

# Combining Models

Oliver Schulte - CMPT 726

Bishop PRML Ch. 14

# Outline

Boosting

Mixture of Experts

# Combining Models

- Motivation: let's say we have a number of models for a problem
  - e.g. Regression with polynomials (different degree)
  - e.g. Classification with support vector machines (kernel type, parameters)
- Often, improved performance can be obtained by combining different models. (More on next slide).
- But how do we combine classifiers?

# Why Combining Works

Intuitively, two reasons.

1. *Portfolio Diversification*: if you combine options that on average perform equally well, you keep the same average performance but you *lower your risk* (i.e., variance).
  - E.g., invest in Gold and in Equities.
2. The **Boosting Theorem**: Suppose you have a learning algorithm  $L$  with the following **PAC** guarantee: Given a sample of size  $n$  drawn from *any* distribution, the algorithm is guaranteed to produce a classifier that with high confidence has accuracy  $> 1/2$ . Then you can repeatedly run  $L$  and combine the resulting classifiers in such a way that with sufficiently many classifiers, with high confidence you can achieve *any* desired degree of accuracy  $< 1$ .

# Committees

- A combination of models is often called a **committee**
- Simplest way to combine models is to just average them together:

$$y_{COM}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x})$$

- It turns out this simple method is better than (or same as) the individual models on average (in **expectation**)
  - And usually slightly better

## Error of Individual Models

- Consider individual models  $y_m(\mathbf{x})$ , assume they can be written as true value plus error:

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x})$$

- The expected value of the error of an individual model is then:

$$\mathbb{E}_{\mathbf{x}}[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2] = \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})^2]$$

- The average error made by an individual model is then:

$$E_{AV} = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})^2]$$

## Error of Individual Models

- Consider individual models  $y_m(\mathbf{x})$ , assume they can be written as true value plus error:

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x})$$

- The expected value of the error of an individual model is then:

$$\mathbb{E}_{\mathbf{x}}[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2] = \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})^2]$$

- The average error made by an individual model is then:

$$E_{AV} = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})^2]$$

## Error of Individual Models

- Consider individual models  $y_m(\mathbf{x})$ , assume they can be written as true value plus error:

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x})$$

- The expected value of the error of an individual model is then:

$$\mathbb{E}_{\mathbf{x}}[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2] = \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})^2]$$

- The average error made by an individual model is then:

$$E_{AV} = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})^2]$$



## Error of Committee

- The committee

$$y_{COM}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x})$$

has expected error

$$E_{COM} = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \left( \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}) \right) - h(\mathbf{x}) \right\}^2 \right]$$

$$= \mathbb{E}_{\mathbf{x}} \left[ \left\{ \left( \frac{1}{M} \sum_{m=1}^M h(\mathbf{x}) + \epsilon_m(\mathbf{x}) \right) - h(\mathbf{x}) \right\}^2 \right]$$

$$= \mathbb{E}_{\mathbf{x}} \left[ \left\{ \left( \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right) + h(\mathbf{x}) - h(\mathbf{x}) \right\}^2 \right] = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right]$$

## Error of Committee

- The committee

$$y_{COM}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x})$$

has expected error

$$E_{COM} = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \left( \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}) \right) - h(\mathbf{x}) \right\}^2 \right]$$

$$= \mathbb{E}_{\mathbf{x}} \left[ \left\{ \left( \frac{1}{M} \sum_{m=1}^M h(\mathbf{x}) + \epsilon_m(\mathbf{x}) \right) - h(\mathbf{x}) \right\}^2 \right]$$

$$= \mathbb{E}_{\mathbf{x}} \left[ \left\{ \left( \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right) + h(\mathbf{x}) - h(\mathbf{x}) \right\}^2 \right] = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right]$$

## Error of Committee

- The committee

$$y_{COM}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x})$$

has expected error

$$E_{COM} = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \left( \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}) \right) - h(\mathbf{x}) \right\}^2 \right]$$

$$= \mathbb{E}_{\mathbf{x}} \left[ \left\{ \left( \frac{1}{M} \sum_{m=1}^M h(\mathbf{x}) + \epsilon_m(\mathbf{x}) \right) - h(\mathbf{x}) \right\}^2 \right]$$

$$= \mathbb{E}_{\mathbf{x}} \left[ \left\{ \left( \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right) + h(\mathbf{x}) - h(\mathbf{x}) \right\}^2 \right] = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right]$$

