SVMs, Duality and the Kernel Trick

Machine Learning – 10701/15781

Carlos Guestrin

Carnegie Mellon University

February 26th, 2007

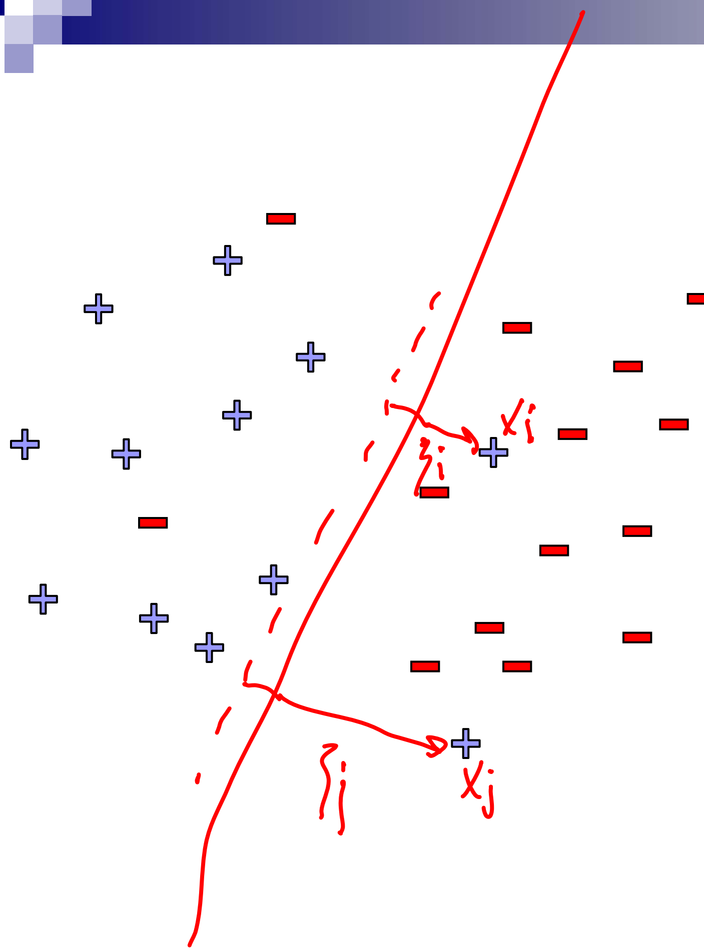
SVMs reminder

slack penalties

$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C \sum_j \xi_j$$

$$\underbrace{\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1}_{\text{each point must have margin } \geq 1} - \xi_j, \quad \forall j$$

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



Today's lecture

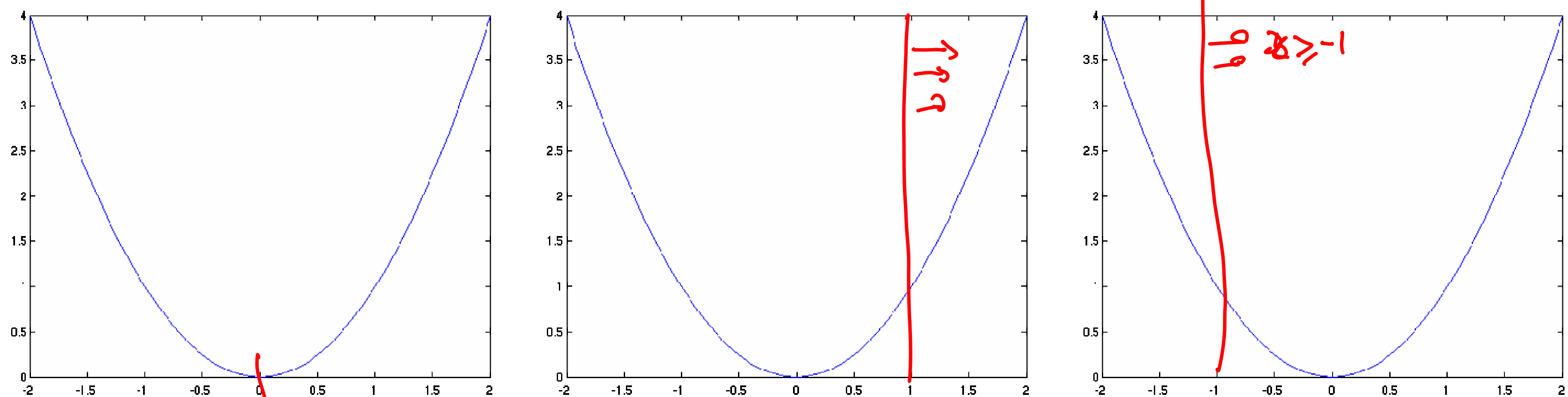
- Learn one of the most interesting and exciting recent advancements in machine learning
  - The "kernel trick"
  - High dimensional feature spaces at no extra cost!
- But first, a detour
  - Constrained optimization!

