Support vector machines (SVMs)
Lecture 2

David Sontag

New York University

Slides adapted from Luke Zettlemoyer, Vibhav Gogate, and Carlos Guestrin

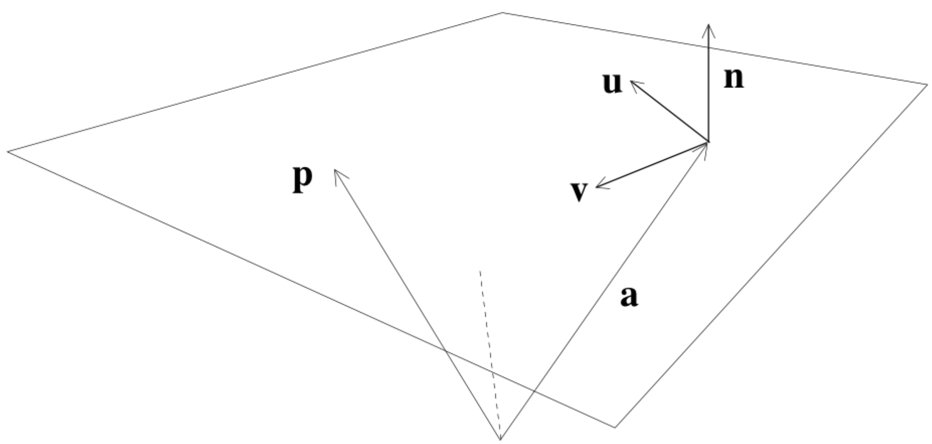
Geometry of linear separators (see blackboard)

A plane can be specified as the set of all points given by:

$$\mathbf{p} = \mathbf{a} + s\mathbf{u} + t\mathbf{v}, \qquad (s,t) \in \mathcal{R}.$$

Vector from origin to a point in the plane

Two non-parallel directions in the plane



Alternatively, it can be specified as:

$$(\mathbf{p} - \mathbf{a}) \cdot \mathbf{n} = 0 \Leftrightarrow \mathbf{p} \cdot \mathbf{n} = \mathbf{a} \cdot \mathbf{n}$$

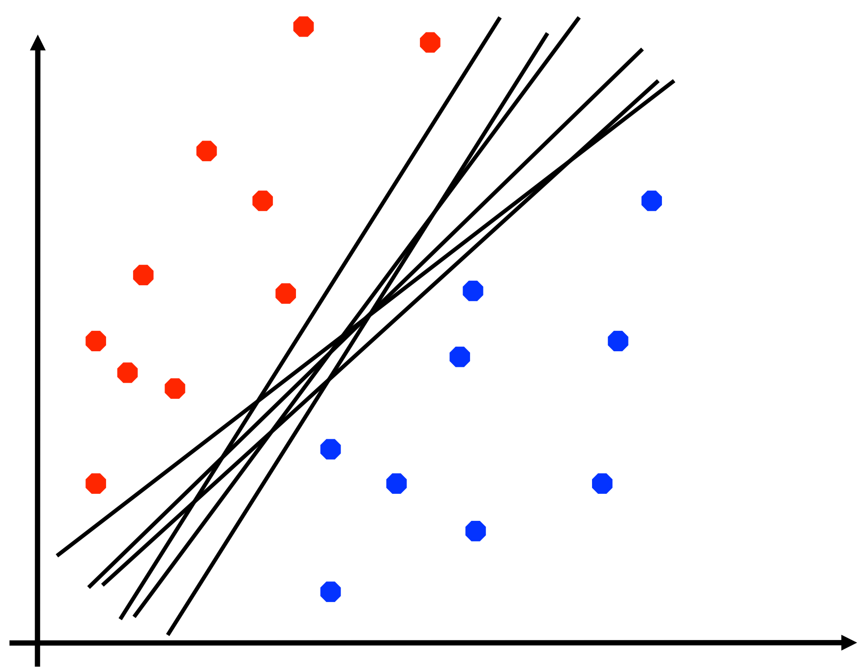
Normal vector
(we will call this w)

Only need to specify this dot product,
a scalar (we will call this the offset, b)

Barber, Section 29.1.1-4

Linear Separators

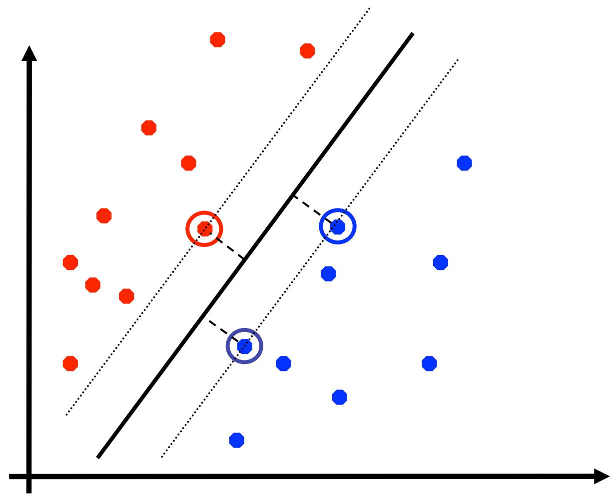
- If training data is linearly separable, perceptron is guaranteed to find *some* linear separator
- Which of these is **optimal**?



Support Vector Machine (SVM)

- SVMs (Vapnik, 1990's) choose the linear separator with the **largest margin**

Robust to outliers!



V. Vapnik

- Good according to intuition, theory, practice

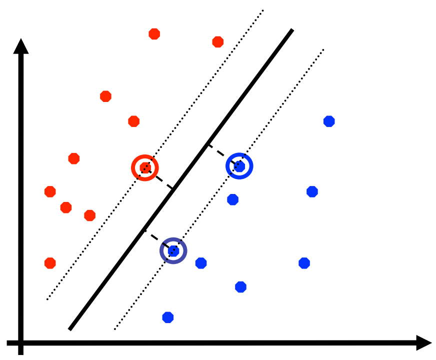
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task

