Introduction to Machine Learning

Combining Models, Bayesian Average and Boosting

Andres Mendez-Vazquez

July 31, 2018

Outline

Outline

Introduction

Observation

- It is often found that improved performance can be obtained by combining multiple classifiers together in some way.

Introduction

Observation

- It is often found that improved performance can be obtained by combining multiple classifiers together in some way.

Example, Committees

- We might train L different classifiers and then make predictions:
  - by using the average of the predictions made by each classifier.

- A error function used to train a particular model depends on the performance of the previous models.

Introduction

Observation

- It is often found that improved performance can be obtained by combining multiple classifiers together in some way.

Example, Committees

- We might train L different classifiers and then make predictions:
  - by using the average of the predictions made by each classifier.

Example, Boosting

- It involves training multiple models in sequence:
  - A error function used to train a particular model depends on the performance of the previous models.

Introduction

Observation

- It is often found that improved performance can be obtained by combining multiple classifiers together in some way.

Example, Committees

- We might train L different classifiers and then make predictions:
  - by using the average of the predictions made by each classifier.

Example, Boosting

- It involves training multiple models in sequence:
  - A error function used to train a particular model depends on the performance of the previous models.

Introduction

Observation

- It is often found that improved performance can be obtained by combining multiple classifiers together in some way.

Example, Committees

- We might train L different classifiers and then make predictions:
  - by using the average of the predictions made by each classifier.

Example, Boosting

- It involves training multiple models in sequence:
  - A error function used to train a particular model depends on the performance of the previous models.

Outline

We could use simple averaging

Given a series of observed samples $\{\widehat{\boldsymbol{x}}_1, \widehat{\boldsymbol{x}}_2, ..., \widehat{\boldsymbol{x}}_N\}$ with noise $\epsilon \sim N(0,1)$

We could use our knowledge on the noise, for example additive:

$$\widehat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \epsilon$$

We can use our knowledge of probability to remove such noise

$$E[\widehat{\boldsymbol{x}}_i] = E[\boldsymbol{x}_i + \epsilon] = E[\boldsymbol{x}_i] + E[\epsilon]$$

Then, because $E[\epsilon] = 0$

$$E[\boldsymbol{x}_i] = E[\widehat{\boldsymbol{x}}_i] \approx \frac{1}{N}\sum_{i=1}^{N} \widehat{\boldsymbol{x}}_i$$

We could use simple averaging

Given a series of observed samples $\{\widehat{\boldsymbol{x}}_1, \widehat{\boldsymbol{x}}_2, ..., \widehat{\boldsymbol{x}}_N\}$ with noise $\epsilon \sim N(0,1)$

We could use our knowledge on the noise, for example additive:

$$\widehat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \epsilon$$

We can use our knowledge of probability to remove such noise

$$E[\widehat{\boldsymbol{x}}_i] = E[\boldsymbol{x}_i + \epsilon] = E[\boldsymbol{x}_i] + E[\epsilon]$$

Then, because $E[\epsilon] = 0$

$$E[\boldsymbol{x}_i] = E[\widehat{\boldsymbol{x}}_i] \approx \frac{1}{N}\sum_{i=1}^{N} \widehat{\boldsymbol{x}}_i$$

We could use simple averaging

Given a series of observed samples $\{\widehat{\boldsymbol{x}}_1, \widehat{\boldsymbol{x}}_2, ..., \widehat{\boldsymbol{x}}_N\}$ with noise $\epsilon \sim N(0,1)$

We could use our knowledge on the noise, for example additive:

$$\widehat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \epsilon$$

We can use our knowledge of probability to remove such noise

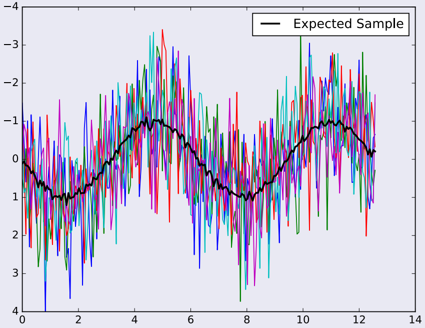
$$E[\widehat{\boldsymbol{x}}_i] = E[\boldsymbol{x}_i + \epsilon] = E[\boldsymbol{x}_i] + E[\epsilon]$$

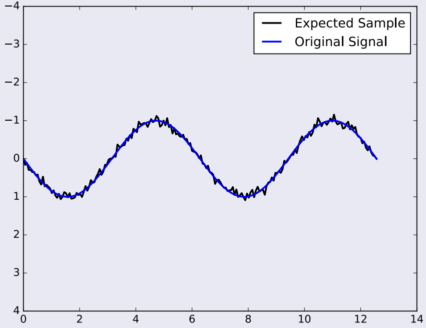
Then, because $E[\epsilon] = 0$

$$E[\boldsymbol{x}_i] = E[\widehat{\boldsymbol{x}}_i] \approx \frac{1}{N}\sum_{i=1}^{N} \widehat{\boldsymbol{x}}_i$$

For Example

We have a nice result





Outline

Beyond Simple Averaging

Instead of averaging the predictions of a set of models

- You can use an alternative form of combination that selects one of the models to make the prediction.

Where

- The choice of model is a function of the input variables.

Thus

- Different Models become responsible for making decisions in different regions of the input space.

Beyond Simple Averaging

Instead of averaging the predictions of a set of models

- You can use an alternative form of combination that selects one of the models to make the prediction.

Where

- The choice of model is a function of the input variables.

Thus

- Different Models become responsible for making decisions in different regions of the input space.

Beyond Simple Averaging

Instead of averaging the predictions of a set of models

- You can use an alternative form of combination that selects one of the models to make the prediction.

Where

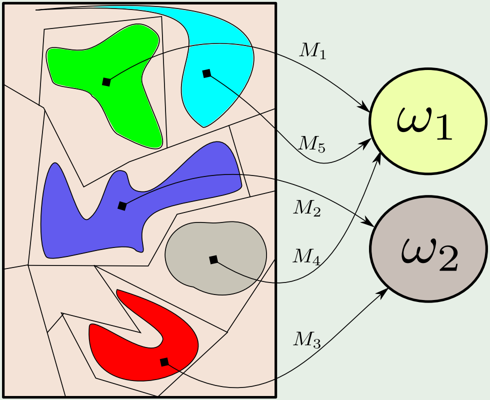
- The choice of model is a function of the input variables.

Thus

- Different Models become responsible for making decisions in different regions of the input space.

Something like this

Models in charge of different set of inputs



Outline

Example, Decision Trees

We can have the decision trees on top of the models

Given a set of models, a model is chosen to take a decision in certain area of the input.

Limitation: It is based on hard splits in which only one model is responsible for making predictions for any given value.

Example, Decision Trees

We can have the decision trees on top of the models

Given a set of models, a model is chosen to take a decision in certain area of the input.

Limitation: It is based on hard splits in which only one model is responsible for making predictions for any given value.

Thus it is better to soften the combination by using

- If we have M classifier for a conditional distribution $p(t|x, k)$.
  - x is the input variable.
  - t is the target variable.
  - $k = 1, 2, ..., M$ indexes the classifiers.

Example, Decision Trees

We can have the decision trees on top of the models

Given a set of models, a model is chosen to take a decision in certain area of the input.

Limitation: It is based on hard splits in which only one model is responsible for making predictions for any given value.

Thus it is better to soften the combination by using

- If we have M classifier for a conditional distribution $p(t|\boldsymbol{x}, k)$.
  - $\boldsymbol{x}$ is the input variable.
  - t is the target variable.
  - $k = 1, 2, ..., M$ indexes the classifiers.

Example, Decision Trees

We can have the decision trees on top of the models

Given a set of models, a model is chosen to take a decision in certain area of the input.

Limitation: It is based on hard splits in which only one model is responsible for making predictions for any given value.

Thus it is better to soften the combination by using

- If we have M classifier for a conditional distribution $p(t|\boldsymbol{x}, k)$.
  - $\boldsymbol{x}$ is the input variable.
  - t is the target variable.
  - $k = 1, 2, \ldots, M$ indexes the classifiers.

Example, Decision Trees

We can have the decision trees on top of the models

Given a set of models, a model is chosen to take a decision in certain area of the input.

Limitation: It is based on hard splits in which only one model is responsible for making predictions for any given value.

Thus it is better to soften the combination by using

- If we have M classifier for a conditional distribution $p(t|\boldsymbol{x}, k)$.
  - $\boldsymbol{x}$ is the input variable.
  - t is the target variable.
  - $k = 1, 2, ..., M$ indexes the classifiers.

Example, Decision Trees

We can have the decision trees on top of the models

Given a set of models, a model is chosen to take a decision in certain area of the input.

Limitation: It is based on hard splits in which only one model is responsible for making predictions for any given value.

Thus it is better to soften the combination by using

- If we have M classifier for a conditional distribution $p(t|\boldsymbol{x}, k)$.
  - $\boldsymbol{x}$ is the input variable.
  - t is the target variable.
  - $k = 1, 2, ..., M$ indexes the classifiers.

Example, Decision Trees

We can have the decision trees on top of the models

Given a set of models, a model is chosen to take a decision in certain area of the input.

Limitation: It is based on hard splits in which only one model is responsible for making predictions for any given value.

Thus it is better to soften the combination by using

- If we have M classifier for a conditional distribution $p(t|\boldsymbol{x}, k)$.
  - $\boldsymbol{x}$ is the input variable.
  - t is the target variable.
  - $k = 1, 2, ..., M$ indexes the classifiers.

This is used in the mixture of distributions

Thus (Mixture of Experts)

$$p\left(t|\boldsymbol{x}\right)=\sum_{k=1}^{M}\pi_k\left(\boldsymbol{x}\right)p\left(t|\boldsymbol{x},k\right) \quad (1)$$

where $\pi_k\left(\boldsymbol{x}\right)=p\left(k|\boldsymbol{x}\right)$ represent the input-dependent mixing coefficients.

This type of models

They can be viewed as mixture distribution in which the component densities and the mixing coefficients are conditioned on the input variables and are known as mixture experts.

This is used in the mixture of distributions

Thus (Mixture of Experts)

$$p(t|\boldsymbol{x}) = \sum_{k=1}^{M} \pi_k(\boldsymbol{x})\, p(t|\boldsymbol{x}, k) \tag{1}$$

where $\pi_k(\boldsymbol{x}) = p(k|\boldsymbol{x})$ represent the input-dependent mixing coefficients.

This type of models

They can be viewed as mixture distribution in which the component densities and the mixing coefficients are conditioned on the input variables and are known as mixture experts.

Outline

It is important to differentiate between them

Although

- Model Combinations and Bayesian Model Averaging look similar.
  - However, they are actually different

For this

We have the following example.

It is important to differentiate between them

Although

- Model Combinations and Bayesian Model Averaging look similar.
  - However, they are actually different

For this

We have the following example.

Example of the Differences

For this consider the following

- Mixture of Gaussians with a binary latent variable z indicating to which component a point belongs to.

Thus the model is specified in terms a joint distribution

$$p(\boldsymbol{x}, \boldsymbol{z})$$

Corresponding density over the observed variable x using marginalization

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z})$$

Example of the Differences

For this consider the following

- Mixture of Gaussians with a binary latent variable $\boldsymbol{z}$ indicating to which component a point belongs to.

Thus the model is specified in terms a joint distribution

$$p\left(\boldsymbol{x}, \boldsymbol{z}\right)$$

Corresponding density over the observed variable $\boldsymbol{x}$ using marginalization

$$p\left(\boldsymbol{x}\right) = \sum_{\boldsymbol{z}} p\left(\boldsymbol{x}, \boldsymbol{z}\right)$$

Example of the Differences

For this consider the following

- Mixture of Gaussians with a binary latent variable z indicating to which component a point belongs to.

Thus the model is specified in terms a joint distribution

$$p\left(\boldsymbol{x}, \boldsymbol{z}\right)$$

Corresponding density over the observed variable x using marginalization

$$p\left(\boldsymbol{x}\right) = \sum_{\boldsymbol{z}} p\left(\boldsymbol{x}, \boldsymbol{z}\right)$$

Example

In the case of Mixture of Gaussian's

$$p\left(\boldsymbol{x}\right) = \sum_{k=1}^{K} \pi_k N\left(\boldsymbol{x}|\mu_k, \Sigma_k\right)$$

This is an example of model combination.

- What about other Models

Example

In the case of Mixture of Gaussian's

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}|\mu_k, \Sigma_k)$$

This is an example of model combination.

- What about other Models

More Models

Now, for independent, identically distributed data
$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

$$p(\boldsymbol{X}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n) = \prod_{n=1}^{N} \left[\sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n) \right]$$

Therefore

Something Notable

- Each observed data point $\boldsymbol{x}_n$ has a corresponding latent variable $\boldsymbol{z}_n$.

