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 $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of examples and $(\pm 1)$ labels and a family of component classifiers such as decision stumps:

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

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) $h(x; \hat{\theta})$ based on the smaller training set

• The resulting combined classifier is obtained by voting

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

![Margin Diagram]

-1 0 1

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

$$
\text{Weight } p(i) \text{ on } (x_i, y_i): \quad p(i) \propto \exp \left\{ -y_i \sum_{k=1}^{m} h(x_i; \hat{\theta}_k) \right\}
$$

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 \((x_1, y_1), \ldots, (x_n, y_n)\) with binary ±1 labels \(y_i\).
  b) A set of “weak” binary (±1) classifiers \(h(x; \theta)\) such as decision stumps

\[
h(x; \theta) = \text{sign}(w_1 x_k - w_0)
\]

  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 $h(x; \hat{\theta}_k)$ such that its weighted training error

$$\epsilon_k = \sum_{i=1}^{n} p_k(i) \left[ [y_i \neq h(x_i; \hat{\theta}_k)] \right]$$

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: $\hat{\alpha}_k = 0.5 \log \left( \frac{(1 - \epsilon_k)/\epsilon_k}{\epsilon_k} \right)$ (decorrelation)

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

- The final classifier after $m$ boosting iterations is given by

$$\hat{h}(x) = \text{sign} \left( \frac{\hat{\alpha}_1 h(x; \hat{\theta}_1) + \ldots + \hat{\alpha}_m h(x; \hat{\theta}_m)}{\hat{\alpha}_1 + \ldots + \hat{\alpha}_m} \right)$$
Boosting: example
Boosting: example cont’d
Boosting: example cont’d
Boosting performance

- Training/test errors for the combined classifier

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

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)

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

- Boosting iterations tend to increase the margin

$$y \left( \frac{\hat{\alpha}_1 h(x; \hat{\theta}_1) + \ldots + \hat{\alpha}_m h(x; \hat{\theta}_m)}{\hat{\alpha}_1 + \ldots + \hat{\alpha}_m} \right)$$
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 \((x_1, y_1), \ldots, (x_n, y_n)\) where the labels are binary \(\pm 1\)
- Linear separator:

\[
f(x; w, w_0) = w_0 + x_1w_1 + \ldots x_dw_d
= w_0 + w^T x
\]

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

\[
y_i [w_0 + w^T x_i] - 1 \geq 0, \quad i = 1, \ldots, n
\]
Support vector machine

- We minimize

\[ \|w\|^2/2 = w^T w/2 = \sum_{j=1}^{d} w_j^2/2 \]

subject to the classification constraints

\[ y_i [w_0 + w^T x_i] - 1 \geq 0, \quad i = 1, \ldots, n \]

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