Support vector machines: 3 key ideas

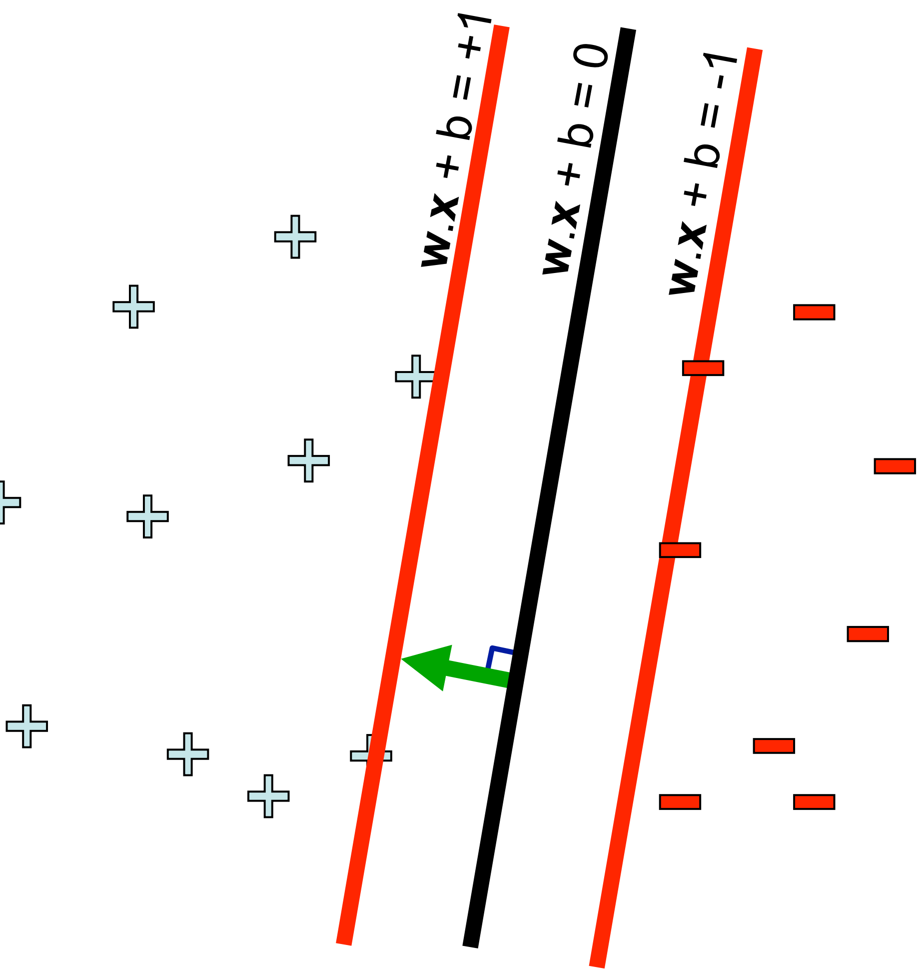
1. Use **optimization** to find solution (i.e. a hyperplane) with few errors

2. Seek **large margin** separator to improve generalization

3. Use **kernel trick** to make large feature spaces computationally efficient



Finding a perfect classifier (when one exists) using linear programming



For every data point (x_t, y_t), enforce the constraint

for y_t = +1, $w \cdot x_t + b \geq 1$

and for y_t = -1, $w \cdot x_t + b \leq -1$

Equivalently, we want to satisfy all of the linear constraints

$$y_t \left(w \cdot x_t + b \right) \geq 1 \quad \forall t$$

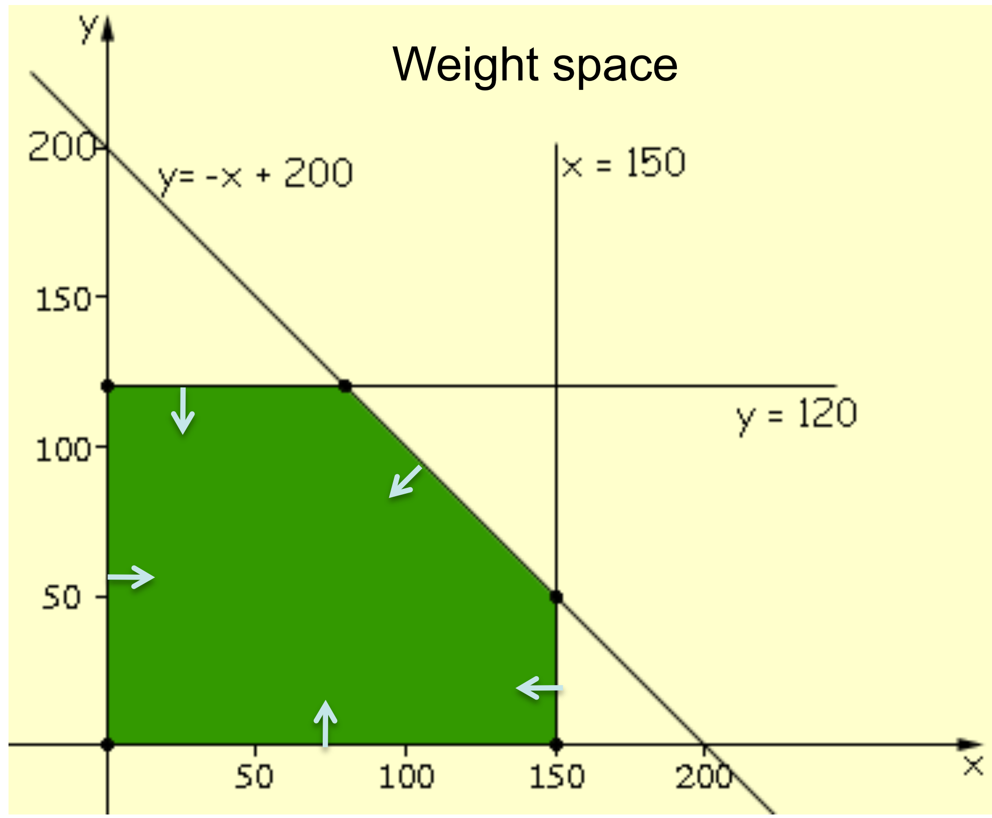
This *linear program* can be efficiently solved using algorithms such as simplex, interior point, or ellipsoid

Finding a perfect classifier (when one exists) using linear programming

Example of 2-dimensional linear programming (feasibility) problem:

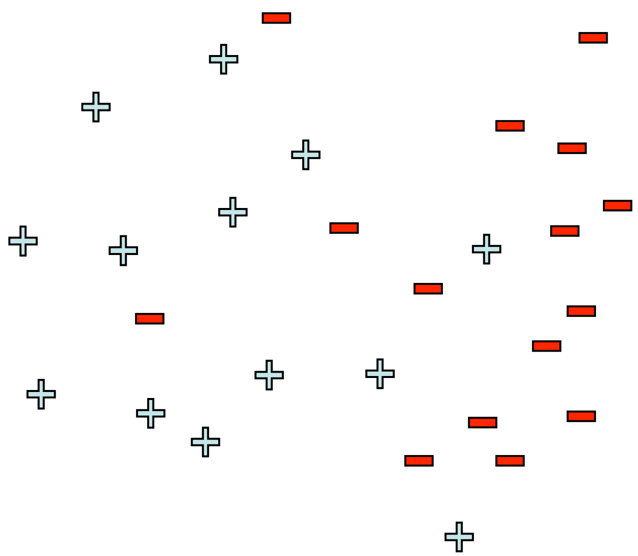
For SVMs, each data point gives one inequality:

$$y_t \left(w \cdot x_t + b \right) \geq 1$$



What happens if the data set is not linearly separable?

