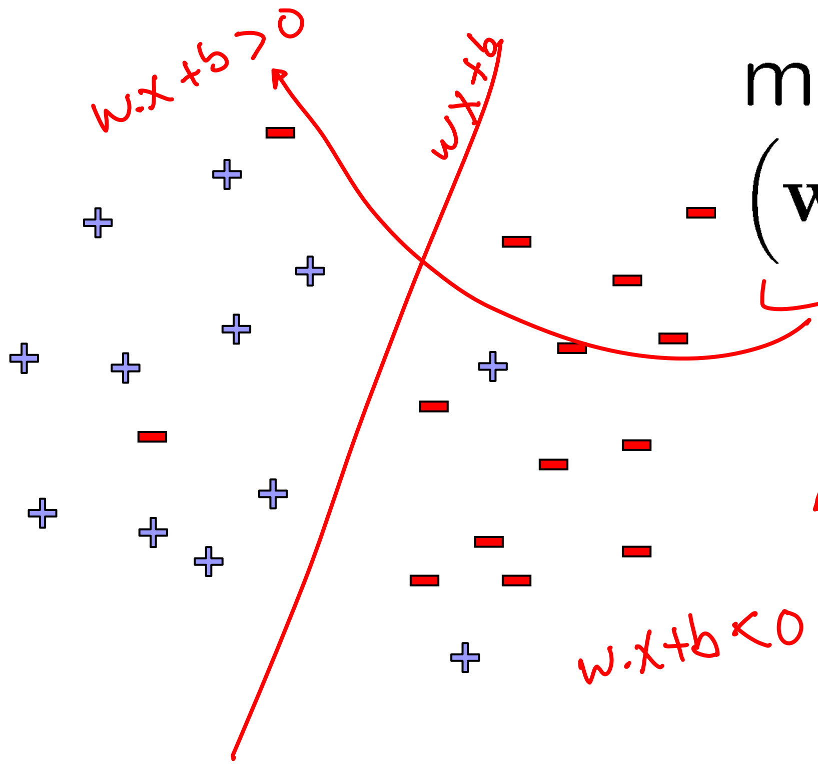
SVMs, Duality and the Kernel Trick

Machine Learning – 10701/15781

Carlos Guestrin
Carnegie Mellon University

February 21st, 2005

SVMs reminder



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

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

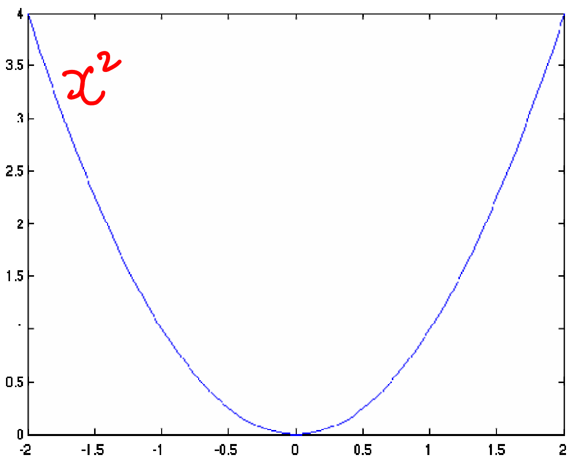
$$\xi_j \geq 0, \ \forall j$$

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!

