6.867 Machine learning and neural networks

Tommi Jaakkola
MIT AI Lab
tommi@ai.mit.edu

Lecture 8: boosting, support vector machines

Topics

- Combination of methods
  - voting methods: bagging and boosting
  - margin and generalization
- Support vector machines
  - “optimal” hyperplane

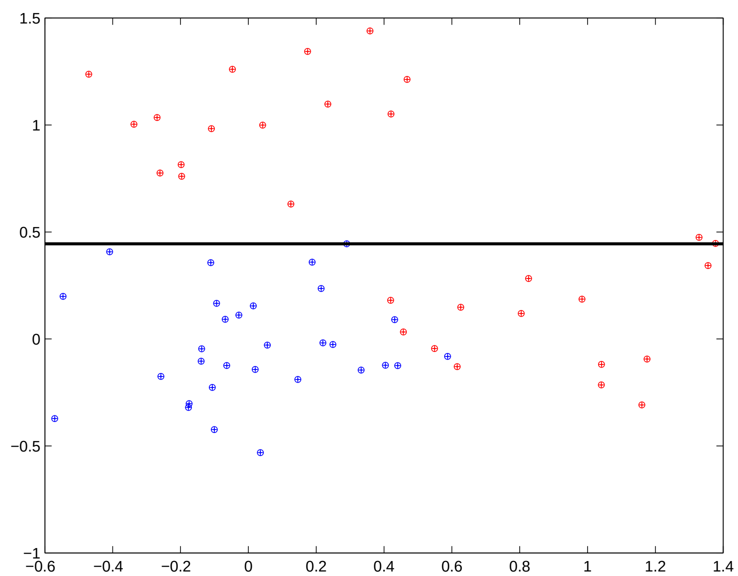
Combination of classifiers

- Suppose we are given a training set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ of examples and (± 1) labels and a family of component classifiers such as *decision stumps*:

$$h(\mathbf{x}; \theta) = \text{sign}(\, w_1\, x_k - w_0\,)$$

where $\theta = \{k, w_1, w_0\}$.

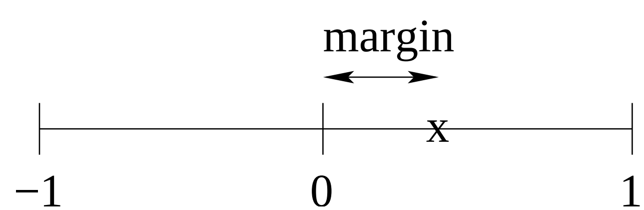
Each decision stump pays attention to only a single component of the input vector



Bagging

- We can combine classifiers to ensure more *robust* predictions (classifications)
- Given a set of n training examples and labels, repeat
  1. resample (with replacement) a smaller training set of $n' < n$ examples
  2. train a new classifier (decision stump) $h(\mathbf{x}; \hat{\theta})$ based on the smaller training set
- The resulting combined classifier is obtained by *voting*

$$\hat{h}(\mathbf{x}) = \text{sign}\left(\frac{1}{m} \sum_{k=1}^{m} h(\mathbf{x}; \hat{\theta}_k) \right)$$



Beyond Bagging: reweighting training examples

- The component classifiers should concentrate more on training examples that are difficult to classify correctly
- We can tune the classifiers towards harder examples by reweighting the training examples (small margin $\Rightarrow$ large weight)

  Example: suppose we already have $h(\mathbf{x}; \hat{\theta}_1), \ldots, h(\mathbf{x}; \hat{\theta}_m)$. We train the next component classifier $h(\mathbf{x}; \theta_{m+1})$ on a reweighted training set

$$\text{Weight } p(i) \text{ on } (\mathbf{x}_i, y_i)\text{:} \quad p(i) \;\propto\; \exp\Big\{ -\overbrace{y_i \sum_{k=1}^{m} h(\mathbf{x}_i; \hat{\theta}_k)}^{\text{margin}} \Big\}$$

  where examples with small or negative classification margins (difficult examples) will have larger weights

Boosting

- A Boosting algorithm sequentially estimates and combines classifiers by reweighting training sets (concentrating on the harder examples)
  - each component classifiers is presented with a slightly different problem
- AdaBoost preliminaries:
  a) Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ with binary ± 1 labels y_i.
  b) A set of "weak" binary (± 1) classifiers $h(\mathbf{x}; \theta)$ such as *decision stumps*

  $$h(\mathbf{x}; \theta) = \text{sign}(\, w_1 \, x_k - w_0 \,)$$

  where $\theta = \{k, w_1, w_0\}$.
  c) Initially all weights are equal: $p(i) = 1/n$.