Minimizing number of errors (0-1 loss)



- Try to find weights that violate as few constraints as possible?

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

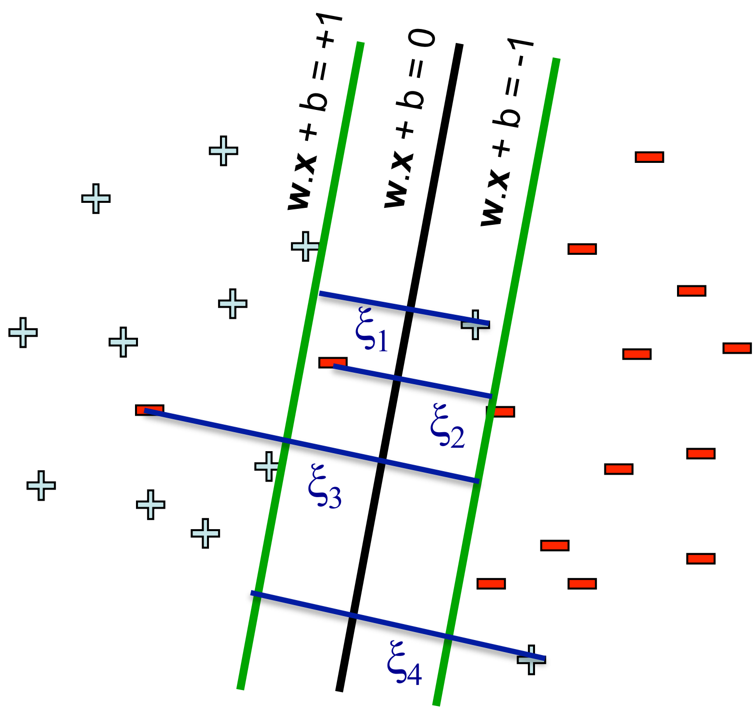
- Formalize this using the 0-1 loss:

$$\min_{\mathbf{w},b} \sum_j \ell_{0,1}(y_j,\, w \cdot x_j + b)$$

where $\ell_{0,1}(y, \hat{y}) = 1[y \neq \text{sign}(\hat{y})]$

- Unfortunately, minimizing 0-1 loss is NP-hard in the worst-case
  - Non-starter. We need another approach.

Key idea #1: Allow for *slack*



$$\text{minimize}_{\mathbf{w},b,\xi} \quad \Sigma_j \, \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad ,\forall j \quad \xi_j \geq 0$$

"slack variables"

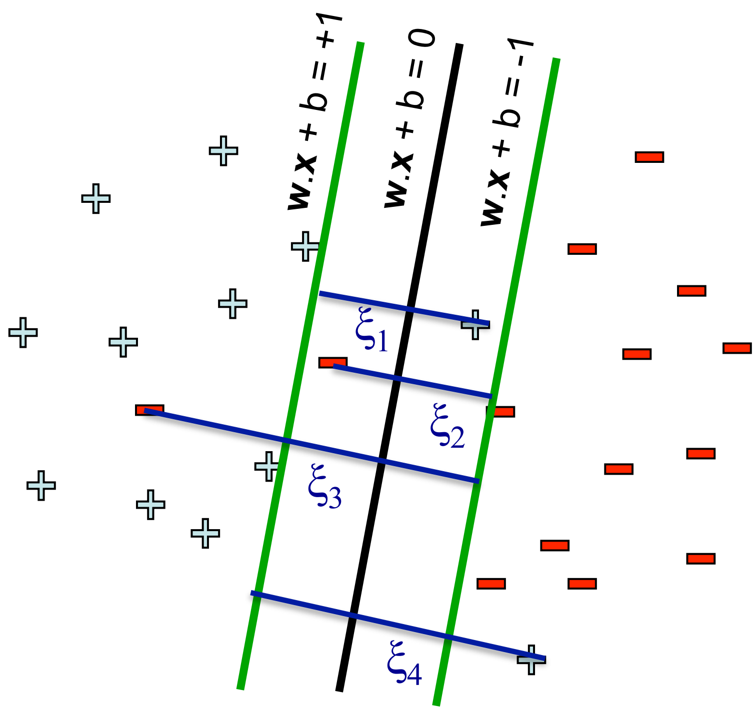
We now have a linear program again, and can efficiently find its optimum

For each data point:

•If functional margin ≥ 1, don't care

•If functional margin < 1, pay linear penalty

Key idea #1: Allow for *slack*



$$\text{minimize}_{\mathbf{w},b,\xi} \quad \Sigma_j \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

"slack variables"

What is the optimal value ξ_j^* as a function of **w*** and b*?

If $(w \cdot x_j + b)\, y_j \geq 1$, then $\xi_j = 0$

If $(w \cdot x_j + b)\, y_j < 1$, then $\xi_j = 1 - (w \cdot x_j + b)\, y_j$

Sometimes written as

$$\left(1 - (w \cdot x_j + b)\, y_j\right)_+ \longleftarrow \xi_j = \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$$

Equivalent hinge loss formulation

$$\text{minimize}_{\mathbf{w},b,\xi} \quad \Sigma_j \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

Substituting $\xi_j = \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$ into the objective, we get:

