \in \text{ training examples}$$

$$\sum_{l=1}^{M} \alpha_l y_l = 0$$

Classification test for new $\mathbf{x}$: $\sum_{l \in \text{SV's}} \alpha_l \; y_l \; \langle \mathbf{x}, \mathbf{x}_l \rangle + b > 0$

both are QP problems with a single local optimum!

What is quadratic programming?

A way to solve optimization problems of the form:

$$\text{Minimize } f(\mathbf{x}) = \mathbf{cx} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\,\mathbf{x}$$

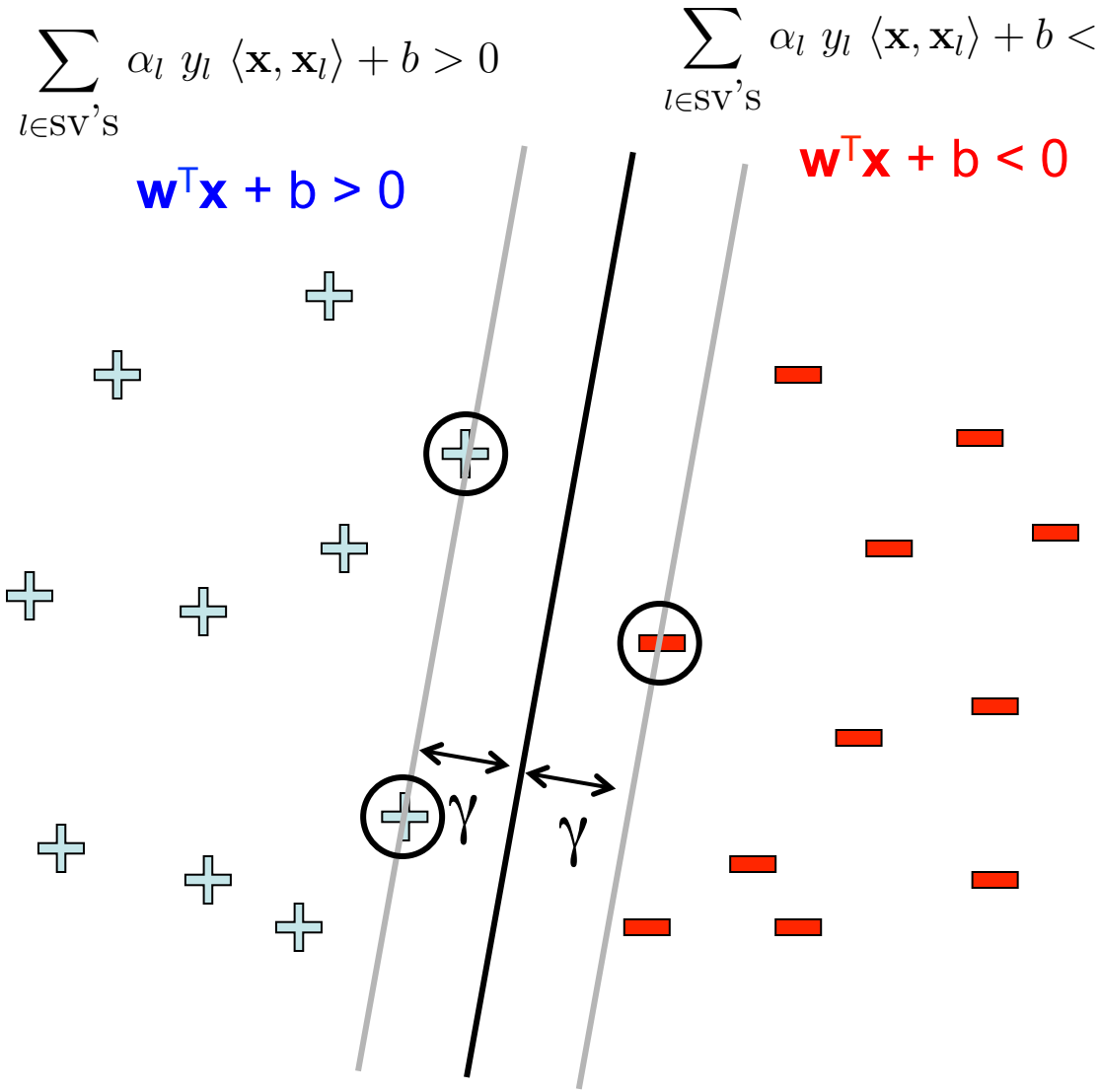
$$\text{subject to } \mathbf{Ax} \le \mathbf{b} \text{ and } \mathbf{x} \ge \mathbf{0}$$

Where **Q** is symmetric matrix.

If f(**x**) is strictly convex for all feasible points (e.g., **Q** is positive semi-definite)

then there is only one local minimum, which is global min.

Support Vectors



Linear hyperplane is fully defined by "support vectors" *

Moving other points a little doesn't effect the decision boundary

only need to store the support vectors to predict labels of new points

How many support vectors in linearly separable case, given d dimensions?

