Combining Models

Greg Mori - CMPT 419/726

Bishop PRML Ch. 14

Outline

Boosting

Decision Trees

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
- But how can we combine them together?

Committees

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

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

- It turns out this simple method is better than (or same as) the individual models on average (in expectation)
  - And usually slightly better
- But there are better methods, which we shall discuss

Error of Individual Models

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

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

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

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

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

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

Error of Individual Models

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

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

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

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

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

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

Error of Individual Models

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

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

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

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

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

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

Error of Committee

- The committee

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

has expected error

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

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

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

Error of Committee

- The committee

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

has expected error

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

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

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

Error of Committee

- The committee

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

has expected error

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

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

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

Committee Error vs. Individual Error

- So, the committee error is

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

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

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

- However, errors are rarely uncorrelated
  - For example, if all errors are the same, $\epsilon_m(\boldsymbol{x}) = \epsilon_n(\boldsymbol{x})$, 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

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

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

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

- However, errors are rarely uncorrelated
  - For example, if all errors are the same, $\epsilon_m(\boldsymbol{x}) = \epsilon_n(\boldsymbol{x})$, 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

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

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

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

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

Outline

Boosting

Decision Trees

Mixture of Experts

Outline

Boosting

Decision Trees

Mixture of Experts

Boosting

- Boosting is a technique for combining classifiers into a committee
  - We describe AdaBoost (adaptive boosting), the most commonly used variant
- 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!

Boosting Model

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

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

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

Boosting Model

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

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

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

Boosting Model

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

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

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

Example - Thresholds

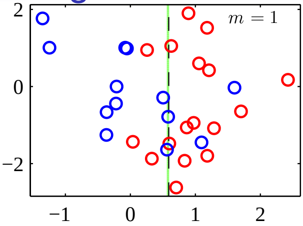
- Let's consider a simple example where weak learners are thresholds
- i.e. Each $y_m(\boldsymbol{x})$ is of the form:

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

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

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

Choosing Weak Learners



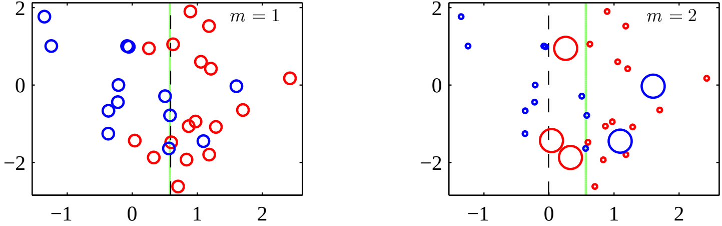
- Boosting is a greedy strategy for building the strong learner

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

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

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

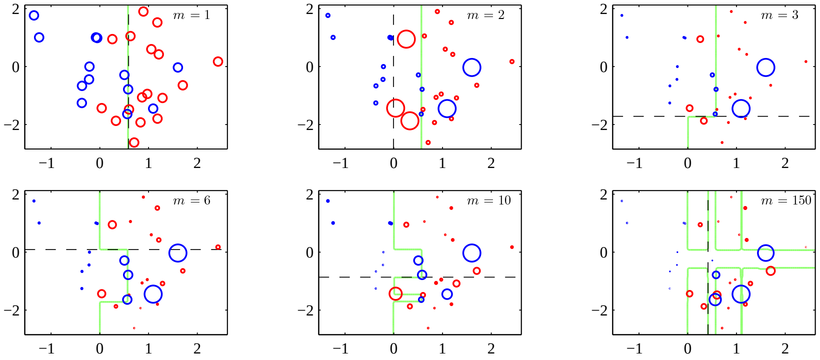
Choosing Weak Learners



- The first weak learner $y_1(\boldsymbol{x})$ made some mistakes
- Choose the second weak learner $y_2(\boldsymbol{x})$ 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 $y_1(\boldsymbol{x})$ got incorrect
- Strong learner now

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

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 α_m be for combining the weak learners $y_m(\boldsymbol{x})$?
- As usual, we define a loss function, and choose these parameters to minimize it

Exponential Loss

- Boosting attempts to minimize the exponential loss

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

  error on n^{th} training example
- Exponential loss is differentiable approximation to 0/1 loss
  - Better for optimization
- Total error