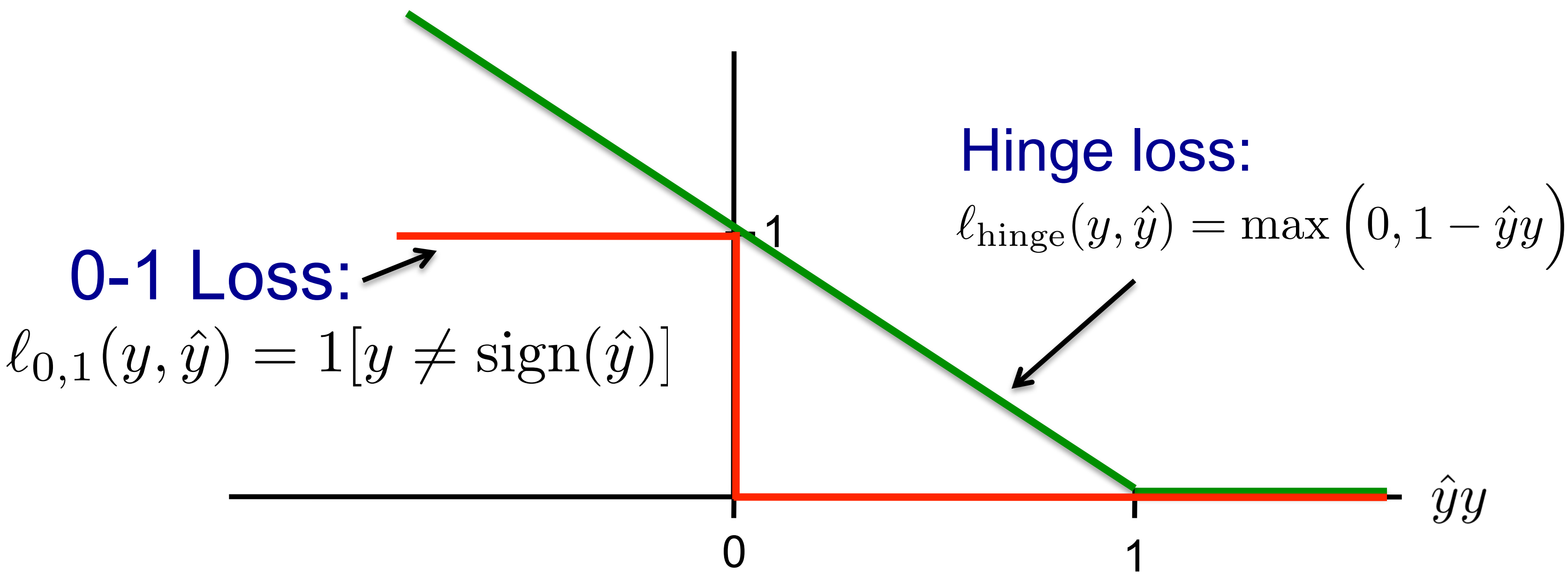
$$\boxed{\min_{w,b} \sum_j \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)}$$

The **hinge loss** is defined as $\ell_{\text{hinge}}(y, \hat{y}) = \max\left(0, 1 - \hat{y}y\right)$

$$\min_{\mathbf{w},b} \sum_j \ell_{\text{hinge}}(y_j,\; w \cdot x_j + b)$$

This is empirical risk minimization,
using the hinge loss

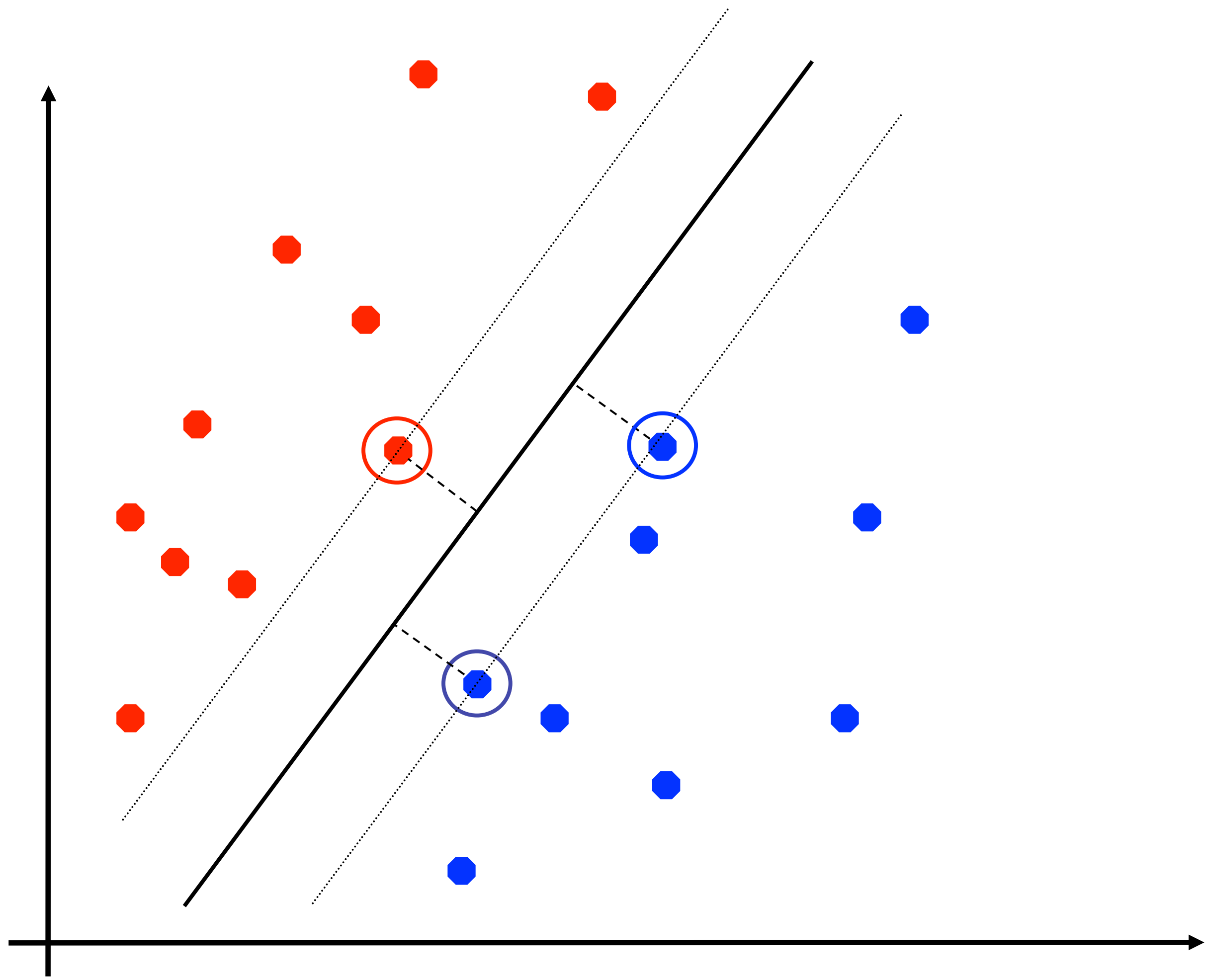
Hinge loss vs. 0/1 loss



Hinge loss upper bounds 0/1 loss!