≤ d+1

* KKT conditions on optimization problem assure other α_i=0

Kernel SVM

And because the dual form depends only on inner products, we can apply the kernel trick to work in a (virtual) projected space $\Phi : X \rightarrow F$

Primal form: solve for $\mathbf{w}, b$ in the projected higher dim. space

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

$$\text{s.t.} \quad y_l(\mathbf{w}^T\Phi(\mathbf{x}_l) + b) \geq 1 \quad \forall l \in \text{ training examples}$$

Classification test for new $\mathbf{x}$ $\quad \mathbf{w}^T\Phi(\mathbf{x}) + b > 0$

Dual form: solve for $\alpha_1 \ldots \alpha_M$ in the original low dim. space

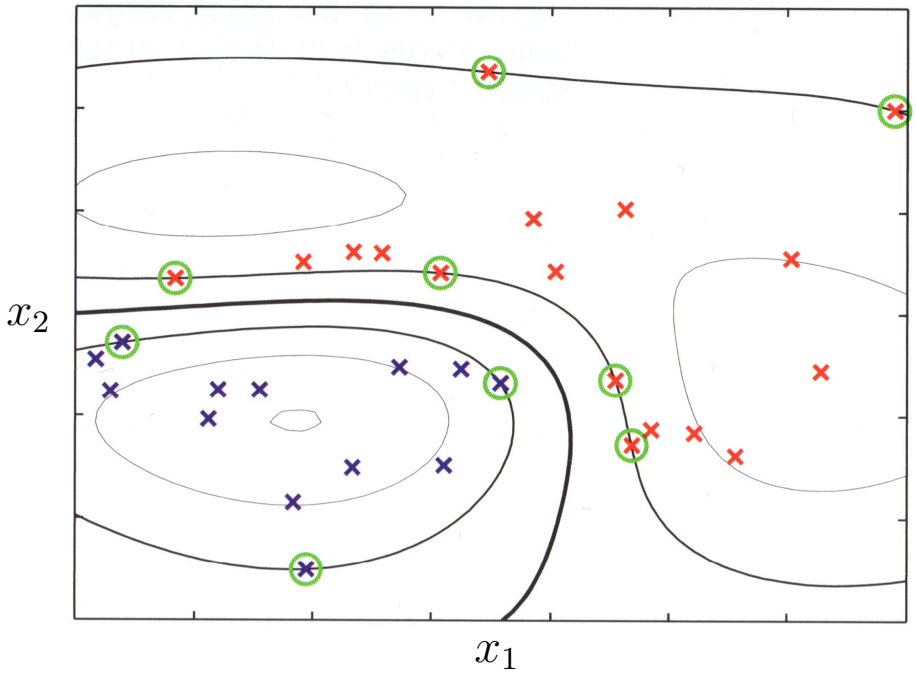
$$\max_{\alpha_1 \ldots \alpha_M} \quad \sum_{l=1}^{M} \alpha_l - \frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{M} \alpha_j\alpha_k y_j y_k \ \kappa(\mathbf{x_j}, \mathbf{x_k})$$

$$\text{s.t.} \quad \alpha_l \geq 0 \qquad \forall l \in \text{ training examples}$$

$$\sum_{l=1}^{M} \alpha_l y_l = 0$$

Classification test for new $\mathbf{x}$: $\quad \sum_{l \in \text{SV's}} \alpha_l \ y_l \ \kappa(\mathbf{x}, \mathbf{x}_l) + b > 0$

SVM Decision Surface using Gaussian Kernel



[from Bishop, figure 7.2]

$$\hat{f}(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

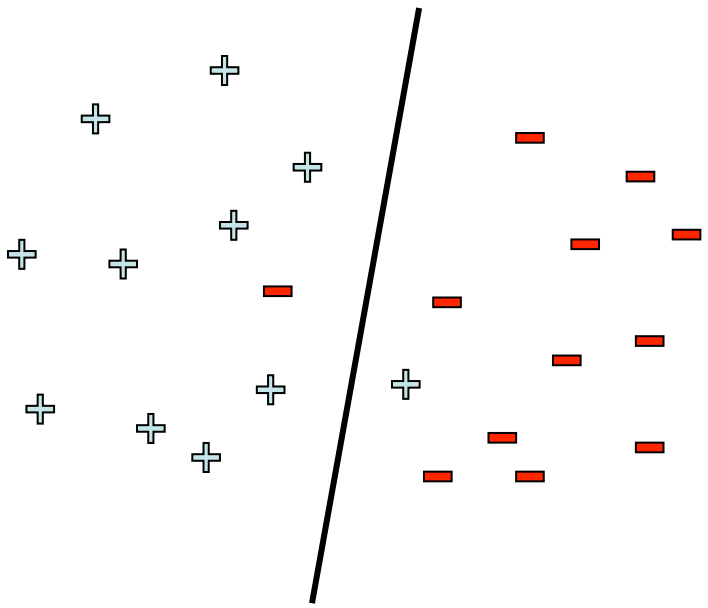
Circled points are the *support vectors*: training examples with non-zero α_l

Points plotted in original 2-D space.

Contour lines show constant $\hat{f}(\mathbf{x})$

$$\hat{f}(\mathbf{x}) = b + \sum_{l=1}^{M} \alpha_l \; y_l \; \kappa(\mathbf{x}, \mathbf{x}_l) = b + \sum_{l=1}^{M} \alpha_l \; y_l \exp(-\|\mathbf{x} - \mathbf{x}_l\|^2 / 2\sigma^2)$$

What if data is not linearly separable?



