Introduction to Machine Learning

Logistic Regression

Andres Mendez-Vazquez

January 26, 2023

Outline

Outline

Assume the following

Let $Y_1, Y_2, ..., Y_N$ independent random variables

Taking values in the set $\{0, 1\}$

Now, you have a set of fixed vectors

$$x_1, x_2, ..., x_N$$

Mapped to a series of numbers by a weight vector w

$$w^T x_1, w^T x_2, ..., w^T x_N$$

Assume the following

Let $Y_1, Y_2, ..., Y_N$ independent random variables

Taking values in the set $\{0, 1\}$

Now, you have a set of fixed vectors

$$\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$$

Mapped to a series of numbers by a weight vector $\boldsymbol{w}$

$$\boldsymbol{w}^T \boldsymbol{x}_1, \boldsymbol{w}^T \boldsymbol{x}_2, ..., \boldsymbol{w}^T \boldsymbol{x}_N$$

Assume the following

Let $Y_1, Y_2, ..., Y_N$ independent random variables

Taking values in the set $\{0, 1\}$

Now, you have a set of fixed vectors

$$\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$$

Mapped to a series of numbers by a weight vector $\boldsymbol{w}$

$$\boldsymbol{w}^T \boldsymbol{x}_1, \boldsymbol{w}^T \boldsymbol{x}_2, ..., \boldsymbol{w}^T \boldsymbol{x}_N$$

In our simplest from [1, 2]

There is a suspected relation

Between $\theta_i = P\left(Y_i = 1\right)$ and $\boldsymbol{w}^T \boldsymbol{x}_i$

- Here Y is the random variable and y is the value that the random variable can take.

Thus we have

$$y = \begin{cases} 1 & w^T x + e > 0 \\ 0 & \text{else} \end{cases}$$

Note: Where e is an error with a certain distribution!!!

In our simplest from [1, 2]

There is a suspected relation

Between $\theta_i = P\left(Y_i = 1\right)$ and $\boldsymbol{w}^T \boldsymbol{x}_i$

- Here Y is the random variable and y is the value that the random variable can take.

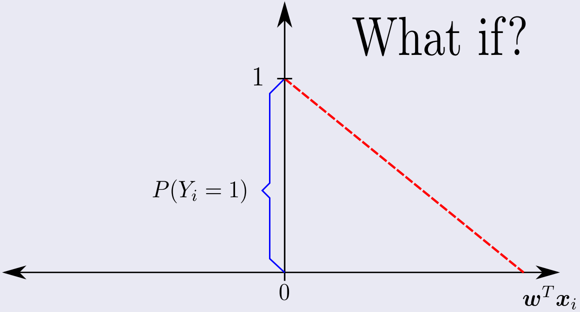
Thus we have

$$y = \begin{cases} 1 & \boldsymbol{w}^T \boldsymbol{x} + e > 0 \\ 0 & \text{else} \end{cases}$$

Note: Where e is an error with a certain distribution!!!

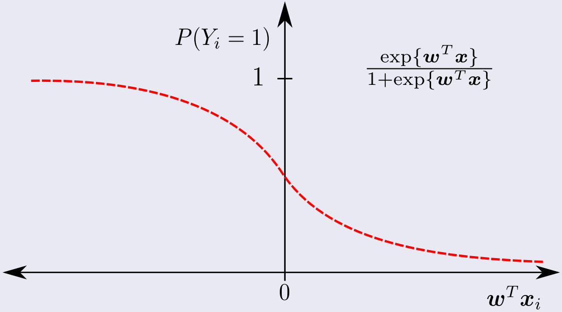
For Example, Graphically

We have



It is better to user a logit version

We have



Logit Distribution

PDF with support $z \in (-\infty, \infty)$,μ location and s scale

$$p\left(x|\mu,s\right)=\frac{\exp\left\{-\frac{z-\mu}{s}\right\}}{s\left(1+\exp\left\{-\frac{z-\mu}{s}\right\}\right)^{2}}$$

With a CDF

$$P\left(Y<z\right)=\int_{-\infty}^{z}p\left(y|\mu,s\right)dy=\frac{1}{1+\exp\left\{-\frac{z-\mu}{s}\right\}}$$

Logit Distribution

PDF with support $z \in (-\infty, \infty)$,μ location and s scale

$$p\left(x|\mu, s\right) = \frac{\exp\left\{-\frac{z-\mu}{s}\right\}}{s\left(1+\exp\left\{-\frac{z-\mu}{s}\right\}\right)^2}$$

With a CDF

$$P\left(Y < z\right) = \int_{-\infty}^{z} p\left(y|\mu, s\right) dy = \frac{1}{1+\exp\left\{-\frac{z-\mu}{s}\right\}}$$

Outline

In Bayesian Classification

Assignment of a pattern

It is performed by using the posterior probabilities, $P(\omega_i|\boldsymbol{x})$

And given K classes, we want

$$\sum_{i=1}^{K} P(\omega_i|\boldsymbol{x}) = 1$$

Such that each

$$0 \leq P(\omega_i|\boldsymbol{x}) \leq 1$$

In Bayesian Classification

Assignment of a pattern

It is performed by using the posterior probabilities, $P\left(\omega_i|\boldsymbol{x}\right)$

And given K classes, we want

$$\sum_{i=1}^{K} P\left(\omega_i|\boldsymbol{x}\right) = 1$$

Such that each

$$0 \leq P\left(\omega_i|\boldsymbol{x}\right) \leq 1$$

In Bayesian Classification

Assignment of a pattern

It is performed by using the posterior probabilities, $P\left(\omega_i|\boldsymbol{x}\right)$

And given K classes, we want

$$\sum_{i=1}^{K} P\left(\omega_i|\boldsymbol{x}\right) = 1$$

Such that each

$$0 \leq P\left(\omega_i|\boldsymbol{x}\right) \leq 1$$

Observation

This is a typical example of the discriminative approach

- Where the distribution of data is of no interest.
  - In the Logistic Regression the Distribution is imposed over the output!!!

Outline

The Model

We have the following under the extended features

$$\log \frac{P(\omega_1|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \boldsymbol{w}_1^T \boldsymbol{x}$$

$$\log \frac{P(\omega_2|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \boldsymbol{w}_2^T \boldsymbol{x}$$

$$\vdots$$

$$\log \frac{P(\omega_{K-1}|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \boldsymbol{w}_{K-1}^T \boldsymbol{x}$$

The Model

We have the following under the extended features

$$\log \frac{P(\omega_1|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \boldsymbol{w}_1^T\boldsymbol{x}$$

$$\log \frac{P(\omega_2|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \boldsymbol{w}_2^T\boldsymbol{x}$$

$$\vdots$$

$$\log \frac{P(\omega_{K-1}|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \boldsymbol{w}_{K-1}^T\boldsymbol{x}$$

The Model

We have the following under the extended features

$$
\begin{aligned}
\log \frac{P(\omega_1|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} &= \boldsymbol{w}_1^T \boldsymbol{x} \\
\log \frac{P(\omega_2|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} &= \boldsymbol{w}_2^T \boldsymbol{x} \\
&\vdots \\
\log \frac{P(\omega_{K-1}|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} &= \boldsymbol{w}_{K-1}^T \boldsymbol{x}
\end{aligned}
$$

Further

We have

The model is specified in terms of $K-1$ log-odds or logit transformations.

And

Although the model uses the last class as the denominator in the odds-ratios.

The choice of denominator is arbitrary

- However, because the estimates are equivariant under this choice.
  - The action taken in a decision problem should not depend on transformation on the measurement used

Further

We have

The model is specified in terms of $K-1$ log-odds or logit transformations.

And

Although the model uses the last class as the denominator in the odds-ratios.

The choice of denominator is arbitrary

- However, because the estimates are equivariant under this choice.
  - The action taken in a decision problem should not depend on transformation on the measurement used

Further

We have

The model is specified in terms of $K-1$ log-odds or logit transformations.

And

Although the model uses the last class as the denominator in the odds-ratios.