## Committee Error vs. Individual Error

- So, the committee error is

$$E_{COM} = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right] = \frac{1}{M^2} \sum_{m=1}^M \sum_{n=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x}) \epsilon_n(\mathbf{x})]$$

- If we assume errors are uncorrelated,  $\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x}) \epsilon_n(\mathbf{x})] = 0$  when  $m \neq n$ , then:

$$E_{COM} = \frac{1}{M^2} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2] = \frac{1}{M} E_{AV}$$

- However, errors are rarely uncorrelated
  - For example, if all errors are the same,  $\epsilon_m(\mathbf{x}) = \epsilon_n(\mathbf{x})$ , then  $E_{COM} = E_{AV}$
  - Using Jensen's inequality (convex functions), can show  $E_{COM} \leq E_{AV}$

## Committee Error vs. Individual Error

- So, the committee error is

$$E_{COM} = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right] = \frac{1}{M^2} \sum_{m=1}^M \sum_{n=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x}) \epsilon_n(\mathbf{x})]$$

- If we assume errors are uncorrelated,  $\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x}) \epsilon_n(\mathbf{x})] = 0$  when  $m \neq n$ , then:

$$E_{COM} = \frac{1}{M^2} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2] = \frac{1}{M} E_{AV}$$

- However, errors are rarely uncorrelated
  - For example, if all errors are the same,  $\epsilon_m(\mathbf{x}) = \epsilon_n(\mathbf{x})$ , then  $E_{COM} = E_{AV}$
  - Using Jensen's inequality (convex functions), can show  $E_{COM} \leq E_{AV}$

## Committee Error vs. Individual Error

- So, the committee error is

$$E_{COM} = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right] = \frac{1}{M^2} \sum_{m=1}^M \sum_{n=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x}) \epsilon_n(\mathbf{x})]$$

- If we assume errors are uncorrelated,  $\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x}) \epsilon_n(\mathbf{x})] = 0$  when  $m \neq n$ , then:

$$E_{COM} = \frac{1}{M^2} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2] = \frac{1}{M} E_{AV}$$

- However, errors are rarely uncorrelated
  - For example, if all errors are the same,  $\epsilon_m(\mathbf{x}) = \epsilon_n(\mathbf{x})$ , then  $E_{COM} = E_{AV}$
  - Using Jensen's inequality (convex functions), can show  $E_{COM} \leq E_{AV}$

# Outline

Boosting

Mixture of Experts

# Outline

Boosting

Mixture of Experts



# Boosting

- **Boosting** is a technique for combining **classifiers** into a **committee**
  - We describe **AdaBoost** (adaptive boosting), the most commonly used variant. (Freund and Schapire 1995, Gödel Prize 2003).
- Boosting is a **meta-learning** technique
  - Combines a set of classifiers trained using their own learning algorithms
  - **Magic: can work well even if those classifiers only perform slightly better than random!**

# Community Notes on Boosting

- Boosting with Decision Trees was used by Dugan O'Neill (SFU, Physics) to find evidence for the top quark. (Yes, this is a big deal.) [http://www.phy.bnl.gov/edg/samba/oneil\\_summary.pdf](http://www.phy.bnl.gov/edg/samba/oneil_summary.pdf).
- PAC Learning was introduced to CS by Leslie Valiant, this year's Turing Award winner. He's giving a distinguished lecture award at SFU this term, don't miss it! October 13.
- Boosting demo <http://cseweb.ucsd.edu/~yfreund/adaboost/index.html>.

# Boosting Model

- We consider two-class classification problems, training data  $(\mathbf{x}_i, t_i)$ , with  $t_i \in \{-1, 1\}$
- In boosting we build a “linear” classifier of the form:

$$y(\mathbf{x}) = \sum_{m=1}^M \alpha_m y_m(\mathbf{x})$$

- A committee of classifiers, with weights
- In boosting terminology:
  - Each  $y_m(\mathbf{x})$  is called a **weak learner** or **base classifier**
  - Final classifier  $y(\mathbf{x})$  is called **strong learner**
- **Learning problem:** how do we choose the weak learners  $y_m(\mathbf{x})$  and weights  $\alpha_m$ ?