Here, we are doing a Combination of Models

- Each Gaussian indexed by z_n is in charge of generating one section of the sample space

Therefore

Something Notable

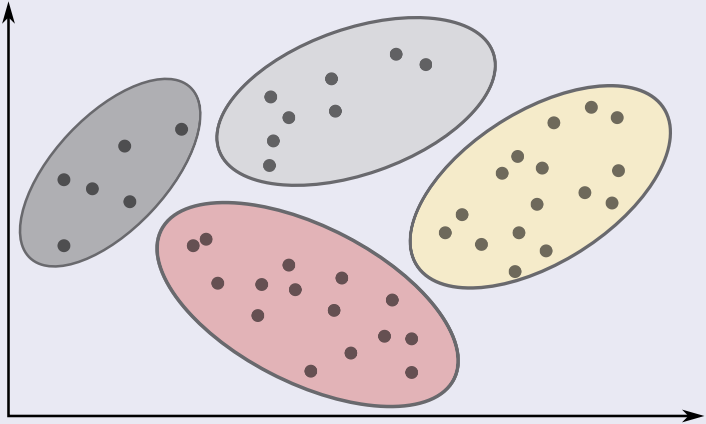
- Each observed data point $\boldsymbol{x}_n$ has a corresponding latent variable $\boldsymbol{z}_n$.

Here, we are doing a Combination of Models

- Each Gaussian indexed by $\boldsymbol{z}_n$ is in charge of generating one section of the sample space

Example

We have



Outline

Now, suppose

We have several different models indexed by $h = 1, ..., H$ with prior probabilities

- One model might be a mixture of Gaussians and another model might be a mixture of Cauchy distributions

The Marginal Distribution is

$$p(X) = \sum_{h=1}^{H} p(X,h) = \sum_{h=1}^{H} \underbrace{p(X|h)\,p(h)}_{\approx p(h|X)}$$

- This is an example of **Bayesian model averaging**

Now, suppose

We have several different models indexed by $h = 1, ..., H$ with prior probabilities

- One model might be a mixture of Gaussians and another model might be a mixture of Cauchy distributions

The Marginal Distribution is

$$p(X) = \sum_{h=1}^{H} p(X, h) = \sum_{h=1}^{H} \underbrace{p(X|h)\, p(h)}_{\approx p(h|X)}$$

- This is an example of **Bayesian model averaging**

Bayesian Model Averaging

Remark

- The summation over h means that just one model is responsible for generating the whole data set.

Observation

- The probability over h simply reflects our uncertainty of which is the correct model to use.

Bayesian Model Averaging

Remark

- The summation over h means that just one model is responsible for generating the whole data set.

Observation

- The probability over h simply reflects our uncertainty of which is the correct model to use.

Thus, as the size of the data set increases

- This uncertainty reduces
  - Posterior probabilities $p(h|X)$ become increasingly focused on just one of the models.

Bayesian Model Averaging

Remark

- The summation over h means that just one model is responsible for generating the whole data set.

Observation

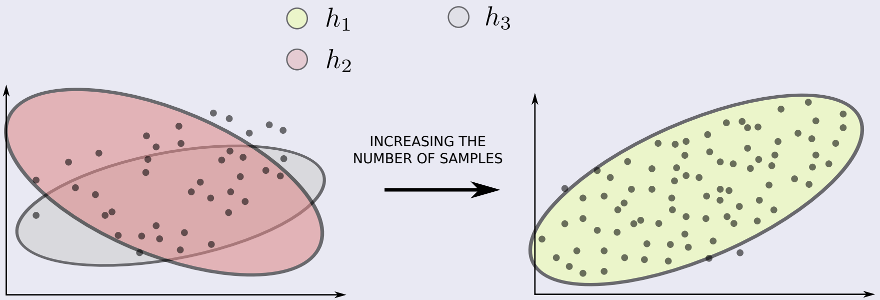
- The probability over h simply reflects our uncertainty of which is the correct model to use.

Thus, as the size of the data set increases

- This uncertainty reduces
  - Posterior probabilities $p(h|X)$ become increasingly focused on just one of the models.

Example

We have



Outline

The Differences

Bayesian model averaging

- The whole data set is generated by a single model h.

Model combination

- Different data points within the data set can potentially be generated from different by different components.

The Differences

Bayesian model averaging

- The whole data set is generated by a single model h.

Model combination

- Different data points within the data set can potentially be generated from different by different components.

Outline

Committees

Idea, the simplest way to construct a committee

- It is to average the predictions of a set of individual models.

Committees

Idea, the simplest way to construct a committee

- It is to average the predictions of a set of individual models.

Thinking as a frequentist

- This is coming from taking in consideration the trade-off between bias and variance.

- The variance component that represents the sensitivity of the model to the individual data points.

Committees

Idea, the simplest way to construct a committee

- It is to average the predictions of a set of individual models.

Thinking as a frequentist

- This is coming from taking in consideration the trade-off between bias and variance.

Where the error in the model into

- The bias component that arises from differences **between the model and the true function to be predicted**.
- The variance component that represents **the sensitivity of the model to the individual data points**.

Committees

Idea, the simplest way to construct a committee

- It is to average the predictions of a set of individual models.

Thinking as a frequentist

- This is coming from taking in consideration the trade-off between bias and variance.

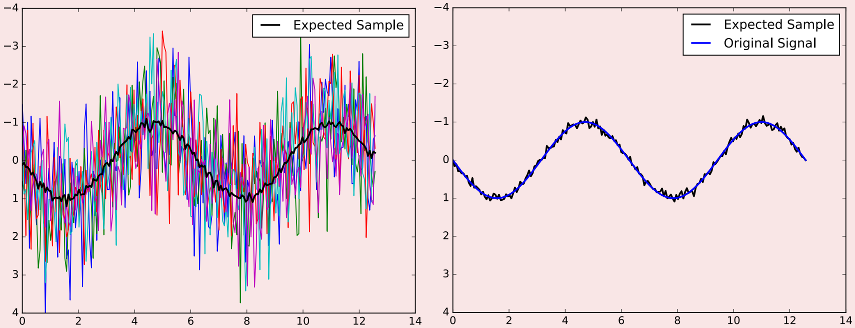
Where the error in the model into

- The bias component that arises from differences **between the model and the true function to be predicted**.
- The variance component that represents **the sensitivity of the model to the individual data points**.

For example

When we averaged a set of low-bias models

- We obtained accurate predictions of the underlying sinusoidal function from which the data were generated.



However

Big Problem

- We have normally a single data set

Thus

- We need to introduce certain variability between the different committee members.

One approach

- You can use bootstrap data sets.

However

Big Problem

- We have normally a single data set

Thus

- We need to introduce certain variability between the different committee members.

One approach

- You can use bootstrap data sets.

Outline

The Idea of Bootstrap

We denote the training set by $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_N\}$

- Where $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$

The basic idea is to randomly draw datasets with replacement from the training data

- Each sample the same size as the original training set.

This is done B times

- Producing B bootstrap datasets.

The Idea of Bootstrap

We denote the training set by $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_N\}$

- Where $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$

The basic idea is to randomly draw datasets with replacement from the training data

- Each sample the same size as the original training set.

This is done B times

- Producing B bootstrap datasets.

The Idea of Bootstrap

We denote the training set by $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_{\mathbf{2}}, ..., \boldsymbol{z}_N\}$

- Where $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$

The basic idea is to randomly draw datasets with replacement from the training data

- Each sample the same size as the original training set.

This is done B times

- Producing B bootstrap datasets.

Then

Then a quantity is computed

- $S(Z)$ is any quantity computed from the data Z

From the bootstrap sampling

- We can estimate any aspect of the distribution of $S(Z)$.

Then

Then a quantity is computed

- $S(Z)$ is any quantity computed from the data Z

From the bootstrap sampling

- We can estimate any aspect of the distribution of $S(Z)$.

Then

Then a quantity is computed

- $S(Z)$ is any quantity computed from the data Z

From the bootstrap sampling

- We can estimate any aspect of the distribution of $S(Z)$.

Then

we refit the model to each of the bootstrap datasets

- You generate $S\left(Z^{*b}\right)$ to refit the model to this dataset.

Then

- You examine the behavior of the fits over the B replications.

Then

we refit the model to each of the bootstrap datasets

- You generate $S\left(Z^{*b}\right)$ to refit the model to this dataset.

Then

- You examine the behavior of the fits over the B replications.

For Example

Its variance

$$\widehat{Var}\left[S\left(Z\right)\right]=\frac{1}{B-1}\sum_{b=1}^{B}\left(S\left(Z^{*b}\right)-\overline{S}^{*}\right)^{2}$$

Where

$$\overline{S}^{*}=\frac{1}{B}\sum_{b=1}^{B}S\left(Z^{*b}\right)$$

For Example

Its variance

$$\widehat{Var}\left[S\left(Z\right)\right]=\frac{1}{B-1}\sum_{b=1}^{B}\left(S\left(Z^{*b}\right)-\overline{S}^{*}\right)^{2}$$

Where

$$\overline{S}^{*}=\frac{1}{B}\sum_{b=1}^{B}S\left(Z^{*b}\right)$$

Outline

Relation with Monte-Carlo Estimation

Note that $\widehat{Var}\left[S\left(Z\right)\right]$

- It can be thought of as a Monte-Carlo estimate of the variance of $S\left(Z\right)$ under sampling.

This is coming

- From the empirical distribution function $\widehat{F}$ for the data $Z = \{z_1, z_2, ..., z_N\}$

Relation with Monte-Carlo Estimation

Note that $\widehat{Var}\left[S\left(Z\right)\right]$

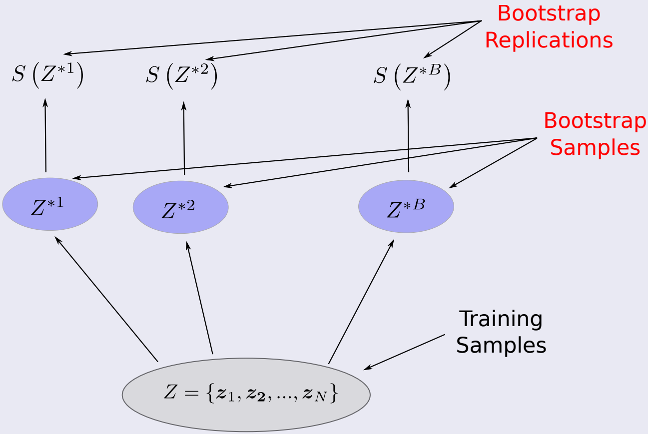
- It can be thought of as a Monte-Carlo estimate of the variance of $S\left(Z\right)$ under sampling.

This is coming

- From the empirical distribution function $\widehat{F}$ for the data $Z = \left\{\boldsymbol{z}_1, \boldsymbol{z}_{\boldsymbol{2}}, ..., \boldsymbol{z}_N\right\}$

For Example

Schematic of the bootstrap process



Thus

Use each of them to train a copy $y_b(\boldsymbol{x})$ of a predictive regression model to predict a single continuous variable

Then,

$$y_{com}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} y_b(\boldsymbol{x}) \tag{2}$$

This is also known as **Bootstrap Aggregation or Bagging.**

What do we with this samples?

Now, assume a true regression function $h(\boldsymbol{x})$ and a estimation $y_b(\boldsymbol{x})$

$$y_b(\boldsymbol{x}) = h(\boldsymbol{x}) + \epsilon_b(\boldsymbol{x}) \tag{3}$$

The average sum-of-squares error over the data takes the form

$$E_{\boldsymbol{x}}\left[\left(y_b(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right] = E_{\boldsymbol{x}}\left[\epsilon_b^2(\boldsymbol{x})\right] \tag{4}$$

What is $E_{\boldsymbol{x}}$?

It denotes a frequentest expectation with respect to the distribution of the input vector $\boldsymbol{x}$.

What do we with this samples?

Now, assume a true regression function $h(\boldsymbol{x})$ and a estimation $y_b(\boldsymbol{x})$

$$y_b(\boldsymbol{x}) = h(\boldsymbol{x}) + \epsilon_b(\boldsymbol{x}) \tag{3}$$

The average sum-of-squares error over the data takes the form

$$E_{\boldsymbol{x}}\left[(y_b(\boldsymbol{x}) - h(\boldsymbol{x}))^2\right] = E_{\boldsymbol{x}}\left[\epsilon_b^2(\boldsymbol{x})\right] \tag{4}$$

What is $E_{\boldsymbol{x}}$?

It denotes a frequentest expectation with respect to the distribution of the input vector $\boldsymbol{x}$.

What do we with this samples?

Now, assume a true regression function $h(\boldsymbol{x})$ and a estimation $y_b(\boldsymbol{x})$

$$y_b(\boldsymbol{x}) = h(\boldsymbol{x}) + \epsilon_b(\boldsymbol{x}) \tag{3}$$

The average sum-of-squares error over the data takes the form

$$E_{\boldsymbol{x}}\left[\left(y_b(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right] = E_{\boldsymbol{x}}\left[\epsilon_b^2(\boldsymbol{x})\right] \tag{4}$$

What is $E_{\boldsymbol{x}}$?

It denotes a frequentest expectation with respect to the distribution of the input vector $\boldsymbol{x}$.