➡ It is the tightest *convex* upper bound on the 0/1 loss

Key idea #2: seek large margin

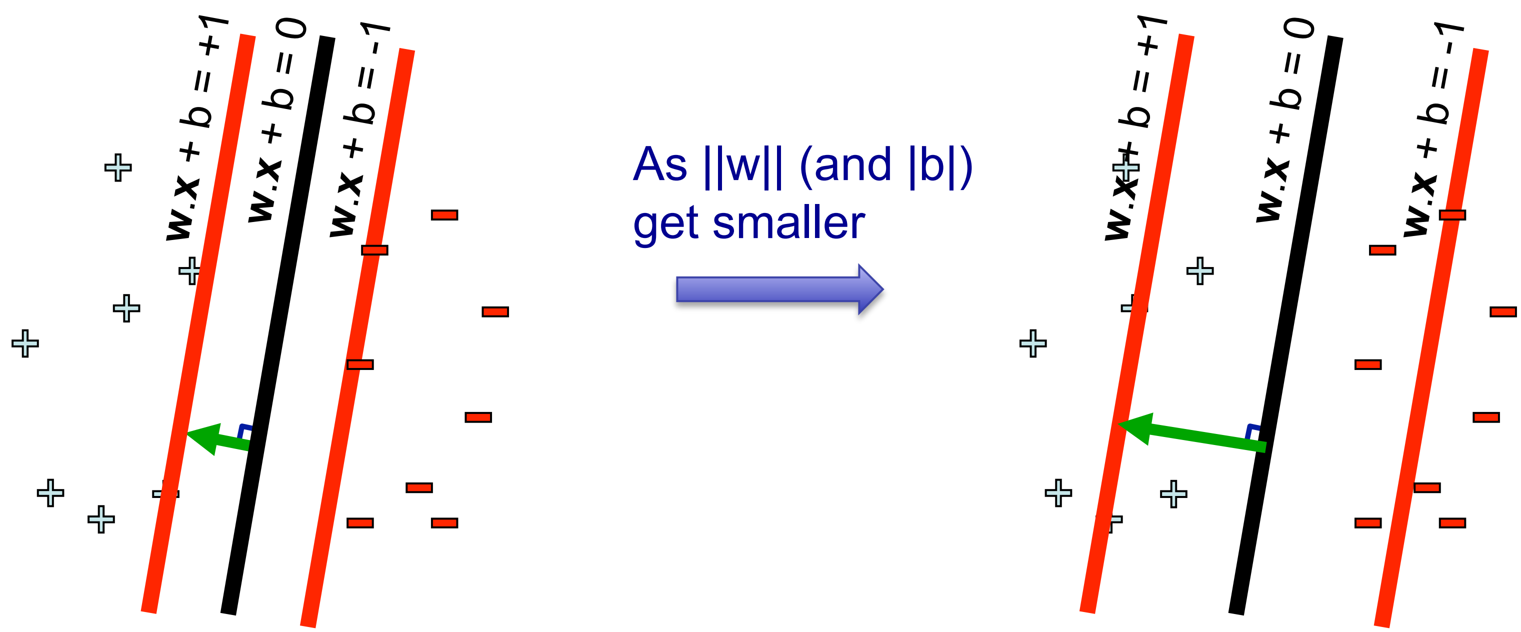


Key idea #2: seek large margin

- Suppose again that the data is linearly separable and we are solving a feasibility problem, with constraints

$$y_t\,(w \cdot x_t + b) \geq 1 \quad \forall t$$

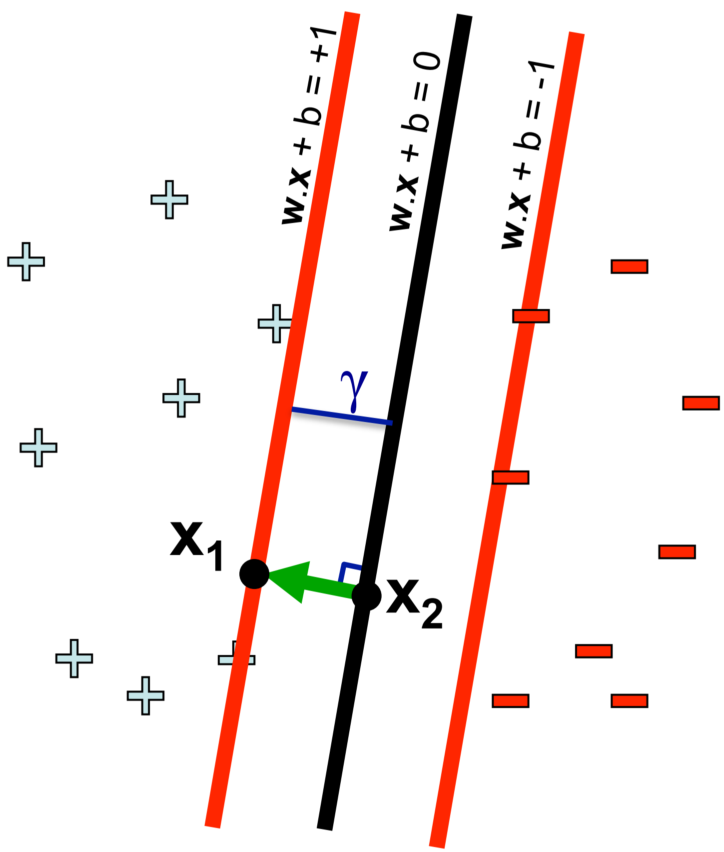
- If the length of the weight vector ||w|| is **too small**, the optimization problem is infeasible! Why?



