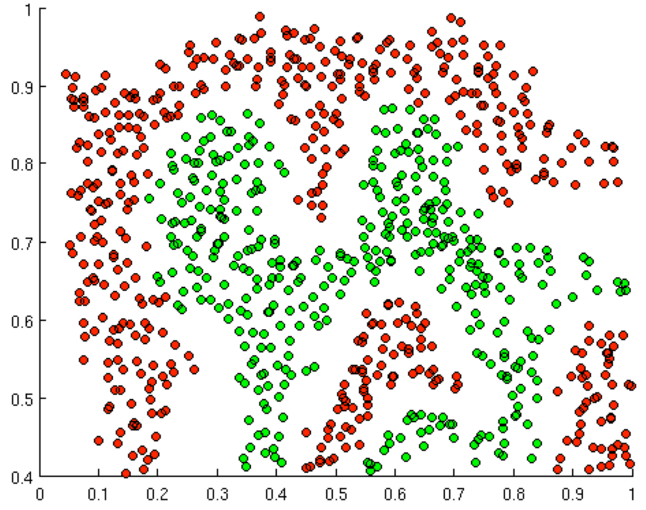
BOOSTING

J. Elder

CSE 4404/5327 Introduction to Machine Learning and Pattern Recognition

Limitations of Classifiers

- All of the classifiers we have studied have limitations: if the problem is 'interesting', they don't tend to perform perfectly.
- Often these limitations stem from the inability of a single classifier to mold a decision surface into exactly the right shape.
- As a consequence, the classifier may perform well for part of the input space, but poorly in another part of the input space.
- How can we overcome this problem?