Meaning

Thus, the average error is

$$E_{AV} = \frac{1}{B}\sum_{b=1}^{b} E_{\boldsymbol{x}}\left[\left\{\epsilon_b\left(\boldsymbol{x}\right)\right\}^2\right] \tag{5}$$

Similarly the Expected error over the committee

$$E_{COM} = E_{\boldsymbol{x}}\left[\left\{\frac{1}{B}\sum_{b=1}^{B}\left(y_m\left(\boldsymbol{x}\right) - h\left(\boldsymbol{x}\right)\right)\right\}^2\right] = E_{\boldsymbol{x}}\left[\left\{\frac{1}{B}\sum_{b=1}^{B}\epsilon_b\left(\boldsymbol{x}\right)\right\}^2\right] \tag{6}$$

Meaning

Thus, the average error is

$$E_{AV} = \frac{1}{B}\sum_{b=1}^{b} E_{\boldsymbol{x}}\left[\{\epsilon_b(\boldsymbol{x})\}^2\right] \tag{5}$$

Similarly the Expected error over the committee

$$E_{COM} = E_{\boldsymbol{x}}\left[\left\{\frac{1}{B}\sum_{b=1}^{B}(y_m(\boldsymbol{x}) - h(\boldsymbol{x}))\right\}^2\right] = E_{\boldsymbol{x}}\left[\left\{\frac{1}{B}\sum_{b=1}^{B}\epsilon_b(\boldsymbol{x})\right\}^2\right] \tag{6}$$

Assume that the errors have zero mean and are uncorrelated

Assume that the errors have zero mean and are uncorrelated

- Something Reasonable to assume given the way we produce the Bootstrap Samples

$$
\begin{aligned}
E_{\boldsymbol{x}}\left[\epsilon_b\left(\boldsymbol{x}\right)\right] =& 0\\
E_{\boldsymbol{x}}\left[\epsilon_b\left(\boldsymbol{x}\right)\epsilon_l\left(\boldsymbol{x}\right)\right] =& 0,\ \text{for } b \neq l
\end{aligned}
$$

Then

We have that

$$
\begin{aligned}
E_{COM} &= \frac{1}{b^2} E_{\boldsymbol{x}} \left[\left\{ \sum_{b=1}^{B} \left(\epsilon_b \left(\boldsymbol{x} \right) \right) \right\}^2 \right] \\
&= \frac{1}{B^2} E_{\boldsymbol{x}} \left[\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) + \sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right] \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + E_{\boldsymbol{x}} \left(\sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + \sum_{\substack{h=1 \\ h \neq k}}^{M} \sum_{k=1}^{M} E_{\boldsymbol{x}} \left(\epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\} = \frac{1}{B} \left\{ \frac{1}{B} E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\}
\end{aligned}
$$

Then

We have that

$$
\begin{aligned}
E_{COM} &= \frac{1}{b^2} E_{\boldsymbol{x}} \left[\left\{ \sum_{b=1}^{B} \left(\epsilon_b \left(\boldsymbol{x} \right) \right) \right\}^2 \right] \\
&= \frac{1}{B^2} E_{\boldsymbol{x}} \left[\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) + \sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right] \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + E_{\boldsymbol{x}} \left(\sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + \sum_{\substack{h=1 \\ h \neq k}}^{M} \sum_{k=1}^{M} E_{\boldsymbol{x}} \left(\epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\} = \frac{1}{B} \left\{ \frac{1}{B} E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\}
\end{aligned}
$$

Then

We have that

$$
\begin{aligned}
E_{COM} &= \frac{1}{b^2} E_{\boldsymbol{x}} \left[\left\{ \sum_{b=1}^{B} \left(\epsilon_b \left(\boldsymbol{x} \right) \right) \right\}^2 \right] \\
&= \frac{1}{B^2} E_{\boldsymbol{x}} \left[\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) + \sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right] \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + E_{\boldsymbol{x}} \left(\sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + \sum_{\substack{h=1 \\ h \neq k}}^{M} \sum_{k=1}^{M} E_{\boldsymbol{x}} \left(\epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\} = \frac{1}{B} \left\{ \frac{1}{B} E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\}
\end{aligned}
$$

Then

We have that

$$
\begin{aligned}
E_{COM} &= \frac{1}{b^2} E_{\boldsymbol{x}} \left[\left\{ \sum_{b=1}^{B} \left(\epsilon_b \left(\boldsymbol{x} \right) \right) \right\}^2 \right] \\
&= \frac{1}{B^2} E_{\boldsymbol{x}} \left[\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) + \sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right] \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + E_{\boldsymbol{x}} \left(\sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + \sum_{\substack{h=1 \\ h \neq k}}^{M} \sum_{k=1}^{M} E_{\boldsymbol{x}} \left(\epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\} = \frac{1}{B} \left\{ \frac{1}{B} E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\}
\end{aligned}
$$

Then

We have that

$$
\begin{aligned}
E_{COM} &= \frac{1}{b^2} E_{\boldsymbol{x}} \left[\left\{ \sum_{b=1}^{B} \left(\epsilon_b \left(\boldsymbol{x} \right) \right) \right\}^2 \right] \\
&= \frac{1}{B^2} E_{\boldsymbol{x}} \left[\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) + \sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right] \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + E_{\boldsymbol{x}} \left(\sum_{\substack{h=1 \\ h \neq k}}^{B} \sum_{k=1}^{B} \epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) + \sum_{\substack{h=1 \\ h \neq k}}^{M} \sum_{k=1}^{M} E_{\boldsymbol{x}} \left(\epsilon_h \left(\boldsymbol{x} \right) \epsilon_k \left(\boldsymbol{x} \right) \right) \right\} \\
&= \frac{1}{B^2} \left\{ E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\} = \frac{1}{B} \left\{ \frac{1}{B} E_{\boldsymbol{x}} \left(\sum_{b=1}^{B} \epsilon_b^2 \left(\boldsymbol{x} \right) \right) \right\}
\end{aligned}
$$

We finally obtain

We obtain

$$E_{COM} = \frac{1}{B} E_{AV} \tag{7}$$

Looks great BUT!!!

Unfortunately, it depends on the key assumption that the errors at the individual Bootstrap Models are uncorrelated.

We finally obtain

We obtain

$$E_{COM} = \frac{1}{B} E_{AV} \tag{7}$$

Looks great BUT!!!

Unfortunately, it depends on the key assumption that the errors at the individual Bootstrap Models are uncorrelated.

Thus

The Reality!!!

The errors are typically highly correlated, and the reduction in overall error is generally small.

Something Notable

However, It can be shown that the expected committee error will not exceed the expected error of the constituent models, so

$$E_{COM} \leq E_{AV} \tag{8}$$

However, we need something better

A more sophisticated technique known as **boosting**.

Thus

The Reality!!!

The errors are typically highly correlated, and the reduction in overall error is generally small.

Something Notable

However, It can be shown that the expected committee error will not exceed the expected error of the constituent models, so

$$E_{COM} \leq E_{AV} \tag{8}$$

However, we need something better

A more sophisticated technique known as **boosting**.

Thus

The Reality!!!

The errors are typically highly correlated, and the reduction in overall error is generally small.

Something Notable

However, It can be shown that the expected committee error will not exceed the expected error of the constituent models, so

$$E_{COM} \leq E_{AV} \tag{8}$$

However, we need something better

A more sophisticated technique known as **boosting**.

Outline

Boosting

What Boosting does?

It combines several classifiers to produce a form of a committee.

We will describe AdaBoost

"Adaptive Boosting" developed by Freund and Schapire (1995).

Boosting

What Boosting does?

It combines several classifiers to produce a form of a committee.

We will describe AdaBoost

"Adaptive Boosting" developed by Freund and Schapire (1995).

Sequential Training

Main difference between boosting and committee methods

The base classifiers are trained in sequence.

Explanation

Consider a two-class classification problem:

1. Samples $x_1, x_2, ..., x_N$
2. Binary labels (-1,1) $t_1, t_2, ..., t_N$

Sequential Training

Main difference between boosting and committee methods

The base classifiers are trained in sequence.

Explanation

Consider a two-class classification problem:

1. Samples $\boldsymbol{x}_1, \boldsymbol{x_2}, ..., \boldsymbol{x_N}$
2. Binary labels (-1,1) $t_1, t_2, ..., t_N$

Outline

Cost Function

Now

You want to put together a set of M experts able to recognize the most difficult inputs in an accurate way!!!

Thus

For each pattern $\boldsymbol{x}_i$ each expert classifier outputs a classification $y_j(\boldsymbol{x}_i) \in \{-1, 1\}$

The final decision of the committee of M experts is $sign(C(\boldsymbol{x}_i))$

$$C(\boldsymbol{x}_i) = \alpha_1 y_1(\boldsymbol{x}_i) + \alpha_2 y_2(\boldsymbol{x}_i) + \dots + \alpha_M y_M(\boldsymbol{x}_i) \tag{9}$$

Cost Function

Now

You want to put together a set of M experts able to recognize the most difficult inputs in an accurate way!!!

Thus

For each pattern $\boldsymbol{x}_i$ each expert classifier outputs a classification $y_j(\boldsymbol{x}_i) \in \{-1, 1\}$

The final decision of the committee of M experts is $sign(C(\boldsymbol{x}_i))$

$$C(\boldsymbol{x}_i) = \alpha_1 y_1(\boldsymbol{x}_i) + \alpha_2 y_2(\boldsymbol{x}_i) + \ldots + \alpha_M y_M(\boldsymbol{x}_i) \tag{9}$$

Cost Function

Now

You want to put together a set of M experts able to recognize the most difficult inputs in an accurate way!!!

Thus

For each pattern $\boldsymbol{x}_i$ each expert classifier outputs a classification $y_j(\boldsymbol{x}_i) \in \{-1, 1\}$

The final decision of the committee of M experts is $sign(C(\boldsymbol{x}_i))$

$$C(\boldsymbol{x}_i) = \alpha_1 y_1(\boldsymbol{x}_i) + \alpha_2 y_2(\boldsymbol{x}_i) + ... + \alpha_M y_M(\boldsymbol{x}_i) \quad (9)$$

Now

Adaptive Boosting

It works even with a continuum of classifiers.

However

For the sake of simplicity, we will assume that the set of expert is finite.

Now

Adaptive Boosting

It works even with a continuum of classifiers.

However

For the sake of simplicity, we will assume that the set of expert is finite.

Outline

Getting the correct classifiers

We want the following

- We want to review possible element members.
- Select them, if they have certain properties.
- Assigning a weight to their contribution to the set of experts.

Getting the correct classifiers

We want the following

- We want to review possible element members.
- Select them, if they have certain properties.
- Assigning a weight to their contribution to the set of experts.

Getting the correct classifiers

We want the following

- We want to review possible element members.
- Select them, if they have certain properties.
- Assigning a weight to their contribution to the set of experts.

Now

Selection is done the following way

Testing the classifiers in the pool using a training set T of N multidimensional data points $\boldsymbol{x}_i$:

- For each point x_i we have a label $t_i = 1$ or $t_i = -1$.

Now

Selection is done the following way

Testing the classifiers in the pool using a training set T of N multidimensional data points $\boldsymbol{x}_i$:

- For each point x_i we have a label $t_i = 1$ or $t_i = -1$.

Assigning a cost for actions

We test and rank all classifiers in the expert pool by

- Charging a cost $\exp\{\beta\}$ any time a classifier fails (a miss).
- Charging a cost $\exp\{-\beta\}$ any time a classifier provides the right label (a hit).

Now

Selection is done the following way

Testing the classifiers in the pool using a training set T of N multidimensional data points $\boldsymbol{x}_i$:

- For each point x_i we have a label $t_i = 1$ or $t_i = -1$.

Assigning a cost for actions

We test and rank all classifiers in the expert pool by

- Charging a cost $\exp\{\beta\}$ any time a classifier fails (a miss).
- Charging a cost $\exp\{-\beta\}$ any time a classifier provides the right label (a hit).

Now

Selection is done the following way

Testing the classifiers in the pool using a training set T of N multidimensional data points $\boldsymbol{x}_i$:

- For each point x_i we have a label $t_i = 1$ or $t_i = -1$.

Assigning a cost for actions

We test and rank all classifiers in the expert pool by

- Charging a cost $\exp\{\beta\}$ any time a classifier fails (a miss).
- Charging a cost $\exp\{-\beta\}$ any time a classifier provides the right label (a hit).

Now

Selection is done the following way

Testing the classifiers in the pool using a training set T of N multidimensional data points $\boldsymbol{x}_i$:

- For each point x_i we have a label $t_i = 1$ or $t_i = -1$.

Assigning a cost for actions

We test and rank all classifiers in the expert pool by

- Charging a cost $\exp\{\beta\}$ any time a classifier fails (a miss).
- Charging a cost $\exp\{-\beta\}$ any time a classifier provides the right label (a hit).

Remarks about β

We require $\beta > 0$

- Thus misses are penalized more heavily penalized than hits

Remarks about β

We require $\beta > 0$

- Thus misses are penalized more heavily penalized than hits

Although

- It looks strange to penalize hits,
- However, as long that the penalty of a success is smaller than the penalty for a miss:

$$\exp\{-\beta\} < \exp\{\beta\}$$

Remarks about β

We require $\beta > 0$

- Thus misses are penalized more heavily penalized than hits

Although

- It looks strange to penalize hits,
- However, as long that the penalty of a success is smaller than the penalty for a miss:

$$\exp\{-\beta\} < \exp\{\beta\}$$

Why?