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

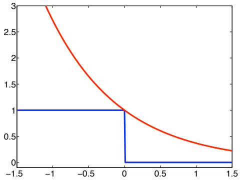


figure from G. Shakhnarovich

Exponential Loss

- Boosting attempts to minimize the exponential loss

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

error on n^{th} training example

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

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

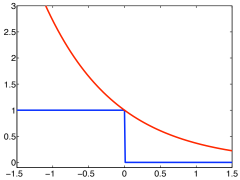


figure from G. Shakhnarovich

Minimizing Exponential Loss

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

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

Minimizing Exponential Loss

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

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

Minimizing Exponential Loss

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

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

Weighted Loss

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

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

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

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

Minimization wrt y_m

- Consider the weighted loss

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

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

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

- 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

Minimization wrt y_m

- Consider the weighted loss

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

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

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

- 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

Minimization wrt y_m

- Consider the weighted loss

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

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

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

- 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

Choosing α_m

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

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

where we define ϵ_m to be the weighted error of y_m

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

Choosing α_m

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

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

where we define ϵ_m to be the weighted error of y_m

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

Choosing α_m

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

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

where we define ϵ_m to be the weighted error of y_m

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

AdaBoost Summary

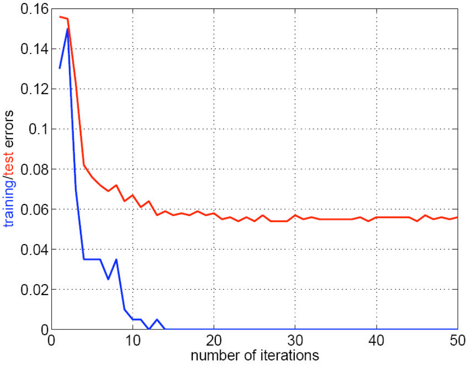
- Initialize weights $w_n^{(1)} = 1/N$
- For $m = 1, \ldots, M$ (and while $\epsilon_m < 1/2$)
  - Find weak learner $y_m(\boldsymbol{x})$ with minimum weighted error

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

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

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

AdaBoost Behaviour



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

from G. Shakhnarovich

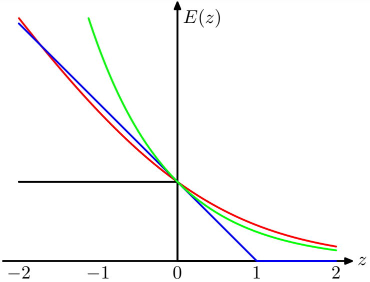
Boosting the Margin

- Define the margin of an example:

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

- 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 behaviour compared to logistic regression cross-entropy loss for badly mis-classified examples

Outline

Boosting

Decision Trees

Mixture of Experts

Carving Up Input Space

- The boosting method for building a committee builds a model:

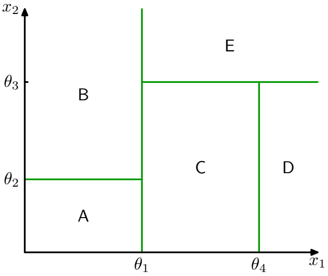
$$y(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x})$$

- Note that the committee is built over all input space
  - Though it can of course behave differently in different regions
- Instead, we could explicitly carve up input space into different regions $\mathcal{R}_m$ and have different committee members act in different regions:

$$y(\boldsymbol{x}) = \sum_{m=1}^{M} 1_{\mathcal{R}_m}(\boldsymbol{x}) y_m(\boldsymbol{x})$$

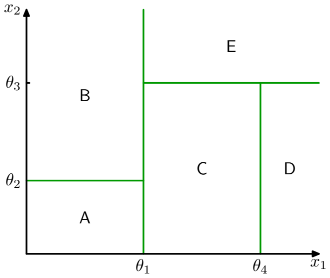
where $1_{\mathcal{R}_m}(\cdot)$ is the indicator function (0 or 1)

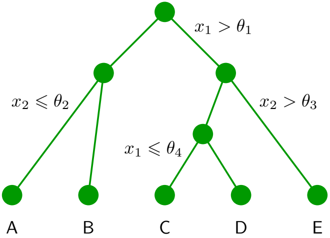
Tree-based Models



- A common method for carving up input space is to use axis-aligned cuboid-shaped regions
- Each model $y_m(\boldsymbol{x})$ would only be responsible for one subregion

Decision Trees





- These splits are commonly chosen in a top-down fashion to form a binary tree
  - These are known as decision trees

Building Decision Trees

- Given a dataset, the learning problem is to decide which is the best tree
- There are (exponentially-exponentially) many different trees to choose from
- Brute force impossible, so use a greedy strategy
    - Start with an empty tree
    - Choose a dimension i and value θ on which to split
    - Make recursive calls
        - Some training examples $\boldsymbol{X}_L$ go down left branch, recursive call with those
        - Other training examples $\boldsymbol{X}_R$ go down right branch, a second recursive call with those