The choice of denominator is arbitrary

- However, because the estimates are equivariant under this choice.
  - The action taken in a decision problem should not depend on transformation on the measurement used

Now

How do we find the terms?

$$P\left(\omega_1|\boldsymbol{x}\right), P\left(\omega_2|\boldsymbol{x}\right), ..., P\left(\omega_K|\boldsymbol{x}\right)$$

It is possible to show that

We have that, for $l = 1, 2, ..., K-1$

$$\frac{P(\omega_l|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

Therefore

$$\sum_{l=1}^{K-1}\frac{P(\omega_l|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

Thus

$$\frac{1-P(\omega_K|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

It is possible to show that

We have that, for $l = 1, 2, ..., K-1$

$$\frac{P(\omega_l|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

Therefore

$$\sum_{l=1}^{K-1}\frac{P(\omega_l|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

Thus

$$\frac{1-P(\omega_K|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

It is possible to show that

We have that, for $l = 1, 2, ..., K-1$

$$\frac{P(\omega_l|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

Therefore

$$\sum_{l=1}^{K-1}\frac{P(\omega_l|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

Thus

$$\frac{1-P(\omega_K|\boldsymbol{x})}{P(\omega_K|\boldsymbol{x})} = \sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

Basically

We have, (Take a look a the board)

$$P\left(\omega_K|\boldsymbol{x}\right) = \frac{1}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}$$

Then

$$\frac{P\left(\omega_i|\boldsymbol{x}\right)}{\frac{1}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}} = \exp\left\{\boldsymbol{w}_i^T\boldsymbol{x}\right\}$$

For $i = 1, 2, ..., k-1$

$$P\left(\omega_i|\boldsymbol{x}\right) = \frac{\exp\left\{\boldsymbol{w}_i^T\boldsymbol{x}\right\}}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}$$

Basically

We have, (Take a look a the board)

$$P\left(\omega_K|\boldsymbol{x}\right)=\frac{1}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}$$

Then

$$\frac{P\left(\omega_i|\boldsymbol{x}\right)}{\frac{1}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}}=\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

For $i=1,2,...,k-1$

$$P\left(\omega_i|\boldsymbol{x}\right)=\frac{\exp\left\{\boldsymbol{w}_i^T\boldsymbol{x}\right\}}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}$$

Basically

We have, (Take a look a the board)

$$P\left(\omega_K|\boldsymbol{x}\right)=\frac{1}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}$$

Then

$$\frac{P\left(\omega_i|\boldsymbol{x}\right)}{\frac{1}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}}=\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}$$

For $i=1,2,...,k-1$

$$P\left(\omega_i|\boldsymbol{x}\right)=\frac{\exp\left\{\boldsymbol{w}_i^T\boldsymbol{x}\right\}}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}$$

Additionally

For K

$$P\left(\omega_K|\boldsymbol{x}\right) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp\left\{\boldsymbol{w}_l^T \boldsymbol{x}\right\}}$$

Easy to see

They sum to one.

Additionally

For K

$$P\left(\omega_K|\boldsymbol{x}\right) = \frac{1}{1+\sum_{l=1}^{K-1}\exp\left\{\boldsymbol{w}_l^T\boldsymbol{x}\right\}}$$

Easy to see

They sum to one.

A Note in Notation

Given all these parameters, we summarized them

$$\Theta = \{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_{K-1}\}$$

Therefore

$$P\left(\omega_l | \boldsymbol{X} = \boldsymbol{x}\right) = p_l\left(\boldsymbol{x} | \Theta\right)$$

A Note in Notation

Given all these parameters, we summarized them

$$\Theta = \{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_{K-1}\}$$

Therefore

$$P\left(\omega_l | \boldsymbol{X} = \boldsymbol{x}\right) = p_l\left(\boldsymbol{x} | \Theta\right)$$

Outline

In the two class case

We have

$$P_1\left(\omega_1|\boldsymbol{x}\right) = \frac{\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

$$P_2\left(\omega_2|\boldsymbol{x}\right) = \frac{1}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

A similar model

$$P_1\left(\omega_1|\boldsymbol{x}\right) = \frac{\exp\left\{-\boldsymbol{w}^T\boldsymbol{x}\right\}}{1+\exp\left\{-\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

$$P_2\left(\omega_2|\boldsymbol{x}\right) = \frac{1}{1+\exp\left\{-\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

In the two class case

We have

$$P_1\left(\omega_1|\boldsymbol{x}\right) = \frac{\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

$$P_2\left(\omega_2|\boldsymbol{x}\right) = \frac{1}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

A similar model

$$P_1\left(\omega_1|\boldsymbol{x}\right) = \frac{\exp\left\{-\boldsymbol{w}^T\boldsymbol{x}\right\}}{1+\exp\left\{-\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

$$P_2\left(\omega_2|\boldsymbol{x}\right) = \frac{1}{1+\exp\left\{-\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

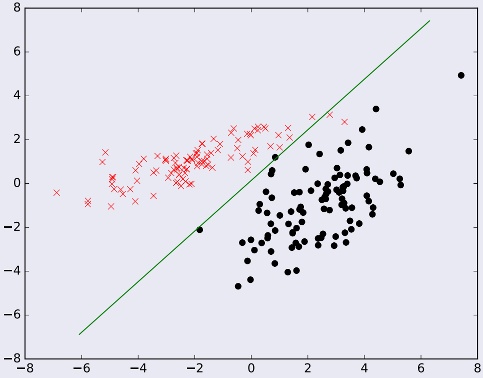
Outline

1. Logistic Regression
   - Introduction
   - Constraints
   - The Initial Model
   - The Two Case Class
   - Graphic Interpretation
   - Fitting The Model
     - The Two Class Case
     - The Final Log-Likelihood
     - The Newton-Raphson Algorithm
     - Matrix Notation

2. More on Optimization Methods
   - Can we do better?
   - Using Cholesky Decomposition
     - Cholesky Decomposition
     - The Proposed Method
   - Quasi-Newton Method
     - The Second Order Approximation
     - The BFGS Algorithm
   - A Neat Trick: Coordinate Ascent
     - Coordinate Ascent Algorithm
   - Conclusion

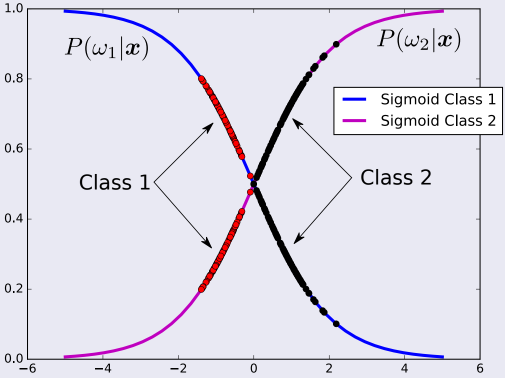
We have the following split

Using $f(x) = w^T x$



Then, we have the mapping to

We have



Outline

1. **Logistic Regression**
   - Introduction
   - Constraints
   - The Initial Model
   - The Two Case Class
   - Graphic Interpretation
   - **Fitting The Model**
     - The Two Class Case
     - The Final Log-Likelihood
     - The Newton-Raphson Algorithm
     - Matrix Notation

2. More on Optimization Methods
   - Can we do better?
   - Using Cholesky Decomposition
     - Cholesky Decomposition
     - The Proposed Method
   - Quasi-Newton Method
     - The Second Order Approximation
     - The BFGS Algorithm
   - A Neat Trick: Coordinate Ascent
     - Coordinate Ascent Algorithm
   - Conclusion

A Classic application of Maximum Likelihood

Given a sequence of smaples iid

$$\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$$

We have the following pdf

$$p\left(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N | \Theta\right) = \prod_{i=1}^{N} p\left(x_i | \Theta\right)$$

A Classic application of Maximum Likelihood

Given a sequence of smaples iid

$$\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$$

We have the following pdf

$$p\left(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N|\Theta\right) = \prod_{i=1}^{N} p\left(x_i|\Theta\right)$$

$P(g = v|X)$ Distribution

Where $P(g = v|X)$ completely specify the conditional distribution

We have a multinomial distribution which under the log-likelihood of N observations:

$$\mathcal{L}(\Theta) = \log p(x_1, x_2, ..., x_N|\Theta) = \log \prod_{i=1}^{N} p_{g_i}(x_i|\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i|\theta)$$

Where

g_i represent the class that x_i belongs.

$$g_i = \begin{cases} 1 & \text{if } x_i \in \text{Class } 1 \\ 2 & \text{if } x_i \in \text{Class } 2 \end{cases}$$

$P\left(g=v|X\right)$ Distribution

Where $P\left(g=v|X\right)$ completely specify the conditional distribution

We have a multinomial distribution which under the log-likelihood of N observations:

$$\mathcal{L}\left(\Theta\right)=\log p\left(\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N|\Theta\right)=\log\prod_{i=1}^{N}p_{g_i}\left(\boldsymbol{x}_i|\theta\right)=\sum_{i=1}^{N}\log p_{g_i}\left(\boldsymbol{x}_i|\theta\right)$$

Where

g_i represent the class that x_i belongs.

$$g_i=\begin{cases}1 & \text{if } x_i\in\text{Class 1}\\ 2 & \text{if } x_i\in\text{Class 2}\end{cases}$$

$P\left(g=v|X\right)$ Distribution

Where $P\left(g=v|X\right)$ completely specify the conditional distribution

We have a multinomial distribution which under the log-likelihood of N observations:

$$\mathcal{L}\left(\Theta\right) = \log p\left(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N|\Theta\right) = \log \prod_{i=1}^{N} p_{g_i}\left(\boldsymbol{x}_i|\theta\right) = \sum_{i=1}^{N} \log p_{g_i}\left(\boldsymbol{x}_i|\theta\right)$$

Where

g_i represent the class that x_i belongs.

$$g_i = \begin{cases} 1 & \text{if } x_i \in \text{Class 1} \\ 2 & \text{if } x_i \in \text{Class 2} \end{cases}$$

