Machine learning: lecture 11

Tommi S. Jaakkola
MIT CSAIL
tommi@csail.mit.edu

Topics

- Combination of classifiers: boosting
  - decision stumps, reweighting, adaboost
  - examples,
  - margin and performance

Review: combination of classifiers

- We can combine multiple "weak" classifiers to produce a single "strong" classifier:

$$h_m(\mathbf{x}) = h(\mathbf{x}; \theta_1) + \ldots + h(\mathbf{x}; \theta_m)$$

where the predicted label for $\mathbf{x}$ is the sign of the discriminant function $h_m(\mathbf{x})$.

- If each component classifier returns only ± 1 it is beneficial to allow some of them to have more "votes" than others:

$$h_m(\mathbf{x}) = \alpha_1\, h(\mathbf{x}; \theta_1) + \ldots + \alpha_m\, h(\mathbf{x}; \theta_m)$$

where the (non-negative) votes α_i can be used to emphasize components that are more reliable than others

- We wish to "modularize" the estimation problem

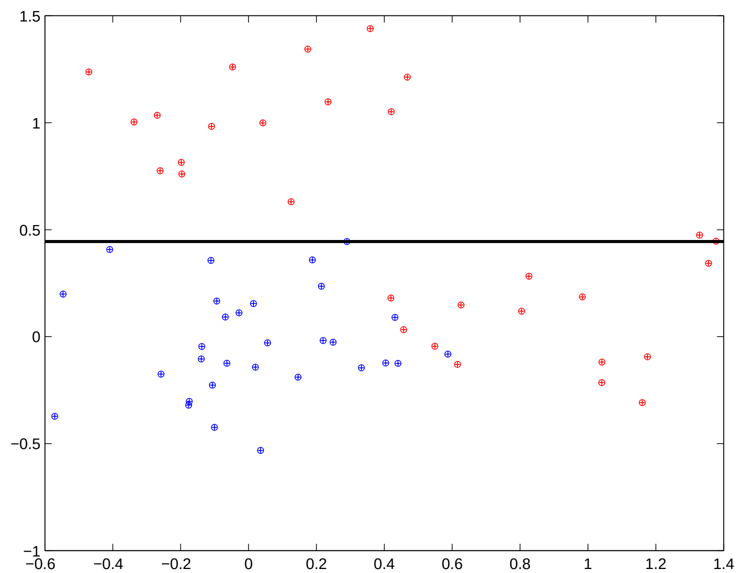
Review: decision stumps as weak classifiers

- Consider the following simple family of component classifiers generating ± 1 labels:

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

where $\theta = \{k, w_1, w_0\}$. These are called *decision stumps*.

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



Review: measure of classification loss

- We need a loss function that permits us to summarize how the already included components in $h_{m-1}(\mathbf{x})$ affect the training of a new component $\alpha_m \, h(\mathbf{x}; \theta_m)$.
- The exponential loss function

$$\exp\{\, -y\, h_m(\mathbf{x})\,\}$$

suffices for this purpose. The loss is large when the ± 1 label y disagrees with the sign of the discriminant function

$$h_m(\mathbf{x}) = \alpha_1 \, h(\mathbf{x}; \theta_1) + \ldots + \alpha_m \, h(\mathbf{x}; \theta_m)$$

Review: modularity

- The empirical exponential loss is "modular":

$$\sum_{i=1}^{n} \exp\{\, -y_i h_m(\mathbf{x}_i)\,\}$$

$$= \sum_{i=1}^{n} \exp\{\, -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\}$$

$$= \sum_{i=1}^{n} \underbrace{\exp\{\, -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\}$$

$$= \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\}$$

The combined classifier based on $m-1$ iterations defines a weighted loss criterion for the next simple classifier to add

Enforcing stronger modularity

- We can further simplify the estimation criterion for the new component classifiers (not the votes)