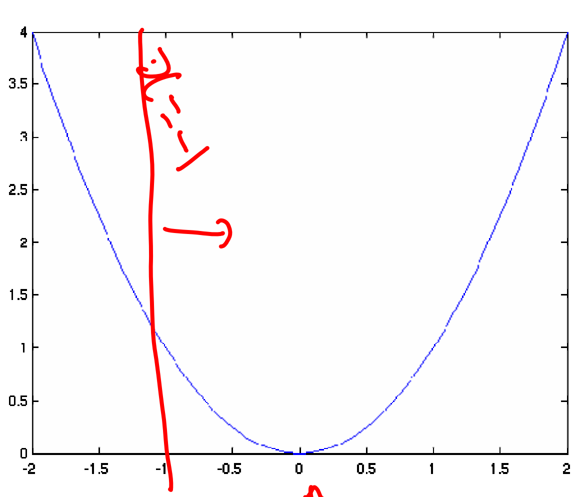
Constrained optimization

$$\min x^2$$
$$x \geq b$$



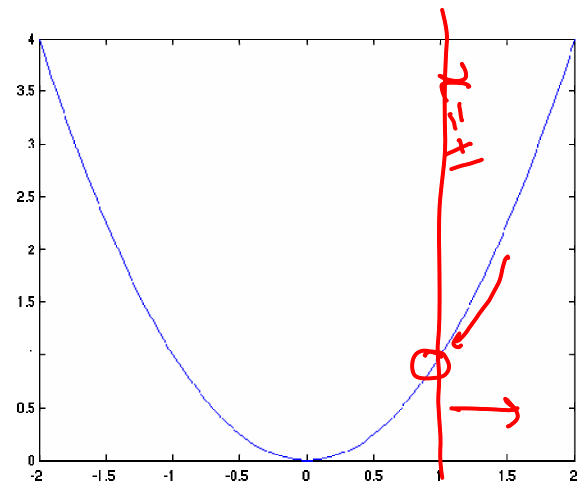
$x=0$

$$\frac{d x^2}{dx} = 2x = 0$$



$b=-1$

$x=0$



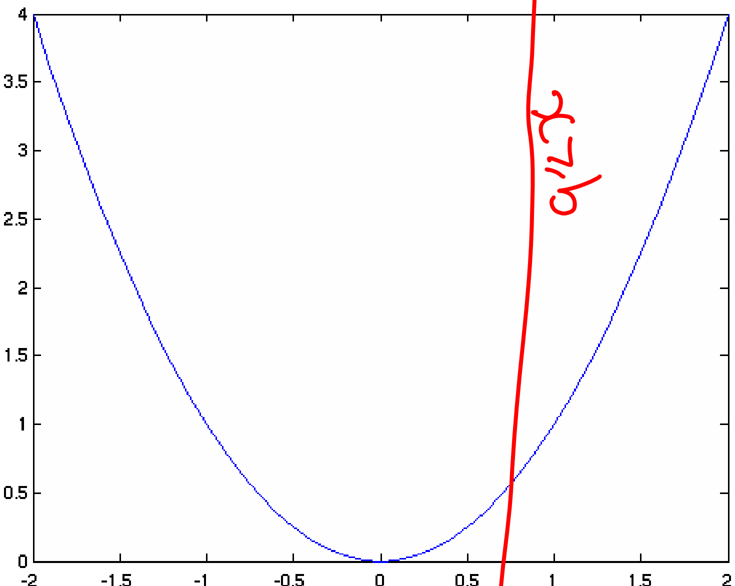
$b=+1$

$$\min x^2$$
$$x \geq +1$$

$$\boxed{x=1}$$

$\rightarrow \alpha$ Formulation

Lagrange multipliers – Dual variables



$\frac{\partial L}{\partial \alpha}$

$\min x^2$

$x \geq b$

$L(x,\alpha) = x^2 - \alpha(x-b)$

$\alpha \geq 0$

if $x < b$

$x-b \rightarrow$ negative

pay $-\alpha(x-b)$

if $x \geq b$

$x-b$ is positive

set $\alpha = 0$

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

$\frac{\partial L}{\partial \alpha}\left(x=\frac{\alpha}{2}, \alpha\right) = \frac{\partial}{\partial \alpha}\left[\left(\frac{\alpha}{2}\right)^2 - \alpha\left(\frac{\alpha}{2} - b\right)\right]$

$= \frac{\partial}{\partial \alpha}\left(\alpha b - \frac{\alpha^2}{4}\right) = b - \frac{\alpha}{2} = 0 \Rightarrow \boxed{\alpha = 2b}$

