Support Vector Machines

Aarti Singh and Eric Xing

Machine Learning 10-701/15-781
Oct 8, 2012

MACHINE LEARNING DEPARTMENT
Carnegie Mellon.
School of Computer Science

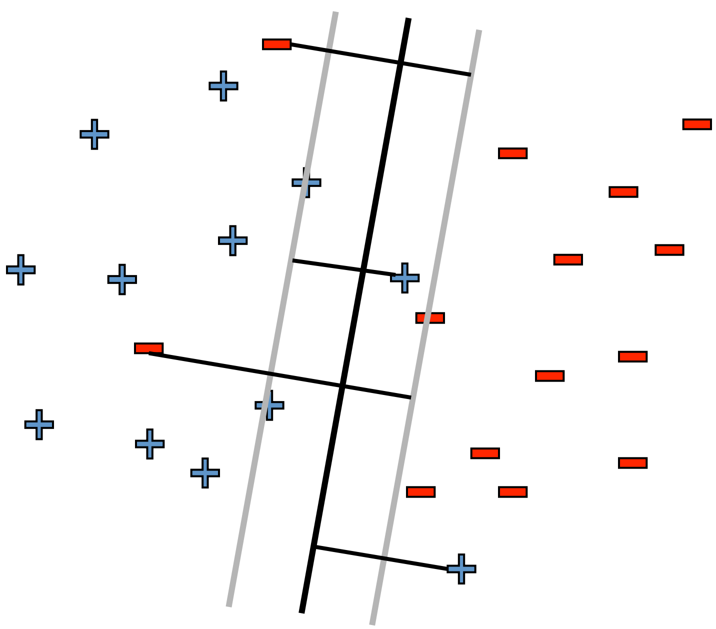
SVMs reminder

Regularization   Hinge loss

$$\min_{\mathbf{w},b,\xi_j} \underbrace{\mathbf{w}.\mathbf{w}}_{} + C \underbrace{\Sigma \xi_j}_{}$$

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

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



Soft margin approach

Hard margin approach: $C = \infty$

Why not C = 0?

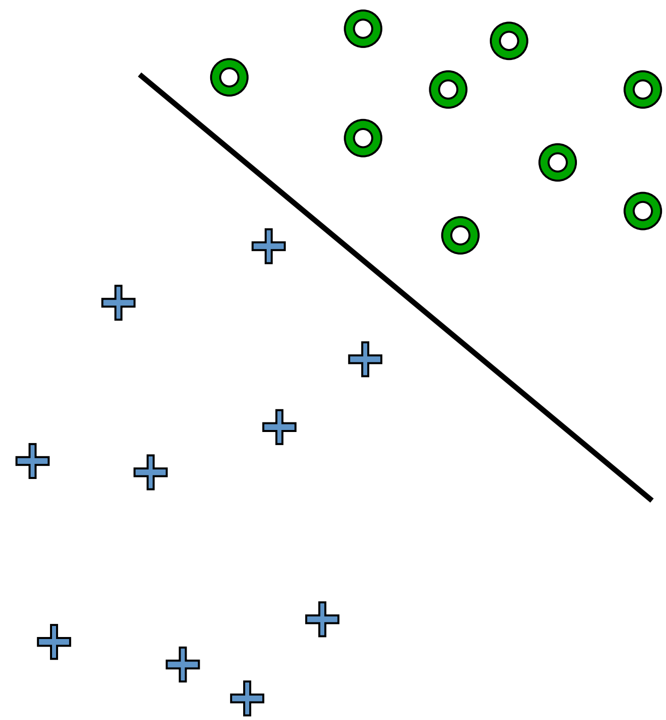
How does C control model complexity?

Margin – 2 class vs multi-class

2 class SVM:

Confidence = Distance from decision boundary

$$\mathbf{w} \cdot \mathbf{x}_j + b$$



y = sign($\mathbf{w} \cdot \mathbf{x} + b$)

Margin – 2 class vs multi-class

2 class SVM:

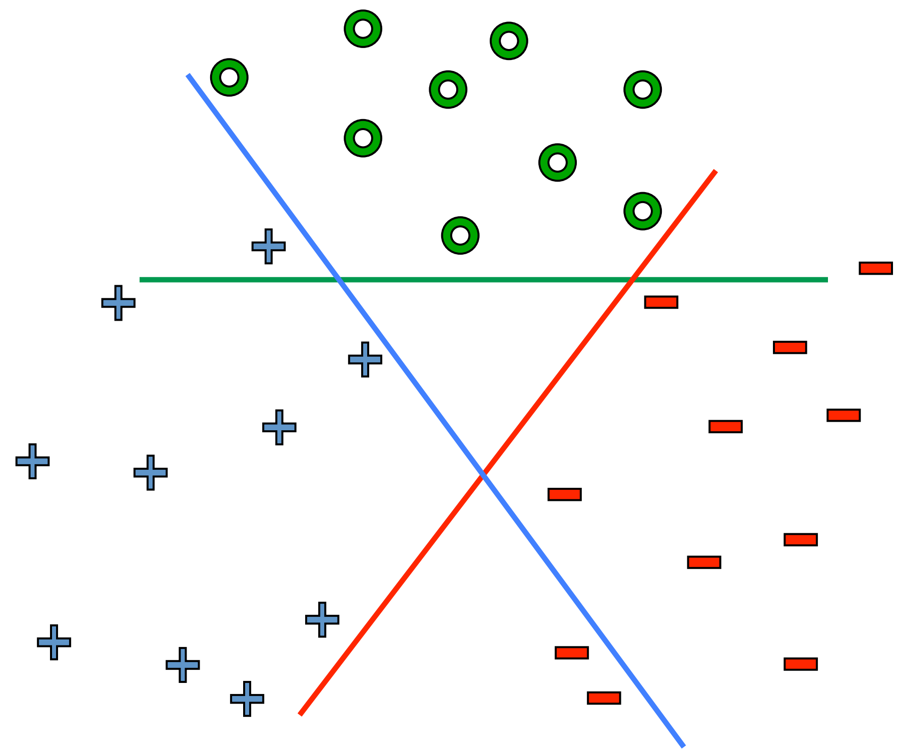
Confidence = Distance from decision boundary

$$\mathbf{w} \cdot \mathbf{x}_j + b$$

Multi-class SVM:

Confidence = Gap between distance to correct class and nearest other class

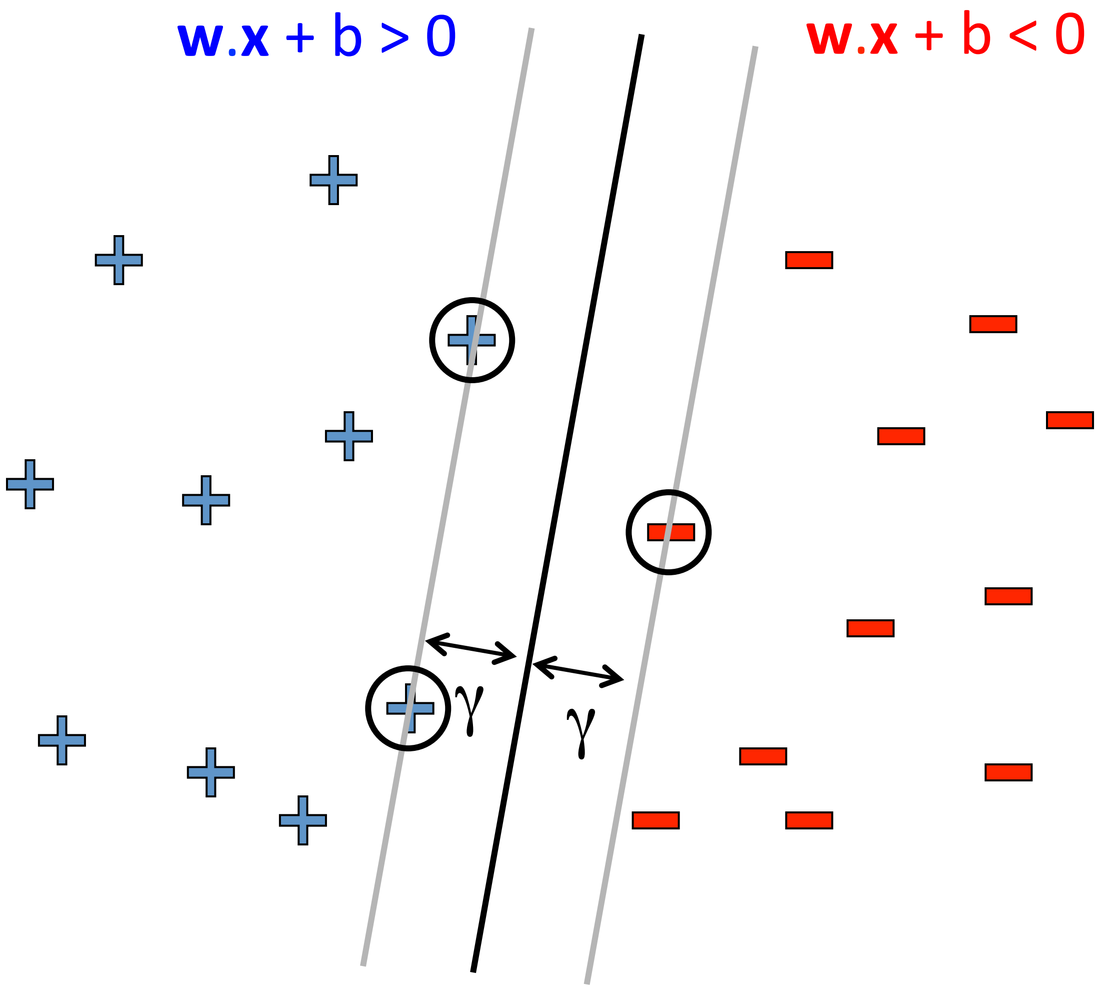
$$\mathbf{w}^{(y_j)} \cdot \mathbf{x}_j + b - (\mathbf{w}^{(y')} \cdot \mathbf{x}_j + b)$$



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

What does decision boundary look like?