$P(g = v|X)$ Distribution

Where $P(g = v|X)$ completely specify the conditional distribution

We have a multinomial distribution which under the log-likelihood of N observations:

$$\mathcal{L}(\Theta) = \log p(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N|\Theta) = \log \prod_{i=1}^{N} p_{g_i}(\boldsymbol{x}_i|\theta) = \sum_{i=1}^{N} \log p_{g_i}(\boldsymbol{x}_i|\theta)$$

Where

g_i represent the class that x_i belongs.

$$g_i = \begin{cases} 1 & \text{if } x_i \in \text{Class 1} \\ 2 & \text{if } x_i \in \text{Class 2} \end{cases}$$

$P\left(g=v|X\right)$ Distribution

Where $P\left(g=v|X\right)$ completely specify the conditional distribution

We have a multinomial distribution which under the log-likelihood of N observations:

$$\mathcal{L}\left(\Theta\right)= \log p\left(\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N|\Theta\right)= \log\prod_{i=1}^{N}p_{g_i}\left(\boldsymbol{x}_i|\theta\right)=\sum_{i=1}^{N}\log p_{g_i}\left(\boldsymbol{x}_i|\theta\right)$$

Where

g_i represent the class that $\boldsymbol{x}_i$ belongs.

$$g_i=\begin{cases}1 & \text{if } \boldsymbol{x}_i\in \text{Class } 1\\ 2 & \text{if } \boldsymbol{x}_i\in \text{Class } 2\end{cases}$$

How do we integrate this into a Cost Function?

Clearly, we have two distributions

We need to represent the distributions into the functions $p_{g_i}(\boldsymbol{x}_i|\theta)$.

Why not to have all the distributions into this function

$$p_{g_i}(\boldsymbol{x}_i|\theta) = \prod_{l=1}^{K-1} p(\boldsymbol{x}_i|\boldsymbol{w}_l)^{I\{x_i \in \omega_l\}}$$

It is easy with the two classes

Given that we have a binary situation!!!

How do we integrate this into a Cost Function?

Clearly, we have two distributions

We need to represent the distributions into the functions $p_{g_i}\left(\boldsymbol{x}_i|\theta\right)$.

Why not to have all the distributions into this function

$$p_{g_i}\left(\boldsymbol{x}_i|\theta\right)=\prod_{l=1}^{K-1}p\left(\boldsymbol{x}_i|\boldsymbol{w}_l\right)^{I\{\boldsymbol{x}_i\in\omega_l\}}$$

It is easy with the two classes

Given that we have a binary situation!!!

How do we integrate this into a Cost Function?

Clearly, we have two distributions

We need to represent the distributions into the functions $p_{g_i}\left(\boldsymbol{x}_i|\theta\right)$.

Why not to have all the distributions into this function

$$p_{g_i}\left(\boldsymbol{x}_i|\theta\right) = \prod_{l=1}^{K-1} p\left(\boldsymbol{x}_i|\boldsymbol{w}_l\right)^{I\{\boldsymbol{x}_i \in \omega_l\}}$$

It is easy with the two classes

Given that we have a binary situation!!!

Outline

Given the two case

We have then a Bernoulli distribution

$$p_1\left(\boldsymbol{x}_i|\boldsymbol{w}\right) = \left[\frac{\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}\right]^{y_i}$$

$$p_2\left(\boldsymbol{x}_i|\boldsymbol{w}\right) = \left[\frac{1}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}\right]^{1-y_i}$$

With

$$y_i = 1 \text{ if } x_i \in \text{Class } 1$$

$$y_i = 0 \text{ if } x_i \in \text{Class } 2$$

Given the two case

We have then a Bernoulli distribution

$$p_1(\boldsymbol{x}_i|\boldsymbol{w}) = \left[\frac{\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}\right]^{y_i}$$

$$p_2(\boldsymbol{x}_i|\boldsymbol{w}) = \left[\frac{1}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}\right]^{1-y_i}$$

With

$$y_i = 1 \text{ if } \boldsymbol{x}_i \in \text{Class } 1$$

$$y_i = 0 \text{ if } \boldsymbol{x}_i \in \text{Class } 2$$

Outline

We have the following

Cost Function

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{N} \{y_i \log p_1(\boldsymbol{x}_i|\boldsymbol{w}) + (1-y_i)\log(1-p_1(\boldsymbol{x}_i|\boldsymbol{w}))\}$$

After some reductions

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{N} \left\{ y_i \boldsymbol{w}^T \boldsymbol{x}_i - \log\left(1 + \exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}\right)\right\}$$

We have the following

Cost Function

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{N} \{y_i \log p_1(\boldsymbol{x}_i|\boldsymbol{w}) + (1 - y_i) \log (1 - p_1(\boldsymbol{x}_i|\boldsymbol{w}))\}$$

After some reductions

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{N} \left\{ y_i \boldsymbol{w}^T \boldsymbol{x}_i - \log \left(1 + \exp \left\{ \boldsymbol{w}^T \boldsymbol{x}_i \right\} \right) \right\}$$

Now, we derive and set it to zero

We have

$$\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_{i=1}^{N} \boldsymbol{x}_i \left(y_i - \frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1 + \exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right) = 0$$

Which are $d+1$ equations nonlinear

$$\sum_{i=1}^{N} \boldsymbol{x}_i \left(y_i - p\left(\boldsymbol{x}_i|\boldsymbol{w}\right)\right) = \begin{pmatrix} \sum_{i=1}^{N} 1 \times \left(y_i - \frac{\exp\{\boldsymbol{w}^T\boldsymbol{x}_i\}}{1+\exp\{\boldsymbol{w}^T\boldsymbol{x}_i\}} \right) \\ \sum_{i=1}^{N} x_1^i \left(y_i - \frac{\exp\{\boldsymbol{w}^T\boldsymbol{x}_i\}}{1+\exp\{\boldsymbol{w}^T\boldsymbol{x}_i\}} \right) \\ \sum_{i=1}^{N} x_2^i \left(y_i - \frac{\exp\{\boldsymbol{w}^T\boldsymbol{x}_i\}}{1+\exp\{\boldsymbol{w}^T\boldsymbol{x}_i\}} \right) \\ \vdots \\ \sum_{i=1}^{N} x_d^i \left(y_i - \frac{\exp\{\boldsymbol{w}^T\boldsymbol{x}_i\}}{1+\exp\{\boldsymbol{w}^T\boldsymbol{x}_i\}} \right) \end{pmatrix} = 0$$

It is know as a scoring function.

Now, we derive and set it to zero

We have

$$\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_{i=1}^{N} \boldsymbol{x}_i \left(y_i - \frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1 + \exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right) = 0$$

Which are $d+1$ equations nonlinear

$$\sum_{i=1}^{N} \boldsymbol{x}_i \left(y_i - p\left(\boldsymbol{x}_i | \boldsymbol{w}\right)\right) = \begin{pmatrix} \sum_{i=1}^{N} 1 \times \left(y_i - \frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1+\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right) \\ \sum_{i=1}^{N} x_1^i \left(y_i - \frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1+\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right) \\ \sum_{i=1}^{N} x_2^i \left(y_i - \frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1+\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right) \\ \vdots \\ \sum_{i=1}^{N} x_d^i \left(y_i - \frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1+\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right) \end{pmatrix} = \boldsymbol{0}$$

It is know as a scoring function.

Finally

In other words you

1. $d + 1$ nonlinear equations in $\boldsymbol{w}$.
2. For example, from the first equation:

$$\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p\left(\boldsymbol{x}_i \middle| \boldsymbol{w}\right)$$

   - The expected number of class ones matches the observed number.
   - And hence also class twos.

Finally

In other words you

1. $d+1$ nonlinear equations in $\boldsymbol{w}$.
2. For example, from the first equation:

$$\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p\left(\boldsymbol{x}_i | \boldsymbol{w}\right)$$

- The expected number of class ones matches the observed number.
- And hence also class twos.

Finally

In other words you

1. $d + 1$ nonlinear equations in $\boldsymbol{w}$.
2. For example, from the first equation:

$$\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p\left(\boldsymbol{x}_i | \boldsymbol{w}\right)$$

- The expected number of class ones matches the observed number.
- And hence also class twos.

Finally

In other words you

1. $d + 1$ nonlinear equations in $\boldsymbol{w}$.
2. For example, from the first equation:

$$\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p\left(\boldsymbol{x}_i | \boldsymbol{w}\right)$$

   - The expected number of class ones matches the observed number.
   - And hence also class twos.