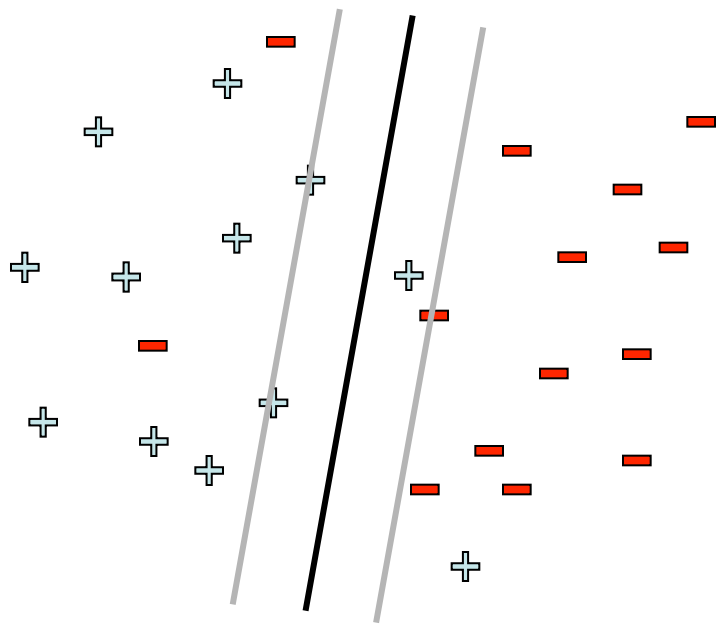
Use features of features
of features of features….

$x_1^2, x_2^2, x_1x_2, \ldots., \exp(x_1)$

But run risk of overfitting!

What if data is still not linearly separable? e.g., very noisy, inherently not separable

Allow "error" in classification



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

Maximize margin and minimize # mistakes on training data

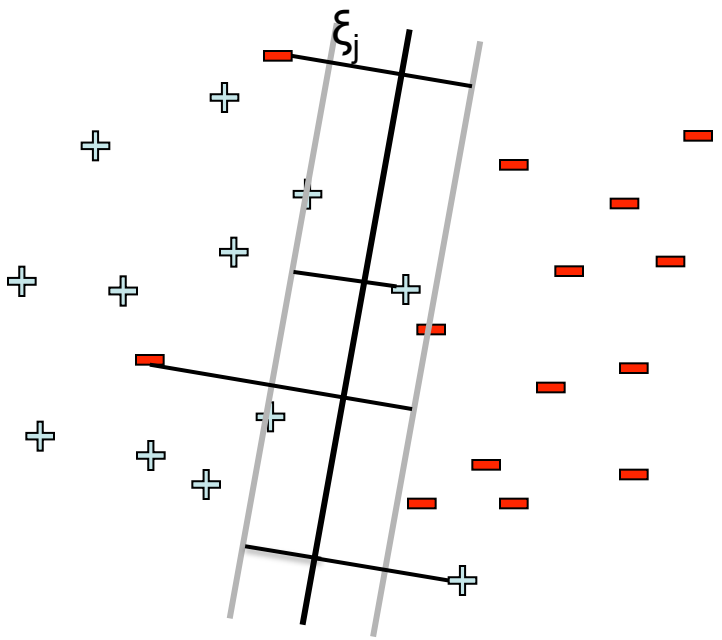
C - tradeoff parameter

Not QP ☹

0/1 loss (doesn't distinguish between near miss and bad mistake)

Support Vector Machine with soft margins

Allow “error” in classification



Soft margin approach

$$\min_{\mathbf{w},b} \ \mathbf{w}^T\mathbf{w} + C \sum_j \xi_j$$

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

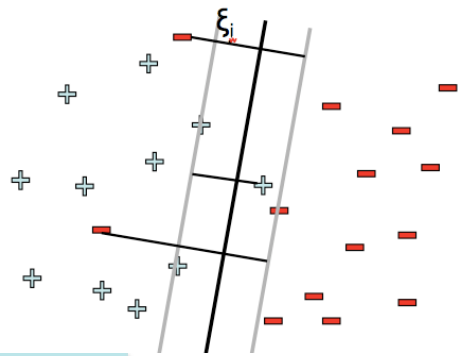
$$\xi_j \geq 0 \quad \forall j$$

ξ_j - “slack” variables
= (>1 if x_j misclassifed)
pay linear penalty if mistake

C - tradeoff parameter (chosen by cross-validation)

Still QP ☺

Primal and Dual Forms for Soft Margin SVM



Primal form: solve for $\mathbf{w}$, b in the projected higher dim. space

$$\min_{\mathbf{w},b} \quad \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{l=1}^{M}\xi_l$$

$$\text{s.t.} \quad y_l(\mathbf{w}^T\Phi(\mathbf{x}_l) + b) \geq 1 - \xi_l \quad \forall l \in \text{ training examples}$$

$$\xi_l \geq 0 \quad \forall l \in \text{ training examples}$$

Dual form: solve for $\alpha_1 \ldots \alpha_M$ in the original low dim. space

$$\max_{\alpha_1 \ldots \alpha_M} \quad \sum_{l=1}^{M}\alpha_l - \tfrac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{M}\alpha_j\alpha_k y_j y_k \ \kappa(\mathbf{x_j}, \mathbf{x_k})$$

$$\text{s.t.} \quad 0 \leq \alpha_l \leq C \quad \forall l \in \text{ training examples}$$

$$\sum_{l=1}^{M}\alpha_l y_l = 0$$

both are QP problems with a single local optimum ☺

Support Vector Machine with Kernels!