Support Vectors – Hard margin SVM

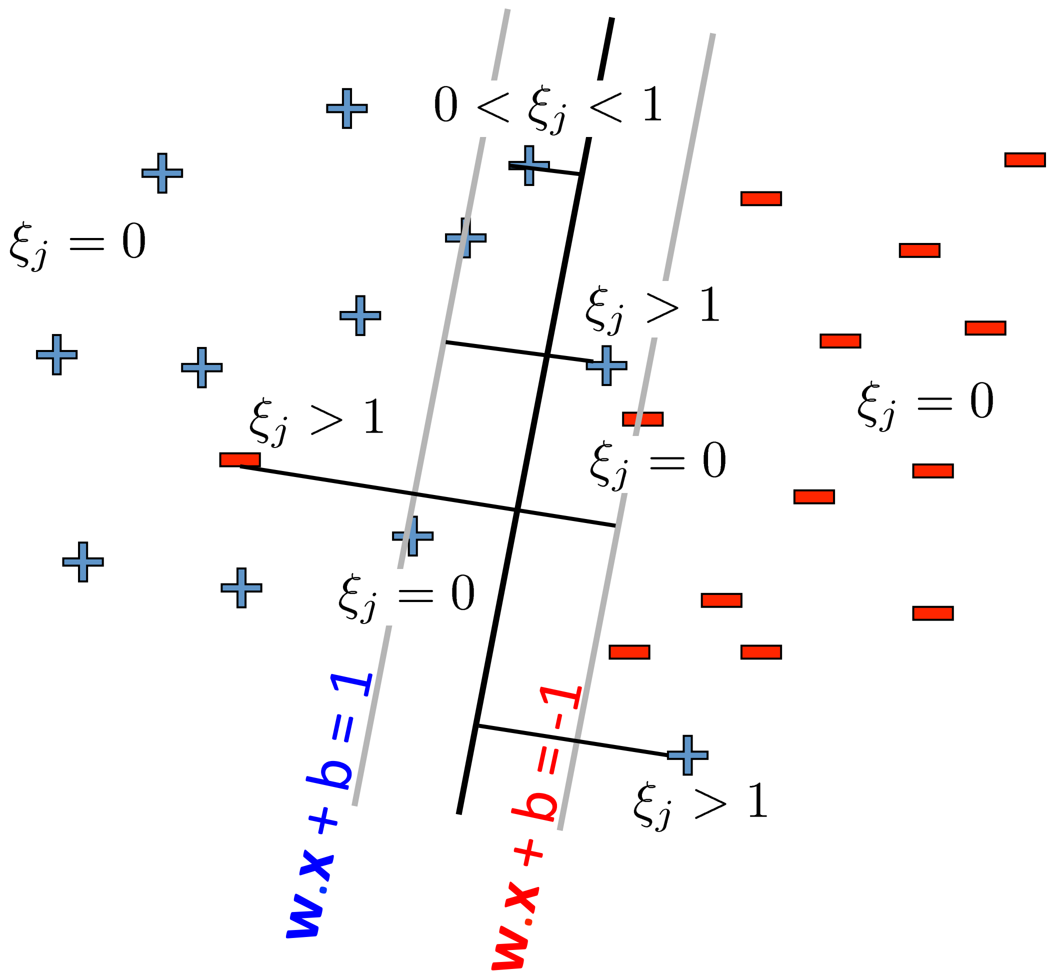


Linear hyperplane defined by "support vectors"

$$i: (\mathbf{w}.\mathbf{x}_i + b)\, y_i = 1$$

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

Support vectors - Soft-margin SVM



Linear hyperplane defined by "support vectors"

i: ($\mathbf{w}.\mathbf{x}_i$+b) y_i = 1-ξ_i

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

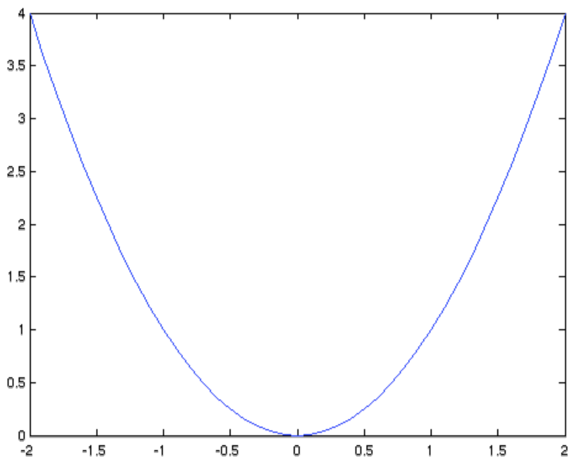
Today's Lecture

- Learn one of the most interesting and exciting advancements in machine learning
  - The "kernel trick"
  - High dimensional feature spaces at no extra cost!
- But first, a detour
  - Constrained optimization!