The AdaBoost algorithm

1: Find the k^{th} classifier $h(\mathbf{x}; \widehat{\theta}_k)$ such that its *weighted training error*

$$\epsilon_k = \sum_{i=1}^{n} p_k(i) \, [\![y_i \neq h(\mathbf{x}_i; \widehat{\theta}_k)]\!]$$

is better than chance. Here $[\![y \neq y']\!] = 1$ if the argument $y \neq y'$ is true and zero otherwise.

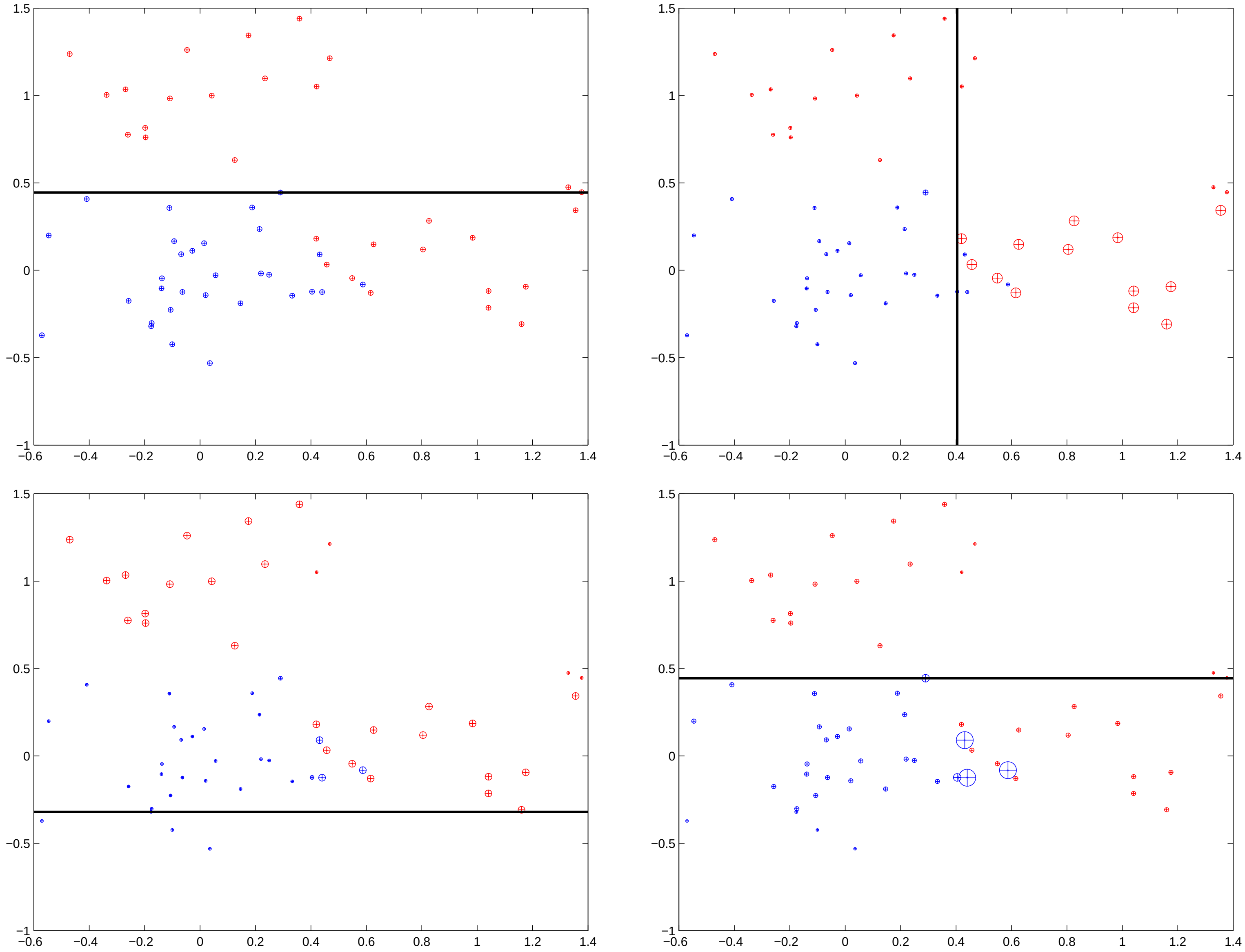
2: Determine how many "votes" to give to the new component classifier: $\widehat{\alpha}_k = 0.5 \log(\,(1-\epsilon_k)/\epsilon_k\,)$ (decorrelation)

3: Update example weights: $p_{k+1}(i) = p_k(i) \cdot \exp(\,-\widehat{\alpha}_k \, y_i \, h(\mathbf{x}_i; \widehat{\theta}_k)\,)$ and renormalize the new weights to one.