  Rationale: When $\alpha_m \approx 0$ (low confidence votes)

  $$\exp\{-y_i \alpha_m h(\mathbf{x}_i; \theta_m)\} \approx 1 - y_i \alpha_m h(\mathbf{x}_i; \theta_m)$$

  and our empirical loss criterion reduces to

  $$\approx \sum_{i=1}^{n} W_i^{(m-1)} \left(1 - y_i \alpha_m h(\mathbf{x}_i; \theta_m)\right) =$$

  $$= \sum_{i=1}^{n} W_i^{(m-1)} - \alpha_m \left(\sum_{i=1}^{n} W_i^{(m-1)} y_i h(\mathbf{x}_i; \theta_m) \right)$$

  So we can choose each new component classifier to optimize a weighted classification error

Boosting

- A Boosting algorithm sequentially estimates and combines classifiers by reweighting training examples
  - each component classifier is presented with a slightly different problem depending on the weights
  - the weights focus each new component on harder examples
- To get started we need:
  - a family of "weak" binary (± 1) classifiers $h(\mathbf{x}; \theta)$ such as *decision stumps*
  - normalized weights $\tilde{W}_i^{(0)}$ on the training examples, initially set to be uniform ($\tilde{W}_i^{(0)} = 1/n$)

The AdaBoost algorithm

1) At the k^{th} iteration we find any classifier $h(\mathbf{x};\hat{\theta}_k)$ for which the *weighted classification error* ϵ_k

$$\epsilon_k = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(k-1)} y_i h(\mathbf{x}_i;\hat{\theta}_k)\right)$$

is better than chance.

The AdaBoost algorithm

1) At the k^{th} iteration we find any classifier $h(\mathbf{x};\hat{\theta}_k)$ for which the *weighted classification error* ϵ_k

$$\epsilon_k = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(k-1)}\, y_i h(\mathbf{x}_i;\hat{\theta}_k)\right)$$

is better than chance.

2) The new component classifier is assigned "votes" based on its performance

$$\hat{\alpha}_k = 0.5\,\log(\,(1-\epsilon_k)/\epsilon_k\,)$$

where $\hat{\alpha}_k$ minimizes the weighted exponential loss

$$\sum_{i=1}^{n} \tilde{W}_i^{(k-1)}\,\exp\{\,-y_i \alpha_k h(\mathbf{x}_i;\hat{\theta}_k)\,\}$$

The AdaBoost algorithm

1) At the k^{th} iteration we find any classifier $h(\mathbf{x};\hat{\theta}_k)$ for which the *weighted classification error* ϵ_k

$$\epsilon_k = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(k-1)} y_i h(\mathbf{x}_i;\hat{\theta}_k)\right)$$

is better than chance.