Constrained Optimization

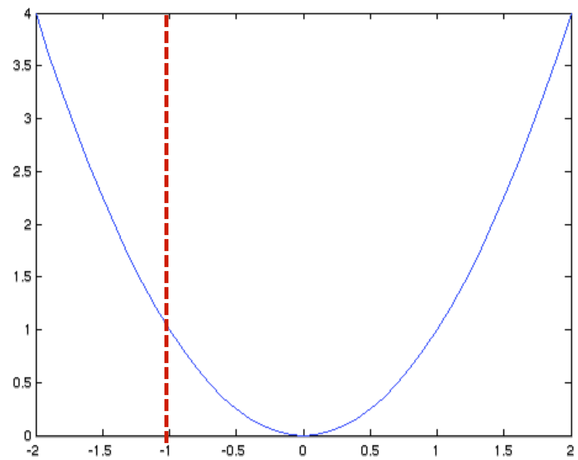
$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

$\min_x \; x^2$



$x^* = 0$

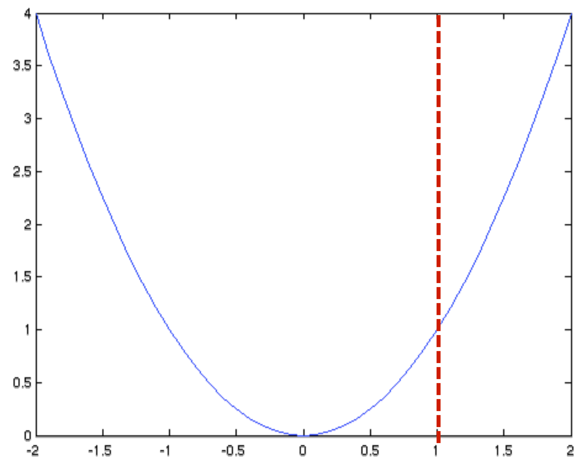
$\min_x \; x^2$
s.t. $\quad x \geq -1$



$x^* = 0$

Constraint inactive

$\min_x \; x^2$
s.t. $\quad x \geq 1$



$x^* = 1$

Constraint active

Constrained Optimization

$$\min_x f(x)$$
$$s.t.\ g(x) \leq 0$$
$$h(x) = 0$$

Convex optimization if

f, g are convex

h is affine

Lagrange dual function:

$$\mathcal{L}(x, \alpha, \beta) = f(x) + \alpha g(x) + \beta h(x)$$

$\alpha \geq 0, \beta$: Lagrange multipliers

Lemma:

$$\max_{\alpha \geq 0, \beta} \mathcal{L}(x, \alpha, \beta) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ \infty & \text{otherwise} \end{cases}$$

Constrained Optimization

$$\min_x f(x)$$
$$s.t.\ g(x) \leq 0$$
$$h(x) = 0$$

$$\equiv \min_x \max_{\alpha \geq 0, \beta} \mathcal{L}(x, \alpha, \beta)$$

Lagrange dual function:

$$\mathcal{L}(x, \alpha, \beta) = f(x) + \alpha g(x) + \beta h(x)$$

$\alpha \geq 0, \beta$: Lagrange multipliers

Lemma:

$$\max_{\alpha \geq 0, \beta} \mathcal{L}(x, \alpha, \beta) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ \infty & \text{otherwise} \end{cases}$$

Primal and Dual problems

Primal problem:

$$\min_x f(x)$$
$$s.t. \; g(x) \leq 0$$
$$h(x) = 0$$

$$\equiv \min_x \max_{\alpha \geq 0, \beta} \mathcal{L}(x, \alpha, \beta)$$

Dual problem:

$$\max_{\alpha \geq 0, \beta} \min_x \mathcal{L}(x, \alpha, \beta)$$

Weak duality:

$$d^* = \max_{\alpha \geq 0, \beta} \min_x \mathcal{L}(x, \alpha, \beta) \;\leq\; \min_x \max_{\alpha \geq 0, \beta} \mathcal{L}(x, \alpha, \beta) = p^*$$