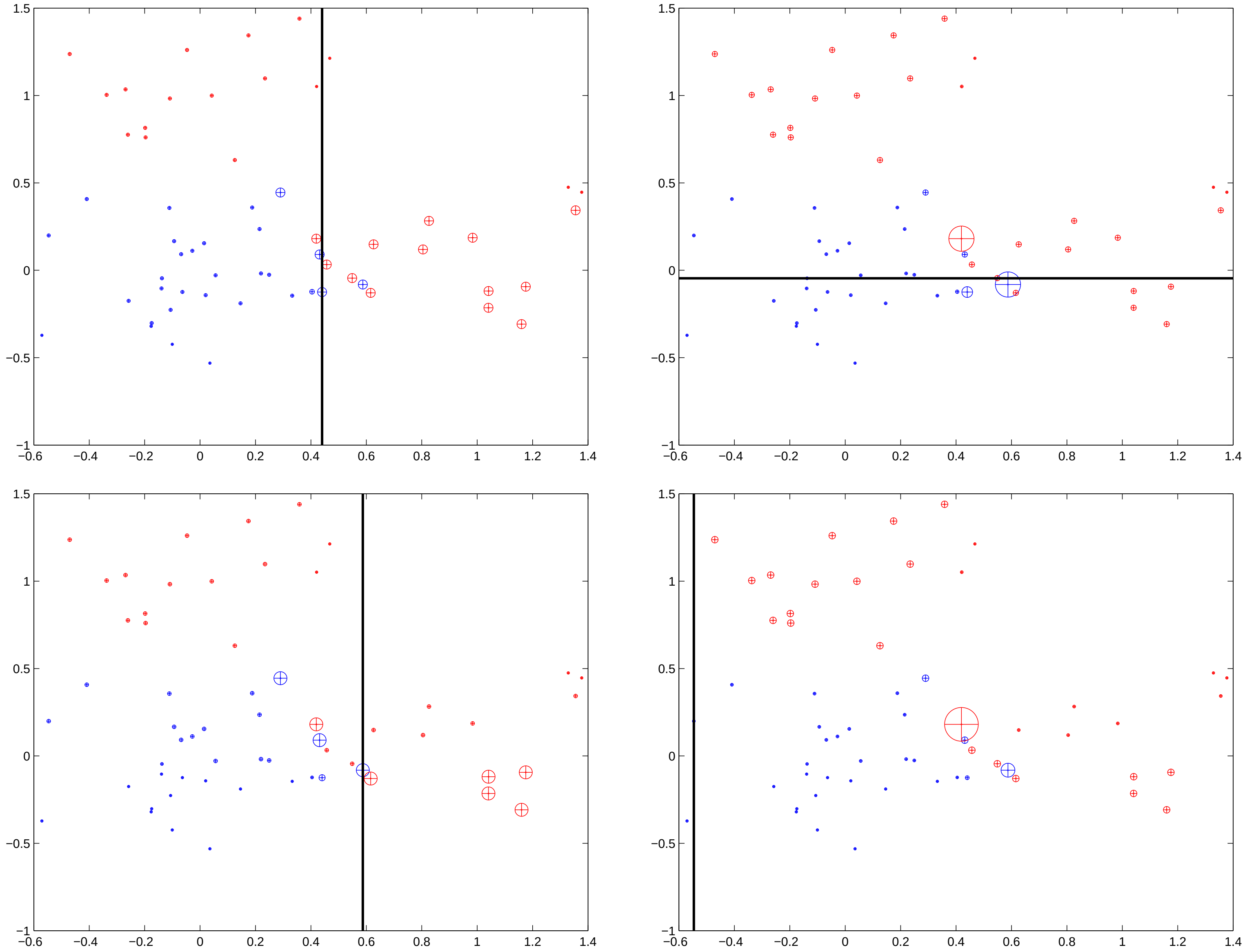
- The final classifier after m boosting iterations is given by

$$\widehat{h}(\mathbf{x}) = \mathrm{sign}\left(\frac{\widehat{\alpha}_1 h(\mathbf{x}; \widehat{\theta}_1) + \ldots + \widehat{\alpha}_m h(\mathbf{x}; \widehat{\theta}_m)}{\widehat{\alpha}_1 + \ldots + \widehat{\alpha}_m} \right)$$