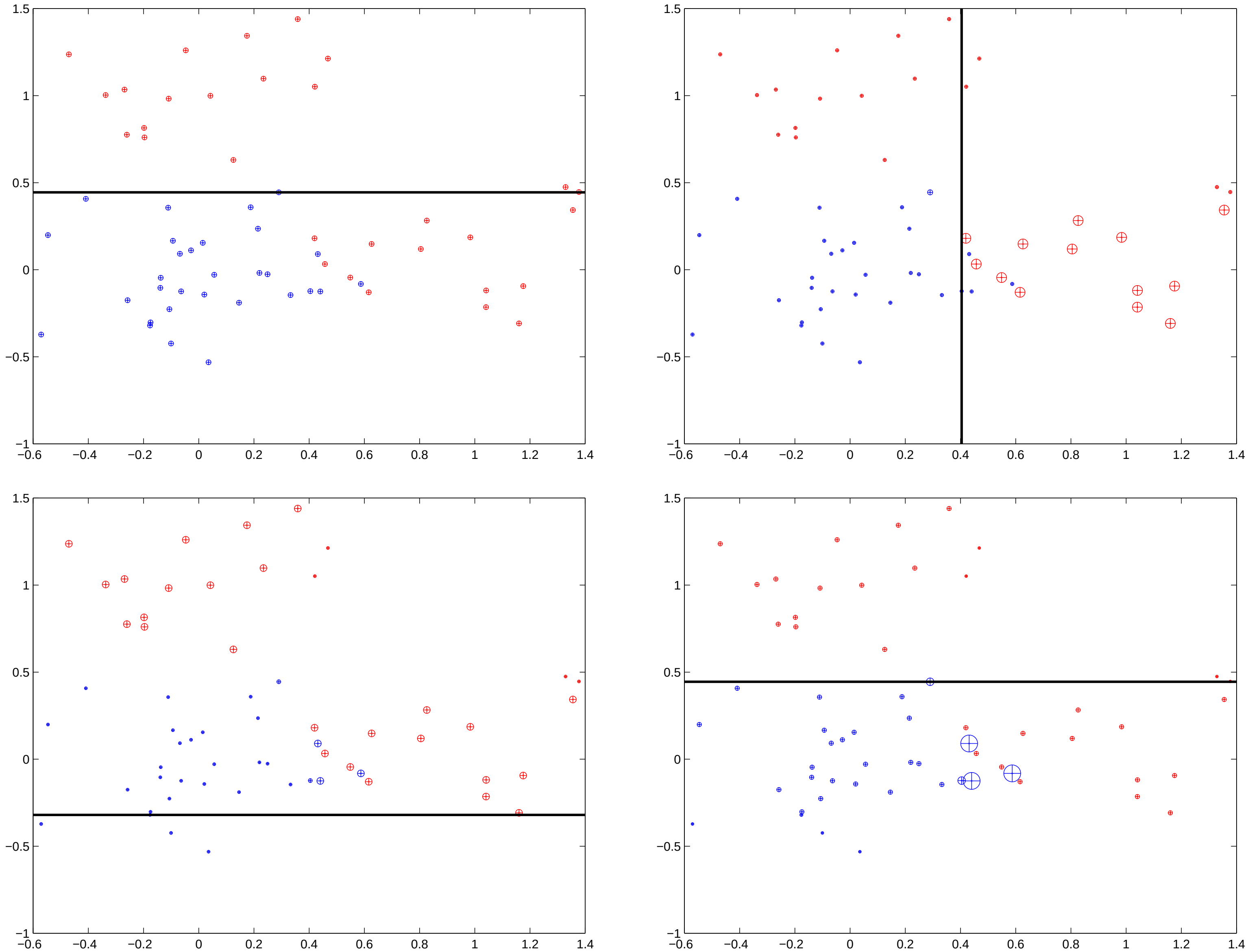
2) The new component classifier is assigned "votes" based on its performance

$$\hat{\alpha}_k = 0.5 \log(\,(1-\epsilon_k)/\epsilon_k\,)$$

3) The weights on the training examples are updated according to (c is chosen so that the new weights $\tilde{W}_i^{(k)}$ sum to one):

$$\tilde{W}_i^{(k)} = c \cdot \tilde{W}_i^{(k-1)} \cdot \exp\{\,-y_i\hat{\alpha}_k h(\mathbf{x}_i;\hat{\theta}_k)\,\}$$