Some restrictions apply

see slides for details...

Constrained optimization

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



$b=1$

$b=-1$

suppose no constraints

min: $x=0$

$b>0$ $\quad x \geq b$

min: $x=1$

$x=b$

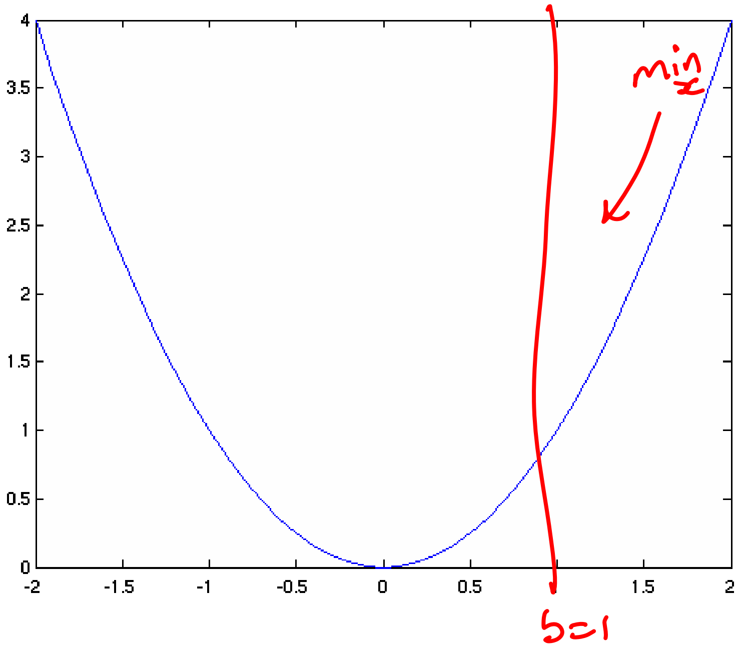
$b<0$

min: $x=0$

constraint irrelevant

Lagrange multipliers – Dual variables

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



Moving the constraint to objective function

Lagrangian:

objective

$$L(x, \alpha) = x^2 - \underbrace{\alpha(x - b)}_{\text{constraint}}$$
$$\text{s.t.} \quad \alpha \geq 0$$

suppose I pick:

$x < b$ violates the constraint — $\alpha \to +\infty \Rightarrow L \to +\infty$

$x > b$ don't violate the constraint — constraint not important — $\alpha = 0 \Rightarrow L(x, \alpha) = x^2$

$x = b$ constraint important — α anything — $L(x, \alpha) = x^2$

Solve:

$$\min_x \max_\alpha \quad L(x, \alpha)$$
$$\text{s.t.} \quad \alpha \geq 0$$

Lagrange multipliers – Dual variables

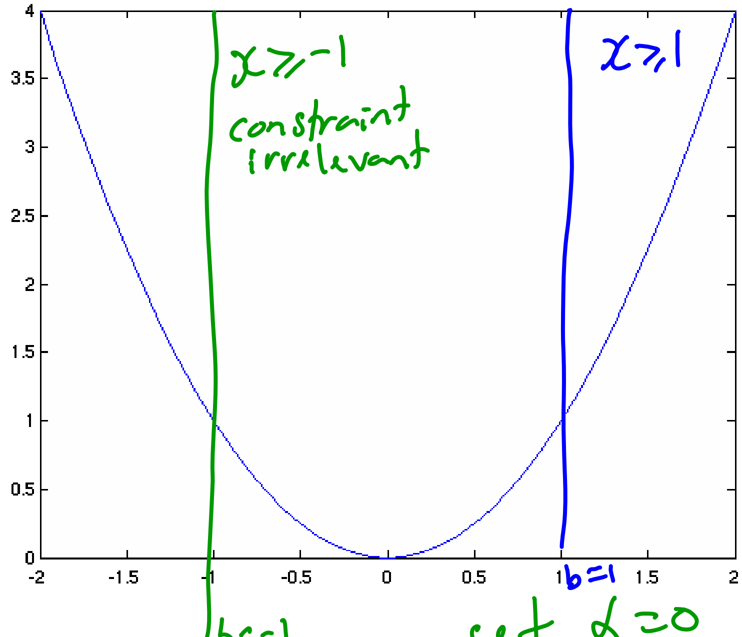
$L(x,\alpha)=$

Solving: $\min_x \max_\alpha \; x^2 - \alpha(x-b)$

s.t. $\alpha \geq 0$

$\frac{\partial L}{\partial x} = 2x - \alpha = 0$ $\qquad \alpha = 2x$!!

$\frac{\partial L}{\partial \alpha} = -(x-b)$



set $\alpha = 0$
plays no
role in
Lagrangian

no way $\frac{\partial L}{\partial \alpha} = 0$
unless $x = -1$
but $x=-1$, $\alpha < 0$ unhappy!!

$\frac{\partial L}{\partial \alpha} = 0 \Rightarrow x = 1$

$\frac{\partial L}{\partial x} = 0 \Rightarrow \alpha = 2 \Rightarrow \alpha > 0$

OK!

constraint plays role $\Rightarrow \alpha > 0$

Dual SVM derivation (1) – the linearly separable case

equations look nicer

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j \in \text{Dataset}$$

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

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

$$\frac{\partial L}{\partial w} = w - \sum_j \alpha_j x_j y_j = 0$$

$$\Rightarrow \ \overbrace{w = \sum_j \alpha_j x_j y_j} \ \Rightarrow$$

give me α's
give you w's

x_j is d-dim
vector for d features for point j

Dual SVM derivation (2) – the linearly separable case

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

constraint for point j is irrelevant: $\alpha_j = 0$
$\Rightarrow (w.x_j + b) y_j > 1$