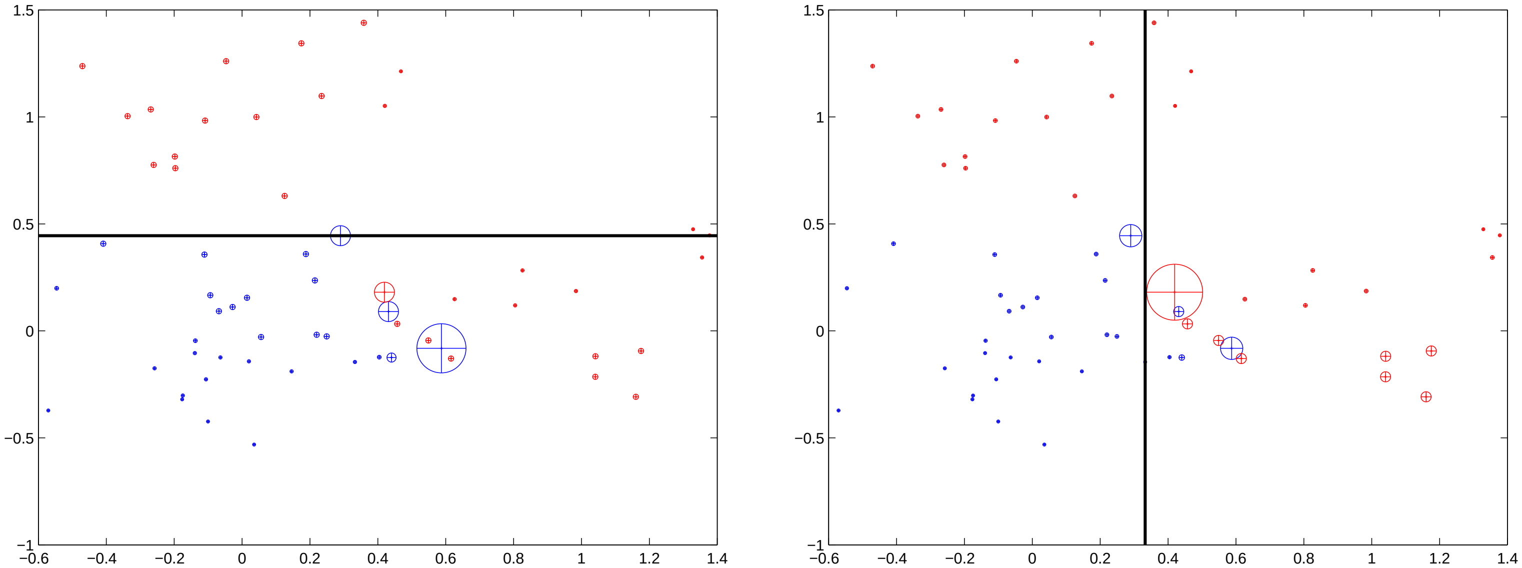
Boosting: example



Boosting: example cont'd



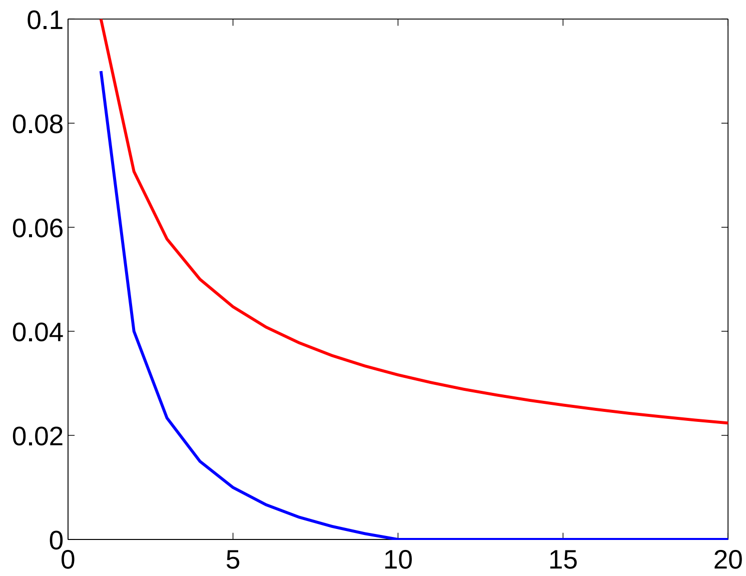
Boosting: example cont'd



Boosting performance

- Training/test errors for the *combined classifier*

$$\hat{h}(\mathbf{x}) = \text{sign}\left(\frac{\hat{\alpha}_1 h(\mathbf{x}; \hat{\theta}_1) + \ldots + \hat{\alpha}_m h(\mathbf{x}; \hat{\theta}_m)}{\hat{\alpha}_1 + \ldots + \hat{\alpha}_m} \right)$$