What is γ (geometric margin) as a function of **w**?

$$\gamma_i = \text{Distance to } i\text{'th data point}$$

$$\gamma = \min_i \gamma_i$$



$$w \cdot x_1 + b = 1$$

-

$$w \cdot x_2 + b = 0$$

$$w \cdot (x_1 - x_2) = 1$$

Plug in

We also know that:

$$x_1 - x_2 = \gamma \frac{w}{||w||}$$

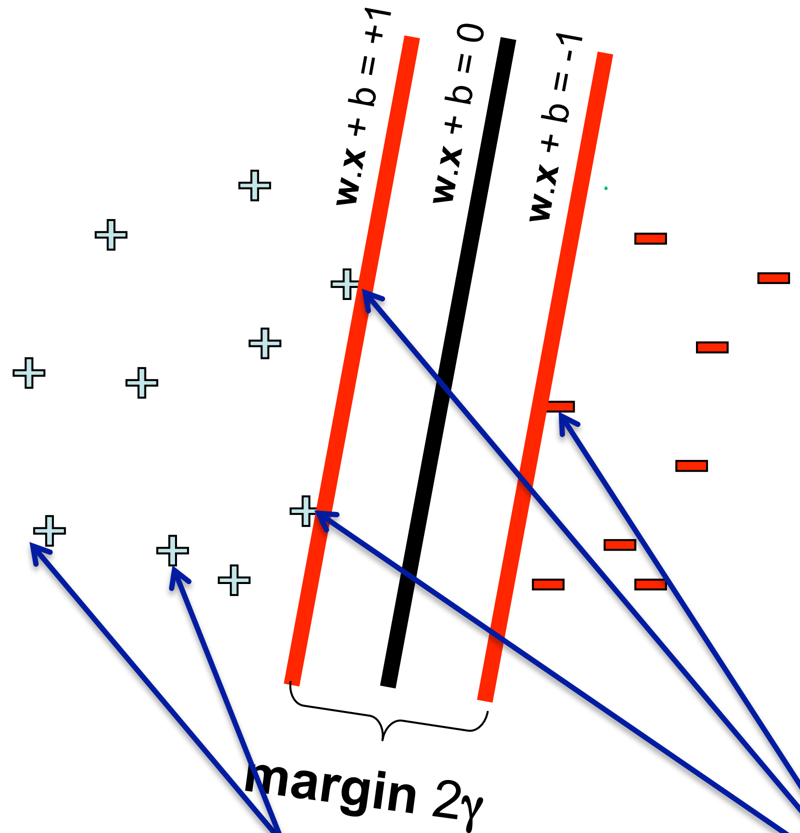
$$1 = w \cdot \left(\gamma \frac{w}{||w||}\right) = \frac{\gamma}{||w||} w \cdot w = \gamma ||w||$$

So, $\gamma = \dfrac{1}{||w||}$

(assuming there is a data point on the **w.x** + b = +1 or -1 line)

Final result: can maximize γ by minimizing $||w||_2$!!!

(Hard margin) support vector machines



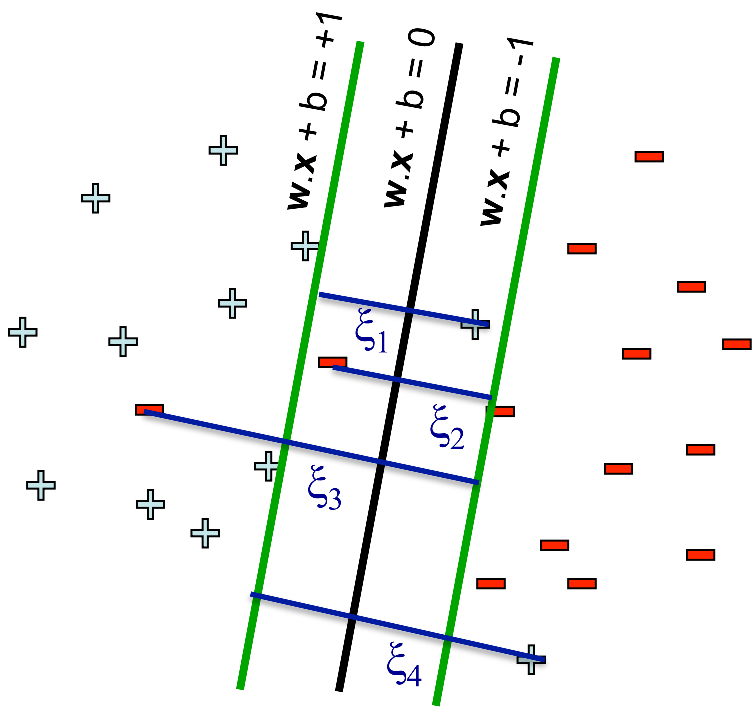
$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \ \forall j$$

- Example of a **convex optimization** problem
  - A quadratic program
  - Polynomial-time algorithms to solve!
- Hyperplane defined by **support vectors**
  - Could use them as a lower-dimension basis to write down line, although we haven't seen how yet
- More on these later

Non-support Vectors:
- everything else
- moving them will not change **w**

Support Vectors:
- data points on the canonical lines

Allowing for slack: “Soft margin SVM”



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

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad ,\forall j \quad \xi_j \geq 0$$

“slack variables”

Slack penalty $C > 0$:

- $C=\infty$ → have to separate the data!
- $C=0$ → ignores the data entirely!
- **Select using cross-validation**

For each data point:

- If margin ≥ 1, don’t care
- If margin < 1, pay linear penalty

Equivalent formulation using hinge loss

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

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

Substituting $\xi_j = \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$ into the objective, we get:

$$\boxed{\min ||w||^2 + C \sum_j \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)}$$

The **hinge loss** is defined as $\ell_{\text{hinge}}(y, \hat{y}) = \max\left(0, 1 - \hat{y}y\right)$

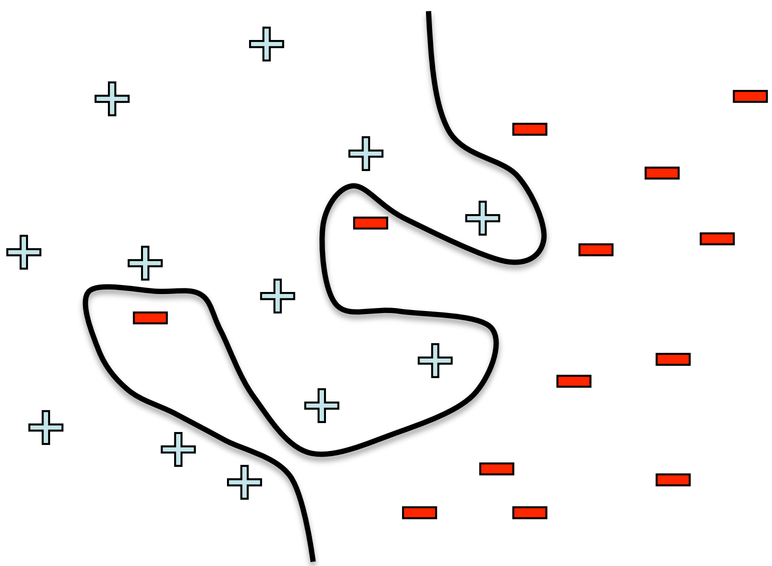
$$\min_{\mathbf{w},b} ||w||_2^2 + C \underbrace{\sum_j \ell_{\text{hinge}}(y_j,\, w \cdot x_j + b)}$$

This is called **regularization**; used to prevent overfitting!