constraint is relevant: $\alpha_j > 0$
$(w.x_j + b) y_j = 1$

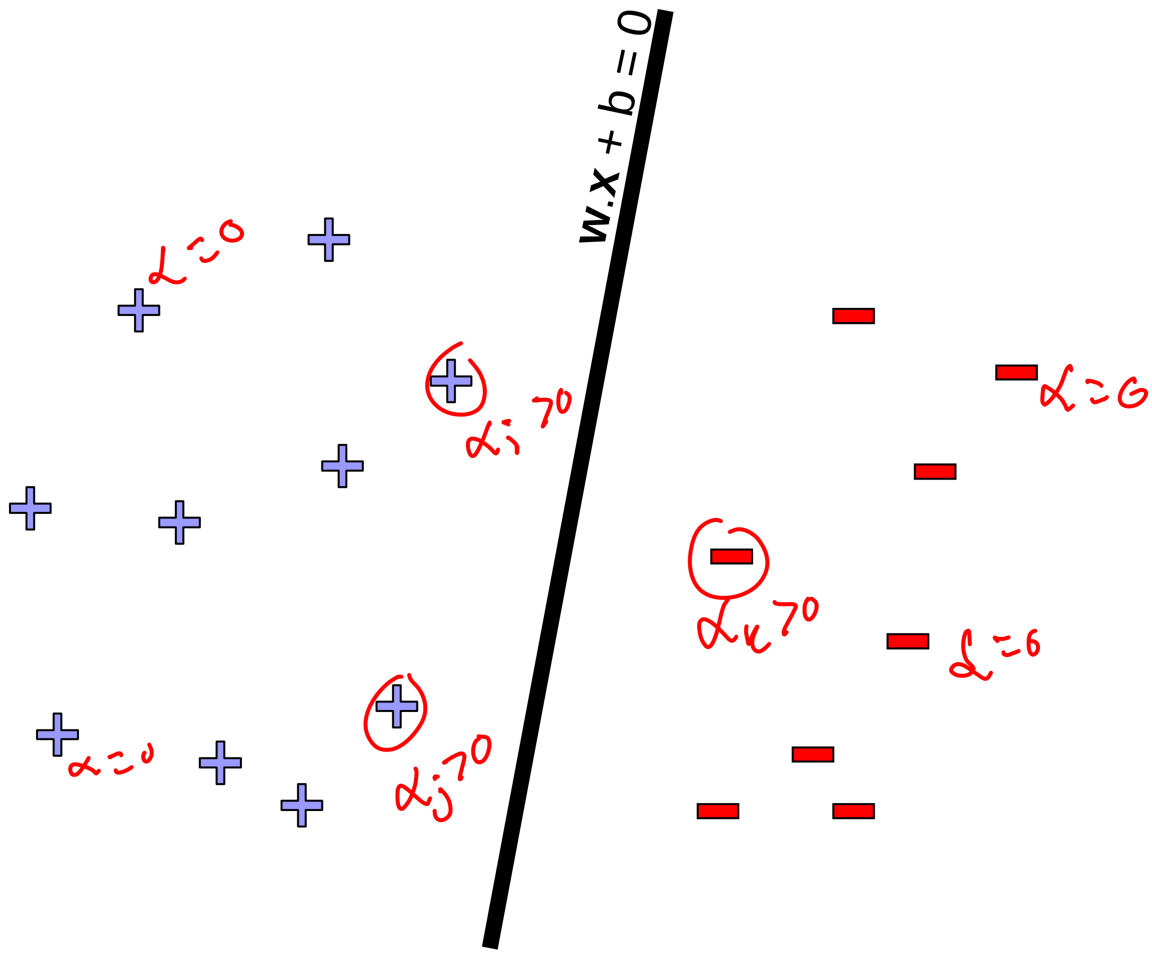
Finding b: find any point where $\alpha_j > 0$
and $b.y_j = 1 - w x_j y_j$

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

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

$$b = y_k - \mathbf{w}.\mathbf{x}_k$$
for any k where $\alpha_k > 0$

Dual SVM interpretation



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

$$w = \sum_{\substack{j:\alpha_j>0 \\ \text{or} \\ j \in \text{Support vectors}}} \alpha_j y_j x_j$$

weight are linear combination of feature values of support vectors

Dual SVM formulation – the 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 svm

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

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

for any k where $\alpha_k > 0$

solve this ⇒ get α's

⇒ get w and b

⇒ solved the SVM problem

Solve by Quadratic Programming ...

Dual SVM derivation – the non-separable case

slack penalty

slack variables

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_j \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j \quad \leftarrow \alpha_j$$

$$\xi_j \geq 0, \ \forall j \quad \leftarrow \mu_j$$

$$L(w,\alpha,\mu) = \frac{1}{2} ww + C\sum_j \xi_j - \sum_j \alpha_j\left[(wx_j + b)y_j - 1 + \xi_j\right] - \sum_j \mu_j\left[\xi_j - 0\right]$$