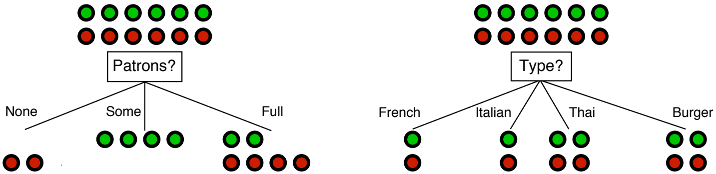
Example - Waiting for Table

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

from Russell and Norvig AIMA

- Classification problem - t_n is whether or not one should wait for a table at a restaurant
- In this example attributes (components of $\boldsymbol{x}_n$) are discrete; can be continuous too

Choosing a Dimension



- Of all the dimensions one could choose to put at root of decision tree, which is best?
- Compare using Patrons? versus Type?
  - Patrons? looks better – more information about classification

Information

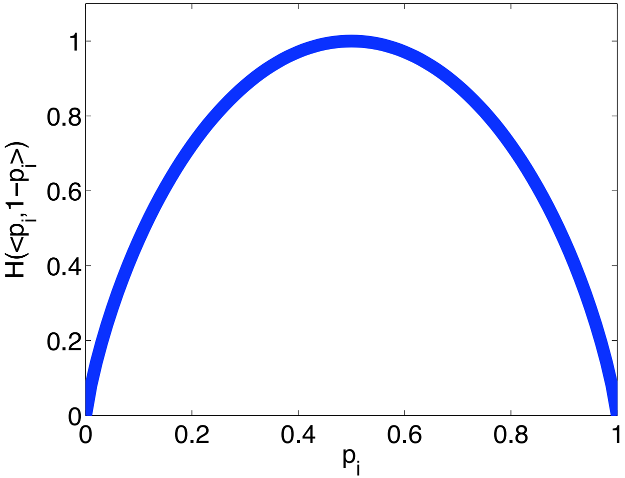
- Information answers questions
- The more clueless I am about the answer initially, the more information is in the answer
- Scale: 1 bit = answer to Boolean question with prior $p(x = true) = 0.5$
- For a K-class classification problem, we have a prior $p(x = k) = \pi_k$
- Information in answer is

$$H(x) = -\sum_{k=1}^{K} \pi_k \log_2 \pi_k$$

known as entropy of prior

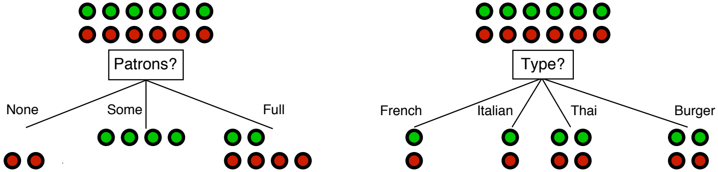
- A good dimension produces a split that reduces entropy

Entropy



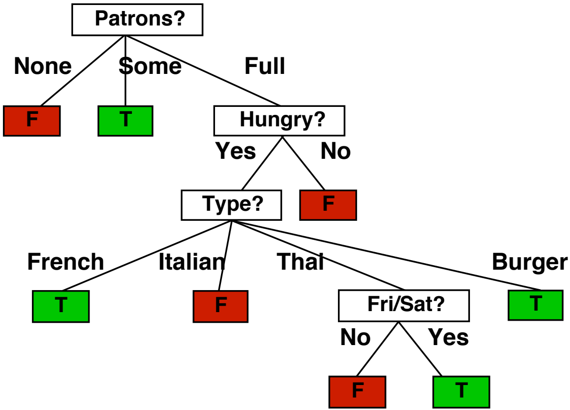
- Entropy for binary classification (boolean prior)
- $H(x) = -\pi_1 \log_2 \pi_1 - (1 - \pi_1) \log_2 (1 - \pi_1)$

Choosing a Dimension



- Compare using Patrons? versus Type?
  - Patrons? has average entropy of 0.459 bits
  - Type? has average entropy of 1 bit
- Put Patrons? at root of tree
  - Make recursive calls using training examples that fall down each path

Learnt Tree



- At each leaf have an expert
- In this case, just report what type of examples are in this region of input space
  - More generally, could stop earlier, build classifier in each region

Outline

Boosting

Decision Trees

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:

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

- Note that all models $p_m(t|\boldsymbol{x})$ are used
- But contributions $\pi_m(\boldsymbol{x})$ depend on input variable $\boldsymbol{x}$
  - These coefficients $\pi_m(\boldsymbol{x})$ are known as gating functions
  - Each $p_m(t|\boldsymbol{x})$ 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)