$\alpha \geq 0$

if $b > 0$, the $\alpha = 2b$

$\Rightarrow \underline{x = b}$

if $b < 0$, then $\alpha = 2b < 0$

$\alpha \geq 0 \Rightarrow \alpha = 0 \quad x = 0$

Dual SVM derivation (1) – the linearly separable case

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

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

data points

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

weights

data points

$$\frac{\partial L}{\partial w} = w - \sum_j \alpha_j x_j . y_j = 0 \Rightarrow w = \sum_j \alpha_j \overbrace{y_j x_j}^{\text{data points}}$$

Dual SVM derivation (2) – the linearly separable case

where does b come from

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

when $\alpha_j = 0$, I don't care about the constraint!

Take a constraint j (or a training example) where $\alpha_j > 0 \Rightarrow$ point j is close to hyperplane.

$(w.x_j + b).y_j = 1$

$b = \frac{1}{y_j} - w.x_j$, notice $\frac{1}{y_j} = y_j$, $y_j = \{-1, +1\}$

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

minimize$_{\mathbf{w}}$ $\frac{1}{2}\mathbf{w}.\mathbf{w}$

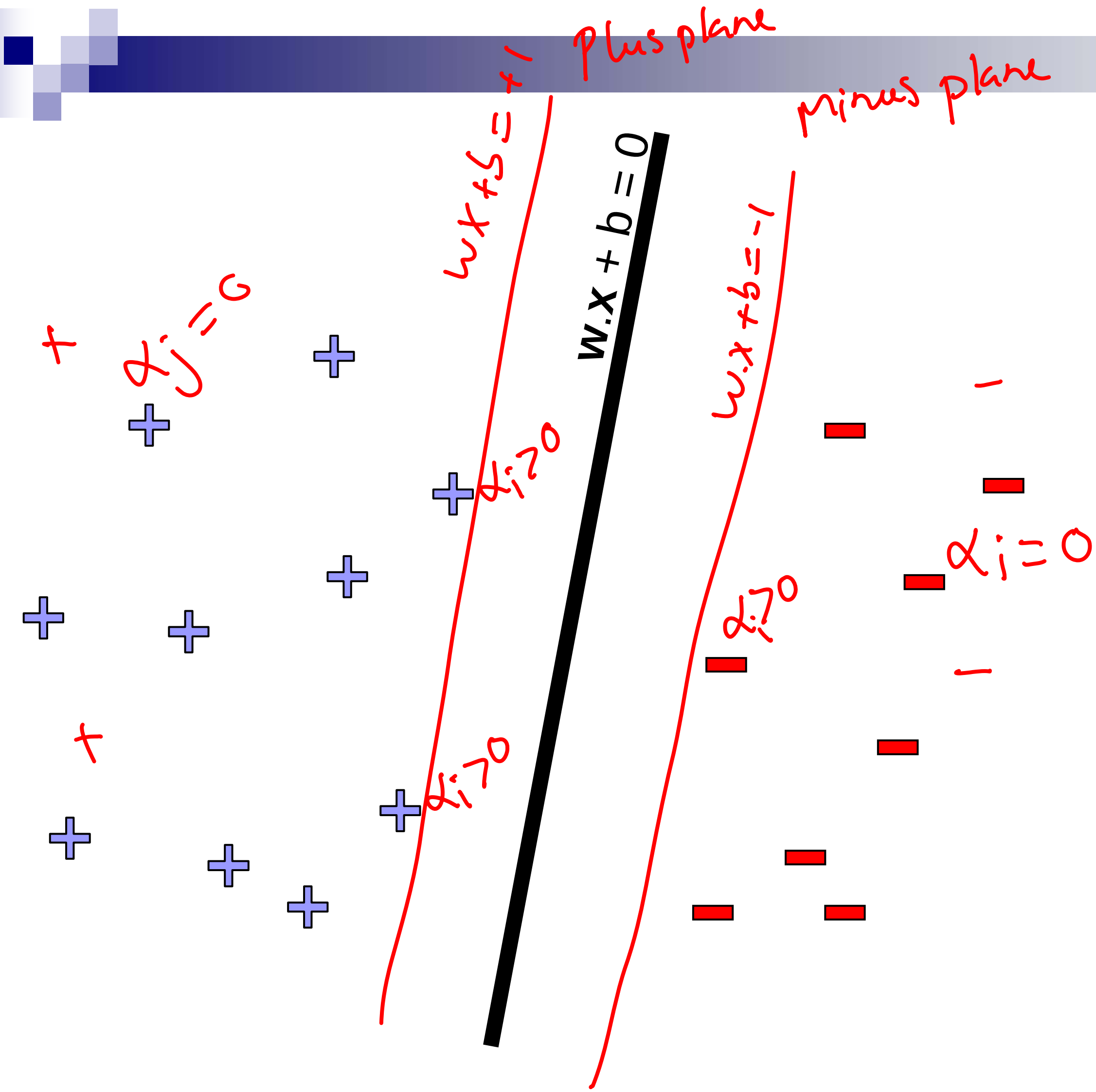
$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j$