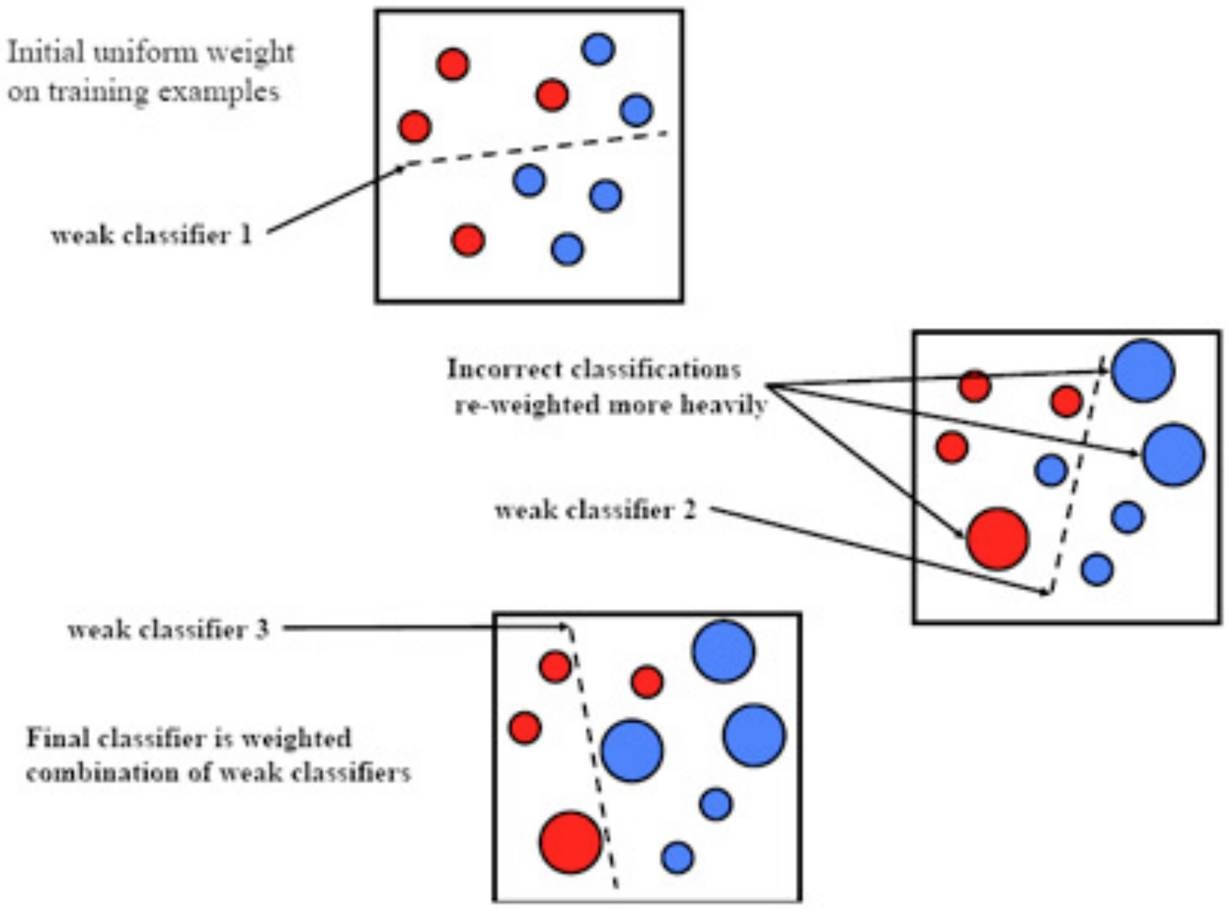
Combining Classifiers

- One natural idea is to try to combine multiple classifiers, each of which captures some aspect of the problem.
- Together, the ensemble of classifiers may be able to model complex decision surfaces.

Boosting

- Boosting is one method for combining multiple 'weak' classifiers into a single 'strong' classifier.
- The approach is greedy. We begin by selecting the best weak classifier (i.e., the one with lowest error on the training dataset).
- We then repeat the following 3 steps until no improvement can be made or an error criterion has been reached:
    1. Reweight the input vectors, up-weighting those that were classified incorrectly by the last classifier selected.
    2. Again select the best weak classifier, on the reweighted training data.
    3. Linearly combine this classifier with the classifiers already selected, with an appropriate weighting.

Adaboost: Reweighting


Initial uniform weight on training examples
weak classifier 1
Incorrect classifications re-weighted more heavily
weak classifier 2
weak classifier 3
Final classifier is weighted combination of weak classifiers

$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

Adaboost

Yoav Freund and Robert E. Schapire (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*,148–156.

Yoav Freund

Robert Schapire

AdaBoost

- AdaBoost (**Ada**ptive **Boost**ing) is probably the most popular form of boosting.

  Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ be the training data, with $y_i \in \{-1, 1\}$.

- We seek an optimal classifier of the form

  $$F(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k \phi(\mathbf{x};\theta_k), \quad \alpha_k > 0 \forall k.$$

  Here, each $\phi(\mathbf{x};\theta_k)$ is a weak classifier that evaluates to -1 or +1, and input vectors $\mathbf{x}$ are classified according to the sign of $F(\mathbf{x})$.

  The parameters α_k and θ_k are selected at each stage to minimize

  $$\sum_{i=1}^{N} \exp\left(-y_i F(\mathbf{x}_i)\right).$$

AdaBoost: Learning the Parameters

$$F(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k \phi\left(\mathbf{x};\theta_k\right)$$

The parameters α_k and θ_k are selected at each stage to minimize

$$\sum_{i=1}^{N} \exp\left(-y_i F\left(\mathbf{x}_i\right)\right).$$

Note that learning is greedy:

at each stage k only the parameters α_k and θ_k are learned.

Parameters at previous stages $1\ldots k-1$ remain fixed.

At stage k we can thus define the problem as

finding the parameters α_k and θ_k that minimize

$$J\left(\alpha,\theta\right) = \sum_{i=1}^{N} \exp\left(-y_i\left(F_{k-1}\left(\mathbf{x}_i\right) + \alpha\phi\left(\mathbf{x}_i;\theta\right)\right)\right).$$

AdaBoost: Learning the Parameters

$$J(\alpha,\theta) = \sum_{i=1}^{N} \exp\left(-y_i\left(F_{k-1}(\mathbf{x}_i) + \alpha\phi(\mathbf{x}_i;\theta)\right)\right)$$

$$= \sum_{i=1}^{N} w_i^{(k)} \exp\left(-y_i\alpha\phi(\mathbf{x}_i;\theta)\right)$$

where

$$w_i^{(k)} \triangleq \exp\left(-y_i F_{k-1}(\mathbf{x}_i)\right)$$

AdaBoost: Learning the Parameters

$$J(\alpha,\theta) = \sum_{i=1}^{N} w_i^{(k)} \exp\left(-y_i \alpha \phi(\mathbf{x}_i;\theta)\right)$$

Observe that $\exp\left(-y_i \alpha \phi(\mathbf{x}_i;\theta)\right)$ evaluates to

$e^{-\alpha}$ for input vectors that are correctly classified and

$e^{+\alpha}$ for vectors that are incorrectly classified.