Outline

To solve the previous equations [3]

We use the Newton-Raphson Method to find the roots or zeros

It comes from the first Taylor Approximation

$$f(x+h) \approx f(x) + hf'(x)$$

Thus we have for a root r of function f

We have

1. Assume a good estimate of r, x_0
2. Thus we have $r = x_0 + h$
3. Or $h = r - x_0$

To solve the previous equations [3]

We use the Newton-Raphson Method to find the roots or zeros

It comes from the first Taylor Approximation

$$f(x+h) \approx f(x) + hf'(x)$$

Thus we have for a root r of function f

We have

1. Assume a good estimate of r, x_0
2. Thus we have $r = x_0 + h$
3. Or $h = r - x_0$

To solve the previous equations [3]

We use the Newton-Raphson Method to find the roots or zeros

It comes from the first Taylor Approximation

$$f\left(x+h\right) \approx f\left(x\right) + hf'\left(x\right)$$

Thus we have for a root r of function f

We have

1. Assume a good estimate of r, x_0
2. Thus we have $r = x_0 + h$
3. Or $h = r - x_0$

To solve the previous equations [3]

We use the Newton-Raphson Method to find the roots or zeros

It comes from the first Taylor Approximation

$$f(x+h) \approx f(x) + hf'(x)$$

Thus we have for a root r of function f

We have

1. Assume a good estimate of r, x_0
2. Thus we have $r = x_0 + h$
3. Or $h = r - x_0$

To solve the previous equations [3]

We use the Newton-Raphson Method to find the roots or zeros

It comes from the first Taylor Approximation

$$f(x+h) \approx f(x) + hf'(x)$$

Thus we have for a root r of function f

We have

1. Assume a good estimate of r, x_0
2. Thus we have $r = x_0 + h$
3. Or $h = r - x_0$

We have then

From Taylor

$$0 = f(r) = f(x_0 + h) \approx f(x_0) + hf'(x_0)$$

Thus, as long $f'(x_0)$ is not close to 0

$$h \approx -\frac{f(x_0)}{f'(x_0)}$$

Thus

$$r = x_0 + h \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

We have then

From Taylor

$$0 = f(r) = f(x_0 + h) \approx f(x_0) + hf'(x_0)$$

Thus, as long $f'(x_0)$ is not close to 0

$$h \approx -\frac{f(x_0)}{f'(x_0)}$$

Thus

$$r = x_0 + h \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

We have then

From Taylor

$$0 = f(r) = f(x_0 + h) \approx f(x_0) + hf'(x_0)$$

Thus, as long $f'(x_0)$ is not close to 0

$$h \approx -\frac{f(x_0)}{f'(x_0)}$$

Thus

$$r = x_0 + h \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

We have our final improving

We have

$$x_1 \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

Then, on the scoring function

For this, we need the Hessian of the function

$$\frac{\partial^2 \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T} = -\sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^T \left[\frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1 + \exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right] \left[1 - \frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1 + \exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right]$$

Thus, we have at a starting point w^{old}

$$w^{new} = w^{old} - \left(\frac{\partial \mathcal{L}(w)}{\partial w \partial w^T} \right)^{-1} \frac{\partial \mathcal{L}(w)}{\partial w}$$

Then, on the scoring function

For this, we need the Hessian of the function

$$\frac{\partial^2 \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T} = -\sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^T \left[\frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1 + \exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right] \left[1 - \frac{\exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}}{1 + \exp\left\{\boldsymbol{w}^T \boldsymbol{x}_i\right\}} \right]$$

Thus, we have at a starting point $\boldsymbol{w}^{old}$

$$\boldsymbol{w}^{new} = \boldsymbol{w}^{old} - \left(\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T} \right)^{-1} \frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w}}$$

Outline

We can rewrite all as matrix notations

Assume

1. Let $\boldsymbol{y}$ denotes the vector of y_i
2. X is the data matrix $N \times (d+1)$
3. $\boldsymbol{p}$ the vector of fitted probabilities with the i^{th} element $p\left(x_i|w^{old}\right)$
4. W a $N \times N$ diagonal matrix of weights with the i^{th} diagonal element

$$p\left(x_i|w^{old}\right)\left[1-p\left(x_i|w^{old}\right)\right]$$

We can rewrite all as matrix notations

Assume

1. Let $\boldsymbol{y}$ denotes the vector of y_i
2. X is the data matrix $N \times (d+1)$
3. $\boldsymbol{p}$ the vector of fitted probabilities with the i^{th} element $p\left(x_i|w^{old}\right)$
4. W a $N \times N$ diagonal matrix of weights with the i^{th} diagonal element

$$p\left(x_i|w^{old}\right)\left[1 - p\left(x_i|w^{old}\right)\right]$$

We can rewrite all as matrix notations

Assume

1. Let $\boldsymbol{y}$ denotes the vector of y_i
2. X is the data matrix $N \times (d+1)$
3. $\boldsymbol{p}$ the vector of fitted probabilities with the i^{th} element $p\left(\boldsymbol{x}_i|\boldsymbol{w}^{old}\right)$
4. W a $N \times N$ diagonal matrix of weights with the i^{th} diagonal element

$$p\left(\boldsymbol{x}_i|\boldsymbol{w}^{old}\right)\left[1 - p\left(\boldsymbol{x}_i|\boldsymbol{w}^{old}\right)\right]$$

We can rewrite all as matrix notations

Assume

1. Let $\boldsymbol{y}$ denotes the vector of y_i
2. X is the data matrix $N \times (d+1)$
3. $\boldsymbol{p}$ the vector of fitted probabilities with the i^{th} element $p\left(\boldsymbol{x}_i|\boldsymbol{w}^{old}\right)$
4. W a $N \times N$ diagonal matrix of weights with the i^{th} diagonal element

$$p\left(\boldsymbol{x}_i|\boldsymbol{w}^{old}\right)\left[1 - p\left(\boldsymbol{x}_i|\boldsymbol{w}^{old}\right)\right]$$

Then, we have

For each updating term

$$
\begin{aligned}
\frac{\partial \mathcal{L}\left(\boldsymbol{w}\right)}{\partial \boldsymbol{w}} &= X^{T}\left(\boldsymbol{y}-\boldsymbol{p}\right)\\
\frac{\partial \mathcal{L}\left(\boldsymbol{w}\right)}{\partial \boldsymbol{w}\partial \boldsymbol{w}^{T}} &= -X^{T}WX
\end{aligned}
$$

Then, we have

Then, the Newton Step

$$
\begin{aligned}
\boldsymbol{w}^{new} &= \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= I\boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T I \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= \left(X^T W X\right)^{-1} X^T W X \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T W W^{-1} \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= \left(X^T W X\right)^{-1} X^T W \left[X\boldsymbol{w}^{old} + W^{-1} \left(\boldsymbol{y} - \boldsymbol{p}\right)\right] \\
&= \left(X^T W X\right)^{-1} X^T W \boldsymbol{z}
\end{aligned}
$$

Then, we have

Then, the Newton Step

$$
\begin{aligned}
\boldsymbol{w}^{new} &= \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= I\boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T I \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= \left(X^T W X\right)^{-1} X^T W X \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T W W^{-1} \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= \left(X^T W X\right)^{-1} X^T W \left[X \boldsymbol{w}^{old} + W^{-1} \left(\boldsymbol{y} - \boldsymbol{p}\right)\right] \\
&= \left(X^T W X\right)^{-1} X^T W \boldsymbol{z}
\end{aligned}
$$

Then, we have

Then, the Newton Step

$$
\begin{aligned}
\boldsymbol{w}^{new} &= \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= I\boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T I \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= \left(X^T W X\right)^{-1} X^T W X \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T W W^{-1} \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= \left(X^T W X\right)^{-1} X^T W \left[X \boldsymbol{w}^{old} + W^{-1} \left(\boldsymbol{y} - \boldsymbol{p}\right)\right] \\
&= \left(X^T W X\right)^{-1} X^T W \boldsymbol{z}
\end{aligned}
$$

Then, we have

Then, the Newton Step

$$
\begin{aligned}
\boldsymbol{w}^{new} &= \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T (\boldsymbol{y} - \boldsymbol{p}) \\
&= I\boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T I (\boldsymbol{y} - \boldsymbol{p}) \\
&= \left(X^T W X\right)^{-1} X^T W X \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T W W^{-1} (\boldsymbol{y} - \boldsymbol{p}) \\
&= \left(X^T W X\right)^{-1} X^T W \left[X \boldsymbol{w}^{old} + W^{-1} (\boldsymbol{y} - \boldsymbol{p})\right] \\
&= \left(X^T W X\right)^{-1} X^T W \boldsymbol{z}
\end{aligned}
$$

Then, we have

Then, the Newton Step

$$
\begin{aligned}
\boldsymbol{w}^{new} &= \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= I \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T I \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= \left(X^T W X\right)^{-1} X^T W X \boldsymbol{w}^{old} + \left(X^T W X\right)^{-1} X^T W W^{-1} \left(\boldsymbol{y} - \boldsymbol{p}\right) \\
&= \left(X^T W X\right)^{-1} X^T W \left[X \boldsymbol{w}^{old} + W^{-1} \left(\boldsymbol{y} - \boldsymbol{p}\right)\right] \\
&= \left(X^T W X\right)^{-1} X^T W \boldsymbol{z}
\end{aligned}
$$

Then

We have

Re-expressed the Newton step as a weighted least squares step.

With a the adjusted response as

$$z = Xw^{old} + W^{-1}(y - p)$$

Then

We have

Re-expressed the Newton step as a weighted least squares step.