# Boosting Model

- We consider two-class classification problems, training data  $(\mathbf{x}_i, t_i)$ , with  $t_i \in \{-1, 1\}$
- In boosting we build a “linear” classifier of the form:

$$y(\mathbf{x}) = \sum_{m=1}^M \alpha_m y_m(\mathbf{x})$$

- A committee of classifiers, with weights
- In boosting terminology:
  - Each  $y_m(\mathbf{x})$  is called a **weak learner** or **base classifier**
  - Final classifier  $y(\mathbf{x})$  is called **strong learner**
- **Learning problem:** how do we choose the weak learners  $y_m(\mathbf{x})$  and weights  $\alpha_m$ ?

# Boosting Model

- We consider two-class classification problems, training data  $(\mathbf{x}_i, t_i)$ , with  $t_i \in \{-1, 1\}$

- In boosting we build a “linear” classifier of the form:

$$y(\mathbf{x}) = \sum_{m=1}^M \alpha_m y_m(\mathbf{x})$$

- A committee of classifiers, with weights
- In boosting terminology:
  - Each  $y_m(\mathbf{x})$  is called a **weak learner** or **base classifier**
  - Final classifier  $y(\mathbf{x})$  is called **strong learner**
- **Learning problem: how do we choose the weak learners  $y_m(\mathbf{x})$  and weights  $\alpha_m$ ?**

## Boosting Intuition

- The weights  $\alpha_k$  reflect the training error of the different classifiers.
- Classifier  $\alpha_{k+1}$  is trained on *weighted examples*, where instances misclassified by the committee

$$y^k(\mathbf{x}) = \sum_{m=1}^k \alpha_m y_m(\mathbf{x})$$

receive higher weight.

- The instance weights can be interpreted as *resampling*: build a new sample where instances with higher weight occur more frequently.

## Example - Thresholds

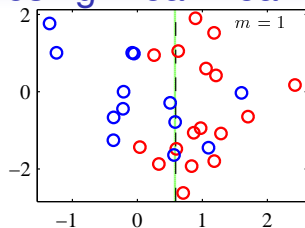
- Let's consider a simple example where weak learners are thresholds
- i.e. Each  $y_m(\mathbf{x})$  is of the form:

$$y_m(\mathbf{x}) = x_i > \theta$$

- To allow different directions of threshold, include  $p \in \{-1, +1\}$ :

$$y_m(\mathbf{x}) = px_i > p\theta$$

## Choosing Weak Learners



- Boosting is a greedy strategy for building the strong learner

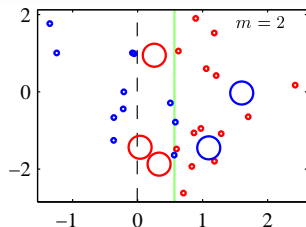
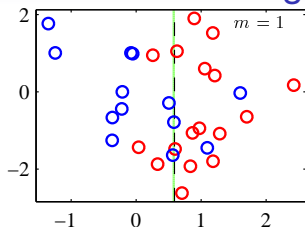
$$y(\mathbf{x}) = \sum_{m=1}^M \alpha_m y_m(\mathbf{x})$$

- Start by choosing the **best** weak learner, use it as  $y_1(\mathbf{x})$ 
  - **Best** is defined as that which minimizes number of mistakes made (0-1 classification loss)
- i.e. Search over all  $p, \theta, i$  to find best