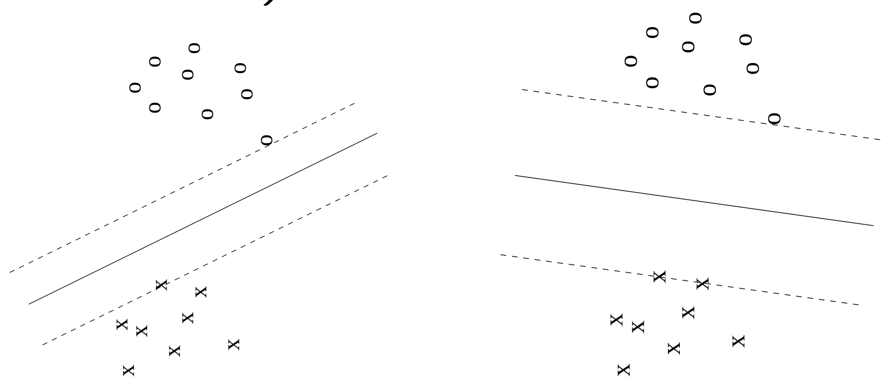


What about the component classifiers (decision stumps)?

- Even after the training error of the combined classifier goes to zero, boosting iterations can still improve the generalization error!

Classification margin

(this is only an illustration; margins from boosted decision stumps would look a bit different)