Strong Duality & KKT conditions

Strong duality:

$$d^* = \max_{\alpha \geq 0, \beta} \min_x \mathcal{L}(x, \alpha, \beta) = \min_x \max_{\alpha \geq 0, \beta} \mathcal{L}(x, \alpha, \beta) = p^*$$

Holds if primal solution x^* and dual solution (α^*, β^*) satisfy KKT (Karush-Kuhn-Tucker) conditions:

$$\nabla \mathcal{L}(x^*, \alpha^*, \beta^*) = 0$$

$$\alpha^* \geq 0$$

$$g(x^*) \leq 0$$

$$h(x^*) = 0$$

Complementary slackness $\alpha^* g(x^*) = 0$

$\alpha^* > 0 \Rightarrow g(x^*) = 0$
Constraint is active

Constraint not active
$g(x^*) < 0 \Rightarrow \alpha^* = 0$

Strong Duality & KKT conditions

Strong duality:

$$d^* = p^*$$

always →

KKT conditions hold

← for convex optimization

For constrained convex optimization, primal and dual problems are equivalent.

Dual SVM – linearly separable case

- Primal problem:

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \ \forall j$$

w - weights on features

- Dual problem:

Lagrangian dual function

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$
$$\alpha_j \geq 0, \ \ \forall j$$

α - weights on training pts

Dual SVM – linearly separable case

- Dual problem:

$$\max_{\alpha} \min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$

$$\alpha_j \geq 0, \ \forall j$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \qquad \Rightarrow \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\frac{\partial L}{\partial b} = 0 \qquad \Rightarrow \sum_j \alpha_j y_j = 0$$

Dual SVM – linearly separable case

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \mathbf{x}_i.\mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Dual problem is also QP

Solution gives α_js →

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

for any k where $\alpha_k > 0$

Use support vectors to compute b

Dual SVM – non-separable case

- Primal problem:

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_j \xi_j$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j$$
$$\xi_j \geq 0, \ \forall j$$

α_j

μ_j

Lagrange Multipliers

- Dual problem:

$$\max_{\alpha,\mu} \min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha, \mu)$$
$$s.t. \alpha_j \geq 0 \quad \forall j$$
$$\mu_j \geq 0 \quad \forall j$$

Dual SVM – non-separable case

maximize$_\alpha$ $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i . \mathbf{x}_j$

$\sum_i \alpha_i y_i = 0$

$C \geq \alpha_i \geq 0$

comes from $\frac{\partial L}{\partial \mu} = 0$

Intuition:

Earlier - If constraint violated, $\alpha_i \rightarrow \infty$

Now - If constraint violated, $\alpha_i \leq C$ (effect of a point on line (**w**) is bounded