With a the adjusted response as

$$\boldsymbol{z} = X\boldsymbol{w}^{old} + W^{-1}\left(\boldsymbol{y} - \boldsymbol{p}\right)$$

This New Algorithm

It is know as

Iteratively Re-weighted Least Squares or IRLS

After all at each iteration, it solves

A weighted Least Square Problem

$$w^{new} \leftarrow \arg\min_{w} (z - Xw)^T W (z - Xw)$$

This New Algorithm

It is know as

Iteratively Re-weighted Least Squares or IRLS

After all at each iteration, it solves

A weighted Least Square Problem

$$\boldsymbol{w}^{new} \leftarrow \arg\min_{\boldsymbol{w}} (\boldsymbol{z} - X\boldsymbol{w})^T W (\boldsymbol{z} - X\boldsymbol{w})$$

Observations

Good Starting Point $w = 0$

However, convergence is never guaranteed!!!

However

- Typically the algorithm does converge, since the log-likelihood is concave.
- But overshooting can occur.

Observations

Good Starting Point $w = 0$

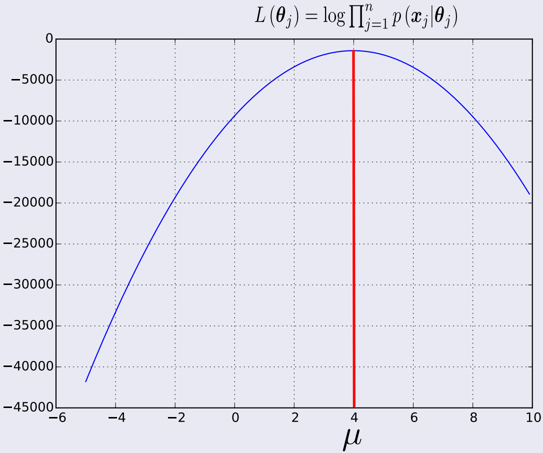
However, convergence is never guaranteed!!!

However

- Typically the algorithm does converge, since the log-likelihood is concave.
- But overshooting can occur.

Observations

$L\left(\boldsymbol{\theta}_j\right) = \log \prod_{j=1}^{n} p\left(\boldsymbol{x}_j | \boldsymbol{\theta}_j\right)$

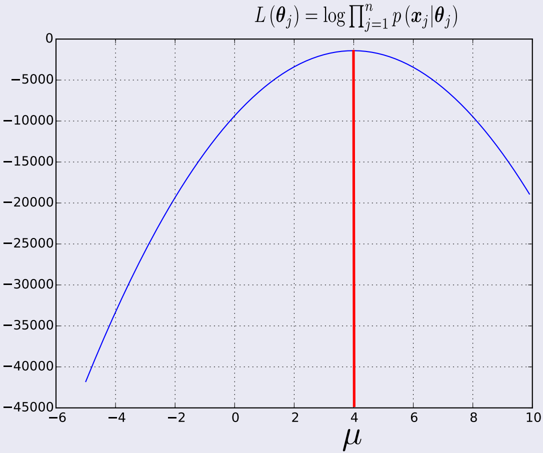


Halving Solve the Problem

Perfect!!!

Observations

$L\left(\boldsymbol{\theta}_j\right) = \log \prod_{j=1}^{n} p\left(\boldsymbol{x}_j|\boldsymbol{\theta}_j\right)$



Halving Solve the Problem

Perfect!!!

Outline

The final question

After all, we always want to have a better solution

- We know that $\left(\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T}\right)^{-1}$ takes $O\left(d^3\right)$.... and we want something better!!!

We have the following methods

- Colesky Decomposition
- Quasi-Newton Method
- Coordinate Ascent

The final question

After all, we always want to have a better solution

- We know that $\left(\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T}\right)^{-1}$ takes $O\left(d^3\right)$.... and we want something better!!!

We have the following methods

- Colesky Decomposition
- Quasi-Newton Method
- Coordinate Ascent

The final question

After all, we always want to have a better solution

- We know that $\left(\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T}\right)^{-1}$ takes $O\left(d^3\right)$.... and we want something better!!!

We have the following methods

- Colesky Decomposition
- Quasi-Newton Method
- Coordinate Ascent

The final question

After all, we always want to have a better solution

- We know that $\left(\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T}\right)^{-1}$ takes $O\left(d^3\right)$.... and we want something better!!!

We have the following methods

- Colesky Decomposition
- Quasi-Newton Method
- Coordinate Ascent

Outline

We can decompose the matrix

Given $A = X^T W X$ and $Y = X^T W \boldsymbol{z}$, you have

$$Ax = Y$$

We want to obtain

$$x = A^{-1}Y$$

We can decompose the matrix

Given $A = X^T W X$ and $Y = X^T W \boldsymbol{z}$, you have

$$Ax = Y$$

We want to obtain

$$x = A^{-1} Y$$

This can be seen as a system of linear equations

As you can see

- We start with a set of linear equations with $d+1$ unknowns:

$$x_1, x_2, ..., x_{d+1} \begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1d+1}x_{d+1} & = y_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2d+1}x_{d+1} & = y_2 \\ \vdots & \vdots \\ a_{d+11}x_1 + a_{d+12}x_2 + ... + a_{d+1d+1}x_n & = y_{d+1} \end{cases}$$

Thus

- A set of values for $x_1, x_2, ..., x_n$ that satisfy all of the equations simultaneously is said to be a solution to these equations.

This can be seen as a system of linear equations

As you can see

- We start with a set of linear equations with $d+1$ unknowns:

$$x_1, x_2, ..., x_{d+1} \begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1d+1}x_{d+1} & = y_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2d+1}x_{d+1} & = y_2 \\ \vdots & \vdots \\ a_{d+11}x_1 + a_{d+12}x_2 + ... + a_{d+1d+1}x_n & = y_{d+1} \end{cases}$$

Thus

- A set of values for $x_1, x_2, ..., x_n$ that satisfy all of the equations simultaneously is said to be a solution to these equations.

This can be seen as a system of linear equations

As you can see

- We start with a set of linear equations with $d+1$ unknowns:

$$x_1, x_2, ..., x_{d+1} \begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1d+1}x_{d+1} & = y_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2d+1}x_{d+1} & = y_2 \\ \vdots & \vdots \\ a_{d+11}x_1 + a_{d+12}x_2 + ... + a_{d+1d+1}x_n & = y_{d+1} \end{cases}$$

Thus

- A set of values for $x_1, x_2, ..., x_n$ that satisfy all of the equations simultaneously is said to be a solution to these equations.

Outline

What is the Cholesky Decomposition? [4]

It is a method that factorize a matrix

- $A \in \mathbb{R}^{d+1 \times d+1}$ is a positive definite Hermitian matrix

Positive definite matrix

$$x^T A x > 0 \text{ for all } x \in \mathbb{R}^{d+1 \times d+1}$$

Hermitian matrix in the Real Domain (Symmetric Matrix)

$$A = A^T$$

What is the Cholesky Decomposition? [4]

It is a method that factorize a matrix

- $A \in \mathbb{R}^{d+1 \times d+1}$ is a positive definite Hermitian matrix

Positive definite matrix

$$\boldsymbol{x}^T A \boldsymbol{x} > 0 \text{ for all } \boldsymbol{x} \in \mathbb{R}^{d+1 \times d+1}$$

Hermitian matrix in the Real Domain (Symmetric Matrix)

$$A = A^T$$

What is the Cholesky Decomposition? [4]

It is a method that factorize a matrix

- $A \in \mathbb{R}^{d+1 \times d+1}$ is a positive definite Hermitian matrix

Positive definite matrix

$$\boldsymbol{x}^T A \boldsymbol{x} > 0 \text{ for all } \boldsymbol{x} \in \mathbb{R}^{d+1 \times d+1}$$

Hermitian matrix in the Real Domain (Symmetric Matrix)

$$A = A^T$$

Therefore

Cholesky decomposes A into lower or upper triangular matrix and their conjugate transpose

$$A = LL^T$$
$$A = R^T R$$

Thus, we can use the Cholensky decomposition

- The Cholensky decomposition is of order $O\left(d^3\right)$ and requires $\frac{1}{6}d^3$ FLOP operations.

Therefore

Cholesky decomposes A into lower or upper triangular matrix and their conjugate transpose

$$A = LL^T$$
$$A = R^T R$$

Thus, we can use the Cholensky decomposition

- The Cholensky decomposition is of order $O\left(d^3\right)$ and requires $\frac{1}{6}d^3$ FLOP operations.