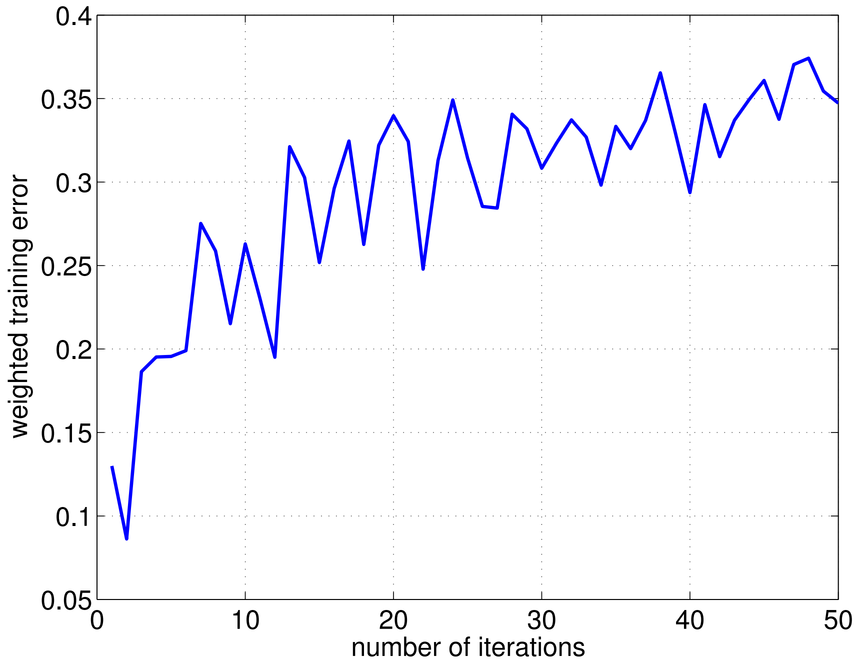
Boosting: example



"Typical" performance

- Weighted error of each new component classifier

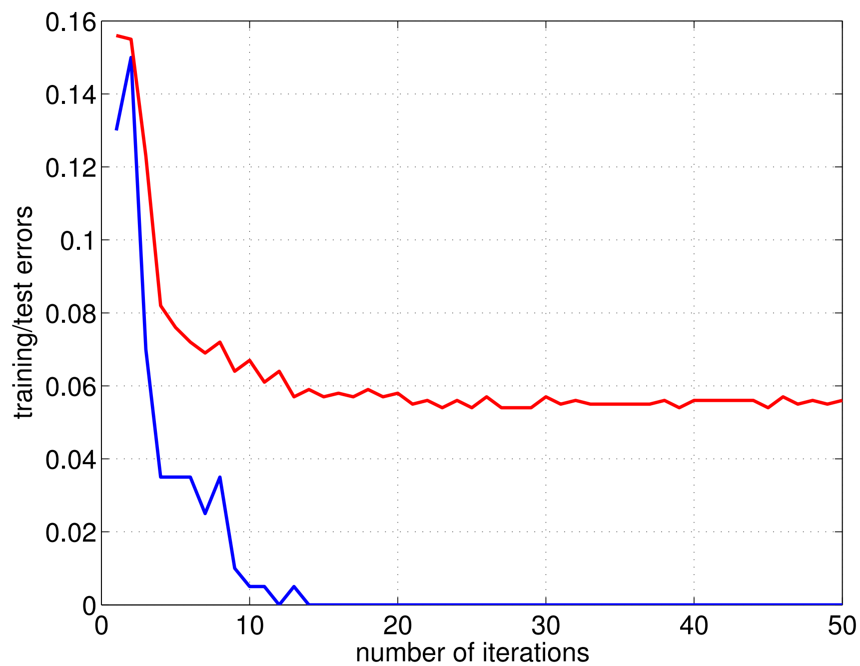
$$\epsilon_k = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(k-1)} y_i h(\mathbf{x}_i; \hat{\theta}_k)\right)$$



"Typical" performance cont'd

- Training and test errors of the *combined classifier*

$$\hat{h}_m(\mathbf{x}) = \hat{\alpha}_1 h(\mathbf{x}; \hat{\theta}_1) + \ldots + \hat{\alpha}_m h(\mathbf{x}; \hat{\theta}_m)$$



- Why should the test error go down after we already have zero training error?