This part is empirical risk minimization, using the hinge loss

What if the data is not linearly separable?

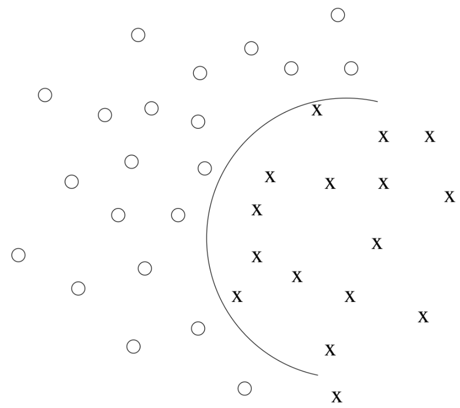
Use features of features of features of features….



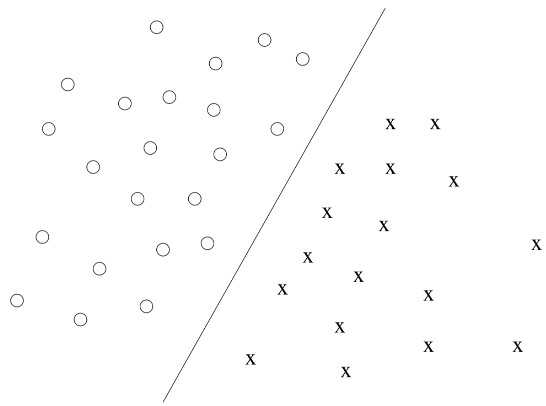
$$\phi(x) = \begin{pmatrix} x^{(1)} \\ \ldots \\ x^{(n)} \\ x^{(1)}x^{(2)} \\ x^{(1)}x^{(3)} \\ \ldots \\ e^{x^{(1)}} \\ \ldots \end{pmatrix}$$

Feature space can get really large really quickly!

Example



Non-linear separator in the original $\mathbf{x}$-space



Linear separator in the feature ϕ-space

[Tommi Jaakkola]

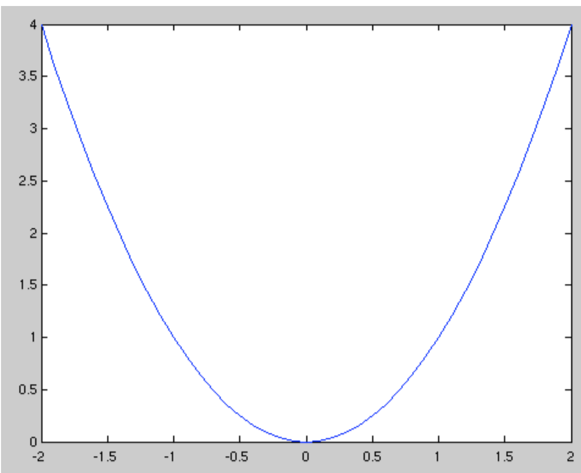
What's Next!

- Learn one of the most interesting and exciting recent advancements in machine learning
  - Key idea #3: the "kernel trick"
  - High dimensional feature spaces at no extra cost
- But first, a detour
  - Constrained optimization!

Constrained optimization

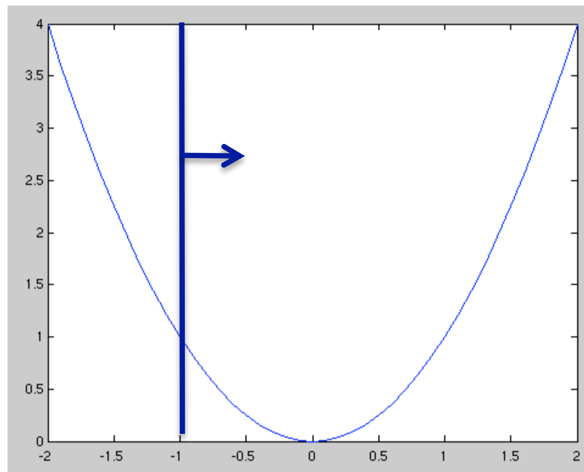
$$\boxed{\begin{array}{ll} \min_x & x^2 \\ \text{s.t.} & x \geq b \end{array}}$$

No Constraint



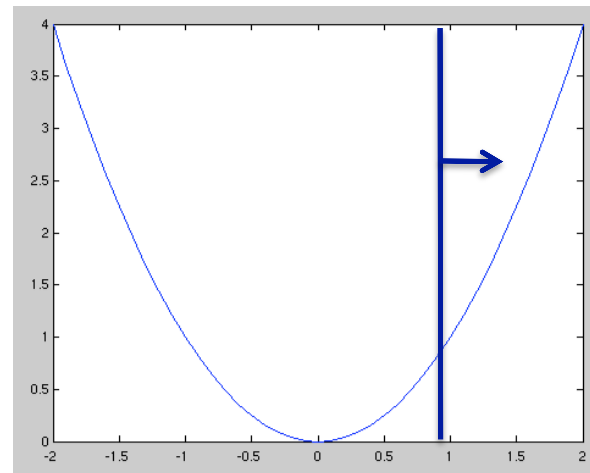
$x^*=0$

$x \geq -1$



$x^*=0$

$x \geq 1$

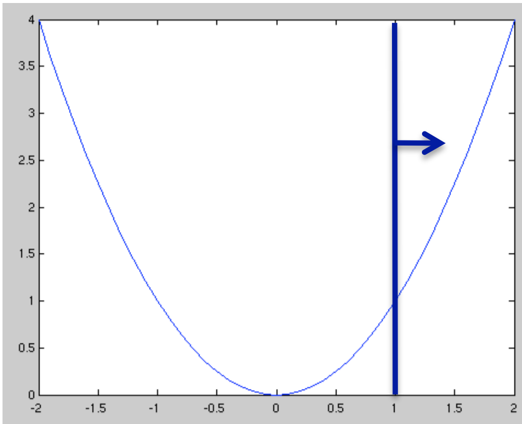


$x^*=1$

How do we solve with constraints?

→ Lagrange Multipliers!!!