Dual problem is also QP

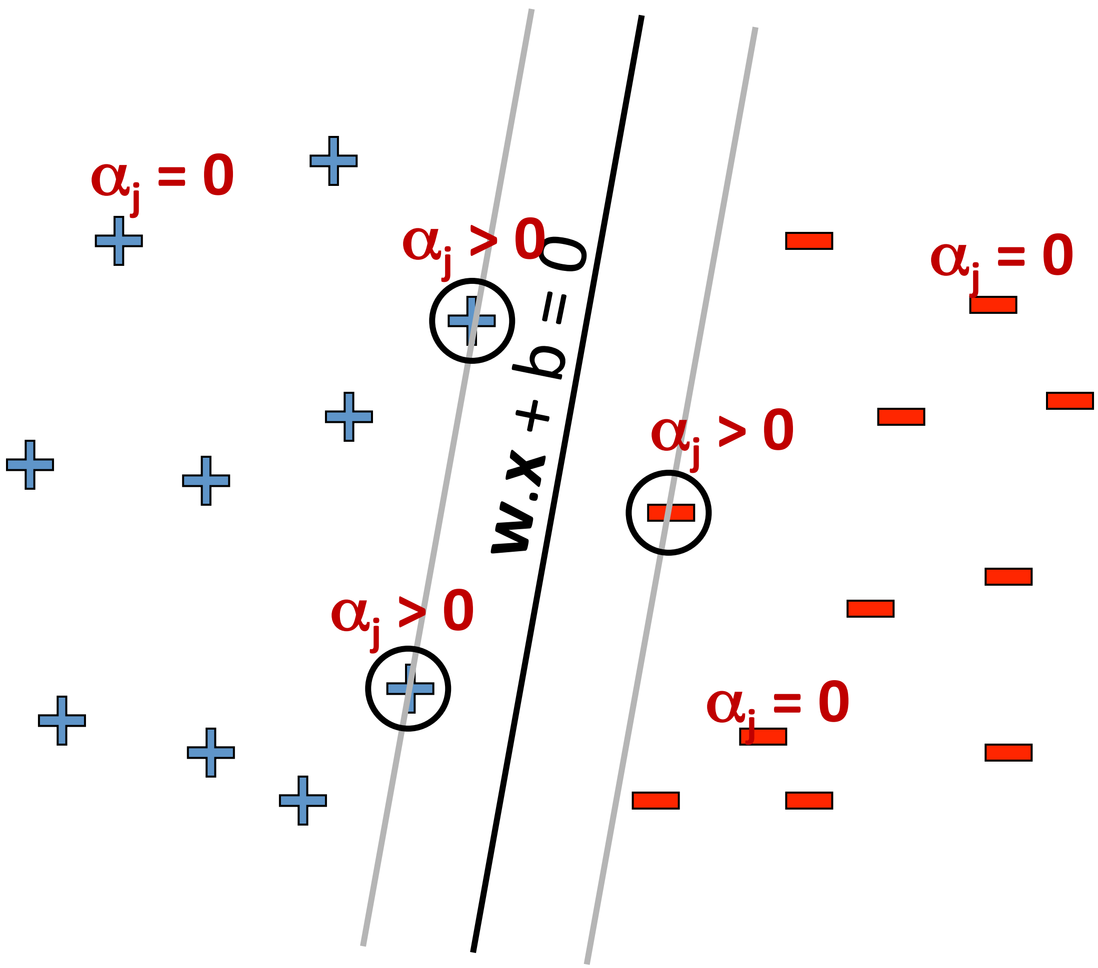
Solution gives α_js →

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

for any k where $C > \alpha_k > 0$

Dual SVM Interpretation: Sparsity



$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

Only few α_js can be non-zero : where constraint is tight

$$(\mathbf{w}.\mathbf{x}_j + b)y_j = 1$$

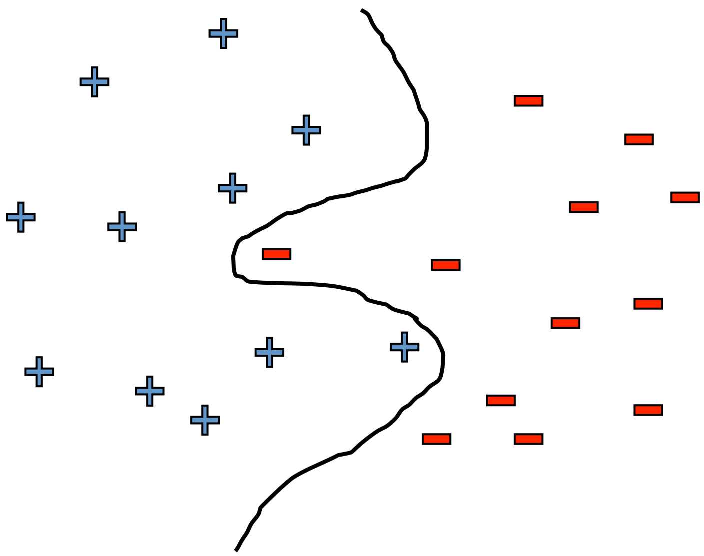
Support vectors – training points j whose α_js are non-zero

So why solve the dual SVM?

- There are some quadratic programming algorithms that can solve the dual faster than the primal, specially in high dimensions m>>n

- But, more importantly, the “**kernel trick**”!!!