Outline

We have

The matrices $A \in \mathbb{R}^{d+1 \times d+1}$ and $X = A^{-1}$

$$AX = I$$

From Cholensky, the decomposition of A

$$R^T R X = I$$

If we define $RX = B$

$$R^T B = I$$

We have

The matrices $A \in \mathbb{R}^{d+1 \times d+1}$ and $X = A^{-1}$

$$AX = I$$

From Cholensky, the decomposition of A

$$R^T RX = I$$

If we define $RX = B$

$$R^T B = I$$

We have

The matrices $A \in \mathbb{R}^{d+1 \times d+1}$ and $X = A^{-1}$

$$AX = I$$

From Cholesky, the decomposition of A

$$R^T RX = I$$

If we define $RX = B$

$$R^T B = I$$

Now

If $B = \left(R^T\right)^{-1} = L^{-1}$ for $L = R^T$

1. We note that the inverse of the lower triangular matrix L is lower triangular.
2. The diagonal entries of L^{-1} are the reciprocal of diagonal entries of L

$$\begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ a_{2,1} & a_{2,2} & 0 & \vdots \\ \vdots & \vdots & \ddots & 0 \\ a_{d+1,1} & a_{d+1,2} & \cdots & a_{d+1,d+1} \end{pmatrix} \begin{pmatrix} b_{1,1} & 0 & \cdots & 0 \\ b_{2,1} & b_{2,2} & 0 & \vdots \\ \vdots & \vdots & \ddots & 0 \\ b_{d+1,1} & b_{d+1,2} & \cdots & b_{d+1,d+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Now

Now, we construct the following matrix S with entries

$$s_{i,j} = \begin{cases} \frac{1}{l_{i,i}} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Now, we have

The matrix S is the correct solution to upper diagonal element of the matrix B

i.e. $s_{ij} = b_{ij}$ for $i \leq j \leq d+1$

Then, we use backward substitution to solve $x_{i,j}$ at equation $R\boldsymbol{x}_j = \boldsymbol{s}_j$

Assuming:

$$X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{d+1}]$$
$$S = [\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_{d+1}]$$

Now, we have

The matrix S is the correct solution to upper diagonal element of the matrix B

i.e. $s_{ij} = b_{ij}$ for $i \leq j \leq d+1$

Then, we use backward substitution to solve $x_{i,j}$ at equation $R\boldsymbol{x}_i = \boldsymbol{s}_i$

Assuming:

$$X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{d+1}]$$
$$S = [\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_{d+1}]$$

Back Substitution

Back substitution

Since R is upper-triangular, we can rewrite the system $R\boldsymbol{x}_i = \boldsymbol{s}_i$ as

$$\begin{aligned}
r_{1,1}x_{1,i} + r_{1,2}x_{2,i} + \ldots + r_{1,d-1}x_{d-1,i} + r_{1,d}x_{d,i} + r_{1,d+1}x_{d+1,i} &= s_{1,i} \\
r_{2,2}x_{2} + \ldots + r_{2,d-1}x_{d-1,i} + r_{2,d}x_{d,i} + r_{2,d+1}x_{d+1,i} &= s_{2,i} \\
&\vdots \\
r_{d-1,d-1}x_{d-1,i} + r_{d-1,d}x_{d,i} + r_{d-1,d+1}x_{d+1,i} &= s_{d-1,i} \\
r_{d,d}x_{d,i} + r_{d,d+1}x_{d,i} &= s_{d,i} \\
r_{d+1,d+1}x_{d+1,i} &= s_{d+1,i}
\end{aligned}$$

Then

We solve only for x_{ij} such that

- We have $i < j \leq N$ (Upper triangle elements).

$x_{ji} = \overline{x_{ij}}$

- In our case the same value given that we live on the reals.

Then

We solve only for x_{ij} such that

- We have $i < j \leq N$ (Upper triangle elements).

$x_{ji} = \overline{x_{ij}}$

- In our case the same value given that we live on the reals.

Complexity

Equation solving requires

- $\frac{1}{3}(d+1)^3$ multiply operations.

The total number of multiply operations for matrix inverse

- Including Cholesky decomposition is $\frac{1}{2}(d+1)^3$

Therefore

- **We have complexity $O(d^3)$!!! Per iteration!!! But actually $\frac{1}{2}(d+1)^3$ multiply operations**

Complexity

Equation solving requires

- $\frac{1}{3}(d+1)^3$ multiply operations.

The total number of multiply operations for matrix inverse

- Including Cholesky decomposition is $\frac{1}{2}(d+1)^3$

Therefore

- We have complexity $O(d^3)$!!! Per iteration!!! But actually $\frac{1}{2}(d+1)^3$ multiply operations

Complexity

Equation solving requires

- $\frac{1}{3}(d+1)^3$ multiply operations.

The total number of multiply operations for matrix inverse

- Including Cholesky decomposition is $\frac{1}{2}\left(d+1\right)^3$

Therefore

- **We have complexity $O\left(d^3\right)$!!! Per iteration!!! But actually $\frac{1}{2}\left(d+1\right)^3$ multiply operations**

Outline

However

As Good Computer Scientists

We want to obtain a better complexity as $O\left(d^2\right)$!!!

We can obtain such improvements

Using Quasi Newton Methods

Let's us to develop the solution

- For the most popular one
  - Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

However

As Good Computer Scientists

We want to obtain a better complexity as $O\left(d^2\right)$!!!

We can obtain such improvements

Using Quasi Newton Methods

Let's us to develop the solution

- For the most popular one
  - Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

However

As Good Computer Scientists

We want to obtain a better complexity as $O\left(d^2\right)$!!!

We can obtain such improvements

Using Quasi Newton Methods

Let's us to develop the solution

- For the most popular one
  - Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

Outline

The Second Order Approximation

We have

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T \boldsymbol{H} f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k)$$

We develop a new equation based in the previous idea by using $\boldsymbol{x} = \boldsymbol{x}_k + \boldsymbol{p}$

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)\boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T H_k \boldsymbol{p}$$

Here

- H_k is an $d+1 \times d+1$ symmetric positive definite matrix that will be updated through the entire process

The Second Order Approximation

We have

$$f\left(\boldsymbol{x}\right) \approx f\left(\boldsymbol{x}_k\right) + \nabla f\left(\boldsymbol{x}_k\right)\left(\boldsymbol{x} - \boldsymbol{x}_k\right) + \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x}_k\right)^T \boldsymbol{H} f\left(\boldsymbol{x}_k\right)\left(\boldsymbol{x} - \boldsymbol{x}_k\right)$$

We develop a new equation based in the previous idea by using $\boldsymbol{x} = \boldsymbol{x}_k + \boldsymbol{p}$

$$f\left(\boldsymbol{x}\right) \approx f\left(\boldsymbol{x}_k\right) + \nabla f\left(\boldsymbol{x}_k\right)\boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T H_k \boldsymbol{p}$$

Here

- H_k is an $d+1 \times d+1$ symmetric positive definite matrix that will be updated through the entire process

The Second Order Approximation

We have

$$f\left(\boldsymbol{x}\right) \approx f\left(\boldsymbol{x}_k\right) + \nabla f\left(\boldsymbol{x}_k\right)\left(\boldsymbol{x} - \boldsymbol{x}_k\right) + \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x}_k\right)^T \boldsymbol{H} f\left(\boldsymbol{x}_k\right)\left(\boldsymbol{x} - \boldsymbol{x}_k\right)$$

We develop a new equation based in the previous idea by using $\boldsymbol{x} = \boldsymbol{x}_k + \boldsymbol{p}$

$$f\left(\boldsymbol{x}\right) \approx f\left(\boldsymbol{x}_k\right) + \nabla f\left(\boldsymbol{x}_k\right)\boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T H_k \boldsymbol{p}$$

Here

- H_k is an $d + 1 \times d + 1$ symmetric positive definite matrix that will be updated through the entire process

For the BFGS

Then, the inverse update of it $H_k = B_k^{-1}$

In BFGS we go directly for the inverse by setting up:

$$\begin{aligned} \min_{H} \|H - H_k\| \\ s.t.\ H = H^T \\ H y_k = s_k \end{aligned}$$

with

$$\begin{aligned} s_k &= \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \\ y_k &= \nabla f\left(\boldsymbol{x}_{k+1}\right) - \nabla f\left(\boldsymbol{x}_k\right) \end{aligned}$$

Then, we have

A unique solution will be

$$H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T \quad (1)$$

where $\rho_k = \frac{1}{y_k^T s_k}$

Then, we have

A unique solution will be

$$H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T \quad (1)$$

where $\rho_k = \frac{1}{y_k^T s_k}$

Complexity of Generating H_{k+1}

We notice that the complexity of calculating

$$s_k s_k^T, s_k s_k^T, s_k y_k^T$$

- It is $O\left(d^2\right)$

Why? For Example

$$\begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_d \end{pmatrix} \begin{pmatrix} s_1 & s_2 & \cdots & s_d \end{pmatrix} = \begin{pmatrix} s_1^2 & s_1 s_2 & \cdots & s_1 s_d \\ s_2 s_1 & s_2^2 & \cdots & s_2 s_d \\ \vdots & \vdots & \ddots & \vdots \\ s_d s_1 & s_d s_2 & \cdots & s_d^2 \end{pmatrix}$$

-Equal to d^2 m

Complexity of Generating H_{k+1}

We notice that the complexity of calculating

$$s_k s_k^T, s_k s_k^T, s_k y_k^T$$

- It is $O\left(d^2\right)$

Why? For Example

$$\begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_d \end{pmatrix} \begin{pmatrix} s_1 & s_2 & \cdots & s_d \end{pmatrix} = \begin{pmatrix} s_1^2 & s_1 s_2 & \cdots & s_1 s_d \\ s_2 s_1 & s_2^2 & \cdots & s_2 s_d \\ \vdots & \vdots & \ddots & \vdots \\ s_d s_1 & s_d s_2 & \cdots & s_d^2 \end{pmatrix}$$ -Equal to d^2 m