- Primal form:

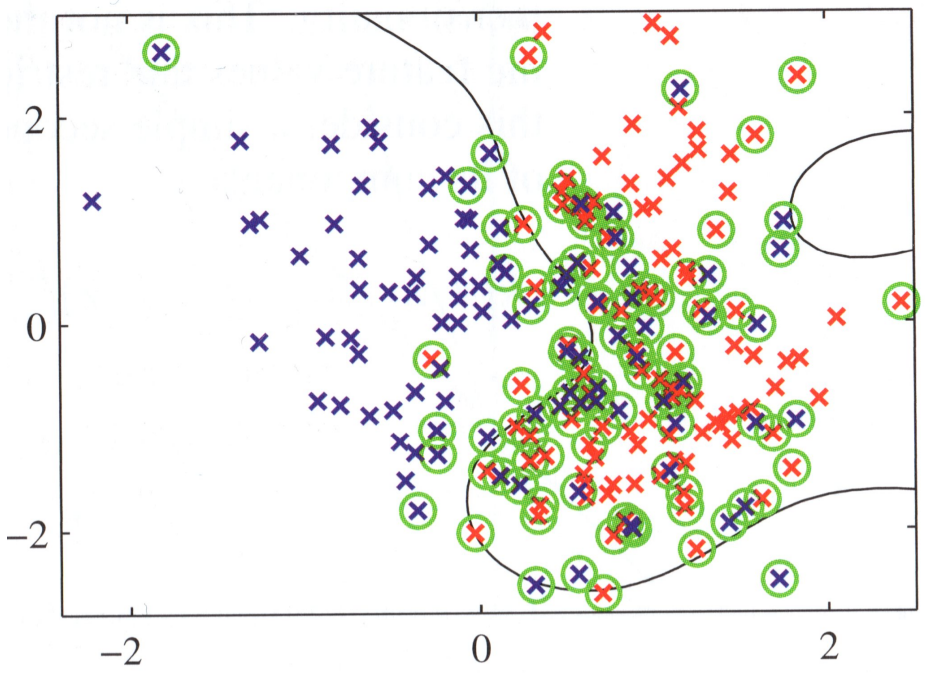
$$\begin{aligned} & \min_{\boldsymbol{w}} \quad \left[\tfrac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i\right] \\ \text{s.t.} \quad & y_i \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) \geq 1 - \xi_i \qquad i = 1, \ldots, m \\ & \xi_i \geq 0 \qquad i = 1, \ldots, m \end{aligned}$$

- Dual form:

$$\begin{aligned} & \max_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{m}\alpha_i - \tfrac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)\right] \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \quad i = 1, \ldots, m \end{aligned}$$

where $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle$ and $\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle = \sum_{i=1}^{m} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x})$.

SVM Soft Margin Decision Surface using Gaussian Kernel



[from Bishop, figure 7.4]

$$\hat{f}(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

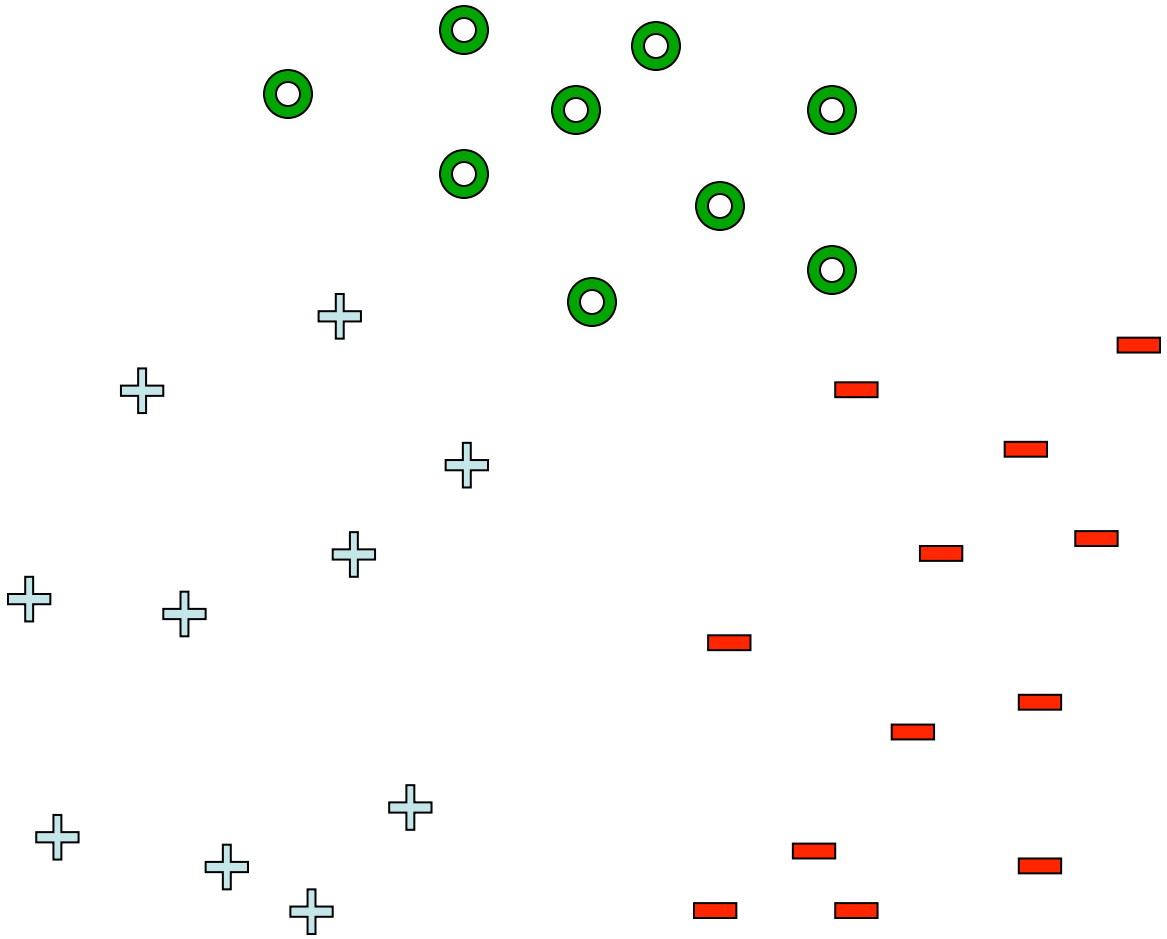
Circled points are the *support vectors*: training examples with non-zero α_l

Points plotted in original 2-D space.

