Support vector machines (SVMs) Lecture 3

David Sontag
New York University

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

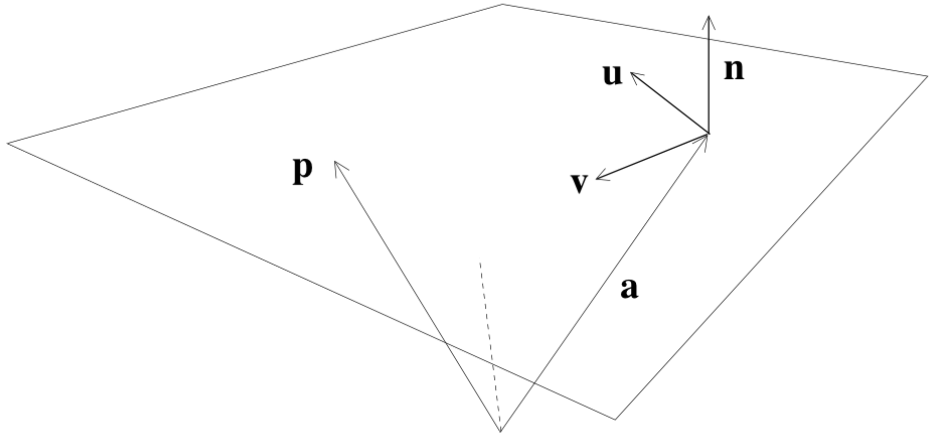
Geometry of linear separators (see blackboard)

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:

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

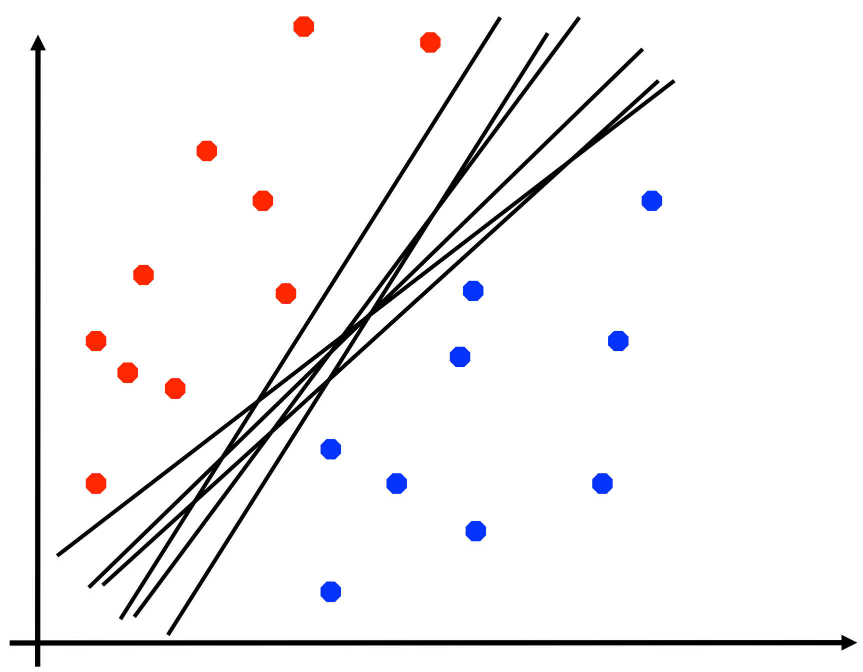
Normal vector
(we will call this w)