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

for any k where $\alpha_k > 0$

Average b over all points where $\alpha_k > 0$

Dual SVM interpretation



weights

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

w is going to be a weighted combination of points near the plane.

Dual SVM formulation – the linearly separable case

quadratic terms

$$\text{minimize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2} \underbrace{\sum_{i,j}}_{\sum_{i=1}^{N} \sum_{j=1}^{N}} \overbrace{\alpha_i \alpha_j}^{} y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

if I solve this Dual problem

I get α_j, plug ⟶

$$\boxed{\begin{array}{l} \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \\ b = y_k - \mathbf{w}.\mathbf{x}_k \\ \text{for any } k \text{ where } \alpha_k > 0 \end{array}}$$

Dual SVM derivation – the non-separable case

$$\text{minimize}_{\mathbf{w}} \quad \tfrac{1}{2}\mathbf{w}.\mathbf{w} + \underbrace{C\sum_j \xi_j}_{\text{slack penalty}}$$

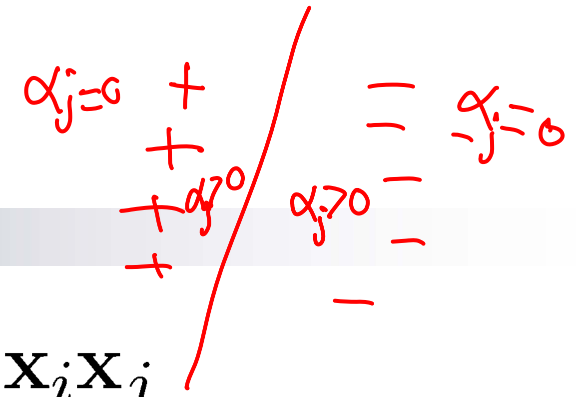
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j$$

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

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

$$+ \sum_j \mu_j \cdot (\xi_j - 0)$$

Dual SVM formulation – the non-separable case



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

$$\sum_i \alpha_i y_i = 0$$

$$\underbrace{C \geq}_{\text{this is the only difference with the non-separable case}} \alpha_i \geq 0$$