$\vdots$

Dual SVM formulation – the non-separable case

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

$$\sum_i \alpha_i y_i = 0$$

$$\underline{C \geq} \alpha_i \geq 0 \quad \forall_j$$

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

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

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

compared to separable case: only difference

$\alpha_i \leq C \quad \forall i$

intuitively, don't give me alphas that are too large

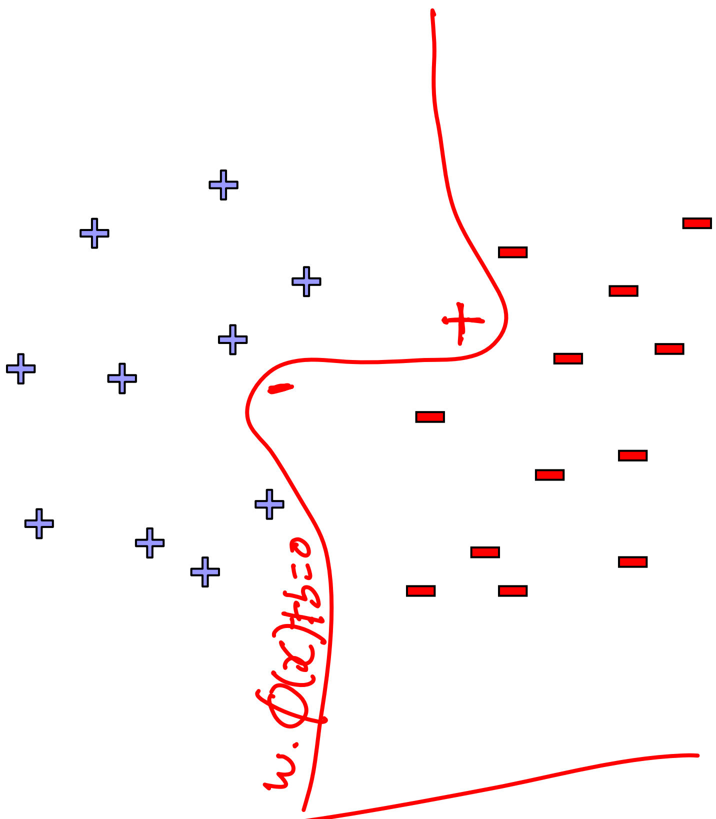
Announcements

- Class projects out later this week

Why did we learn about the dual SVM?

- There are some quadratic programming algorithms that can solve the dual faster than the primal
- But, more importantly, the "**kernel trick**"!!!
  - Another little detour…

Reminder from last time: What if the data is not linearly separable?



find w : $\dim(w) = d$

Use features of features of features of features….

$$x = \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{2}$$

$$\Phi(\mathbf{x}) : R^m \mapsto F$$

$$\phi(x) = \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ \sin x_1 \\ e^{-x_1/x_2} \\ \vdots \end{pmatrix}}_{d}$$