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

Barber, Section A.1.1-4

Linear Separators

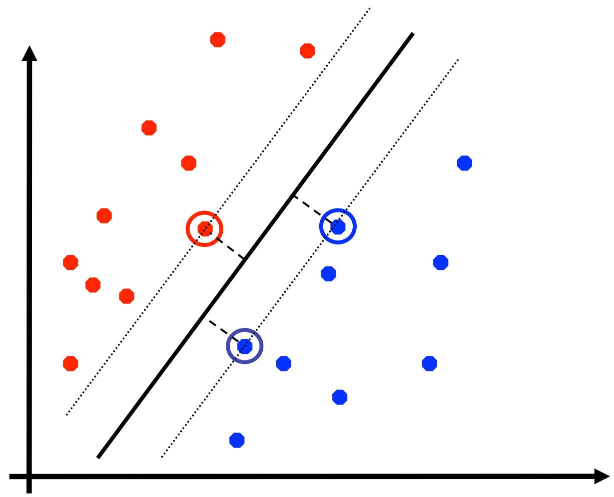
- If training data is linearly separable, perceptron is guaranteed to find *some* linear separator
- Which of these is **optimal**?



Support Vector Machine (SVM)

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

Robust to outliers!



V. Vapnik

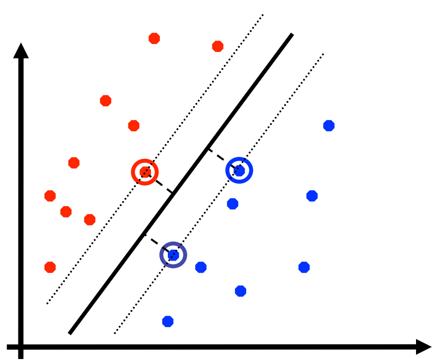
- Good according to intuition, theory, practice
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task

Support vector machines: 3 key ideas

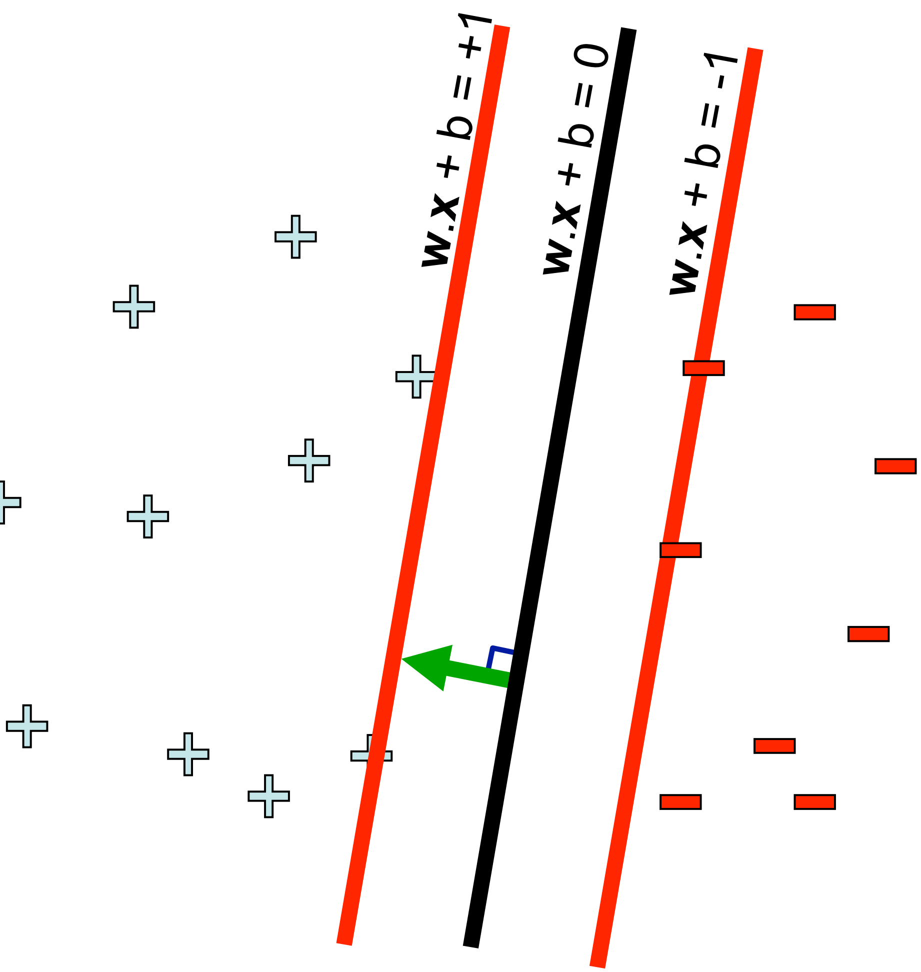
1. Use **optimization** to find solution (i.e. a hyperplane) with few errors