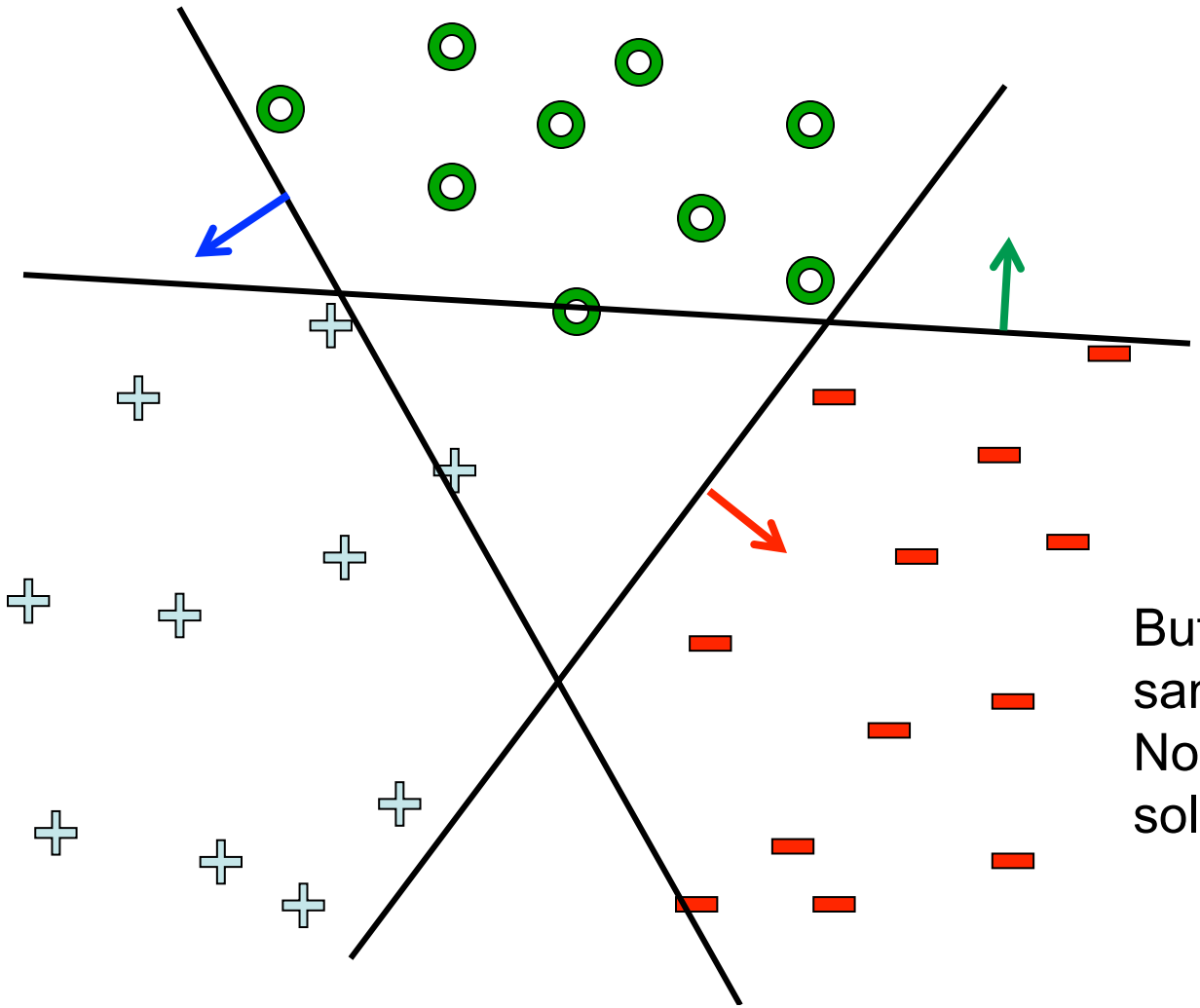
Contour lines show constant $\hat{f}(\mathbf{x})$

$$\hat{f}(\mathbf{x}) = b + \sum_{l=1}^{M} \alpha_l \; y_l \; \kappa(\mathbf{x}, \mathbf{x}_l) = b + \sum_{l=1}^{M} \alpha_l \; y_l \exp(-\|\mathbf{x} - \mathbf{x}_l\|^2 / 2\sigma^2)$$

What about multiple classes?