AdaBoost and margin

- We can write the combined classifier in a more useful form by dividing the predictions by the "total number of votes":

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

- This allows us to define a clear notion of "voting margin" that the combined classifier achieves for each training example:

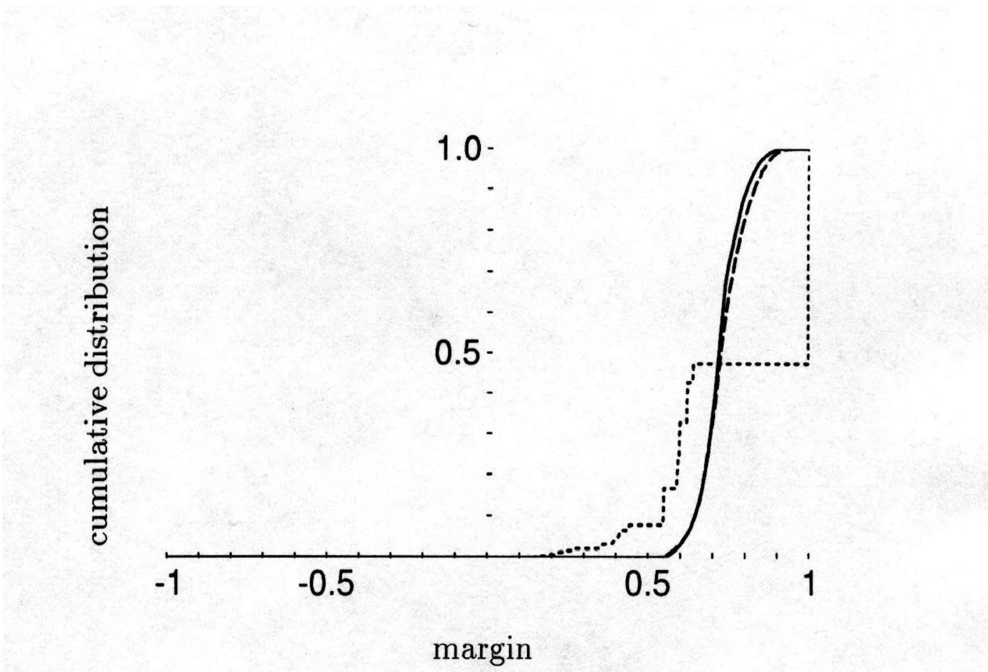
$$\text{margin}(\mathbf{x}_i) = y_i \cdot \hat{h}_m(\mathbf{x}_i)$$

  The margin lies in $[-1, 1]$ and is negative for all misclassified examples.

AdaBoost and margin

- Successive boosting iterations improve the majority vote or margin for the training examples

$$\text{margin}(\mathbf{x}_i) \;=\; y_i \left[\frac{\hat{\alpha}_1 h(\mathbf{x}_i;\hat{\theta}_1) + \ldots + \hat{\alpha}_m h(\mathbf{x}_i;\hat{\theta}_m)}{\hat{\alpha}_1 + \ldots + \hat{\alpha}_m} \right]$$



Can we improve the combination?

- As a result of running the boosting algorithm for m iterations, we essentially generate a new feature representation for the data

$$\phi_i(\mathbf{x}) = h(\mathbf{x}; \hat{\theta}_i), i = 1, \ldots, m$$

- Perhaps we can do better by separately estimating a new set of "votes" for each component. In other words, we could estimate a linear classifier of the form

$$f(\mathbf{x}; \alpha) = \alpha_1 \phi_1(\mathbf{x}) + \ldots \alpha_m \phi_m(\mathbf{x})$$