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

$$\text{for } y_t = +1, \quad w \cdot x_t + b \geq 1$$

$$\text{and for } y_t = -1, \quad w \cdot x_t + b \leq -1$$

Equivalently, we want to satisfy all of the linear constraints

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

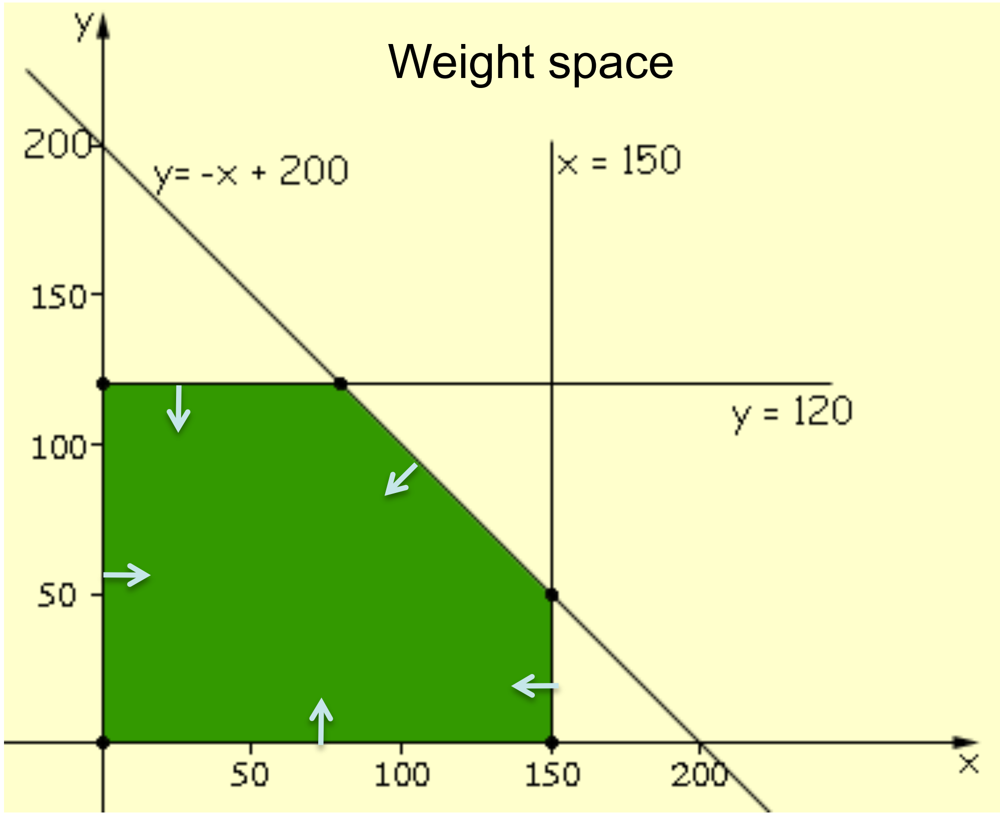
This *linear program* can be efficiently solved using algorithms such as simplex, interior point, or ellipsoid