One against all



Could try to learn
3 separate classifiers:
Class k vs. rest

$$(\mathbf{w}_k, b_k)_{k=1,2,3}$$

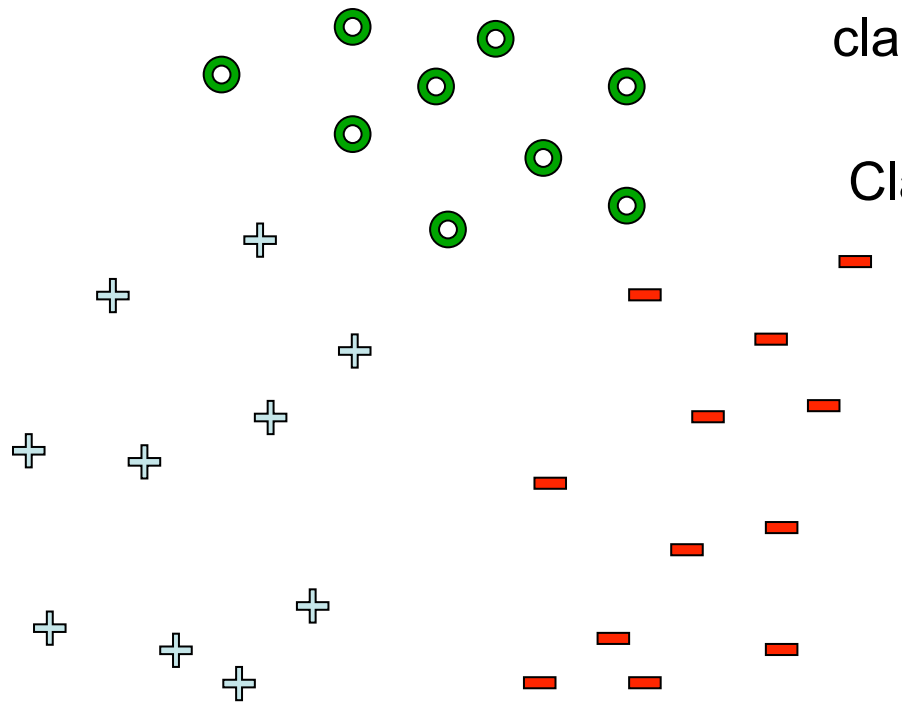
$$y = \arg\max_k \mathbf{w}_k^T x + b_k$$

But $\mathbf{w}_k$'s might not be on the same scale.
Note: (a**w**).x + (ab) is also a solution

Learn single, Multi-class SVM

Simultaneously learn 3 sets of weights

$$\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')}.\mathbf{x}_j + b^{(y')} + 1, \ \forall y' \neq y_j, \ \forall j$$



Margin = gap between correct class and nearest other class

Classification rule:

$y = \arg\max_k \mathbf{w}^{(k)}.x + b^{(k)}$

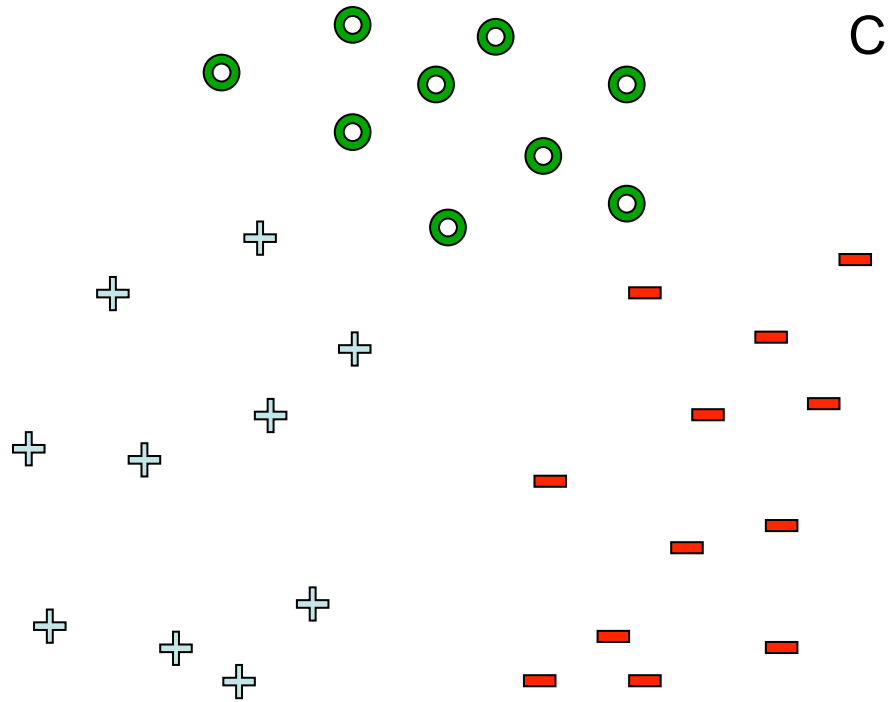
Learn single Multi-class SVM

Simultaneously learn 3 sets of weights

$$\text{minimize}_{\mathbf{w},b} \quad \sum_y \mathbf{w}^{(y)}.\mathbf{w}^{(y)} + C \sum_j \sum_{y \neq y_j} \xi_j^{(y)}$$

such that: $\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y)}.\mathbf{x}_j + b^{(y)} + 1 - \xi_j^{(y)}, \ \forall y \neq y_j, \ \forall j$

$$\xi_j^{(y)} \geq 0 \qquad , \ \forall y \neq y_j, \ \forall j$$



Classification rule:

$y = \arg\max_k \mathbf{w}^{(k)}.x + b^{(k)}$

Joint optimization: $\mathbf{w}_k$s are on the same scale.

SVM Summary

- Objective: maximize margin between decision surface and data
- Primal and dual formulations
  - dual represents classifier decision in terms of *support vectors*
- Kernel SVM's
  - learn linear decision surface in high dimension space, working in original low dimension space
- Handling noisy data: soft margin "slack variables"
  - again primal and dual forms
- SVM algorithm: Quadratic Program optimization
  - single global optimum

Maximizing Margin as an Objective Function

- We've talked about many learning algorithms, with different objective functions

- 0-1 loss
- sum sq error
- maximum log data likelihood
- MAP
- maximum margin

How are these all related?