Then, we have that

Computation $\rho_k = \frac{1}{y_k^T s_k}$ as a constant

Complexity $O\left(d^2\right)$

Now for $\left(I - \rho_k y_k s_k^T\right)$

Complexity $O\left(d^2\right)$

Then, we have that

Computation $\rho_k = \frac{1}{y_k^T s_k}$ as a constant

Complexity $O\left(d^2\right)$

Now for $\left(I - \rho_k y_k s_k^T\right)$

Complexity $O\left(d^2\right)$

Then, we have that

Computation $\rho_k = \frac{1}{y_k^T s_k}$ as a constant

Complexity $O\left(d^2\right)$

Now for $\left(I - \rho_k y_k s_k^T\right)$

Complexity $O\left(d^2\right)$

Finally, we have

If we expand the equation $\left(I-\rho_k s_k y_k^T\right) H_k \left(I-\rho_k y_k s_k^T\right)$

$$\left(I-\rho_k s_k y_k^T\right) H_k \left(I-\rho_k y_k s_k^T\right) = \left(H_k - \rho_k s_k \left[y_k^T H_k\right]\right) - \left(\rho_k \left[H_k y_k\right] s_k^T - \rho\right.$$

Where

1. $\left[y_k^T H_k\right]$ and $\left[H_k y_k\right]$ has complexity $O\left(d^2\right)$
2. Thus $s_k \left[y_k^T H_k\right]$ and $\left[y_k^T H_k y_k\right]$ has complexity $O\left(d^2\right)$

Thus

- H_{k+1} has a complexity of $O\left(d^2\right)$ by avoiding inverting a Hessian Matrix

Finally, we have

If we expand the equation $\left(I-\rho_k s_k y_k^T\right) H_k \left(I-\rho_k y_k s_k^T\right)$

$$\left(I-\rho_k s_k y_k^T\right) H_k \left(I-\rho_k y_k s_k^T\right) = \left(H_k - \rho_k s_k \left[y_k^T H_k\right]\right) - \left(\rho_k \left[H_k y_k\right] s_k^T - \rho\right.$$

Where

1. $\left[y_k^T H_k\right]$ and $\left[H_k y_k\right]$ has complexity $O\left(d^2\right)$
2. Thus $s_k \left[y_k^T H_k\right]$ and $\left[y_k^T H_k y_k\right]$ has complexity $O\left(d^2\right)$

Thus

- H_{k+1} has a complexity of $O\left(d^2\right)$ by avoiding inverting a Hessian Matrix

Finally, we have

If we expand the equation $\left(I-\rho_k s_k y_k^T\right) H_k \left(I-\rho_k y_k s_k^T\right)$

$$\left(I-\rho_k s_k y_k^T\right) H_k \left(I-\rho_k y_k s_k^T\right) = \left(H_k - \rho_k s_k \left[y_k^T H_k\right]\right) - \left(\rho_k \left[H_k y_k\right] s_k^T - \rho\right.$$

Where

1. $\left[y_k^T H_k\right]$ and $\left[H_k y_k\right]$ has complexity $O\left(d^2\right)$
2. Thus $s_k \left[y_k^T H_k\right]$ and $\left[y_k^T H_k y_k\right]$ has complexity $O\left(d^2\right)$

Thus

- H_{k+1} has a complexity of $O\left(d^2\right)$ by avoiding inverting a Hessian Matrix

Problem

There is no magic formula to find an initial H_0

We can use specific information about the problem:

- For instance by setting it to the inverse of an approximate Hessian calculated by finite differences at x_0
- In our case, we have $\frac{\partial \mathcal{L}(w)}{\partial w \partial w^T}$, or in matrix format $X^T W X$, we could get initial setup

Problem

There is no magic formula to find an initial H_0

We can use specific information about the problem:

- For instance by setting it to the inverse of an approximate Hessian calculated by finite differences at $\boldsymbol{x}_0$
- In our case, we have $\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T}$, or in matrix format $X^T W X$, we could get initial setup

Problem

There is no magic formula to find an initial H_0

We can use specific information about the problem:

- For instance by setting it to the inverse of an approximate Hessian calculated by finite differences at $\boldsymbol{x}_0$
- In our case, we have $\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T}$, or in matrix format $X^T W X$, we could get initial setup

Outline

Algorithm (BFGS Method)

Quasi-Newton Algorithm

- Starting point x_0, Convergence tolerance e, Inverse Hessian approximation H_0

1. $k \leftarrow 0$
2. while $\|\nabla f(x_{k+1})\| > e$
3. Compute search direction $p_k = -H_k \nabla \|\nabla f(x_{k+1})\|$
4. Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is obtained from a linear search (Under Wolfe conditions).
5. Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
6. Compute $H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T$
7. $k \leftarrow k + 1$

Algorithm (BFGS Method)

Quasi-Newton Algorithm

- Starting point x_0, Convergence tolerance e, Inverse Hessian approximation H_0

1. $k \leftarrow 0$
2. while $\|\nabla f(x_{k+1})\| > e$
3. Compute search direction $p_k = -H_k \nabla \|\nabla f(x_{k+1})\|$
4. Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is obtained from a linear search (Under Wolfe conditions).
5. Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
6. Compute $H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T$
7. $k \leftarrow k + 1$

Algorithm (BFGS Method)

Quasi-Newton Algorithm

- Starting point x_0, Convergence tolerance e, Inverse Hessian approximation H_0

1. $k \leftarrow 0$
2. while $\|\nabla f(\boldsymbol{x}_{k+1})\| > e$
3. Compute search direction $p_k = -H_k \nabla \|\nabla f(x_{k+1})\|$
4. Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is obtained from a linear search (Under Wolfe conditions).
5. Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
6. Compute $H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T$
7. $k \leftarrow k + 1$

Algorithm (BFGS Method)

Quasi-Newton Algorithm

- Starting point x_0, Convergence tolerance e, Inverse Hessian approximation H_0

1. $k \leftarrow 0$
2. while $\|\nabla f\left(\boldsymbol{x}_{k+1}\right)\| > e$
3. Compute search direction $\boldsymbol{p}_k = -H_k \nabla \|\nabla f\left(\boldsymbol{x}_{k+1}\right)\|$
4. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$ where α_k is obtained from a linear search (Under Wolfe conditions).
5. Define $s_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $y_k = \nabla f\left(\boldsymbol{x}_{k+1}\right) - \nabla f\left(\boldsymbol{x}_k\right)$
6. Compute $H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T$
7. $k \leftarrow k + 1$

Algorithm (BFGS Method)

Quasi-Newton Algorithm

- Starting point x_0, Convergence tolerance e, Inverse Hessian approximation H_0

1. $k \leftarrow 0$
2. while $\|\nabla f(\boldsymbol{x}_{k+1})\| > e$
3. Compute search direction $\boldsymbol{p}_k = -H_k \nabla \|\nabla f(\boldsymbol{x}_{k+1})\|$
4. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$where α_k is obtained from a linear search (Under Wolfe conditions).
5. Define $s_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $y_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$
6. Compute $H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T$
7. $k \leftarrow k + 1$

Algorithm (BFGS Method)

Quasi-Newton Algorithm

- Starting point x_0, Convergence tolerance e, Inverse Hessian approximation H_0

1. $k \leftarrow 0$
2. while $\|\nabla f(\boldsymbol{x}_{k+1})\| > e$
3. Compute search direction $\boldsymbol{p}_k = -H_k \nabla \|\nabla f(\boldsymbol{x}_{k+1})\|$
4. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$where α_k is obtained from a linear search (Under Wolfe conditions).
5. Define $s_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $y_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$
6. Compute $H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T$
7. $k \leftarrow k+1$

Algorithm (BFGS Method)

Quasi-Newton Algorithm

- Starting point x_0, Convergence tolerance e, Inverse Hessian approximation H_0

1. $k \leftarrow 0$
2. while $\|\nabla f(\boldsymbol{x}_{k+1})\| > e$
3. Compute search direction $\boldsymbol{p}_k = -H_k \nabla \|\nabla f(\boldsymbol{x}_{k+1})\|$
4. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$where α_k is obtained from a linear search (Under Wolfe conditions).
5. Define $s_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $y_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$
6. Compute $H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T$
7. $k \leftarrow k + 1$

Algorithm (BFGS Method)

Quasi-Newton Algorithm

- Starting point x_0, Convergence tolerance e, Inverse Hessian approximation H_0

1. $k \leftarrow 0$
2. while $\left\|\nabla f\left(\boldsymbol{x}_{k+1}\right)\right\| > e$
3. Compute search direction $\boldsymbol{p}_k = -H_k \nabla \left\|\nabla f\left(\boldsymbol{x}_{k+1}\right)\right\|$
4. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$where α_k is obtained from a linear search (Under Wolfe conditions).
5. Define $s_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $y_k = \nabla f\left(\boldsymbol{x}_{k+1}\right) - \nabla f\left(\boldsymbol{x}_k\right)$
6. Compute $H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho s_k s_k^T$
7. $k \leftarrow k + 1$

Complexity

The

- Cost of update or inverse update is $O\left(d^2\right)$ operations per iteration.