- if we assign cost a to misses and cost b to hits, where $a > b > 0$.
- We can rewrite such costs as $a = c^d$ and $b = c^{-d}$ for constants c and d.
    - It does not compromise generality.

Remarks about β

We require $\beta > 0$

- Thus misses are penalized more heavily penalized than hits

Although

- It looks strange to penalize hits,
- However, as long that the penalty of a success is smaller than the penalty for a miss:

$$\exp\left\{-\beta\right\} < \exp\left\{\beta\right\}$$

Why?

- if we assign cost a to misses and cost b to hits, where $a > b > 0$.
- We can rewrite such costs as $a = c^d$ and $b = c^{-d}$ for constants c and d.
  - It does not compromise generality.

Remarks about β

We require $\beta > 0$

- Thus misses are penalized more heavily penalized than hits

Although

- It looks strange to penalize hits,
- However, as long that the penalty of a success is smaller than the penalty for a miss:

$$\exp\{-\beta\} < \exp\{\beta\}$$

Why?

- if we assign cost a to misses and cost b to hits, where $a > b > 0$.
- We can rewrite such costs as $a = c^d$ and $b = c^{-d}$ for constants c and d.
  - It does not compromise generality.

Remarks about β

We require $\beta > 0$

- Thus misses are penalized more heavily penalized than hits

Although

- It looks strange to penalize hits,
- However, as long that the penalty of a success is smaller than the penalty for a miss:

$$\exp\left\{-\beta\right\} < \exp\left\{\beta\right\}$$

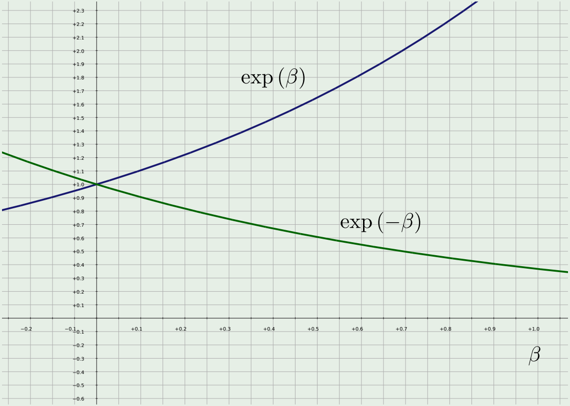
Why?

- if we assign cost a to misses and cost b to hits, where $a > b > 0$.
- We can rewrite such costs as $a = c^d$ and $b = c^{-d}$ for constants c and d.
  - It does not compromise generality.

Exponential Loss Function

This kind of error function is different from Squared Euclidean distance

- The classification target is called an exponential loss function.
- AdaBoost uses exponential error loss as error criterion.



Outline

Selection of the Classifier

We need to have a way to select the best Classifier in the Pool

- When we test the M classifiers in the pool, we build a matrix S

Then

- We record the misses (with a ONE) and hits (with a ZERO) of each classifiers.

Selection of the Classifier

We need to have a way to select the best Classifier in the Pool

- When we test the M classifiers in the pool, we build a matrix S

Then

- We record the misses (with a ONE) and hits (with a ZERO) of each classifiers.

The Matrix S

Row i in the matrix is reserved for the data point $\boldsymbol{x}_i$

- Column m is reserved for the mth classifier in the pool.

Classifiers

	1	2	$\cdots$	M
$\boldsymbol{x}_1$	0	1	$\cdots$	1
$\boldsymbol{x}_2$	0	0	$\cdots$	1
$\boldsymbol{x}_3$	1	1	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\boldsymbol{x}_N$	0	0	$\cdots$	0

Something interesting about the S

The sum along the rows is the sum at the empirical risk

$$\text{ER}\left(y_j\right) = \frac{1}{N}\sum_{i=1}^{N} S_{ij} \text{ with } j = 1, ..., M$$

Therefore, the candidate to be used at certain iteration

- It is the classifier y_j with the smallest empirical risk!!!

Something interesting about the S

The sum along the rows is the sum at the empirical risk

$$\mathsf{ER}\left(y_j\right) = \frac{1}{N}\sum_{i=1}^{N} S_{ij} \text{ with } j = 1, ..., M$$

Therefore, the candidate to be used at certain iteration

- It is the classifier y_j with the smallest empirical risk!!!

Main Idea

What does AdaBoost want to do?

The main idea of AdaBoost is to proceed systematically by extracting one classifier from the pool in each of M iterations.

Main Idea

What does AdaBoost want to do?

The main idea of AdaBoost is to proceed systematically by extracting one classifier from the pool in each of M iterations.

Thus

The elements in the data set are weighted according to their current relevance (or urgency) at each iteration.

- Just 1, or $\frac{1}{N}$ if we want to have a total sum of 1 for all weights

Main Idea

What does AdaBoost want to do?

The main idea of AdaBoost is to proceed systematically by extracting one classifier from the pool in each of M iterations.

Thus

The elements in the data set are weighted according to their current relevance (or urgency) at each iteration.

Thus at the beginning of the iterations

All data samples are assigned the same weight:

- Just 1, or $\frac{1}{N}$, if we want to have a total sum of 1 for all weights.

Main Idea

What does AdaBoost want to do?

The main idea of AdaBoost is to proceed systematically by extracting one classifier from the pool in each of M iterations.

Thus

The elements in the data set are weighted according to their current relevance (or urgency) at each iteration.

Thus at the beginning of the iterations

All data samples are assigned the same weight:

- Just 1, or $\frac{1}{N}$, if we want to have a total sum of 1 for all weights.

The process of the weights

As the selection progresses

- The more difficult samples, those where the committee still performs badly, are assigned larger and larger weights.

The process of the weights

As the selection progresses

- The more difficult samples, those where the committee still performs badly, are assigned larger and larger weights.

The selection process concentrates in selecting new classifiers

- For the committee by focusing on those which can help with the still misclassified examples.

- Classifiers being selected should complement each other in an optimal way.

The process of the weights

As the selection progresses

- The more difficult samples, those where the committee still performs badly, are assigned larger and larger weights.

The selection process concentrates in selecting new classifiers

- For the committee by focusing on those which can help with the still misclassified examples.

Then

- The best classifiers are those which can provide new insights to the committee.
- Classifiers being selected should complement each other in an optimal way.

The process of the weights

As the selection progresses

- The more difficult samples, those where the committee still performs badly, are assigned larger and larger weights.

The selection process concentrates in selecting new classifiers

- For the committee by focusing on those which can help with the still misclassified examples.

Then

- The best classifiers are those which can provide new insights to the committee.
- Classifiers being selected should complement each other in an optimal way.

Outline

Selecting New Classifiers

What we want

In each iteration, we rank all classifiers, so that we can select the current best out of the pool.

At mth iteration

We have already included $m-1$ classifiers in the committee and we want to select the next one.

Thus, we have the following cost function which is actually the output of the committee

$$C_{(m-1)}\left(x_i\right) = \alpha_1 y_1\left(x_i\right) + \alpha_2 y_2\left(x_i\right) + \ldots + \alpha_{m-1} y_{m-1}\left(x_i\right) \tag{10}$$

Selecting New Classifiers

What we want

In each iteration, we rank all classifiers, so that we can select the current best out of the pool.

At mth iteration

We have already included $m-1$ classifiers in the committee and we want to select the next one.

Thus, we have the following cost function which is actually the output of the committee

$$C_{(m-1)}(x_i) = \alpha_1 y_1(x_i) + \alpha_2 y_2(x_i) + \dots + \alpha_{m-1} y_{m-1}(x_i) \tag{10}$$

Selecting New Classifiers

What we want

In each iteration, we rank all classifiers, so that we can select the current best out of the pool.

At mth iteration

We have already included $m-1$ classifiers in the committee and we want to select the next one.

Thus, we have the following cost function which is actually the output of the committee

$$C_{(m-1)}(\boldsymbol{x}_i) = \alpha_1 y_1(\boldsymbol{x}_i) + \alpha_2 y_2(\boldsymbol{x}_i) + \ldots + \alpha_{m-1} y_{m-1}(\boldsymbol{x}_i) \tag{10}$$

Thus, we have that

Extending the cost function by the new regression y_m

$$C_{(m)}\left(\boldsymbol{x}_i\right) = C_{(m-1)}\left(\boldsymbol{x}_i\right) + \alpha_m y_m\left(\boldsymbol{x}_i\right) \quad (11)$$

At the first iteration $m = 1$

- $C_{(0)}$ is the zero function.

Thus, the total cost or total error is defined as the exponential error

$$E = \sum_{i=1}^{N} \exp\left\{-t_i\left(C_{(m-1)}\left(\boldsymbol{x}_i\right) + \alpha_m y_m\left(\boldsymbol{x}_i\right)\right)\right\} \quad (12)$$

Thus, we have that

Extending the cost function by the new regression y_m

$$C_{(m)}\left(\boldsymbol{x}_i\right) = C_{(m-1)}\left(\boldsymbol{x}_i\right) + \alpha_m y_m\left(\boldsymbol{x}_i\right) \tag{11}$$

At the first iteration $m = 1$

- $C_{(0)}$ is the zero function.

Thus, the total cost or total error is defined as the exponential error

$$E = \sum_{i=1}^{N} \exp\left\{-t_i\left(C_{(m-1)}\left(\boldsymbol{x}_i\right) + \alpha_m y_m\left(\boldsymbol{x}_i\right)\right)\right\} \tag{12}$$

Thus, we have that

Extending the cost function by the new regression y_m

$$C_{(m)}(\boldsymbol{x}_i) = C_{(m-1)}(\boldsymbol{x}_i) + \alpha_m y_m(\boldsymbol{x}_i) \tag{11}$$

At the first iteration $m = 1$

- $C_{(0)}$ is the zero function.

Thus, the total cost or total error is defined as the exponential error

$$E = \sum_{i=1}^{N} \exp\left\{-t_i\left(C_{(m-1)}(\boldsymbol{x}_i) + \alpha_m y_m(\boldsymbol{x}_i)\right)\right\} \tag{12}$$

Thus

We want to determine

α_m and y_m in optimal way

Thus, rewriting

$$E = \sum_{i=1}^{N} w_i^{(m)} \exp\left\{-t_i \alpha_m y_m(\boldsymbol{x}_i)\right\} \tag{13}$$

Where, for $i = 1, 2, \ldots, N$

$$w_i^{(m)} = \exp\left\{-t_i C_{(m-1)}(\boldsymbol{x}_i)\right\} \tag{14}$$

Thus

We want to determine

α_m and y_m in optimal way

Thus, rewriting

$$E = \sum_{i=1}^{N} w_i^{(m)} \exp\left\{-t_i \alpha_m y_m\left(\boldsymbol{x}_i\right)\right\} \tag{13}$$

Where, for $i = 1, 2, ..., N$

$$w_i^{(m)} = \exp\left\{-t_i C_{(m-1)}\left(\boldsymbol{x}_i\right)\right\} \tag{14}$$

Thus

We want to determine

α_m and y_m in optimal way

Thus, rewriting

$$E = \sum_{i=1}^{N} w_i^{(m)} \exp\left\{-t_i \alpha_m y_m\left(\boldsymbol{x}_i\right)\right\} \tag{13}$$

Where, for $i = 1, 2, ..., N$

$$w_i^{(m)} = \exp\left\{-t_i C_{(m-1)}\left(\boldsymbol{x}_i\right)\right\} \tag{14}$$

Remark

We have that the weight

$$w_i^{(m)} = \exp\left\{-t_i C_{(m-1)}\left(\boldsymbol{x}_i\right)\right\}$$

Needs to be used in someway for the training of the new classifier

- This is of the out most importance!!!

Remark

We have that the weight

$$w_i^{(m)} = \exp\left\{-t_i C_{(m-1)}\left(\boldsymbol{x}_i\right)\right\}$$

Needs to be used in someway for the training of the new classifier

- This is of the out most importance!!!

Therefore

You could use such weight

- As a output in the estimator function when applied to the loss function

$$\sum_{i=1}^{N} \left(y_i - w_i^{(m)} f\left(\boldsymbol{x}_i\right) \right)^2$$

  - The train using that sub-sample

Therefore

You could use such weight

- As a output in the estimator function when applied to the loss function

$$\sum_{i=1}^{N}\left(y_i - w_i^{(m)} f\left(\boldsymbol{x}_i\right)\right)^2$$

You could use such weight

- You could sub-sample with substitution by using the distribution $D_m\left\{w_i^{(m)}\right\}$ of $\boldsymbol{x}_i$
  - The train using that sub-sample

Therefore

You could use such weight

- As a output in the estimator function when applied to the loss function

$$\sum_{i=1}^{N}\left(y_i - w_i^{(m)} f\left(\boldsymbol{x}_i\right)\right)^2$$

You could use such weight

- You could sub-sample with substitution by using the distribution $D_m\left\{w_i^{(m)}\right\}$ of $\boldsymbol{x}_i$
  - The train using that sub-sample

You could apply the weight function to the loss function itself used for training

$$\sum_{i=1}^{N} w_i^{(m)}\left(y_i - w_i f\left(\boldsymbol{x}_i\right)\right)^2$$

Therefore

You could use such weight

- As a output in the estimator function when applied to the loss function

$$\sum_{i=1}^{N}\left(y_i - w_i^{(m)} f\left(\boldsymbol{x}_i\right)\right)^2$$

You could use such weight

- You could sub-sample with substitution by using the distribution $D_m\left\{w_i^{(m)}\right\}$ of $\boldsymbol{x}_i$
  - The train using that sub-sample

You could apply the weight function to the loss function itself used for training

$$\sum_{i=1}^{N} w_i^{(m)}\left(y_i - w_i f\left(\boldsymbol{x}_i\right)\right)^2$$

Thus

In the first iteration $w_i^{(1)} = 1$ for $i = 1, ..., N$

- Meaning all the points have the same importance.