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

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

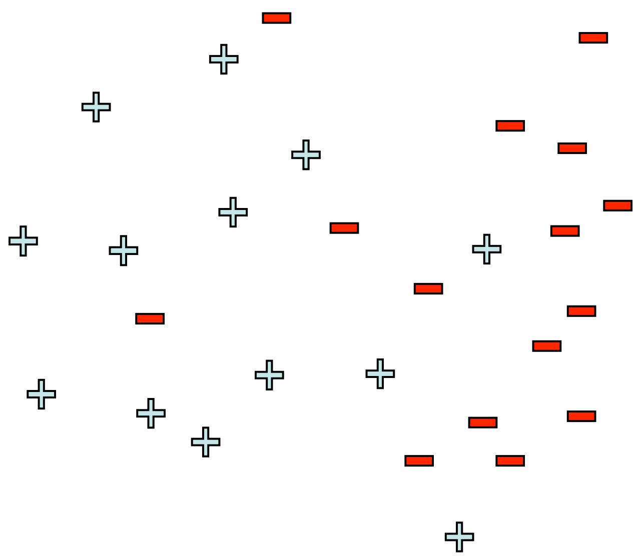
For SVMs, each data point gives one inequality:

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



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?

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

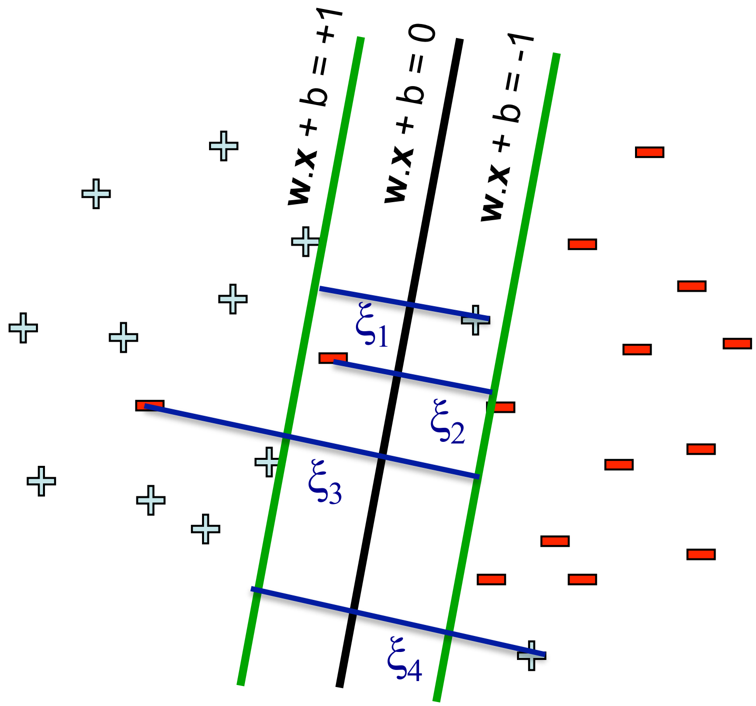
- Formalize this using the 0-1 loss:

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

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*



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

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

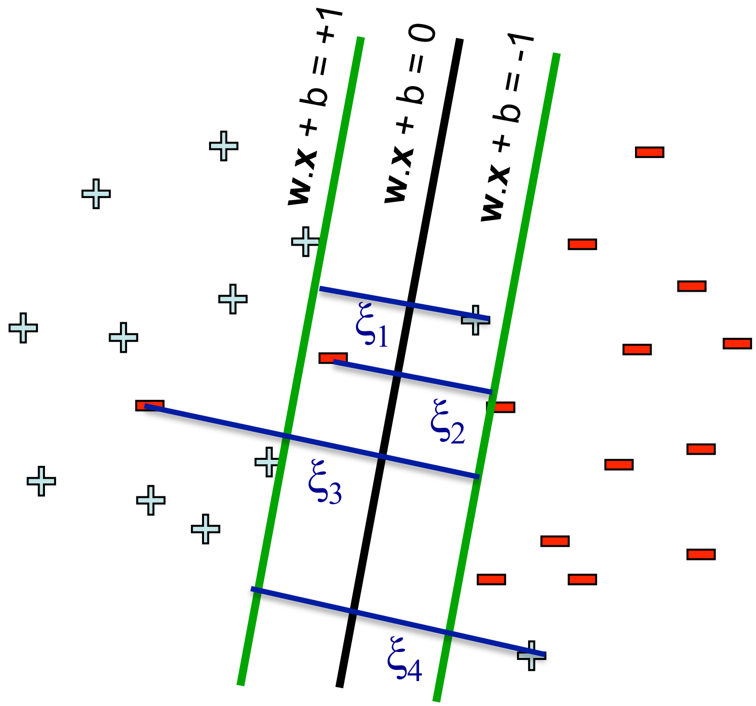
"slack variables"