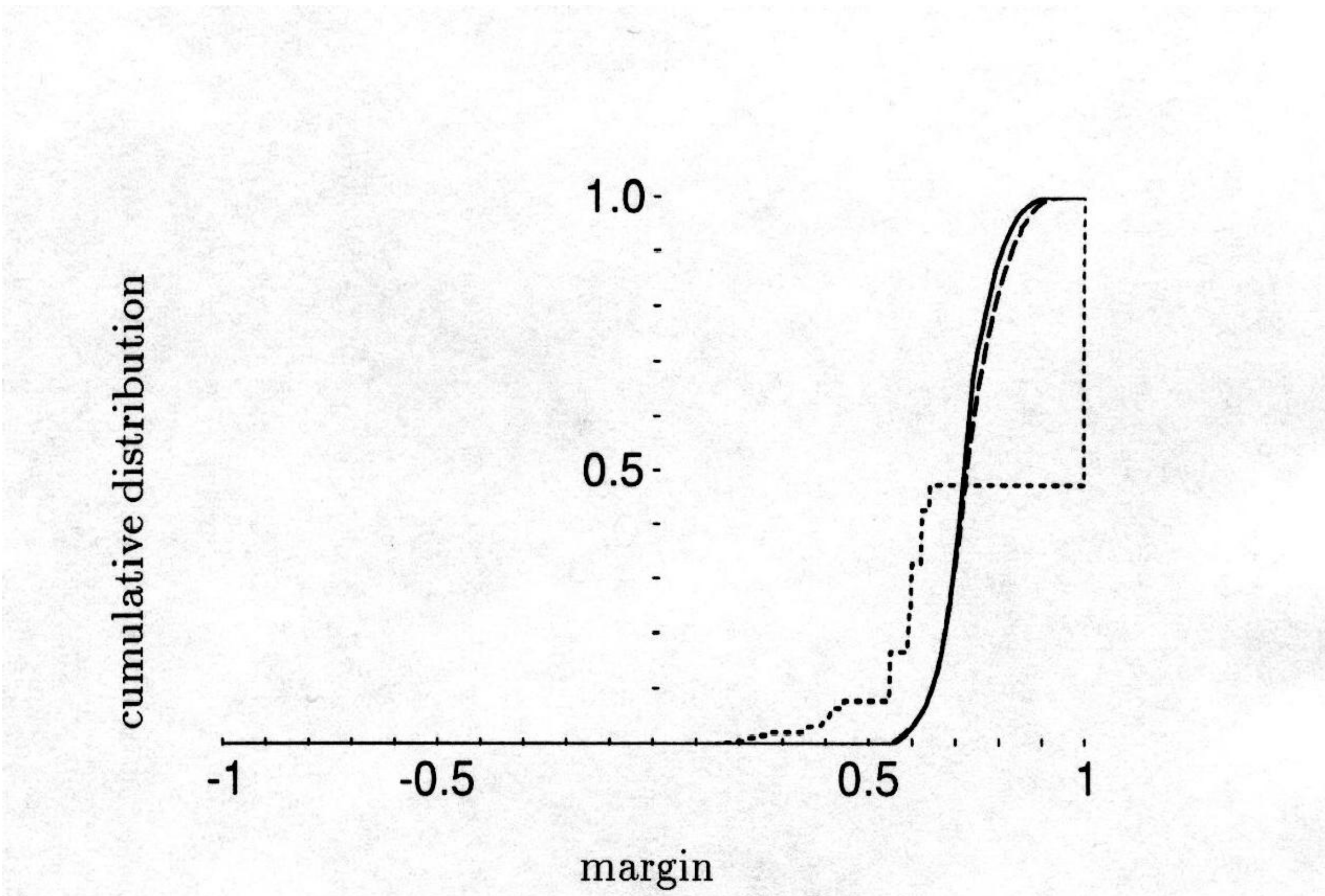
small margin large margin

- The training error is zero in both cases ... why is larger margin better?

Boosting and margin

- Boosting iterations tend to increase the margin

$$y\left(\frac{\widehat{\alpha}_1 h(\mathbf{x};\widehat{\theta}_1)+\ldots+\widehat{\alpha}_m h(\mathbf{x};\widehat{\theta}_m)}{\widehat{\alpha}_1+\ldots+\widehat{\alpha}_m}\right)$$

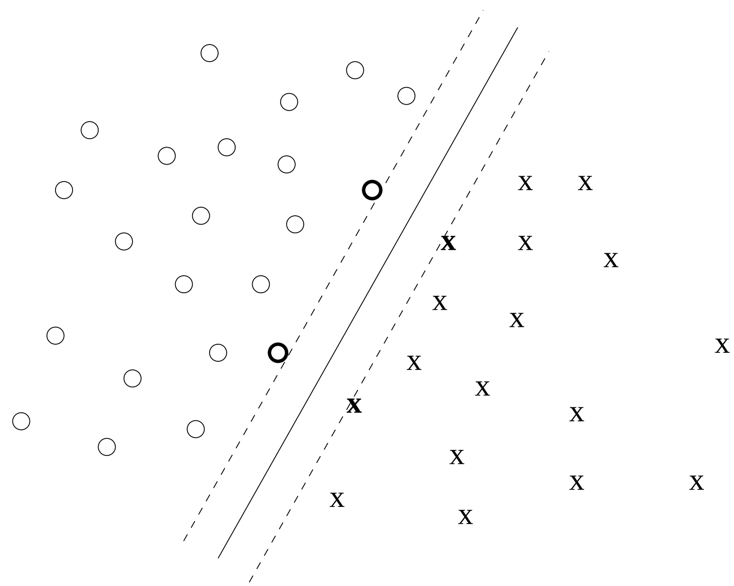


Topics

- Support vector machines
  - “optimal” hyperplane

“Optimal” hyperplane

- Let’s assume for simplicity that the classification problem is *linearly separable*



- Maximum margin hyperplane is maximally removed from all the training examples
- This hyperplane can be defined on the basis of only a few training examples called *support vectors*