During later iterations, the vector $w^{(m)}$

- It represents the weight assigned to each data point in the training set at iteration m.

Thus

In the first iteration $w_i^{(1)} = 1$ for $i = 1, ..., N$

- Meaning all the points have the same importance.

During later iterations, the vector $\boldsymbol{w}^{(m)}$

- It represents the weight assigned to each data point in the training set at iteration m.

Rewriting the Cost Equation

We can split (Eq. 13)

$$E = \sum_{t_i = y_m(\boldsymbol{x}_i)} w_i^{(m)} \exp\left\{-\alpha_m\right\} + \sum_{t_i \neq y_m(\boldsymbol{x}_i)} w_i^{(m)} \exp\left\{\alpha_m\right\} \tag{15}$$

Meaning

The total cost is the weighted cost of all hits plus the weighted cost of all misses.

Rewriting the Cost Equation

We can split (Eq. 13)

$$E=\sum_{t_i=y_m(\boldsymbol{x}_i)} w_i^{(m)}\exp\{-\alpha_m\}+\sum_{t_i\neq y_m(\boldsymbol{x}_i)} w_i^{(m)}\exp\{\alpha_m\} \tag{15}$$

Meaning

The total cost is the weighted cost of all hits plus the weighted cost of all misses.

Therefore

Writing the first summand as $W_c \exp\{-\alpha_m\}$ and the second as $W_e \exp\{\alpha_m\}$

$$E = W_c \exp\{-\alpha_m\} + W_e \exp\{\alpha_m\} \tag{16}$$

Empty

Now, for the selection of y_m

- The exact value of $\alpha_m > 0$ is irrelevant

Since a fixed α_m minimizing E

- It is equivalent to minimizing $\exp\{\alpha_m\} E$

Or in other words

$$\exp\{\alpha_m\} E = W_c + W_e \exp\{2\alpha_m\} \tag{17}$$

Empty

Now, for the selection of y_m

- The exact value of $\alpha_m > 0$ is irrelevant

Since a fixed α_m minimizing E

- It is equivalent to minimizing $\exp\{\alpha_m\} E$

Or in other words

$$\exp\{\alpha_m\} E = W_c + W_e \exp\{2\alpha_m\} \quad (17)$$

Empty

Now, for the selection of y_m

- The exact value of $\alpha_m > 0$ is irrelevant

Since a fixed α_m minimizing E

- It is equivalent to minimizing $\exp\left\{\alpha_m\right\} E$

Or in other words

$$\exp\left\{\alpha_m\right\} E = W_c + W_e \exp\left\{2\alpha_m\right\} \tag{17}$$

Now, we have

Given that $\alpha_m > 0$

$$2\alpha_m > 0$$

We have

$$\exp\left\{2\alpha_m\right\} > \exp\left\{0\right\} = 1$$

Now, we have

Given that $\alpha_m > 0$

$$2\alpha_m > 0$$

We have

$$\exp\left\{2\alpha_m\right\} > \exp\left\{0\right\} = 1$$

Then

We can rewrite (Eq. 17)

$$\exp\{\alpha_m\} E = W_c + W_e - W_e + W_e \exp\{2\alpha_m\} \tag{18}$$

Thus

$$\exp\{\alpha_m\} E = (W_c + W_e) + W_e (\exp\{2\alpha_m\} - 1) \tag{19}$$

Now, $W_c + W_e$ is the total sum W of the weights

- Of all data points which is constant in **the current iteration.**

Then

We can rewrite (Eq. 17)

$$\exp\{\alpha_m\} E = W_c + W_e - W_e + W_e \exp\{2\alpha_m\} \tag{18}$$

Thus

$$\exp\{\alpha_m\} E = (W_c + W_e) + W_e (\exp\{2\alpha_m\} - 1) \tag{19}$$

Now, $W_c + W_e$ is the total sum W of the weights

- Of all data points which is constant in **the current iteration.**

Then

We can rewrite (Eq. 17)

$$\exp\{\alpha_m\} E = W_c + W_e - W_e + W_e \exp\{2\alpha_m\} \tag{18}$$

Thus

$$\exp\{\alpha_m\} E = (W_c + W_e) + W_e (\exp\{2\alpha_m\} - 1) \tag{19}$$

Now, $W_c + W_e$ is the total sum W of the weights

- Of all data points which is constant in **the current iteration.**

Thus

The right hand side of the equation is minimized

- When at the m-th iteration, we pick the classifier with the lowest total cost W_e
  - That is the lowest rate of weighted error.

Intuitively

The next selected y_m should be the one with the lowest penalty given the current set of weights.

Thus

The right hand side of the equation is minimized

- When at the m-th iteration, we pick the classifier with the lowest total cost W_e
  - That is the lowest rate of weighted error.

Intuitively

The next selected y_m should be the one with the lowest penalty given the current set of weights.

Do you remember?

The Matrix S

- We pick the classifier with the lowest total cost W_e

Now, we need to do some updates

- Specifically the value α_m.

Do you remember?

The Matrix S

- We pick the classifier with the lowest total cost W_e

Now, we need to do some updates

- Specifically the value α_m .

Outline

Deriving against the weight α_m

Going back to the original E, we can use the derivative trick

$$\frac{\partial E}{\partial \alpha_m} = -W_c \exp\{-\alpha_m\} + W_e \exp\{\alpha_m\} \tag{20}$$

Making the equation equal to zero and multiplying by $\exp\{\alpha_m\}$

$$-W_c + W_e \exp\{2\alpha_m\} = 0 \tag{21}$$

The optimal value is thus

$$\alpha_m = \frac{1}{2}\ln\left(\frac{W_c}{W_e}\right) \tag{22}$$

Deriving against the weight α_m

Going back to the original E, we can use the derivative trick

$$\frac{\partial E}{\partial \alpha_m} = -W_c \exp\{-\alpha_m\} + W_e \exp\{\alpha_m\} \tag{20}$$

Making the equation equal to zero and multiplying by $\exp\{\alpha_m\}$

$$-W_c + W_e \exp\{2\alpha_m\} = 0 \tag{21}$$

The optimal value is thus

$$\alpha_m = \frac{1}{2}\ln\left(\frac{W_c}{W_e}\right) \tag{22}$$

Deriving against the weight α_m

Going back to the original E, we can use the derivative trick

$$\frac{\partial E}{\partial \alpha_m} = -W_c \exp\left\{-\alpha_m\right\} + W_e \exp\left\{\alpha_m\right\} \tag{20}$$

Making the equation equal to zero and multiplying by $\exp\left\{\alpha_m\right\}$

$$-W_c + W_e \exp\left\{2\alpha_m\right\} = 0 \tag{21}$$

The optimal value is thus

$$\alpha_m = \frac{1}{2} \ln\left(\frac{W_c}{W_e}\right) \tag{22}$$

Now

Making the total sum of all weights

$$W = W_c + W_e \tag{23}$$

We can rewrite the previous equation as

$$\alpha_m = \frac{1}{2}\ln\left(\frac{W - W_e}{W_e}\right) = \frac{1}{2}\ln\left(\frac{1 - e_m}{e_m}\right) \tag{24}$$

With the percentage rate of error given the weights of the data points

$$e_m = \frac{W_e}{W} \tag{25}$$

Now

Making the total sum of all weights

$$W = W_c + W_e \tag{23}$$

We can rewrite the previous equation as

$$\alpha_m = \frac{1}{2} \ln \left(\frac{W - W_e}{W_e} \right) = \frac{1}{2} \ln \left(\frac{1 - e_m}{e_m} \right) \tag{24}$$

With the percentage rate of error given the weights of the data points

$$e_m = \frac{W_e}{W} \tag{25}$$

Now

Making the total sum of all weights

$$W = W_c + W_e \tag{23}$$

We can rewrite the previous equation as

$$\alpha_m = \frac{1}{2}\ln\left(\frac{W - W_e}{W_e}\right) = \frac{1}{2}\ln\left(\frac{1 - e_m}{e_m}\right) \tag{24}$$

With the percentage rate of error given the weights of the data points

$$e_m = \frac{W_e}{W} \tag{25}$$

What about the weights?

Using the equation

$$w_i^{(m)} = \exp\left\{-t_i C_{(m-1)}\left(\boldsymbol{x}_i\right)\right\} \tag{26}$$

And because we have α_m and $y_m\left(\boldsymbol{x}_i\right)$

What about the weights?

Using the equation

$$w_i^{(m)} = \exp\left\{-t_i C_{(m-1)}\left(\boldsymbol{x}_i\right)\right\} \tag{26}$$

And because we have α_m and $y_m\left(\boldsymbol{x}_i\right)$

$$\begin{aligned} w_i^{(m+1)} &= \exp\left\{-t_i C_{(m)}\left(\boldsymbol{x}_i\right)\right\} \\ &= \exp\left\{-t_i\left[C_{(m-1)}\left(\boldsymbol{x}_i\right) + \alpha_m y_m\left(\boldsymbol{x}_i\right)\right]\right\} \\ &= w_i^{(m)} \exp\left\{-t_i \alpha_m y_m\left(\boldsymbol{x}_i\right)\right\} \end{aligned}$$

What about the weights?

Using the equation

$$w_i^{(m)} = \exp\left\{-t_i C_{(m-1)}\left(\boldsymbol{x}_i\right)\right\} \tag{26}$$

And because we have α_m and $y_m\left(\boldsymbol{x}_i\right)$

$$\begin{aligned} w_i^{(m+1)} &= \exp\left\{-t_i C_{(m)}\left(\boldsymbol{x}_i\right)\right\} \\ &= \exp\left\{-t_i\left[C_{(m-1)}\left(\boldsymbol{x}_i\right) + \alpha_m y_m\left(\boldsymbol{x}_i\right)\right]\right\} \\ &= w_i^{(m)} \exp\left\{-t_i \alpha_m y_m\left(\boldsymbol{x}_i\right)\right\} \end{aligned}$$

What about the weights?

Using the equation

$$w_i^{(m)} = \exp\left\{-t_i C_{(m-1)}\left(\boldsymbol{x}_i\right)\right\} \tag{26}$$

And because we have α_m and $y_m\left(\boldsymbol{x}_i\right)$

$$\begin{aligned} w_i^{(m+1)} &= \exp\left\{-t_i C_{(m)}\left(\boldsymbol{x}_i\right)\right\} \\ &= \exp\left\{-t_i\left[C_{(m-1)}\left(\boldsymbol{x}_i\right) + \alpha_m y_m\left(\boldsymbol{x}_i\right)\right]\right\} \\ &= w_i^{(m)} \exp\left\{-t_i \alpha_m y_m\left(\boldsymbol{x}_i\right)\right\} \end{aligned}$$

Sequential Training

Thus

- AdaBoost trains a new classifier using a data set
- There the weighting coefficients are adjusted according to the performance of the previously trained classifier
- To give greater weight to the misclassified data points.

Sequential Training

Thus

- AdaBoost trains a new classifier using a data set
- There the weighting coefficients are adjusted according to the performance of the previously trained classifier
- To give greater weight to the misclassified data points.

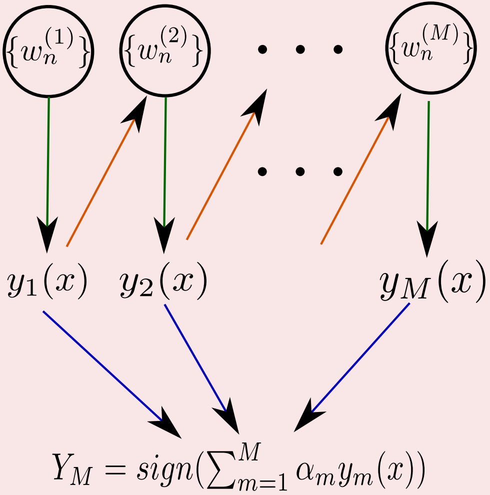
Sequential Training

Thus

- AdaBoost trains a new classifier using a data set
- There the weighting coefficients are adjusted according to the performance of the previously trained classifier
- To give greater weight to the misclassified data points.

Illustration

Schematic Illustration



Outline

AdaBoost Algorithm

Step 1

Initialize $\left\{ w_i^{(1)} \right\}$ to $\frac{1}{N}$

AdaBoost Algorithm

Step 1

Initialize $\left\{ w_i^{(1)} \right\}$ to $\frac{1}{N}$

Step 2

For $m = 1, 2, ..., M$

- Select a weak classifier $y_m(x)$ to the training data by minimizing the weighted error function or

$$\arg\min_{y_m} \sum_{i=1}^{N} w_i^{(m)} I\left(y_m(x_i) \neq t_n\right) = \arg\min_{y_m} \sum_{t_i \neq y_m(x_i)} w_i^{(m)} = \arg\min_{y_m} W_e \quad (27)$$

Where I is an indicator function.