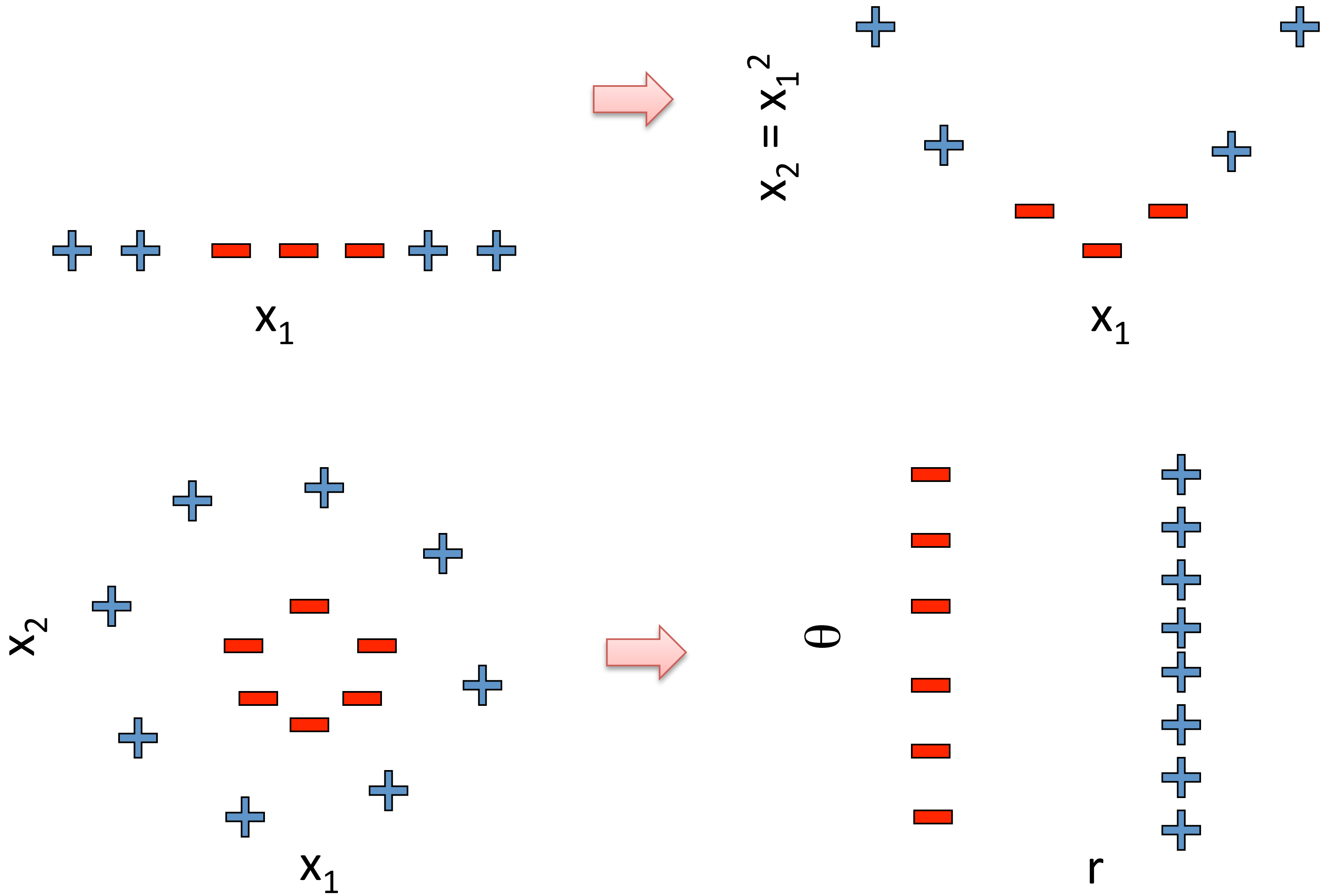
What if data is not linearly separable?



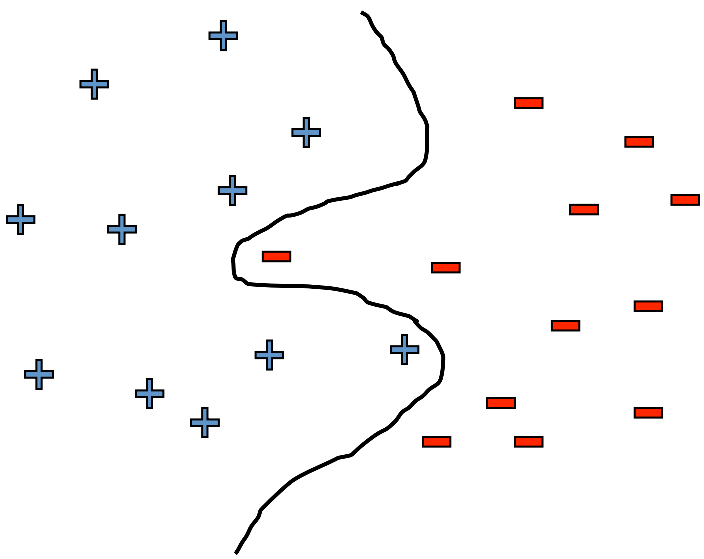
Use features of features of features of features....

$$\Phi(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, ..., \exp(x_1))$$

Non-linearly separable case



What if data is not linearly separable?



Use features of features of features of features….

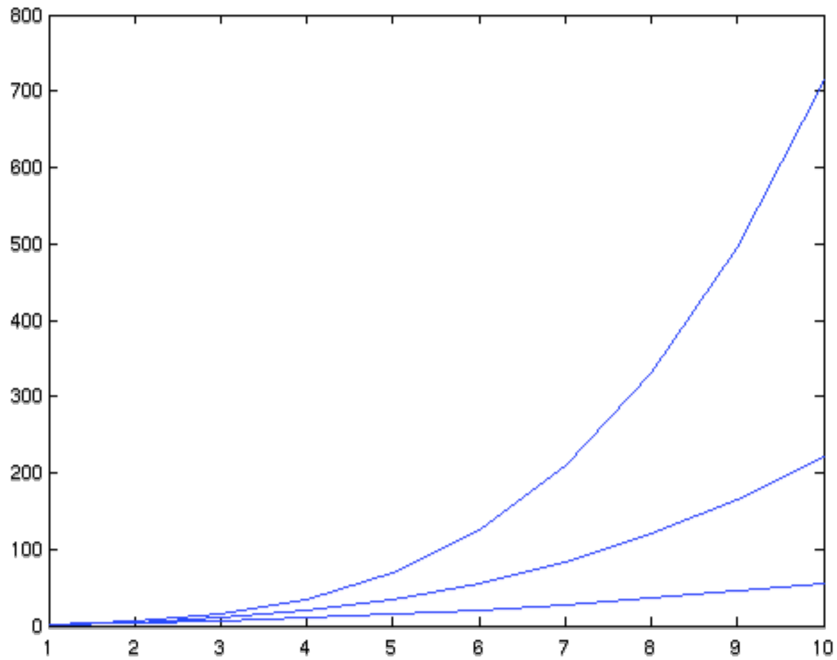
$\Phi(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, \ldots., \exp(x_1))$