$$y_1(\mathbf{x}) = px_i > p\theta$$



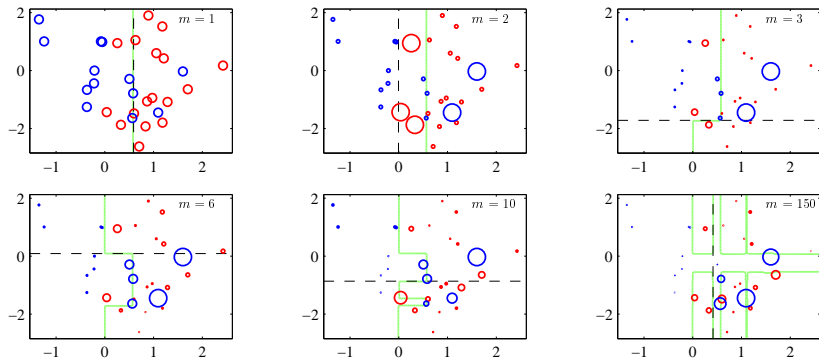
## Choosing Weak Learners



- The first weak learner  $y_1(\mathbf{x})$  made some mistakes
- Choose the second weak learner  $y_2(\mathbf{x})$  to try to get those ones correct
  - **Best** is now defined as that which minimizes **weighted** number of mistakes made
  - Higher weight given to those  $y_1(\mathbf{x})$  got incorrect
- Strong learner now

$$y(\mathbf{x}) = \alpha_1 y_1(\mathbf{x}) + \alpha_2 y_2(\mathbf{x})$$

# Choosing Weak Learners



- Repeat: reweight examples and choose new weak learner based on weights
- **Green line** shows decision boundary of **strong learner**

## What About Those Weights?

- So exactly how should we choose the weights for the examples when classified incorrectly?
- And what should the  $\alpha_m$  be for combining the **weak learners**  $y_m(\mathbf{x})$ ?
- Original approach: make sure the strong learner satisfies the PAC guarantee.
- Alternative view: define a loss function, and choose parameters to minimize it.

# Exponential Loss

- Boosting attempts to minimize the **exponential loss**

$$E_n = \exp\{-t_n y(\mathbf{x}_n)\}$$

error on  $n^{\text{th}}$  training example

- **Exponential loss** is differentiable approximation to 0/1 loss
  - Better for optimization
- Total error

$$E = \sum_{n=1}^N \exp\{-t_n y(\mathbf{x}_n)\}$$

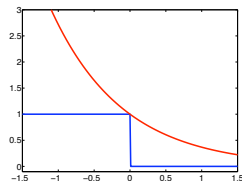


figure from G. Shakhnarovich

# Exponential Loss

- Boosting attempts to minimize the **exponential loss**

$$E_n = \exp\{-t_n y(\mathbf{x}_n)\}$$

error on  $n^{\text{th}}$  training example

- **Exponential loss** is differentiable approximation to 0/1 loss
  - Better for optimization
- Total error

$$E = \sum_{n=1}^N \exp\{-t_n y(\mathbf{x}_n)\}$$

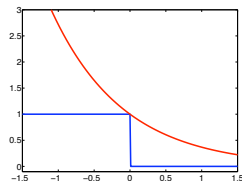


figure from G. Shakhnarovich

## Minimizing Exponential Loss

- Let's assume we've already chosen weak learners  $y_1(\mathbf{x}), \dots, y_{m-1}(\mathbf{x})$  and their weights  $\alpha_1, \dots, \alpha_{m-1}$ 
  - Define  $f_{m-1}(\mathbf{x}) = \alpha_1 y_1(\mathbf{x}) + \dots + \alpha_{m-1} y_{m-1}(\mathbf{x})$
- Just focus on choosing  $y_m(\mathbf{x})$  and  $\alpha_m$ 
  - Greedy optimization strategy
- Total error using exponential loss is:

$$\begin{aligned}
 E &= \sum_{n=1}^N \exp\{-t_n y(\mathbf{x}_n)\} = \sum_{n=1}^N \exp\{-t_n [f_{m-1}(\mathbf{x}_n) + \alpha_m y_m(\mathbf{x})]\} \\
 &= \sum_{n=1}^N \exp\{-t_n f_{m-1}(\mathbf{x}_n) - t_n \alpha_m y_m(\mathbf{x})\} \\
 &= \sum_{n=1}^N \underbrace{\exp\{-t_n f_{m-1}(\mathbf{x}_n)\}}_{\text{weight } w_n^{(m)}} \exp\{-t_n \alpha_m y_m(\mathbf{x})\}
 \end{aligned}$$