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

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

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

use Average for numerical stability

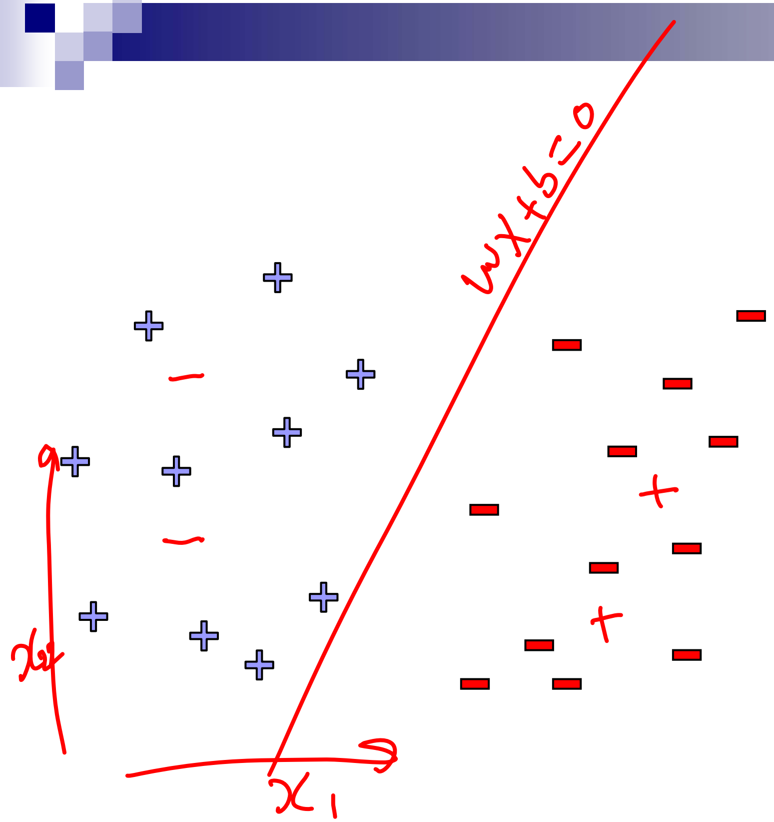
before $\alpha_k > 0$

now choose $C > \alpha_k > 0$

Why did we learn about the dual SVM?

- There are some quadratic programming algorithms that can solve the $\underbrace{\text{dual}}_{\text{version with } \alpha}$ faster than the $\underbrace{\text{primal}}_{\text{version with } w, b}$
- But, more importantly, the "**kernel trick**"!!!
  - Another little detour…

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



Use features of features of features of features….

$$\Phi(\mathbf{x}) : R^m \mapsto F \leftarrow \text{higher dim. space}$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leftarrow \mathbb{R}^2$$

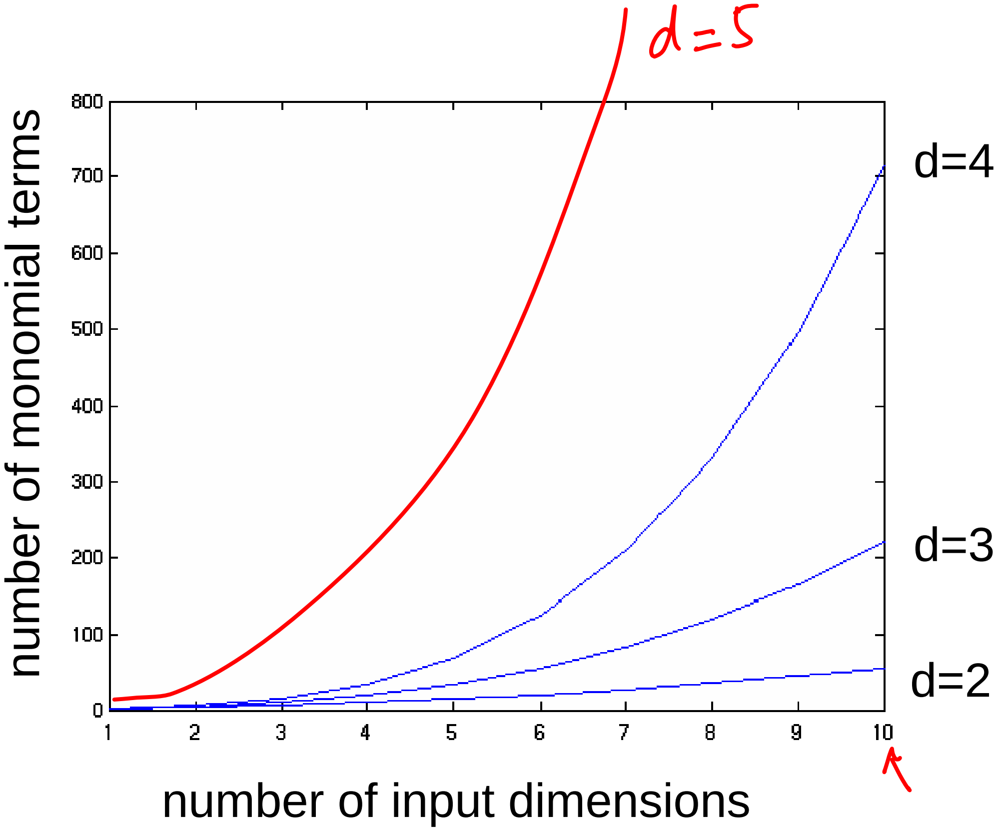
$$\phi(x) = \begin{pmatrix} x_1 \\ x_1^2 \\ x_1 x_2 \\ x_2 \\ x_2^2 \end{pmatrix} \leftarrow \mathbb{R}^5$$