AdaBoost Algorithm

Step 1

Initialize $\left\{ w_i^{(1)} \right\}$ to $\frac{1}{N}$

Step 2

For $m = 1, 2, ..., M$

- Select a weak classifier $y_m(\boldsymbol{x})$ to the training data by minimizing the weighted error function or

$$\arg\min_{y_m} \sum_{i=1}^{N} w_i^{(m)} I\left(y_m(\boldsymbol{x}_i) \neq t_n\right) = \arg\min_{y_m} \sum_{t_i \neq y_m(\boldsymbol{x}_i)} w_i^{(m)} = \arg\min_{y_m} W_e \quad (27)$$

Where I is an indicator function.

AdaBoost Algorithm

Step 1

Initialize $\left\{ w_i^{(1)} \right\}$ to $\frac{1}{N}$

Step 2

For $m = 1, 2, ..., M$

- Select a weak classifier $y_m(\boldsymbol{x})$ to the training data by minimizing the weighted error function or

$$\arg\min_{y_m} \sum_{i=1}^{N} w_i^{(m)} I\left(y_m(\boldsymbol{x}_i) \neq t_n\right) = \arg\min_{y_m} \sum_{t_i \neq y_m(\boldsymbol{x}_i)} w_i^{(m)} = \arg\min_{y_m} W_e \quad (27)$$

  Where I is an indicator function.

AdaBoost Algorithm

Step 2

- Evaluate

$$e_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I\left(y_m\left(\boldsymbol{x_n}\right) \neq t_n\right)}{\sum_{n=1}^{N} w_n^{(m)}} \quad (28)$$

Where I is an indicator function

AdaBoost Algorithm

Step 3

Set the α_m weight to

$$\alpha_m = \frac{1}{2} \ln \left\{ \frac{1 - e_m}{e_m} \right\} \tag{29}$$

AdaBoost Algorithm

Step 3

Set the α_m weight to

$$\alpha_m = \frac{1}{2} \ln \left\{ \frac{1 - e_m}{e_m} \right\} \tag{29}$$

Now update the weights of the data for the next iteration

- If $t_i \neq y_m(\boldsymbol{x}_i)$ i.e. a miss

$$w_i^{(m+1)} = w_i^{(m)} \exp\left\{\alpha_m\right\} = w_i^{(m)} \sqrt{\frac{1 - e_m}{e_m}} \tag{30}$$

- If $t_i == y_m(\boldsymbol{x}_i)$ i.e. a hit

$$w_i^{(m+1)} = w_i^{(m)} \exp\left\{-\alpha_m\right\} = w_i^{(m)} \sqrt{\frac{e_m}{1 - e_m}} \tag{31}$$

AdaBoost Algorithm

Step 3

Set the α_m weight to

$$\alpha_m = \frac{1}{2} \ln \left\{ \frac{1 - e_m}{e_m} \right\} \tag{29}$$

Now update the weights of the data for the next iteration

- If $t_i \neq y_m(\boldsymbol{x}_i)$ i.e. a miss

$$w_i^{(m+1)} = w_i^{(m)} \exp\left\{\alpha_m\right\} = w_i^{(m)} \sqrt{\frac{1 - e_m}{e_m}} \tag{30}$$

- If $t_i == y_m(\boldsymbol{x}_i)$ i.e. a hit

$$w_i^{(m+1)} = w_i^{(m)} \exp\left\{-\alpha_m\right\} = w_i^{(m)} \sqrt{\frac{e_m}{1 - e_m}} \tag{31}$$

Finally, make predictions

For this use

$$Y_M(\boldsymbol{x}) = sign\left(\sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x})\right) \quad (32)$$

Outline

Observations

First

The first base classifier is the usual procedure of training a single classifier.

Second

From (Eq. 30) and (Eq. 31), we can see that the weighting coefficient are increased for data points that are misclassified.

Third

- The quantity ϵ_m represent weighted measures of the error rate.
- Thus α_m gives more weight to the more accurate classifiers.

Observations

First

The first base classifier is the usual procedure of training a single classifier.

Second

From (Eq. 30) and (Eq. 31), we can see that the weighting coefficient are increased for data points that are misclassified.

Third

- The quantity ϵ_m represent weighted measures of the error rate.
- Thus α_m gives more weight to the more accurate classifiers.

Observations

First

The first base classifier is the usual procedure of training a single classifier.

Second

From (Eq. 30) and (Eq. 31), we can see that the weighting coefficient are increased for data points that are misclassified.

Third

- The quantity e_m represent weighted measures of the error rate.
- Thus α_m gives more weight to the more accurate classifiers.

In addition

The pool of classifiers in Step 1 can be substituted by a family of classifiers

One whose members are trained to minimize the error function given the current weights

Now

If indeed a finite set of classifiers is given, we only need to test the classifiers once for each data point

The Scouting Matrix S

It can be reused at each iteration by multiplying the transposed vector of weights $w^{(m)}$ with S to obtain W_e of each machine

In addition

The pool of classifiers in Step 1 can be substituted by a family of classifiers

One whose members are trained to minimize the error function given the current weights

Now

If indeed a finite set of classifiers is given, we only need to test the classifiers once for each data point

The Scouting Matrix S

It can be reused at each iteration by multiplying the transposed vector of weights $w^{(m)}$ with S to obtain W_e of each machine

In addition

The pool of classifiers in Step 1 can be substituted by a family of classifiers

One whose members are trained to minimize the error function given the current weights

Now

If indeed a finite set of classifiers is given, we only need to test the classifiers once for each data point

The Scouting Matrix S

It can be reused at each iteration by multiplying the transposed vector of weights $\boldsymbol{w}^{(m)}$ with S to obtain W_e of each machine

We have then

The following

$$\left[W_e^{(1)}\ W_e^{(2)} \cdots W_e^{M}\right] = \left(\boldsymbol{w}^{(m)}\right)^T S \tag{33}$$

We have then

The following

$$\left[W_e^{(1)}\ W_e^{(2)} \cdots W_e^{M}\right] = \left(\boldsymbol{w}^{(m)}\right)^T S \tag{33}$$

This allows to reformulate the weight update step such that

It only misses lead to weight modification.

- It could be recomputed completely at every iteration, but the iterative construction is more efficient and simple to implement.

We have then

The following

$$\left[W_e^{(1)} \; W_e^{(2)} \cdots W_e^{M} \right] = \left(\boldsymbol{w}^{(m)} \right)^T S \tag{33}$$

This allows to reformulate the weight update step such that

It only misses lead to weight modification.

Note

- Note that the weight vector $\boldsymbol{w}^{(m)}$ is constructed iteratively.
- It could be recomputed completely at every iteration, but the iterative construction is more efficient and simple to implement.

We have then

The following

$$\left[W_e^{(1)}\ W_e^{(2)} \cdots W_e^{M}\right] = \left(\boldsymbol{w}^{(m)}\right)^T S \quad (33)$$

This allows to reformulate the weight update step such that

It only misses lead to weight modification.

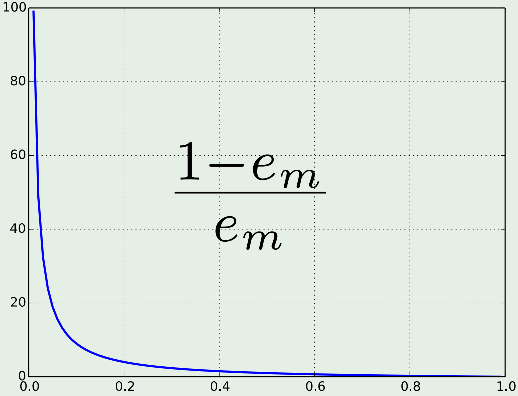
Note

- Note that the weight vector $\boldsymbol{w}^{(m)}$ is constructed iteratively.
- It could be recomputed completely at every iteration, but the iterative construction is more efficient and simple to implement.

Outline

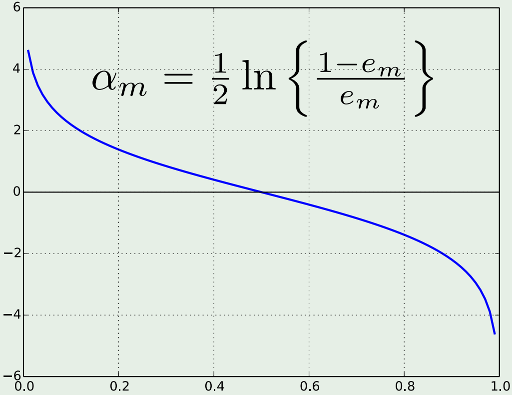
Explanation

Graph for $\frac{1-e_m}{e_m}$ in the range $0 \leq e_m \leq 1$



So we have

Graph for α_m



We have the following cases

We have the following

If $e_m \longrightarrow 1$, we have that all the samples were not correctly classified!!!

Thus

We get that for all miss-classified sample $\lim_{e_m \to 1} \frac{1-e_m}{e_m} \longrightarrow 0$, then $\alpha_m \longrightarrow -\infty$

We have the following cases

We have the following

If $e_m \longrightarrow 1$, we have that all the samples were not correctly classified!!!

Thus

We get that for all miss-classified sample $\lim_{e_m \to 1} \frac{1-e_m}{e_m} \longrightarrow 0$, then $\alpha_m \longrightarrow -\infty$

Now

We get that for all miss-classified sample

$$w_i^{(m+1)} = w_i^{(m)} \exp\left\{\alpha_m\right\} \longrightarrow 0$$

Therefore

- We only need to reverse the answers to get the perfect classifier and select it as the only committee member.

Now

We get that for all miss-classified sample

$$w_i^{(m+1)} = w_i^{(m)} \exp\{\alpha_m\} \longrightarrow 0$$

Therefore

- We only need to reverse the answers to get the perfect classifier and select it as the only committee member.

Now, the Last Case

If $e_m \longrightarrow 1/2$

- We have $\alpha_m \longrightarrow 0$

Thus we have that if the sample is well or bad classified

$$\exp\left\{-\alpha_m t_i y_m\left(\boldsymbol{x}_i\right)\right\} \rightarrow 1 \tag{34}$$

Therefore

- The weight does not change at all.

Now, the Last Case

If $e_m \longrightarrow 1/2$

- We have $\alpha_m \longrightarrow 0$

Thus we have that if the sample is well or bad classified

$$\exp\left\{-\alpha_m t_i y_m\left(\boldsymbol{x}_i\right)\right\} \rightarrow 1 \tag{34}$$

Therefore

- The weight does not change at all.

Now, the Last Case

If $e_m \longrightarrow 1/2$

- We have $\alpha_m \longrightarrow 0$

Thus we have that if the sample is well or bad classified

$$\exp\left\{-\alpha_m t_i y_m\left(\boldsymbol{x_i}\right)\right\} \rightarrow 1 \tag{34}$$

Therefore

- The weight does not change at all.

Thus, we have

What about $e_m \to 0$

- We have that $\alpha_m \to +\infty$

- Thus, the only need m committee members, we do not need another $m+1$ member

Thus, we have

What about $e_m \to 0$

- We have that $\alpha_m \to +\infty$

Thus, we have

Samples always correctly classified

$$w_i^{(m+1)} = w_i^{(m)} \exp\left\{-\alpha_m t_i y_m\left(\boldsymbol{x}_i\right)\right\} \to 0$$

- Thus, the only need m committee members, we do not need another $m+1$ member.

Thus, we have

What about $e_m \to 0$

- We have that $\alpha_m \to +\infty$

Thus, we have

Samples always correctly classified

$$w_i^{(m+1)} = w_i^{(m)} \exp\left\{-\alpha_m t_i y_m\left(\boldsymbol{x}_i\right)\right\} \to 0$$

- Thus, the only need m committee members, we do not need another $m+1$ member.