## Minimizing Exponential Loss

- Let's assume we've already chosen weak learners  $y_1(\mathbf{x}), \dots, y_{m-1}(\mathbf{x})$  and their weights  $\alpha_1, \dots, \alpha_{m-1}$ 
  - Define  $f_{m-1}(\mathbf{x}) = \alpha_1 y_1(\mathbf{x}) + \dots + \alpha_{m-1} y_{m-1}(\mathbf{x})$
- Just focus on choosing  $y_m(\mathbf{x})$  and  $\alpha_m$ 
  - Greedy optimization strategy
- Total error using exponential loss is:

$$\begin{aligned}
 E &= \sum_{n=1}^N \exp\{-t_n y(\mathbf{x}_n)\} = \sum_{n=1}^N \exp\{-t_n [f_{m-1}(\mathbf{x}_n) + \alpha_m y_m(\mathbf{x})]\} \\
 &= \sum_{n=1}^N \exp\{-t_n f_{m-1}(\mathbf{x}_n) - t_n \alpha_m y_m(\mathbf{x})\} \\
 &= \sum_{n=1}^N \underbrace{\exp\{-t_n f_{m-1}(\mathbf{x}_n)\}}_{\text{weight } w_n^{(m)}} \exp\{-t_n \alpha_m y_m(\mathbf{x})\}
 \end{aligned}$$

## Minimizing Exponential Loss

- Let's assume we've already chosen weak learners  $y_1(\mathbf{x}), \dots, y_{m-1}(\mathbf{x})$  and their weights  $\alpha_1, \dots, \alpha_{m-1}$ 
  - Define  $f_{m-1}(\mathbf{x}) = \alpha_1 y_1(\mathbf{x}) + \dots + \alpha_{m-1} y_{m-1}(\mathbf{x})$
- Just focus on choosing  $y_m(\mathbf{x})$  and  $\alpha_m$ 
  - Greedy optimization strategy
- Total error using **exponential loss** is:

$$\begin{aligned}
 E &= \sum_{n=1}^N \exp\{-t_n y(\mathbf{x}_n)\} = \sum_{n=1}^N \exp\{-t_n [f_{m-1}(\mathbf{x}_n) + \alpha_m y_m(\mathbf{x})]\} \\
 &= \sum_{n=1}^N \exp\{-t_n f_{m-1}(\mathbf{x}_n) - t_n \alpha_m y_m(\mathbf{x})\} \\
 &= \sum_{n=1}^N \underbrace{\exp\{-t_n f_{m-1}(\mathbf{x}_n)\}}_{\text{weight } w_n^{(m)}} \exp\{-t_n \alpha_m y_m(\mathbf{x})\}
 \end{aligned}$$



# Weighted Loss

- On the  $m^{\text{th}}$  iteration of boosting, we are choosing  $y_m$  and  $\alpha_m$  to minimize the weighted loss:

$$E = \sum_{n=1}^N w_n^{(m)} \exp\{-t_n \alpha_m y_m(\mathbf{x})\}$$

where  $w_n^{(m)} = \exp\{-t_n f_{m-1}(\mathbf{x}_n)\}$

- Can define these as weights since they are constant wrt  $y_m$  and  $\alpha_m$ 
  - We'll see they're the right weights to use

## Minimization wrt $y_m$

- Consider the weighted loss

$$E = \sum_{n=1}^N w_n^{(m)} e^{-t_n \alpha_m y_m(\mathbf{x})} = e^{-\alpha_m} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m} \sum_{n \in \mathcal{M}_m} w_n^{(m)}$$

where  $\mathcal{T}_m$  is the set of points correctly classified by the choice of  $y_m(\mathbf{x})$ , and  $\mathcal{M}_m$  those that are not

$$\begin{aligned} E &= e^{\alpha_m} \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m} \sum_{n=1}^N w_n^{(m)} (1 - I(y_m(\mathbf{x}_n) \neq t_n)) \\ &= (e^{\alpha_m} - e^{-\alpha_m}) \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m} \sum_{n=1}^N w_n^{(m)} \end{aligned}$$

- Since the second term is a constant wrt  $y_m$  and  $e^{\alpha_m} - e^{-\alpha_m} > 0$  if  $\alpha_m > 0$ , best  $y_m$  minimizes weighted 0-1 loss  $\sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$ .