Feature space becomes really large very quickly!

Higher Order Polynomials

m – input features d – degree of polynomial

$$\text{num. terms } = \binom{d+m-1}{d} = \frac{(d+m-1)!}{d!(m-1)!} \sim m^d$$



grows fast!
d = 6, m = 100
about 1.6 billion terms

Dual formulation only depends on dot-products, not on w!

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{\mathbf{x}_i . \mathbf{x}_j}$$

$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

⇩

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{K(\mathbf{x}_i, \mathbf{x}_j)}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

$\Phi(\mathbf{x})$ – High-dimensional feature space, but never need it explicitly as long as we can compute the dot product fast using some Kernel K

Common 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 – recall series expansion)

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

- Sigmoid

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

Dot Product of Polynomials

$\Phi(\mathbf{x}) = \text{polynomials of degree exactly d}$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

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

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

$$\begin{aligned} &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\ &= (x_1 z_1 + x_2 z_2)^2 \\ &= (\mathbf{x} \cdot \mathbf{z})^2 \end{aligned}$$

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

Finally: The Kernel Trick!

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

$$\mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i)$$

$$b = y_k - \mathbf{w}.\Phi(\mathbf{x}_k)$$

for any k where $C > \alpha_k > 0$

- Never represent features explicitly
  - Compute dot products in closed form
- Constant-time high-dimensional dot-products for many classes of features

- Very interesting theory – Reproducing Kernel Hilbert Spaces
  - Not covered in detail in 10701/15781, more in 10702

Overfitting

- Huge feature space with kernels, what about overfitting???
  - Maximizing margin leads to sparse set of support vectors
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting

What about classification time?

$$\mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i)$$

$$b = y_k - \mathbf{w}.\Phi(\mathbf{x}_k)$$

for any k where $C > \alpha_k > 0$

- For a new input **x**, if we need to represent Φ(**x**), we are in trouble!
- Recall classifier: sign(**w**.Φ(**x**)+b)
- Using kernels we are cool!

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

SVMs with Kernels

- Choose a set of features and kernel function
- Solve dual problem to obtain support vectors α_i
- At classification time, compute:

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)$$

$$b = y_k - \sum_i \alpha_i y_i K(\mathbf{x}_k, \mathbf{x}_i)$$

for any k where $C > \alpha_k > 0$

Classify as

$$sign\left(\mathbf{w} \cdot \Phi(\mathbf{x}) + b\right)$$

SVMs vs. Kernel Regression

SVMs

$$sign\left(\mathbf{w}\cdot\Phi(\mathbf{x})+b\right)$$

or

$$sign\left(\sum_i \alpha_i y_i K(\mathbf{x},\mathbf{x}_i)+b\right)$$

Kernel Regression

$$sign\left(\frac{\sum_i y_i K(\mathbf{x},\mathbf{x}_i)}{\sum_j K(\mathbf{x},\mathbf{x}_j)}\right)$$

Differences:

- SVMs:
  - Learn weights α_I
  - Often sparse solution
- KR:
  - Fixed "weights"
  - Solution may not be sparse
  - Much simpler to implement

SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!

Kernels in Logistic Regression

$$P(Y=1 \mid x, \mathbf{w}) = \frac{1}{1+e^{-(\mathbf{w}\cdot\Phi(\mathbf{x})+b)}}$$

- Define weights in terms of features:

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i)$$

$$\begin{aligned} P(Y=1 \mid x, \mathbf{w}) &= \frac{1}{1+e^{-(\sum_i \alpha_i \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x})+b)}} \\ &= \frac{1}{1+e^{-(\sum_i \alpha_i K(\mathbf{x},\mathbf{x}_i)+b)}} \end{aligned}$$

- Derive simple gradient descent rule on α_i

SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!
Solution sparse		
Semantics of output		

What you need to know…

- Dual SVM formulation
  - How it's derived
- The kernel trick
- Common kernels
- Differences between SVMs and kernel regression
- Differences between SVMs and logistic regression
- Kernelized logistic regression