Outline

This comes from

The paper

- "Additive Logistic Regression: A Statistical View of Boosting" by Friedman, Hastie and Tibshirani

Something Notable

- In this paper, a proof exists to show that boosting algorithms are procedures to fit and additive logistic regression model.

$$E[y|x] = F(x) \text{ with } F(x) = \sum_{m=1}^{M} f_m(x)$$

This comes from

The paper

- "Additive Logistic Regression: A Statistical View of Boosting" by Friedman, Hastie and Tibshirani

Something Notable

- In this paper, a proof exists to show that boosting algorithms are procedures to fit and additive logistic regression model.

$$E\left[y|\boldsymbol{x}\right] = F\left(\boldsymbol{x}\right) \text{ with } F\left(\boldsymbol{x}\right) = \sum_{m=1}^{M} f_m\left(\boldsymbol{x}\right)$$

Consider the Additive Regression Model

We are interested in modeling the mean $E[y|\boldsymbol{x}] = F(\boldsymbol{x})$

- With Additive Model

$$F(\boldsymbol{x}) = \sum_{i=1}^{d} f_i(x_i)$$

Where each $f_i(x_i)$ is a function for each feature input x_i

- A convenient algorithm for updating these models it the **backfitting algorithm** with update:

$$f_i(x_i) = E\left[y - \sum_{k\neq i} f_k(x_k) \,|\, x_i\right]$$

Consider the Additive Regression Model

We are interested in modeling the mean $E\left[y|\boldsymbol{x}\right] = F\left(\boldsymbol{x}\right)$

- With Additive Model

$$F\left(\boldsymbol{x}\right) = \sum_{i=1}^{d} f_i\left(x_i\right)$$

Where each $f_i\left(x_i\right)$ is a function for each feature input x_i

- A convenient algorithm for updating these models it the **backfitting algorithm** with update:

$$f_i\left(x_i\right) = E\left[y - \sum_{k \neq i} f_k\left(x_k\right)|x_i\right]$$

Remarks

An example of these additive models is the matching pursuit

$$f(t) = \sum_{n=-\infty}^{+\infty} a_n g_{\gamma_n}(t)$$

Backfitting ensures that under fairly general conditions

- Backfitting converges to the minimizer of $E\left[(y - f(x))^2\right]$

Remarks

An example of these additive models is the matching pursuit

$$f(t) = \sum_{n=-\infty}^{+\infty} a_n g_{\gamma_n}(t)$$

Backfitting ensures that under fairly general conditions

- Backfitting converges to the minimizer of $E\left[(y - f(\boldsymbol{x}))^2\right]$

In the case of AdaBoost

We have an additive model

- Which considers functions $\{f_m(\boldsymbol{x})\}_{m=1}^{M}$ that take in account all the features - Perceptron, Decision Trees, etc

Each of these functions is characterized by a set of parameters γ_m and multiplier α_m

$$f_m(\boldsymbol{x}) = \alpha_m y_m(\boldsymbol{x}|\gamma_m)$$

With additive model

$$F_M(\boldsymbol{x}) = \alpha_1 y_1(\boldsymbol{x}|\gamma_1) + \cdots + \alpha_M y_M(\boldsymbol{x}|\gamma_M)$$

In the case of AdaBoost

We have an additive model

- Which considers functions $\{f_m(\boldsymbol{x})\}_{m=1}^{M}$ that take in account all the features - Perceptron, Decision Trees, etc

Each of these functions is characterized by a set of parameters γ_m and multiplier α_m

$$f_m(\boldsymbol{x}) = \alpha_m y_m(\boldsymbol{x}|\gamma_m)$$

With additive model

$$F_M(\boldsymbol{x}) = \alpha_1 y_1(\boldsymbol{x}|\gamma_1) + \cdots + \alpha_M y_M(\boldsymbol{x}|\gamma_M)$$

In the case of AdaBoost

We have an additive model

- Which considers functions $\{f_m(\boldsymbol{x})\}_{m=1}^{M}$ that take in account all the features - Perceptron, Decision Trees, etc

Each of these functions is characterized by a set of parameters γ_m and multiplier α_m

$$f_m(\boldsymbol{x}) = \alpha_m y_m(\boldsymbol{x}|\gamma_m)$$

With additive model

$$F_M(\boldsymbol{x}) = \alpha_1 y_1(\boldsymbol{x}|\gamma_1) + \cdots + \alpha_M y_M(\boldsymbol{x}|\gamma_M)$$

Outline

Remark - Moving from Regression to Classification

Given that Regression have wide ranges of outputs

- Logistic Regression is widely used to move Regression to Classification

$$\log \frac{P\left(Y=1|\boldsymbol{x}\right)}{P\left(Y=-1|\boldsymbol{x}\right)} = \sum_{m=1}^{M} f_m\left(\boldsymbol{x}\right)$$

A nice property, the probability estimates lie in [0, 1]

- Now, solving by assuming $P\left(Y=1|\boldsymbol{x}\right) + P\left(Y=-1|\boldsymbol{x}\right) = 1$

$$P\left(Y=1|\boldsymbol{x}\right) = \frac{e^{F(\boldsymbol{x})}}{1+e^{F(\boldsymbol{x})}}$$

Remark - Moving from Regression to Classification

Given that Regression have wide ranges of outputs

- Logistic Regression is widely used to move Regression to Classification

$$\log \frac{P(Y=1|\boldsymbol{x})}{P(Y=-1|\boldsymbol{x})} = \sum_{m=1}^{M} f_m(\boldsymbol{x})$$

A nice property, the probability estimates lie in $[0,1]$

- Now, solving by assuming $P(Y=1|\boldsymbol{x}) + P(Y=-1|\boldsymbol{x}) = 1$

$$P(Y=1|\boldsymbol{x}) = \frac{e^{F(\boldsymbol{x})}}{1+e^{F(\boldsymbol{x})}}$$

Outline

The Exponential Criterion

We have our exponential Criterion under an Expected Value with $y \in \{1, -1\}$

$$J\left(F\right) = E\left[e^{-yF(\boldsymbol{x})}\right]$$

Lemma

- $E\left[e^{-yF(\boldsymbol{x})}\right]$ is minimized at

$$F\left(\boldsymbol{x}\right) = \frac{1}{2}\log\frac{P\left(Y=1|\boldsymbol{x}\right)}{P\left(Y=-1|\boldsymbol{x}\right)}$$

Hence:

$$P\left(Y=1|\boldsymbol{x}\right) = \frac{e^{F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}}$$

$$P\left(Y=-1|\boldsymbol{x}\right) = \frac{e^{-F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}}$$

The Exponential Criterion

We have our exponential Criterion under an Expected Value with $y \in \{1, -1\}$

$$J(F) = E\left[e^{-yF(\boldsymbol{x})}\right]$$

Lemma

- $E\left[e^{-yF(\boldsymbol{x})}\right]$ is minimized at

$$F(\boldsymbol{x}) = \frac{1}{2}\log\frac{P(Y=1|\boldsymbol{x})}{P(Y=-1|\boldsymbol{x})}$$

Hence:

$$P(Y=1|\boldsymbol{x}) = \frac{e^{F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}}$$
$$P(Y=-1|\boldsymbol{x}) = \frac{e^{-F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}}$$

Proof

Given the discrete nature of $y \in \{1, -1\}$

$$\frac{\partial E\left[e^{-yF(\boldsymbol{x})}\right]}{\partial F(\boldsymbol{x})} = -P\left(Y = 1|\boldsymbol{x}\right) e^{-F(\boldsymbol{x})} + P\left(Y = -1|\boldsymbol{x}\right) e^{F(\boldsymbol{x})}$$

Therefore

$$-P\left(Y = 1|\boldsymbol{x}\right) e^{-F(\boldsymbol{x})} + P\left(Y = -1|\boldsymbol{x}\right) e^{F(\boldsymbol{x})} = 0$$

Proof

Given the discrete nature of $y \in \{1, -1\}$

$$\frac{\partial E\left[e^{-yF(\boldsymbol{x})}\right]}{\partial F(\boldsymbol{x})} = -P(Y=1|\boldsymbol{x})\, e^{-F(\boldsymbol{x})} + P(Y=-1|\boldsymbol{x})\, e^{F(\boldsymbol{x})}$$

Therefore

$$-P(Y=1|\boldsymbol{x})\, e^{-F(\boldsymbol{x})} + P(Y=-1|\boldsymbol{x})\, e^{F(\boldsymbol{x})} = 0$$

Then

We have that

$$
\begin{aligned}
P\left(Y=1|\boldsymbol{x}\right)e^{-F(\boldsymbol{x})} &= P\left(Y=-1|\boldsymbol{x}\right)e^{F(\boldsymbol{x})}\\
&= \left[1-P\left(Y=1|\boldsymbol{x}\right)\right]e^{F(\boldsymbol{x})}
\end{aligned}
$$

Solving

$$
e^{F(\boldsymbol{x})} = \left[e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}\right]P\left(Y=1|\boldsymbol{x}\right)
$$

Then

We have that

$$P(Y=1|\boldsymbol{x})\,e^{-F(\boldsymbol{x})} = P(Y=-1|\boldsymbol{x})\,e^{F(\boldsymbol{x})}$$
$$= \left[1-P(Y=1|\boldsymbol{x})\right]e^{F(\boldsymbol{x})}$$

Solving

$$e^{F(\boldsymbol{x})} = \left[e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}\right]P(Y=1|\boldsymbol{x})$$

Finally, we have

The first equation

$$P\left(Y=1|\boldsymbol{x}\right)=\frac{e^{F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}}$$

Similarly

$$P\left(Y=-1|\boldsymbol{x}\right)=\frac{e^{-F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}}$$

Finally, we have

The first equation

$$P\left(Y=1|\boldsymbol{x}\right)=\frac{e^{F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}}$$

Similarly

$$P\left(Y=-1|\boldsymbol{x}\right)=\frac{e^{-F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})}+e^{F(\boldsymbol{x})}}$$

Basically

We have that the $E\left[e^{-yF(\boldsymbol{x})}\right]$

- When you minimize the cost function

Then at the optimal you have the Binary Classification

- Of the Logistic Regression

Basically

We have that the $E\left[e^{-yF(\boldsymbol{x})}\right]$

- When you minimize the cost function

Then at the optimal you have the Binary Classification

- Of the Logistic Regression

Furthermore

Corollary

- If E is replaced by averages over regions of x where $F(x)$ is constant (Similar to a decision tree),
  - The same result applies to the sample proportions of $y = 1$ and $y = -1$

Furthermore

Corollary

- If E is replaced by averages over regions of $\boldsymbol{x}$ where $F(\boldsymbol{x})$ is constant (Similar to a decision tree),
  - The same result applies to the sample proportions of $y = 1$ and $y = -1$

Outline

Finally, The Additive Logistic Regression

Proposition

- The AdaBoost algorithm fits an additive logistic regression model by stage-wise optimization of

$$J(F) = E\left[e^{-yF(\boldsymbol{x})}\right]$$

Proof

- Imagine you have an estimate $F(\boldsymbol{x})$ then we seek an improved estimate:

$$F(\boldsymbol{x}) + f(\boldsymbol{x})$$

Finally, The Additive Logistic Regression

Proposition

- The AdaBoost algorithm fits an additive logistic regression model by stage-wise optimization of

$$J\left(F\right) = E\left[e^{-yF(\boldsymbol{x})}\right]$$

Proof

- Imagine you have an estimate $F\left(\boldsymbol{x}\right)$ then we seek an improved estimate:

$$F\left(\boldsymbol{x}\right) + f\left(\boldsymbol{x}\right)$$

For This

We minimize at each $\boldsymbol{x}$

$$J\left(F\left(\boldsymbol{x}\right)+f\left(\boldsymbol{x}\right)\right)$$

This can be expanded

$$\begin{aligned} J\left(F\left(\boldsymbol{x}\right)+f\left(\boldsymbol{x}\right)\right) &= E\left[e^{-y\left(F\left(\boldsymbol{x}\right)+f\left(\boldsymbol{x}\right)\right)}|\boldsymbol{x}\right] \\ &= e^{-f\left(\boldsymbol{x}\right)}E\left[e^{-yF\left(\boldsymbol{x}\right)}I\left(y=1\right)|\boldsymbol{x}\right]+ \\ &\quad ...e^{f\left(\boldsymbol{x}\right)}E\left[e^{-yF\left(\boldsymbol{x}\right)}I\left(y=-1\right)|\boldsymbol{x}\right] \end{aligned}$$

For This

We minimize at each $\boldsymbol{x}$

$$J\left(F\left(\boldsymbol{x}\right)+f\left(\boldsymbol{x}\right)\right)$$

This can be expanded

$$\begin{aligned} J\left(F\left(\boldsymbol{x}\right)+f\left(\boldsymbol{x}\right)\right) &= E\left[e^{-y(F(\boldsymbol{x})+f(\boldsymbol{x}))}|\boldsymbol{x}\right] \\ &= e^{-f(\boldsymbol{x})}E\left[e^{-yF(\boldsymbol{x})}I\left(y=1\right)|\boldsymbol{x}\right]+ \\ &\ldots e^{f(\boldsymbol{x})}E\left[e^{-yF(\boldsymbol{x})}I\left(y=-1\right)|\boldsymbol{x}\right] \end{aligned}$$

Deriving w.r.t. $f(\boldsymbol{x})$

We get

$$-e^{-f(\boldsymbol{x})}E\left[e^{-yF(\boldsymbol{x})}I\left(y=1\right)|\boldsymbol{x}\right]+e^{f(\boldsymbol{x})}E\left[e^{-yF(\boldsymbol{x})}I\left(y=-1\right)|\boldsymbol{x}\right]=0$$

We have the following

If we divide by $E\left[e^{-yF(\boldsymbol{x})}|\boldsymbol{x}\right]$, the first term

$$\frac{E\left[e^{-yF(\boldsymbol{x})}I\left(y=1\right)|\boldsymbol{x}\right]}{E\left[e^{-yF(\boldsymbol{x})}|\boldsymbol{x}\right]}=E_w\left[I\left(y=1\right)|\boldsymbol{x}\right]$$

Also

$$\frac{E\left[e^{-yF(\boldsymbol{x})}I\left(y=-1\right)|\boldsymbol{x}\right]}{E\left[e^{-yF(\boldsymbol{x})}|\boldsymbol{x}\right]}=E_w\left[I\left(y=-1\right)|\boldsymbol{x}\right]$$