## Minimization wrt $y_m$

- Consider the weighted loss

$$E = \sum_{n=1}^N w_n^{(m)} e^{-t_n \alpha_m y_m(\mathbf{x})} = e^{-\alpha_m} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m} \sum_{n \in \mathcal{M}_m} w_n^{(m)}$$

where  $\mathcal{T}_m$  is the set of points correctly classified by the choice of  $y_m(\mathbf{x})$ , and  $\mathcal{M}_m$  those that are not

$$\begin{aligned} E &= e^{\alpha_m} \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m} \sum_{n=1}^N w_n^{(m)} (1 - I(y_m(\mathbf{x}_n) \neq t_n)) \\ &= (e^{\alpha_m} - e^{-\alpha_m}) \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m} \sum_{n=1}^N w_n^{(m)} \end{aligned}$$

- Since the second term is a constant wrt  $y_m$  and  $e^{\alpha_m} - e^{-\alpha_m} > 0$  if  $\alpha_m > 0$ , best  $y_m$  minimizes weighted 0-1 loss  $\sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$ .

## Minimization wrt $y_m$

- Consider the weighted loss

$$E = \sum_{n=1}^N w_n^{(m)} e^{-t_n \alpha_m y_m(\mathbf{x})} = e^{-\alpha_m} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m} \sum_{n \in \mathcal{M}_m} w_n^{(m)}$$

where  $\mathcal{T}_m$  is the set of points correctly classified by the choice of  $y_m(\mathbf{x})$ , and  $\mathcal{M}_m$  those that are not

$$\begin{aligned} E &= e^{\alpha_m} \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m} \sum_{n=1}^N w_n^{(m)} (1 - I(y_m(\mathbf{x}_n) \neq t_n)) \\ &= (e^{\alpha_m} - e^{-\alpha_m}) \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m} \sum_{n=1}^N w_n^{(m)} \end{aligned}$$

- Since the second term is a constant wrt  $y_m$  and  $e^{\alpha_m} - e^{-\alpha_m} > 0$  if  $\alpha_m > 0$ , best  $y_m$  minimizes weighted 0-1 loss  $\sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$ .

## Choosing $\alpha_m$

- So best  $y_m$  minimizes weighted 0-1 loss **regardless of  $\alpha_m$**
- How should we set  $\alpha_m$  given this best  $y_m$ ?
- Recall from above:

$$\begin{aligned} E &= e^{\alpha_m} \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m} \sum_{n=1}^N w_n^{(m)} (1 - I(y_m(\mathbf{x}_n) \neq t_n)) \\ &= e^{\alpha_m} \epsilon_m + e^{-\alpha_m} (1 - \epsilon_m) \end{aligned}$$

where we define  $\epsilon_m$  to be the **weighted error of  $y_m$**

- Calculus:  $\alpha_m = \frac{1}{2} \ln \frac{1-\epsilon_m}{\epsilon_m}$

## Choosing $\alpha_m$

- So best  $y_m$  minimizes weighted 0-1 loss **regardless of  $\alpha_m$**
- How should we set  $\alpha_m$  given this best  $y_m$ ?
- Recall from above:

$$\begin{aligned} E &= e^{\alpha_m} \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m} \sum_{n=1}^N w_n^{(m)} (1 - I(y_m(\mathbf{x}_n) \neq t_n)) \\ &= e^{\alpha_m} \epsilon_m + e^{-\alpha_m} (1 - \epsilon_m) \end{aligned}$$

where we define  $\epsilon_m$  to be the **weighted error of  $y_m$**

- Calculus:  $\alpha_m = \frac{1}{2} \ln \frac{1-\epsilon_m}{\epsilon_m}$

## Choosing $\alpha_m$

- So best  $y_m$  minimizes weighted 0-1 loss **regardless of  $\alpha_m$**
- How should we set  $\alpha_m$  given this best  $y_m$ ?
- Recall from above:

$$\begin{aligned} E &= e^{\alpha_m} \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m} \sum_{n=1}^N w_n^{(m)} (1 - I(y_m(\mathbf{x}_n) \neq t_n)) \\ &= e^{\alpha_m} \epsilon_m + e^{-\alpha_m} (1 - \epsilon_m) \end{aligned}$$

where we define  $\epsilon_m$  to be the **weighted error of  $y_m$**

- Calculus:  $\alpha_m = \frac{1}{2} \ln \frac{1-\epsilon_m}{\epsilon_m}$

# AdaBoost Summary

- Initialize weights  $w_n^{(1)} = 1/N$
- For  $m = 1, \dots, M$  (and while  $\epsilon_m < 1/2$ )
  - Find weak learner  $y_m(\mathbf{x})$  with minimum weighted error