Thus the cost function can be re-expressed as

$$J(\alpha,\theta) = e^{-\alpha} \sum_{i \in I_c(\theta)} w_i^{(k)} + e^{+\alpha} \sum_{i \in I_e(\theta)} w_i^{(k)},$$

where I_c and I_e are the index sets of inputs correctly and incorrectly classified by the k^{th} weak classifier, respectively.

AdaBoost: Learning the Parameters

$$J(\alpha,\theta) = e^{-\alpha} \sum_{i \in I_c(\theta)} w_i^{(k)} + e^{+\alpha} \sum_{i \in I_e(\theta)} w_i^{(k)},$$

where I_c and I_e are the index sets of inputs correctly and incorrectly classified by the k^{th} weak classifier, respectively.

Thus we seek the parameters that minimize the sum of the weights for the misclassified inputs.

Note that this does not depend upon α! Thus we have:

$$\theta_k = \arg\min_{\theta} \sum_{i \in I_e(\theta)} w_i^{(k)}.$$

AdaBoost: Learning the Parameters

$$J(\alpha,\theta) = e^{-\alpha} \sum_{i\in I_c(\theta)} w_i^{(k)} + e^{+\alpha} \sum_{i\in I_e(\theta)} w_i^{(k)},$$

where I_c and I_e are the index sets of inputs correctly and incorrectly classified by the k^{th} weak classifier, respectively.

Once θ_k is chosen, we need to determine α_k.

$$\frac{\partial}{\partial\alpha} J(\alpha,\theta) = -e^{-\alpha} \sum_{i\in I_c(\theta)} w_i^{(k)} + e^{+\alpha} \sum_{i\in I_e(\theta)} w_i^{(k)} = 0$$

$$\rightarrow e^{2\alpha} = \frac{\sum_{i\in I_c(\theta)} w_i^{(k)}}{\sum_{i\in I_e(\theta)} w_i^{(k)}} \quad \rightarrow \alpha = \frac{1}{2}\log\left(\frac{\sum_{i\in I_c(\theta)} w_i^{(k)}}{\sum_{i\in I_e(\theta)} w_i^{(k)}}\right)$$

AdaBoost: Learning the Parameters

$$\alpha_k = \frac{1}{2}\log\left(\frac{\sum_{i \in I_c(\theta)} w_i^{(k)}}{\sum_{i \in I_e(\theta)} w_i^{(k)}}\right)$$

We define the weighted error rate ε_k as $\varepsilon_k = \frac{\sum_{i \in I_e(\theta)} w_i^{(k)}}{\sum_{i=1}^{N} w_i^{(k)}}$.

Then we have $\alpha_k = \frac{1}{2}\log\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$

AdaBoost: Learning the Parameters

$$\alpha_k = \frac{1}{2}\log\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$$

Note that for the optimal θ, $\varepsilon_k < \frac{1}{2}$.

(Otherwise, we could just flip the sign and have a better weak classifier.)

Thus $\alpha_k > 0$, and is larger for weak classifiers with lower weighted error.

Weights

$$F(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k \phi\left(\mathbf{x}; \theta_k\right)$$

$$w_i^{(k)} \triangleq \exp\left(-y_i F_{k-1}\left(\mathbf{x}_i\right)\right)$$

$$\rightarrow w_i^{(k+1)} \triangleq \exp\left(-y_i F_k\left(\mathbf{x}_i\right)\right) = \exp\left(-y_i F_{k-1}\left(\mathbf{x}_i\right)\right) \exp\left(-y_i \alpha_k \phi\left(\mathbf{x}_i, \theta\right)\right)$$

Thus the weight for an input decreases if the current weak classifier classifies it correctly, and increases if it classifies it incorrectly.

The amount of the change depends on the margin: inputs with a large negative margin increase the weight a lot, inputs with a large positive margin decrease the weight a lot.

Convergence

- Boosting enjoys a very strong convergence property:
  - Convergence is not generally monotonic.
  - However, as the number of iterations $\rightarrow \infty$, error on the training dataset $\rightarrow$ 0:

Let the weighted error rate of the k^{th} weak classifier be $\frac{1}{2} - \gamma_k$.