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

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

Sometimes written as

$$\Big(1 - (w \cdot x_j + b)\, y_j\Big)_+$$

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:

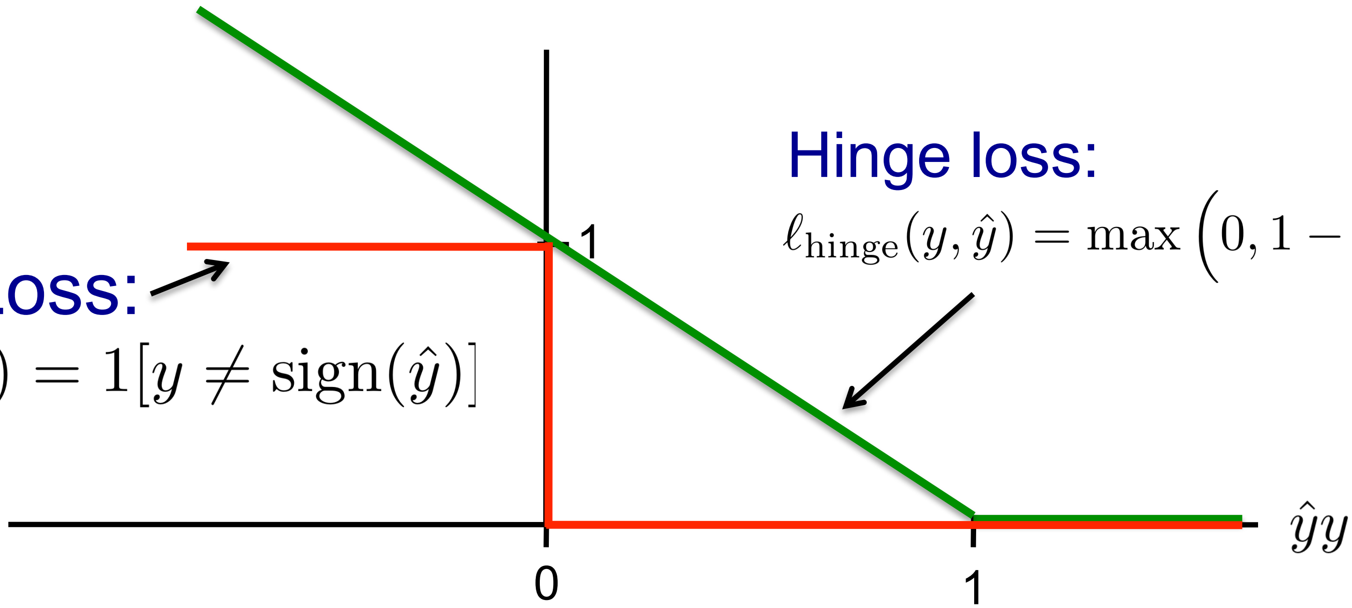
$$\boxed{\min_{w,b} \sum_j \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)}$$

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

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

This is empirical risk minimization,
using the hinge loss

Hinge loss vs. 0/1 loss



Hinge loss:

$$\ell_{\text{hinge}}(y,\hat{y}) = \max\left(0, 1-\hat{y}y\right)$$

0-1 Loss:

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

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