Slack variables – Hinge loss

Complexity penalization

$$\xi_j = \text{loss}(f(x_j), y_j)$$

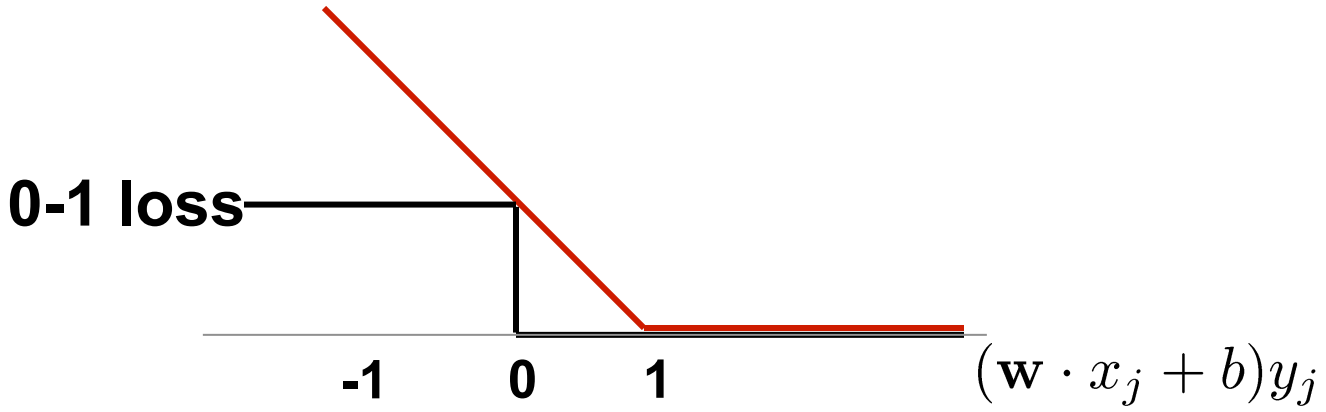
$$f(x_j) = \text{sgn}(\mathbf{w} \cdot \mathrm{x_j} + \mathrm{b})$$

$$\min_{\mathbf{w},b} \ \mathbf{w}^T\mathbf{w} + C \sum_j \xi_j$$

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

$$\xi_j \geq 0 \quad \forall j$$

$$\xi_j = (1 - (\mathbf{w} \cdot x_j + b)y_j))_+$$ ⇐ **Hinge loss**



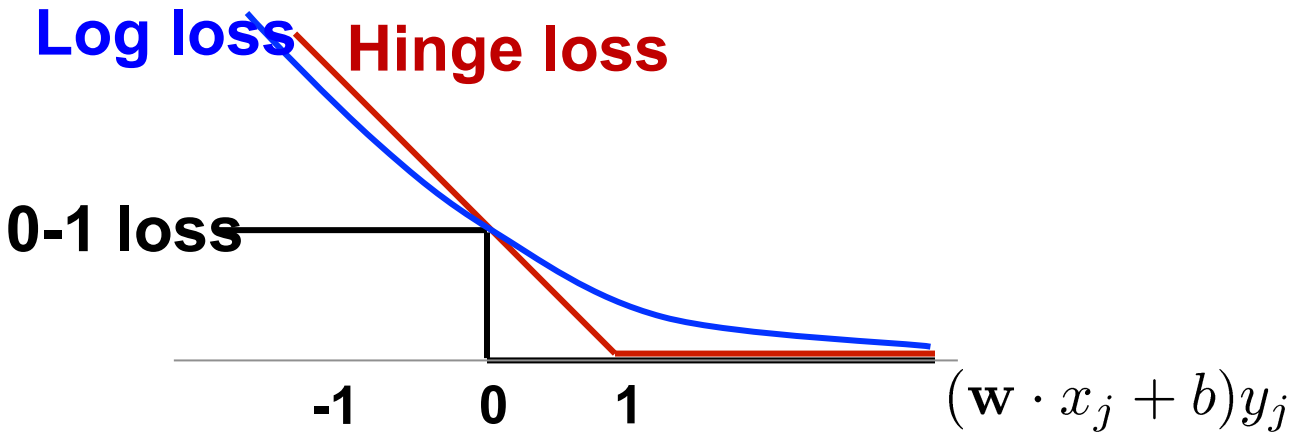
SVM vs. Logistic Regression

SVM : **Hinge loss**

$$\text{loss}(f(x_j), y_j) = (1 - (\mathbf{w} \cdot x_j + b)y_j))_+$$

Logistic Regression : **Log loss** (negative log conditional likelihood)

$$\text{loss}(f(x_j), y_j) = -\log P(y_j \mid x_j, \mathbf{w}, b) = \log(1 + e^{-(\mathbf{w} \cdot x_j + b)y_j})$$

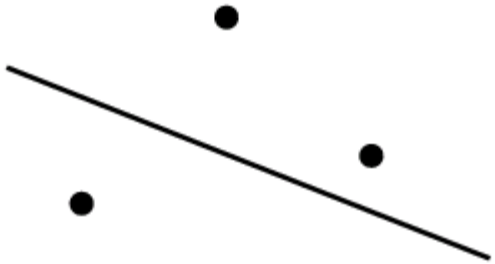


SVM: PAC Results?