For More and better versions (With a Hessian Approximation)

- Nocedal, Jorge & Wright, Stephen J. (1999). Numerical Optimization. Springer-Verlag.

Complexity

The

- Cost of update or inverse update is $O\left(d^2\right)$ operations per iteration.

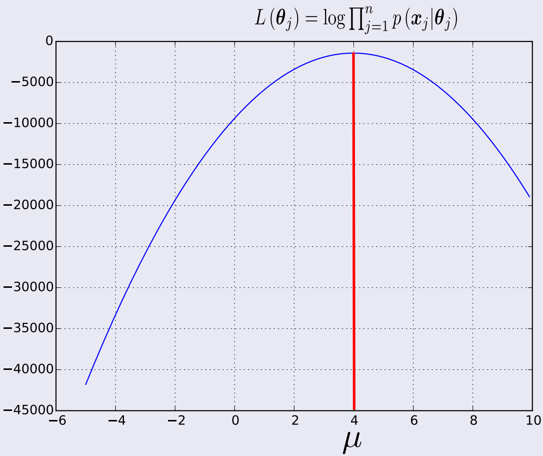
For More and better versions (With a Hessian Approximation)

- Nocedal, Jorge & Wright, Stephen J. (1999). Numerical Optimization. Springer-Verlag.

Outline

Given the following [5]

Because the likelihood is concave



Caution

Here, we change the labeling to $y_i = \pm 1$ with

$$p\left(y_i = \pm 1 | \boldsymbol{x}, \boldsymbol{w}\right) = \sigma\left(y\boldsymbol{w}^T\boldsymbol{x}\right) = \frac{1}{1 + \exp\left\{-y\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

Thus, we have the following log likelihood under regularization $\lambda > 0$

$$\mathcal{L}\left(\boldsymbol{w}\right) = -\sum_{i=1}^{N}\log\left\{1 + \exp\left\{-y_i\boldsymbol{w}^T\boldsymbol{x}_i\right\}\right\} - \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

It is possible to get a Gradient Descent

$$\nabla_{\boldsymbol{w}} l\left(\boldsymbol{w}\right) = \sum_{i=1}^{N}\left\{1 - \frac{1}{1 + \exp\left\{-y_i\boldsymbol{w}^T\boldsymbol{x}_i\right\}}\right\}y_i\boldsymbol{x}_i - \lambda\boldsymbol{w}$$

Caution

Here, we change the labeling to $y_i = \pm 1$ with

$$p\left(y_i = \pm 1 | \boldsymbol{x}, \boldsymbol{w}\right) = \sigma\left(y\boldsymbol{w}^T\boldsymbol{x}\right) = \frac{1}{1 + \exp\left\{-y\boldsymbol{w}^T\boldsymbol{x}\right\}}$$

Thus, we have the following log likelihood under regularization $\lambda > 0$

$$\mathcal{L}\left(\boldsymbol{w}\right) = -\sum_{i=1}^{N} \log\left\{1 + \exp\left\{-y_i\boldsymbol{w}^T\boldsymbol{x}_i\right\}\right\} - \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

It is possible to get a Gradient Descent

$$\nabla_{\boldsymbol{w}} l\left(\boldsymbol{w}\right) = \sum_{i=1}^{N}\left\{1 - \frac{1}{1 + \exp\left\{-y_i\boldsymbol{w}^T\boldsymbol{x}_i\right\}}\right\} y_i\boldsymbol{x}_i - \lambda\boldsymbol{w}$$

Caution

Here, we change the labeling to $y_i = \pm 1$ with

$$p\left(y_i = \pm 1 | \boldsymbol{x}, \boldsymbol{w}\right) = \sigma\left(y \boldsymbol{w}^T \boldsymbol{x}\right) = \frac{1}{1 + \exp\left\{-y \boldsymbol{w}^T \boldsymbol{x}\right\}}$$

Thus, we have the following log likelihood under regularization $\lambda > 0$

$$\mathcal{L}\left(\boldsymbol{w}\right) = -\sum_{i=1}^{N} \log\left\{1 + \exp\left\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\right\}\right\} - \frac{\lambda}{2} \boldsymbol{w}^T \boldsymbol{w}$$

It is possible to get a Gradient Descent

$$\nabla_{\boldsymbol{w}} l\left(\boldsymbol{w}\right) = \sum_{i=1}^{N} \left\{1 - \frac{1}{1 + \exp\left\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\right\}}\right\} y_i \boldsymbol{x}_i - \lambda \boldsymbol{w}$$

Danger Will Robinson!!!

Gradient descent using resembles the Perceptron learning algorithm

Problem!!! It will always converge for a suitable step size, regardless of whether the classes are separable!!!

Outline

Here

We will simplify our work

By stating the algorithm for coordinate ascent

Then

A more precise version will be given

Here

We will simplify our work

By stating the algorithm for coordinate ascent

Then

A more precise version will be given

Coordinate Ascent

Algorithm

- Input Max, an initial $\boldsymbol{w}_0$

1. $counter \leftarrow 0$
2. while $counter < Max$
3. for $i \leftarrow 1, ..., d$
4. Randomly pick i
5. Compute a step size δ^* by approximately maximize $\arg\min_\delta f\left(\boldsymbol{x} + \delta \boldsymbol{e}_i\right)$
6. $x_i \leftarrow x_i + \delta^*$

Where

$$\boldsymbol{e}_i = \begin{pmatrix} 0 & \cdots & 0 & 1 \leftarrow i & 0 & \cdots & 0 \end{pmatrix}^T$$

Coordinate Ascent

Algorithm

- Input Max, an initial $\boldsymbol{w}_0$

1. $counter \leftarrow 0$
2. while $counter < Max$
3. for $i \leftarrow 1, ..., d$
4. Randomly pick i
5. Compute a step size δ^* by approximately maximize $\arg\min_\delta f\left(\boldsymbol{x} + \delta \boldsymbol{e}_i\right)$
6. $x_i \leftarrow x_i + \delta^*$

Where

$$\boldsymbol{e}_i = \begin{pmatrix} 0 & \cdots & 0 & 1 \leftarrow i & 0 & \cdots & 0 \end{pmatrix}^T$$

In the case of Logistic Regression

Thus, we can optimize each w_k alternatively by a coordinate-wise Newton update

$$w_k^{new} = w_k^{old} + \frac{-\lambda w_k^{old} + \sum_{i=1}^{N} \left\{1 - \frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right\} y_i x_{ik}}{\lambda + \sum_{i=1}^{N} x_{ik}^2 \left(\frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right)\left(1 - \frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right)}$$

Complexity of this update

Item	Complexity
$\sum_{i=1}^{N} \left\{1 - \frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right\} y_i x_{ik}$	$O(N)$
$\sum_{i=1}^{N} x_{ik}^2 \left(\frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right)\left(1 - \frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right)$	$O(N)$
Total Complexity	$O(N)$
For all the dimensions	$O(Nd)$

In the case of Logistic Regression

Thus, we can optimize each w_k alternatively by a coordinate-wise Newton update

$$w_k^{new} = w_k^{old} + \frac{-\lambda w_k^{old} + \sum_{i=1}^{N} \left\{1 - \frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right\} y_i x_{ik}}{\lambda + \sum_{i=1}^{N} x_{ik}^2 \left(\frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right)\left(1 - \frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right)}$$

Complexity of this update

Item	Complexity
$\sum_{i=1}^{N} \left\{1 - \frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right\} y_i x_{ik}$	$O(N)$
$\sum_{i=1}^{N} x_{ik}^2 \left(\frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right)\left(1 - \frac{1}{1+\exp\{-y_i \boldsymbol{w}^T \boldsymbol{x}_i\}}\right)$	$O(N)$
Total Complexity	$O(N)$
For all the dimensions	$O(Nd)$

Outline

We have the following Complexities per iteration

Complexities

Method	Per Iteration	Convergence Rate
Cholesky Decomposition	$\frac{d^3}{2} = O\left(d^3\right)$	Quadratic
Quasi-Newton BFGS	$O\left(d^2\right)$	Super-linearly
Coordinate Ascent	$O\left(Nd\right)$	Not established

T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics, Springer New York, 2009.

F. E. Harrell Jr, *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.

J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

A. Krishnamoorthy and D. Menon, "Matrix inversion using cholesky decomposition," in *2013 signal processing: Algorithms, architectures, arrangements, and applications (SPA)*, pp. 70–72, IEEE, 2013.

J. Dhanani and K. Rana, "Logistic regression with stochastic gradient ascent to estimate click through rate," in *Information and Communication Technology for Sustainable Development*, pp. 319–326, Springer, 2018.