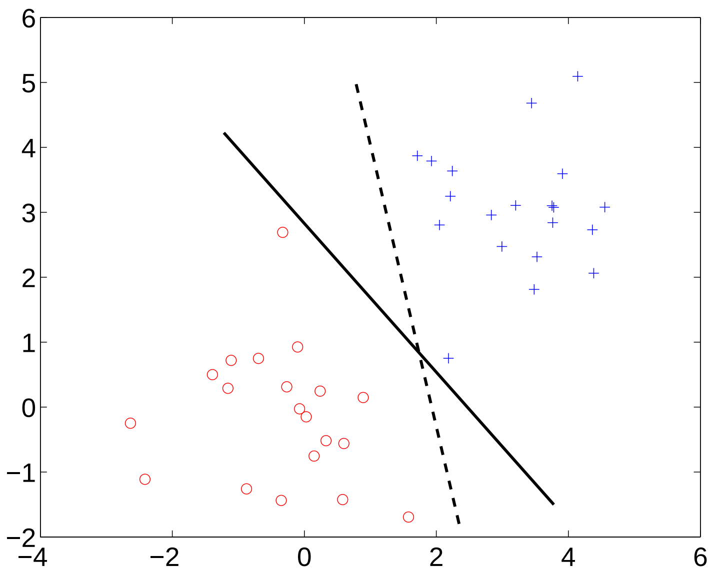
"Optimal" hyperplane cont'd

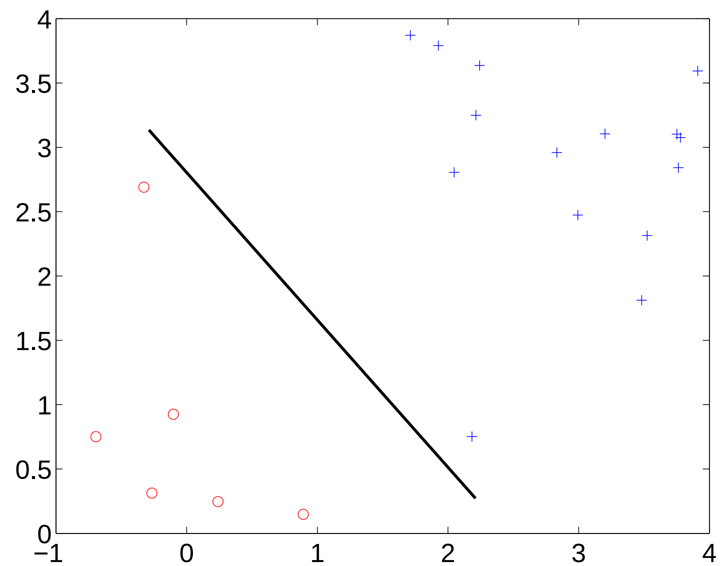
- Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where the labels are binary ± 1
- Linear separator:

$$\begin{aligned} f(\mathbf{x}; \mathbf{w}, w_0) &= w_0 + x_1 w_1 + \ldots x_d w_d \\ &= w_0 + \mathbf{w}^T \mathbf{x} \end{aligned}$$

- We can try to find the "optimal" hyperplane by requiring that the sign of the decision boundary $[w_0 + \mathbf{w}^T \mathbf{x}]$ (clearly) agrees with the training labels

$$y_i \left[w_0 + \mathbf{w}^T \mathbf{x}_i \right] - 1 \geq 0, \quad i = 1, \ldots, n$$





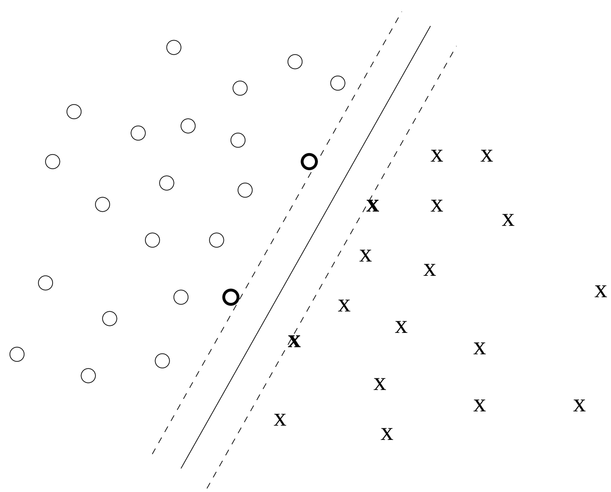
Support vector machine

- We minimize

$$\|\mathbf{w}\|^2/2 = \mathbf{w}^T\mathbf{w}/2 = \sum_{j=1}^{d} w_i^2/2$$

subject to the classification constraints

$$y_i\,[w_0 + \mathbf{w}^T\mathbf{x}_i] - 1 \geq 0, \quad i = 1, \ldots, n$$



- Only a few of the classification constraints are relevant
$\Rightarrow$ support vectors