VC dimension: examples

What is VC dimension of

- $H_2 = \{ ((w_0 + w_1x_1 + w_2x_2)>0 \rightarrow y=1) \}$
  - $VC(H_2)=3$
- For H_n = linear separating hyperplanes in n dimensions, $VC(H_n)=n+1$



$$m \geq \frac{1}{\epsilon}(4\log_2(2/\delta) + 8VC(H)\log_2(13/\epsilon))$$

Margin-based PAC Results

[Shawe-Taylor, Langford, McCallester]

Consider a fixed distribution D on pairs $\langle x,\ y\rangle$ with $x \in R^d$ satisfying $||x|| = 1$ and $y \in \{-1, 1\}$. We are interested in finding a weight vector w with $||w|| = 1$ such that the sign of $w \cdot x$ predicts y. For $\gamma > 0$ the error rate of w on distribution D relative to safety margin γ, denoted $\ell_\gamma(w,\ D)$ is defined as follows.

$$\ell_\gamma(w,\ D) = \mathrm{P}_{\langle x,\ y\rangle \sim D}\left[(w \cdot x)y \leq \gamma\right]$$

Let S be a sample of m pairs drawn IID from the distribution D. The sample S can be viewed as an empirical distribution on pairs. We are interested in bounding $\ell_0(w,\ D)$ in terms of $\ell_\gamma(w,\ S)$ and the margin γ. Bartlett and Shawe-Taylor use fat shattering arguments [2] to show that with probability at least $1 - \delta$ over the choice of the sample S we have the following simultaneously for all weight vectors w with $||w|| = 1$ and margins $\gamma > 0$.

$$\ell_0(w, D) \leq \ell_\gamma(w, S) + 27.18\sqrt{\frac{\log^2 m + 84}{m\gamma^2}} + O\left(\sqrt{\frac{\ln \frac{1}{\delta}}{m}}\right) \qquad (1)$$

recall:

$$error_{true}(h) < error_{train}(h) + \sqrt{\frac{VC(H)(\ln \frac{2m}{VC(H)} + 1) + \ln \frac{4}{\delta}}{m}}$$

Perceptron Algorithm

Perceptron Algorithm: learn $\hat{y} = h(x) = sign(\vec{w} \cdot \vec{x})$, where $\vec{x} =< 1, x_1, \ldots, x_n >$, $\vec{w} =< w_0, w_1 \ldots, w_n >$, $y \in \{-1, +1\}$

Input: $\{< \vec{x}_1, y_1 > \ldots < \vec{x}_m, y_m >\}$

Initialize $\vec{w} = 0$;

repeat

- for $i = 1$ to m
  - **if** $y_i \neq sign(\vec{w} \cdot \vec{x}_i)$ **then** $\vec{w} \leftarrow \vec{w} + y_i \vec{x}_i$;

until converged

Perceptron Algorithm

Perceptron Algorithm: learn $\hat{y} = h(\vec{x}) = sign(\vec{w} \cdot \vec{x})$, where $\vec{x} = [1, x_1, \ldots, x_n]$, $\vec{w} = [w_0, w_1 \ldots, w_n]$, $y \in \{-1, +1\}$

Input: $\{< \vec{x}^{(1)}, y^{(1)} > \ldots < \vec{x}^{(m)}, y^{(m)} >\}$

Initialize $\vec{w} = 0$;

repeat

- for $i = 1$ to m
  - **if** $y^{(i)} \neq sign(\vec{w} \cdot \vec{x}^{(i)})$
    then $\vec{w} \leftarrow \vec{w} + y^{(i)}\vec{x}^{(i)}$;

until converged

Mistake Bounds for Perceptron

When data is linearly separable:

THEOREM 1 (BLOCK, NOVIKOFF) *Let* $\langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \rangle$ *be a sequence of labeled examples with* $||\mathbf{x}_i|| \leq R$. *Suppose that there exists a vector* $\mathbf{u}$ *such that* $||\mathbf{u}|| = 1$ *and* $y_i(\mathbf{u} \cdot \mathbf{x}_i) \geq \gamma$ *for all examples in the sequence. Then the number of mistakes made by the online perceptron algorithm on this sequence is at most* $(R/\gamma)^2$.

Mistake Bounds for Perceptron

When data is linearly separable:

THEOREM 1 (BLOCK, NOVIKOFF) *Let* $\langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$ *be a sequence of labeled examples with* $||\mathbf{x}_i|| \leq R$. *Suppose that there exists a vector* $\mathbf{u}$ *such that* $||\mathbf{u}|| = 1$ *and* $y_i(\mathbf{u} \cdot \mathbf{x}_i) \geq \gamma$ *for all examples in the sequence. Then the number of mistakes made by the online perceptron algorithm on this sequence is at most* $(R/\gamma)^2$.

When not linearly separable: [Freund & Schapire]

THEOREM 2 *Let* $\langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$ *be a sequence of labeled examples with* $||\mathbf{x}_i|| \leq R$. *Let* $\mathbf{u}$ *be any vector with* $||\mathbf{u}|| = 1$ *and let* $\gamma > 0$. *Define the deviation of each example as*

$$d_i = \max\{0, \gamma - y_i(\mathbf{u} \cdot \mathbf{x}_i)\} ,$$

and define $D = \sqrt{\sum_{i=1}^{m} d_i^2}$. *Then the number of mistakes of the online perceptron algorithm on this sequence is bounded by*

$$\left(\frac{R + D}{\gamma}\right)^2 .$$

What you should know

Primal and Dual optimization problems

Kernel functions

Support Vector Machines

- Maximizing margin
- Kernel SVM's
- Noise, slack variables and hinge loss
- Relationship between SVMs and logistic regression
  - 0/1 loss
  - Hinge loss
  - Log loss

Theory shows overfitting, mistakes depends on margin size

Other things to cover

- Support vector regression [see Smola tutorial]
- margin-based PAC bound proof
- margin-based mistake bound perceptron proof

- add a derivation of the dual form, using Lagrangian, mentioning KKT conditions, like the derivation of ‘center of the smallest sphere’ in Shaw-Taylor book, pgs. 197—200.
- add figure like prev. lecture hand drawing showing 2d and 3 d space, and saying “dot product here can be calculated by kernel function here”