Feature space can get really large really quickly!

Higher order polynomials

polynomial degree   input dim.

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



m – input features
d – degree of polynomial

if you give me x and I have to write down $\Phi(x)$, vector is too big

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

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

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

input only shows up here

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

dot-product $x_i \cdot x_j$

If I use $\phi(x)$ as my input

only care about $\phi(x_i) \cdot \phi(x_j) = k(x_i, x_j)$

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

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

Dot-product of polynomials

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

degree $=1$ $\quad u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \quad v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \quad u \cdot v = u_1 v_1 + u_2 v_2$

degree $2 = \phi(u) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix} \quad \phi(v) = \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2 v_1 \\ v_2^2 \end{pmatrix}$

12 multiplies

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

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

3 multiplies

Degree d polynomials $\quad \phi(u) \cdot \phi(v) = (u \cdot v)^d$

Finally: the "kernel trick"!

for each x_i, x_j
compute $k(x_i, x_j)$

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

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

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

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

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

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: $\phi(u) \cdot \phi(v) = \sum_{i=0}^{d} (u \cdot v)^i$
  - Better solution: $(u \cdot v + 1)^2 = (u \cdot v)^2 + u \cdot v + v \cdot u + 1$

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

Common kernels

- Polynomials of degree d

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

- Polynomials of degree up to d (and including)

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

- Gaussian kernels

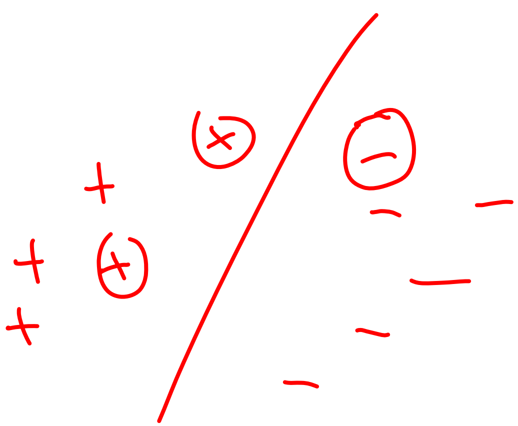
$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||}{2\sigma^2}\right)$$

- 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 Φ(**x**), we are in trouble!
- Recall classifier: sign(**w**.Φ(**x**)+b)
- Using kernels we are cool!

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

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

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

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

$w \cdot \phi(x) + b$ plug in

$$= \sum_i \alpha_i y_i \underbrace{\phi(x_i)\cdot\phi(x)}_{\text{dot product of new instance } X \text{ with old data}} + b$$

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	Hinge loss	Log-loss
High dimensional features with kernels	**Yes!**	~~No~~ yes!

Kernels in logistic regression

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

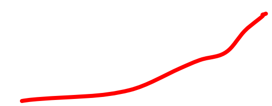
- Define weights in terms of support vectors:

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

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

- 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  doesn't care about correct examples	Log-loss  cares about all examples
High dimensional features with kernels	Yes!	Yes!
Solution sparse	Often yes! many $\alpha_j = 0$	Almost always no! all $\alpha_j \neq 0$

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

Acknowledgment

- SVM applet:
  - http://www.site.uottawa.ca/~gcaron/applets.htm