We have the following

If we divide by $E\left[e^{-yF(\boldsymbol{x})}|\boldsymbol{x}\right]$, the first term

$$\frac{E\left[e^{-yF(\boldsymbol{x})}I\left(y=1\right)|\boldsymbol{x}\right]}{E\left[e^{-yF(\boldsymbol{x})}|\boldsymbol{x}\right]}=E_w\left[I\left(y=1\right)|\boldsymbol{x}\right]$$

Also

$$\frac{E\left[e^{-yF(\boldsymbol{x})}I\left(y=-1\right)|\boldsymbol{x}\right]}{E\left[e^{-yF(\boldsymbol{x})}|\boldsymbol{x}\right]}=E_w\left[I\left(y=-1\right)|\boldsymbol{x}\right]$$

Thus, we have

We apply the natural log to both sides

$$\log e^{-f(\boldsymbol{x})} + \log E_w \left[I \left(y = 1 \right) | \boldsymbol{x} \right] = \log e^{f(\boldsymbol{x})} + \log E_w \left[I \left(y = -1 \right) | \boldsymbol{x} \right]$$

Then

$$2f\left(\boldsymbol{x}\right) = \log E_w \left[I \left(y = 1 \right) | \boldsymbol{x} \right] - \log E_w \left[I \left(y = -1 \right) | \boldsymbol{x} \right]$$

Thus, we have

We apply the natural log to both sides

$$\log e^{-f(\boldsymbol{x})} + \log E_w \left[I \left(y = 1 \right) | \boldsymbol{x} \right] = \log e^{f(\boldsymbol{x})} + \log E_w \left[I \left(y = -1 \right) | \boldsymbol{x} \right]$$

Then

$$2f \left(\boldsymbol{x} \right) = \log E_w \left[I \left(y = 1 \right) | \boldsymbol{x} \right] - \log E_w \left[I \left(y = -1 \right) | \boldsymbol{x} \right]$$

Finally

We have that

$$\widehat{f}\left(\boldsymbol{x}\right) = \frac{1}{2}\log\frac{E_w\left[I\left(y=1\right)|\boldsymbol{x}\right]}{E_w\left[I\left(y=-1\right)|\boldsymbol{x}\right]}$$

In term of probabilities

$$\widehat{f}\left(\boldsymbol{x}\right) = \frac{1}{2}\log\frac{P_w\left(y=1|\boldsymbol{x}\right)}{P_w\left(y=-1|\boldsymbol{x}\right)}$$

Finally

We have that

$$\widehat{f}(\boldsymbol{x}) = \frac{1}{2}\log\frac{E_w\left[I\left(y=1\right)|\boldsymbol{x}\right]}{E_w\left[I\left(y=-1\right)|\boldsymbol{x}\right]}$$

In term of probabilities

$$\widehat{f}(\boldsymbol{x}) = \frac{1}{2}\log\frac{P_w\left(y=1|\boldsymbol{x}\right)}{P_w\left(y=-1|\boldsymbol{x}\right)}$$

The Weight Update

Finally, we have a way to update the weights by setting $w_t(\boldsymbol{x}, y) = e^{-yF(\boldsymbol{x})}$

$$w_{t+1}(\boldsymbol{x}, y) = w_t(\boldsymbol{x}, y)\, e^{-y\widehat{f}(\boldsymbol{x})}$$

Additionally, the weighted conditional mean

Corollary

- At the Optimal $F(\boldsymbol{x})$, the weighted conditional mean of y is 0.

Proof

- When $F(\boldsymbol{x})$ is optimal

$$\frac{\partial J(F(\boldsymbol{x}))}{\partial F(\boldsymbol{x})} = \frac{\partial \left\{ P(Y=1|\boldsymbol{x})\, e^{-yF(\boldsymbol{x})} + P(Y=-1|\boldsymbol{x})\, e^{yF(\boldsymbol{x})} \right\}}{\partial F(\boldsymbol{x})}$$

Additionally, the weighted conditional mean

Corollary

- At the Optimal $F(\boldsymbol{x})$, the weighted conditional mean of y is 0.

Proof

- When $F(\boldsymbol{x})$ is optimal

$$\frac{\partial J(F(\boldsymbol{x}))}{\partial F(\boldsymbol{x})} = \frac{\partial \left\{ P(Y=1|\boldsymbol{x}) e^{-yF(\boldsymbol{x})} + P(Y=-1|\boldsymbol{x}) e^{yF(\boldsymbol{x})} \right\}}{\partial F(\boldsymbol{x})}$$

Therefore

We have

$$\frac{\partial J\left(F\left(\boldsymbol{x}\right)\right)}{\partial F\left(\boldsymbol{x}\right)} = \left[P\left(Y=1|\boldsymbol{x}\right)e^{-yF(\boldsymbol{x})}\right]\{-y\} + \left[P\left(Y=-1|\boldsymbol{x}\right)e^{-yF(\boldsymbol{x})}\right]\{-y\}$$

Therefore

$$E\left[e^{yF(\boldsymbol{x})}y\right] = 0$$

Therefore

We have

$$\frac{\partial J\left(F\left(\boldsymbol{x}\right)\right)}{\partial F\left(\boldsymbol{x}\right)} = \left[P\left(Y = 1|\boldsymbol{x}\right)e^{-yF(\boldsymbol{x})}\right]\{-y\} + \left[P\left(Y = -1|\boldsymbol{x}\right)e^{-yF(\boldsymbol{x})}\right]\{-y\}$$

Therefore

$$E\left[e^{yF(\boldsymbol{x})}y\right] = 0$$

Outline

Here, we decide to use Perceptrons

As Weak Learners

- We could be using a finite number of Perceptrons
- But we want to have a infinitude of possible weak learners
  - Thus avoiding the need of a matrix S

Here, we decide to use Perceptrons

As Weak Learners

- We could be using a finite number of Perceptrons
- But we want to have a infinitude of possible weak learners
  - Thus avoiding the need of a matrix S

Remark

- We need to use a Gradient Based Learner for this

Here, we decide to use Perceptrons

As Weak Learners

- We could be using a finite number of Perceptrons
- But we want to have a infinitude of possible weak learners
  - Thus avoiding the need of a matrix S

Remark

- We need to use a Gradient Based Learner for this

Perceptron

We use the following formula of error per sample

$$E\left(\boldsymbol{w}\right) = \frac{1}{2}\sum_{j=1}^{N}\left(w_j\left(t\right)y_j\left(t\right) - d_j\right)^2$$

- With $y_j\left(t\right) = \varphi\left(\boldsymbol{w}^T\left(t\right)\boldsymbol{x}_j\right)$

Deriving against w_i

$$\frac{\partial E\left(\boldsymbol{w}\right)}{\partial w_i} = \sum_{j=1}^{N}\left(w_j\left(t\right)y_j\left(t\right) - d_j\right)\varphi'\left(\boldsymbol{w}^T\left(t\right)\boldsymbol{x}_j\right)w_j^b x_{ij}$$

Then, using gradient descent, we have the following update

$$w_i\left(n+1\right) = w_i\left(n\right) - \eta\left[\sum_{j=1}^{N}\left(w_j\left(t\right)y_j\left(t\right) - d_j\right)\varphi'\left(\boldsymbol{w}^T\left(t\right)\boldsymbol{x}_j\right)w_i x_{ij}\right]$$

Perceptron

We use the following formula of error per sample

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{j=1}^{N}\left(w_j(t)\,y_j(t) - d_j\right)^2$$

- With $y_j(t) = \varphi\left(\boldsymbol{w}^T(t)\,\boldsymbol{x}_j\right)$

Deriving against w_i

$$\frac{\partial E(\boldsymbol{w})}{\partial w_i} = \sum_{j=1}^{N}\left(w_j(t)\,y_j(t) - d_j\right)\varphi'\left(\boldsymbol{w}^T(t)\,\boldsymbol{x}_j\right) w_j^b x_{ij}$$

Then, using gradient descent, we have the following update

$$w_i(n+1) = w_i(n) - \eta\left[\sum_{j=1}^{N}\left(w_j(t)\,y_j(t) - d_j\right)\varphi'\left(\boldsymbol{w}^T(t)\,\boldsymbol{x}_j\right) w_i x_{ij}\right]$$

Perceptron

We use the following formula of error per sample

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{j=1}^{N}\left(w_j(t)\,y_j(t) - d_j\right)^2$$

- With $y_j(t) = \varphi\left(\boldsymbol{w}^T(t)\,\boldsymbol{x}_j\right)$

Deriving against w_i

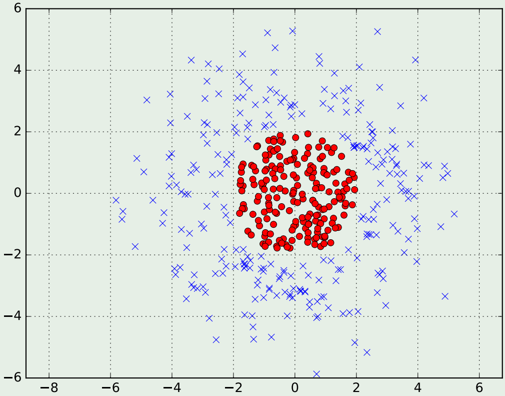
$$\frac{\partial E(\boldsymbol{w})}{\partial w_i} = \sum_{j=1}^{N}\left(w_j(t)\,y_j(t) - d_j\right)\varphi'\left(\boldsymbol{w}^T(t)\,\boldsymbol{x}_j\right)w_j^b x_{ij}$$

Then, using gradient descent, we have the following update

$$w_i(n+1) = w_i(n) - \eta\left[\sum_{j=1}^{N}\left(w_j(t)\,y_j(t) - d_j\right)\varphi'\left(\boldsymbol{w}^T(t)\,\boldsymbol{x}_j\right)w_i x_{ij}\right]$$

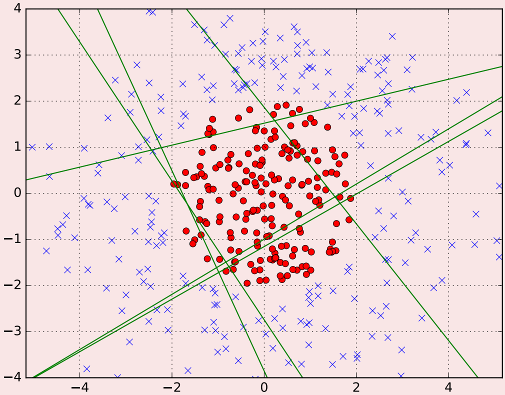
Data Set

Training set with classes $\omega_1 = N(0,1)$ and $\omega_2 = N(0,\sigma^2) - N(0,1)$



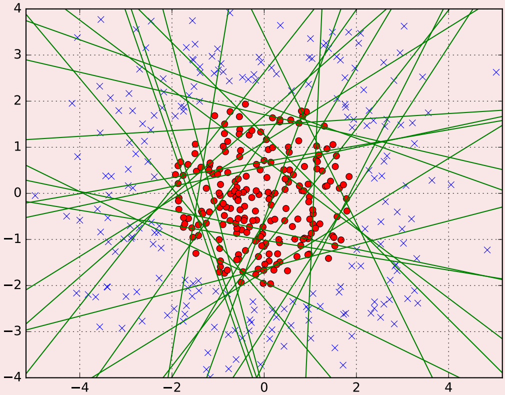
Example

For $m = 10$



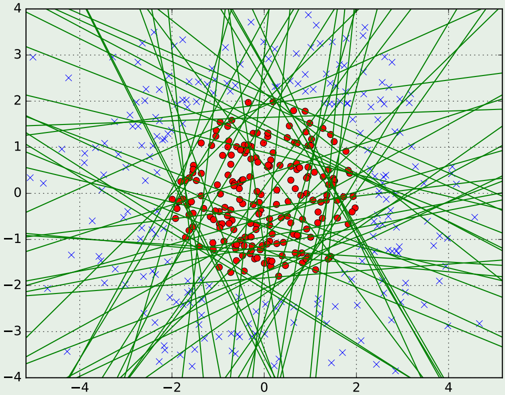
Example

For 40



At the end of the process

For $m = 80$



Final Confusion Matrix

When $m = 80$

	C_1	C_2
C_1	1.0	0.0
C_2	0.0	1.0

However

There are other versions to the Cryptic Phrase

- At "Boosting: Foundation and Algorithms" by Schaphire and Freund
  - "Train weak learner using distribution D_t"

We could re-sample using the distribution w_t

- Basically using sampling with substitution over the data set $\{x_1, x_2, ..., x_N\}$

However

There are other versions to the Cryptic Phrase

- At "Boosting: Foundation and Algorithms" by Schaphire and Freund
  - "Train weak learner using distribution D_t"

We could re-sample using the distribution $\boldsymbol{w}_t$

- Basically using sampling with substitution over the data set $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$

Other Interpretations exist

But you can use a weighted version of the cost function

$$\frac{1}{2}\sum_{j} w_j\left(t\right)\left(y_j\left(t\right)-d_j\right)^2$$

For More, Take a look

- "Boosting Neural Networks" by Holger Schwenk and Yoshua Bengio