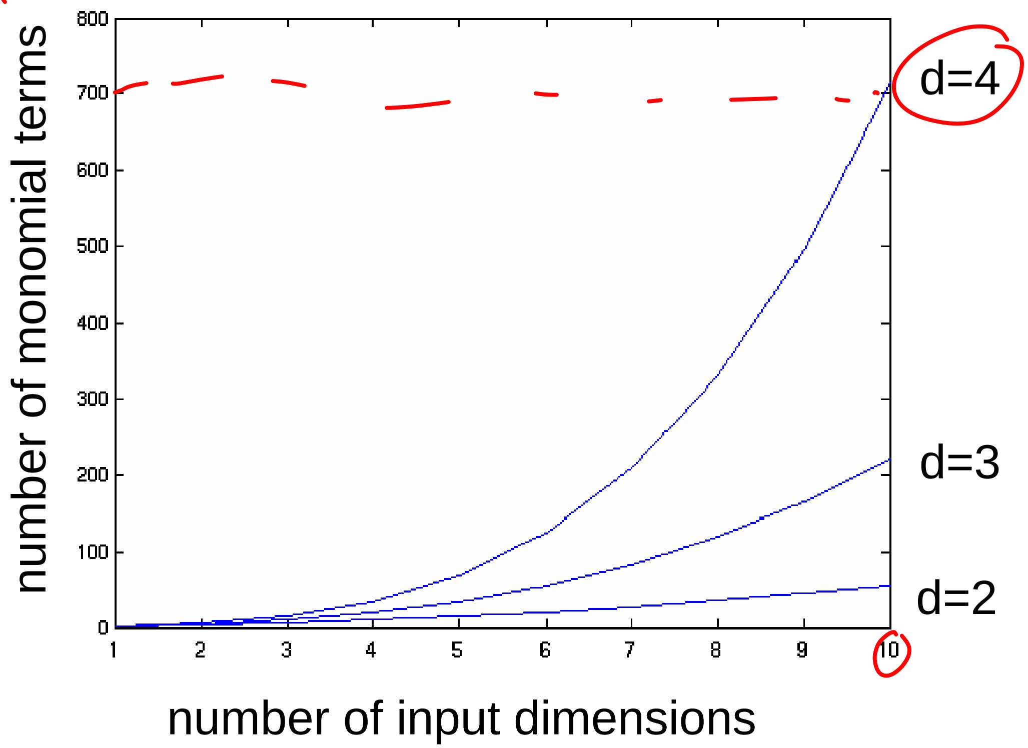
mapping to high dim features

Feature space can get really large really quickly!

Higher order polynomials

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

$|\phi(x)|$



m – input features

d – degree of polynomial

$d=2$
$m=2$

$x = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$

$\phi(x) = \begin{pmatrix} v_1 \\ v_2 \\ v_1 v_2 \\ v_1^2 \\ v_2^2 \end{pmatrix}$

grows fast!

d = 6, m = 100

about 1.6 billion terms

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

$X_i = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{pmatrix}$ $X_j = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_d \end{pmatrix}$

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

$X_i \cdot X_j$

$= v_1\mu_1 + v_2\mu_2 + \cdots + v_d\mu_d$

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

$\phi(X) = \begin{pmatrix} v_1 \\ v_2 \\ v_1 v_2 \\ v_1^2 \\ v_2^2 \\ \vdots \end{pmatrix}$

$\phi(x_i) \cdot \phi(x_j) = K(x_i, x_j)$

how many terms?

m^2

$m \leftarrow$ # data points

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

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

Dot-product of polynomials

$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ $v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \text{polynomials of degree } \underline{\text{d}}$$

$d=1 \quad \phi(u) = u$

$\phi(u) \cdot \phi(v) = u \cdot v = u_1 v_1 + u_2 v_2$

$d=2 \quad \phi(u) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix}$

$\phi(u) \cdot \phi(v) =$

$u_1^2 \cdot v_1^2 + 2 u_1 u_2 \cdot v_1 v_2 + u_2^2 \cdot v_2^2$

$= (u_1 v_1 + u_2 v_2)^2 = (u \cdot v)^2$

degree $= d$ $\quad K(u,v) = \phi(u) \cdot \phi(v) = (u \cdot v)^d$

$\equiv O(d)$ multiplications

Finally: the "kernel trick"!

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

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

$$\sum_i \alpha_i y_i = 0$$

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

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

d : $O(d)$

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

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

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

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

Polynomial kernels

- All monomials of degree d in O(d) operations:

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d = \text{polynomials of degree d}$$

- How about all monomials of degree up to d?
  - Solution 0: $\sum_{i=1}^{d} (u \cdot v)^i \equiv O(d^2)$
  - Better solution: $(u \cdot v + 1)^2 = \underbrace{(u \cdot v)^2 + 2\,uv + 1}_{\text{got polynomials up to } d=2}$

poly degree up to d $\equiv \phi(u) \cdot \phi(v) = (u \cdot v + 1)^d \equiv O(d)$

Common kernels

- Polynomials of degree d

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

- Polynomials of degree up to d

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

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

- Gaussian kernels

Gaussian kernels

equivalent to $\phi(u)$ has infinite dimensionality

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

- Sigmoid

Sigmoid :

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

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 at classification time

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

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

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

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

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

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

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

What's the difference between SVMs and Logistic Regression?

	SVMs	Logistic Regression
Loss function		
High dimensional features with kernels		

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 support vectors:

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

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

- Derive simple gradient descent rule on α_i

What's the difference between SVMs and Logistic Regression? (Revisited)

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

What you need to know

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