Lagrange multipliers – Dual variables



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

Add Lagrange multiplier

Rewrite Constraint

Introduce Lagrangian (objective):

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

We will solve:

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

Add new constraint

Why is this equivalent?

- min is fighting max!

$x<b \rightarrow (x-b)<0 \rightarrow \max_\alpha -\alpha(x-b) = \infty$

- min won't let this happen!

$x>b,\ \alpha \geq 0 \rightarrow (x-b)>0 \rightarrow \max_\alpha -\alpha(x-b) = 0,\ \alpha^*=0$

- min is cool with 0, and $L(x, \alpha)=x^2$ (original objective)

$x=b \rightarrow \alpha$ can be anything, and $L(x, \alpha)=x^2$ (original objective)

The *min* on the outside forces *max* to behave, so constraints will be satisfied.

Dual SVM derivation (1) – the linearly separable case (hard margin SVM)

Original optimization problem:

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

Rewrite constraints

One Lagrange multiplier per example

Lagrangian:

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

Our goal now is to solve: $\min\limits_{\vec{w},b} \; \max\limits_{\vec{\alpha}\geq 0} \; L(\vec{w}, \vec{\alpha})$

Dual SVM derivation (2) – the linearly separable case (hard margin SVM)

(Primal) $$\min_{\vec{w},b} \max_{\vec{\alpha}\geq 0} \frac{1}{2}||\vec{w}||^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1\right]$$

Swap min and max

(Dual) $$\max_{\vec{\alpha}\geq 0} \min_{\vec{w},b} \frac{1}{2}||\vec{w}||^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1\right]$$

Slater's condition from convex optimization guarantees that these two optimization problems are equivalent!

Dual SVM derivation (3) – the linearly separable case (hard margin SVM)

(Dual) $$\max_{\vec{\alpha}\geq 0} \ \min_{\vec{w},b} \ \frac{1}{2}||\vec{w}||^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1 \right]$$

Can solve for optimal **w**, b as function of α:

$$\frac{\partial L}{\partial w} = w - \sum_j \alpha_j y_j x_j \quad \rightarrow \quad \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\frac{\partial L}{\partial b} = -\sum_j \alpha_j y_j \quad \rightarrow \quad \sum_j \alpha_j y_j = 0$$

Substituting these values back in (and simplifying), we obtain:

(Dual) $$\max_{\vec{\alpha}\geq 0,\ \sum_j \alpha_j y_j = 0} \ \sum_j \alpha_j - \frac{1}{2}\sum_{i,j} \underbrace{y_i y_j \alpha_i \alpha_j}_{\text{scalars}} \underbrace{(\vec{x}_i \cdot \vec{x}_j)}_{\text{dot product}}$$

Sums over all training examples

Dual formulation only depends on dot-products of the features!

$$\max_{\vec{\alpha}\geq 0,\ \sum_j \alpha_j y_j = 0} \ \sum_j \alpha_j - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j\right)$$

First, we introduce a *feature mapping*:

$$\mathbf{x}_i \mathbf{x}_j \quad \rightarrow \quad \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$$

Next, replace the dot product with an equivalent *kernel* function:

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

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

$$\sum_i \alpha_i y_i = 0$$

SVM with kernels

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

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

- Never compute features explicitly!!!
  - Compute dot products in closed form
- $O(n^2)$ time in size of dataset to compute objective
  - much work on speeding up

Predict with:

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i K(x_i, x) + b\right]$$

Common kernels

- Polynomials of degree exactly *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 kernels

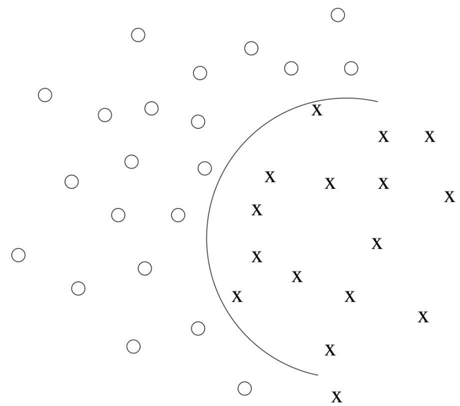
$$K(\vec{u},\vec{v}) = \exp\left(-\frac{||\vec{u}-\vec{v}||_2^2}{2\sigma^2}\right)$$

- Sigmoid

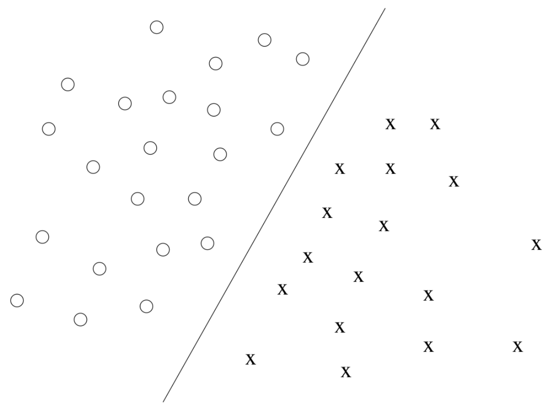
$$K(\mathbf{u},\mathbf{v}) = \tanh(\eta\mathbf{u}\cdot\mathbf{v} + \nu)$$

- And many others: very active area of research!

Quadratic kernel



Non-linear separator in the original $\mathbf{x}$-space



Linear separator in the feature ϕ-space

[Tommi Jaakkola]

Quadratic kernel

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^T \mathbf{z} + c)^2 = \left(\sum_{j=1}^{n} x^{(j)} z^{(j)} + c \right) \left(\sum_{\ell=1}^{n} x^{(\ell)} z^{(\ell)} + c \right) \\
&= \sum_{j=1}^{n} \sum_{\ell=1}^{n} x^{(j)} x^{(\ell)} z^{(j)} z^{(\ell)} + 2c \sum_{j=1}^{n} x^{(j)} z^{(j)} + c^2 \\
&= \sum_{j,\ell=1}^{n} (x^{(j)} x^{(\ell)})(z^{(j)} z^{(\ell)}) + \sum_{j=1}^{n} (\sqrt{2c}\, x^{(j)})(\sqrt{2c}\, z^{(j)}) + c^2,
\end{aligned}
$$

Feature mapping given by:

$$\mathbf{\Phi}(\mathbf{x}) = [x^{(1)2}, x^{(1)} x^{(2)}, ..., x^{(3)2}, \sqrt{2c}x^{(1)}, \sqrt{2c}x^{(2)}, \sqrt{2c}x^{(3)}, c]$$

[Cynthia Rudin]