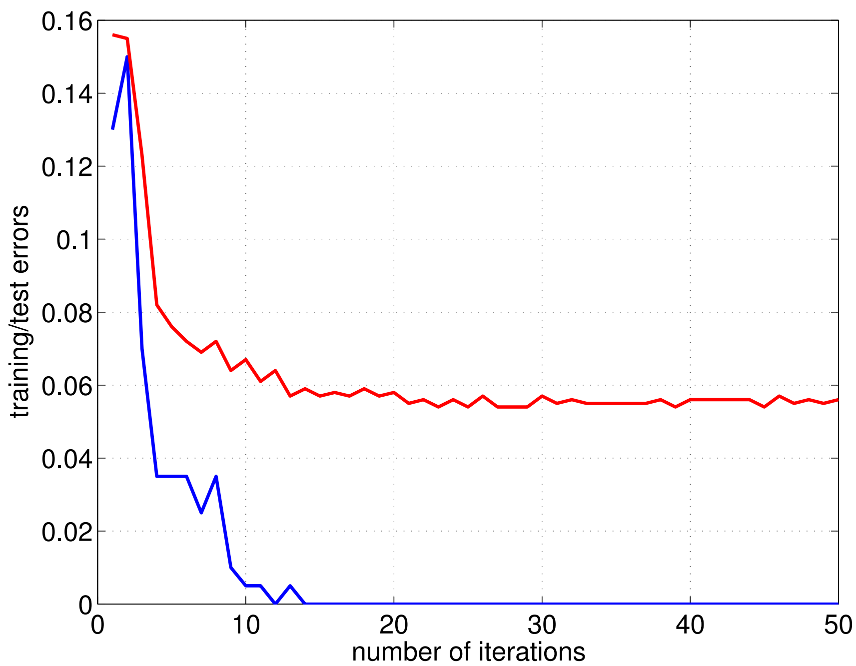
where each parameter α_i can be now any real number (even negative). The parameters would be estimated jointly rather than one after the other as in boosting.

Can we improve the combination?

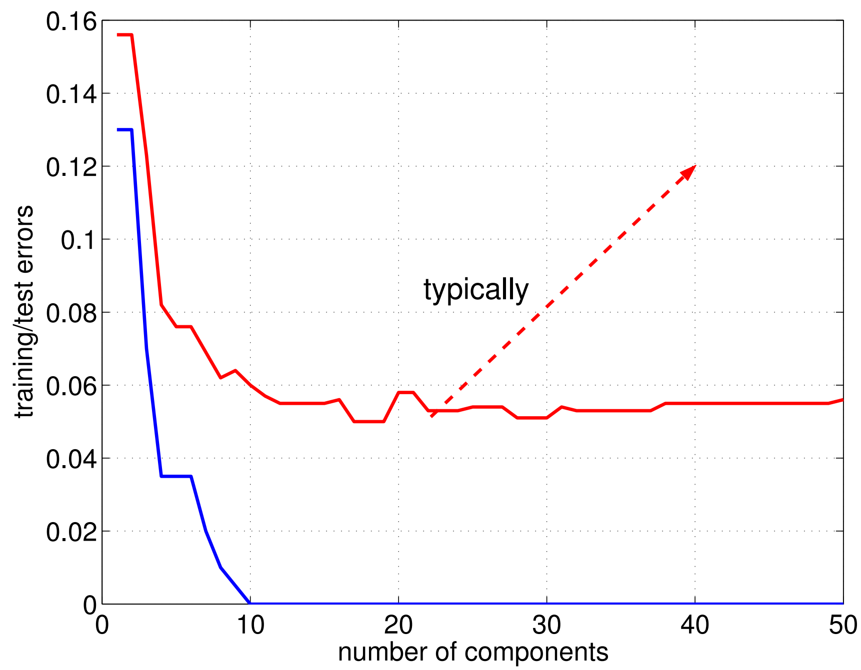
- We could use SVMs in a postprocessing step to reoptimize

$$f(\mathbf{x};\alpha) = \alpha_1\phi_1(\mathbf{x}) + \ldots \alpha_m\phi_m(\mathbf{x})$$

with respect to $\alpha_1, \ldots, \alpha_m$. This is not necessarily a good idea.



boosting



svm postprocessing

Topics

- Combination of classifiers: boosting
  - decision stumps, reweighting, adaboost
  - examples,
  - margin and performance
- Complexity and model selection
  - learning and VC dimension
  - structural risk minimization