Then it can be shown that after K iterations,

the training error of the strong classifier is at most $\exp\left(-2\sum_{k=1}^{K}\gamma_k^2\right)$

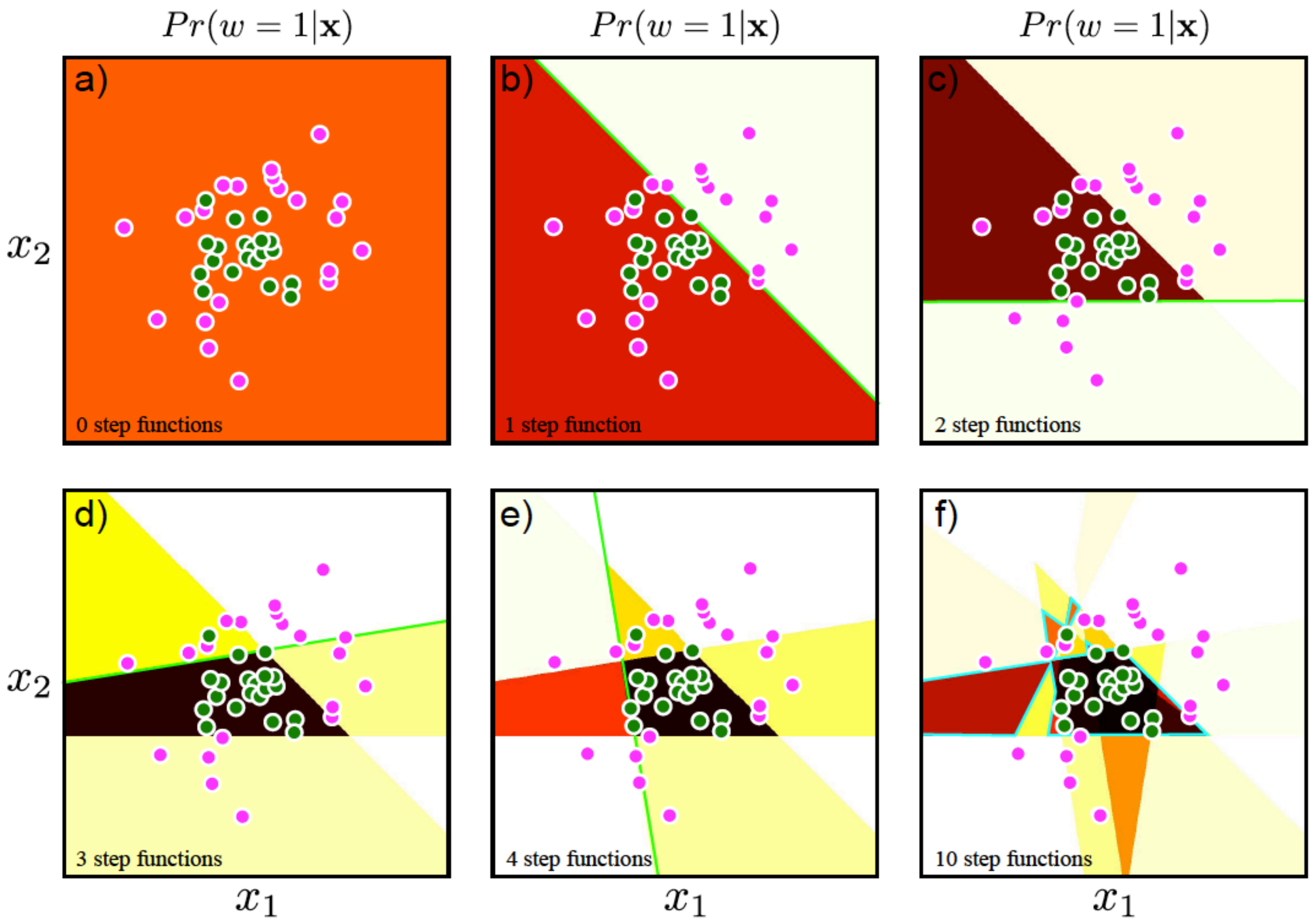
Generalization

- Adaboost is found empirically to have good generalization properties:
  - As the number K of weak classifiers grows, the error on the test set tends to decrease and then level off
  - Unlike many other methods, Adaboost does not always suffer from overlearning, even for large K.
  - The error on the test set tends to decrease even after the error on the training set is 0.

Limitations

- Adaboost is not probabilistic, and thus does not provide a measure of confidence for each classification.
- Can be sensitive to noise and outliers

Example


$Pr(w = 1|\mathbf{x})$
$Pr(w = 1|\mathbf{x})$
$Pr(w = 1|\mathbf{x})$
a)
0 step functions
b)
1 step function
c)
2 step functions
d)
3 step functions
e)
4 step functions
f)
10 step functions
x_2
x_2
x_1
x_1
x_1

- Example

Boosting in MATLAB

- MATLAB’s Statistics Toolbox provides various boosting algorithms through the function **fitensemble()**.