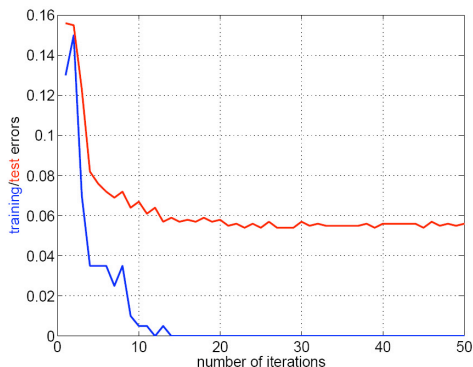
$$\epsilon_m = \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

- Set  $\alpha_m = \frac{1}{2} \ln \frac{1-\epsilon_m}{\epsilon_m}$
  - Update weights  $w_n^{(m+1)} = w_n^{(m)} \exp\{-\alpha_m t_n y_m(\mathbf{x}_n)\}$
  - Normalize weights to sum to one
- Final classifier is

$$y(\mathbf{x}) = \text{sign} \left( \sum_{m=1}^M \alpha_m y_m(\mathbf{x}) \right)$$



# AdaBoost Behaviour



- Typical behaviour:
  - Test error decreases **even after training error is flat** (even zero!)
  - Tends not to overfit

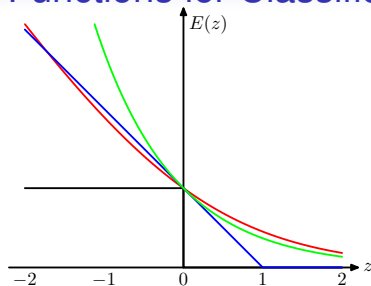
# Boosting the Margin

- Define the **margin** of an example:

$$\gamma(\mathbf{x}_i) = t_i \frac{\alpha_1 y_1(\mathbf{x}_i) + \dots + \alpha_m y_m(\mathbf{x}_i)}{\alpha_1 + \dots + \alpha_m}$$

- Margin is 1 iff all  $y_i$  classify correctly, -1 if none do
- Iterations of AdaBoost increase the margin of training examples (even after training error is zero)

# Loss Functions for Classification



- We revisit a graph from earlier: 0-1 loss, SVM hinge loss, logistic regression cross-entropy loss, and AdaBoost exponential loss are shown
- All are approximations (upper bounds) to 0-1 loss
- Exponential loss leads to simple greedy optimization scheme
- But it has problems with outliers: note different behaviour compared to logistic regression cross-entropy loss for badly mis-classified examples.

# Outline

Boosting

Mixture of Experts

# Mixture of Experts

- The mixture of experts model takes the idea of splitting up regions of space in a probabilistic direction
- The decision on which model to use is probabilistic:

$$p(t|\mathbf{x}) = \sum_{m=1}^M \pi_m(\mathbf{x}) p_m(t|\mathbf{x})$$

- Note that all models  $p_m(t|\mathbf{x})$  are used
- But contributions  $\pi_m(\mathbf{x})$  **depend on input variable  $\mathbf{x}$** 
  - These coefficients  $\pi_m(\mathbf{x})$  are known as **gating functions**
  - Each  $p_m(t|\mathbf{x})$  is an **expert** in a region of input space, the gating functions determine when to use each expert

# Conclusion

- Readings: Ch. 14.3, 14.4
- Methods for combining models
  - Simple averaging into a committee
  - Greedy selection of models to minimize exponential loss ([AdaBoost](#))
  - Select models which are good at particular regions of input space ([decision trees](#), [mixture of experts](#))