Logistic Regression

Instructor: Ping Li

Department of Statistics and Biostatitics

Department of Computer Science

Rutgers University

2015

Calculus Review: Derivatives

Simple derivatives:

$$[\log x]' = \frac{1}{x}, \quad [x^n]' = nx^{n-1}, \quad [e^x]' = e^x, \quad [a^x]' = a^x \log a$$

Chain rule:

$$[f(h(x))]' = f'(h(x))\, h'(x)$$

$$\left[\log\left(ax^2 + e^{2x}\right)\right]' = \frac{1}{(ax^2 + e^{2x})}\left[ax^2 + e^{2x}\right]' = \frac{2ax + 2e^{2x}}{(ax^2 + e^x)}$$

Multivariate derivatives:

$$f(x,y) = a^x + x^n y + cy^2,$$

$$\frac{\partial f(x,y)}{\partial x} = a^x \log a + nx^{n-1}y, \qquad \frac{\partial f(x,y)}{\partial y} = x^n + 2cy$$

Quick Review of Numerical Optimization

Slides 4 - 15 are for reviewing some basic stuff about numerical optimization, which is essential in modern applications.

Maximum Likelihood Estimation (MLE)

Observations x_i, $i = 1$ to n, are i.i.d. samples from a distribution with probability density function $f_X(x; \theta_1, \theta_2, ..., \theta_k)$,
where θ_j, $j = 1$ to k, are parameters to be estimated.

The maximum likelihood estimator seeks the θ to maximize the joint likelihood

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \prod_{i=1}^{n} f_X(x_i; \theta)$$

Or, equivalently, to maximize the log joint likelihood

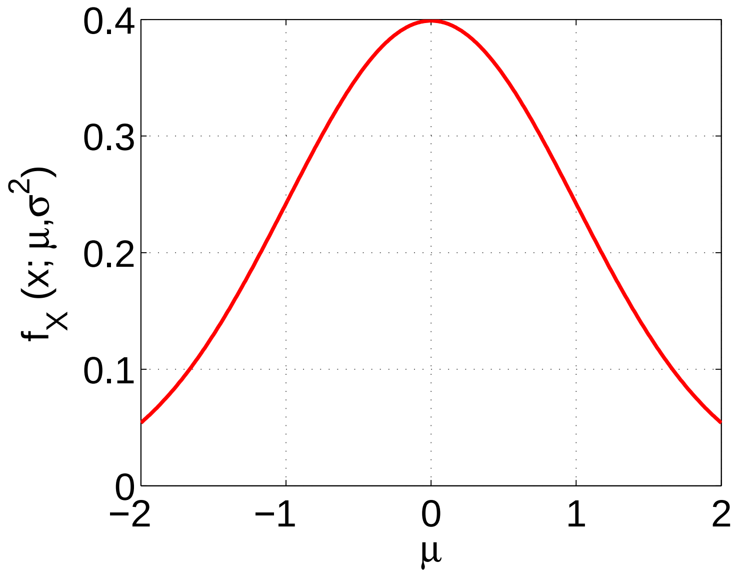
$$\hat{\theta} = \underset{\theta}{\text{argmax}} \sum_{i=1}^{n} \log f_X(x_i; \theta)$$

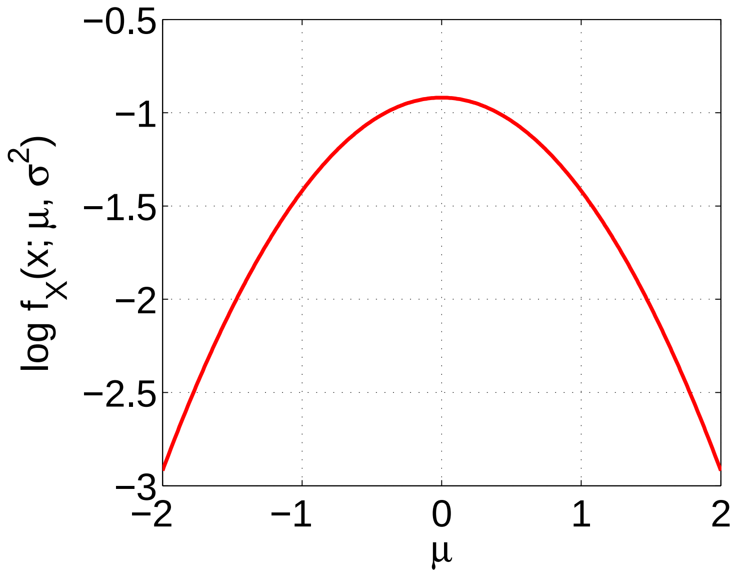
This is a **convex** optimization if f_X is concave or -log-convex.

An Example: Normal Distribution

If $X \sim N\left(\mu, \sigma^2\right)$, then $f_X\left(x; \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Fix $\sigma^2 = 1$, $x = 0$. $f_X\left(x; \mu, \sigma^2\right)$ $\log f_X\left(x; \mu, \sigma^2\right)$





It is Not concave, but it is a -log convex, i.e., a unique MLE solution exists.

$\boxed{\textbf{Example of Exact MLE Solution}}$

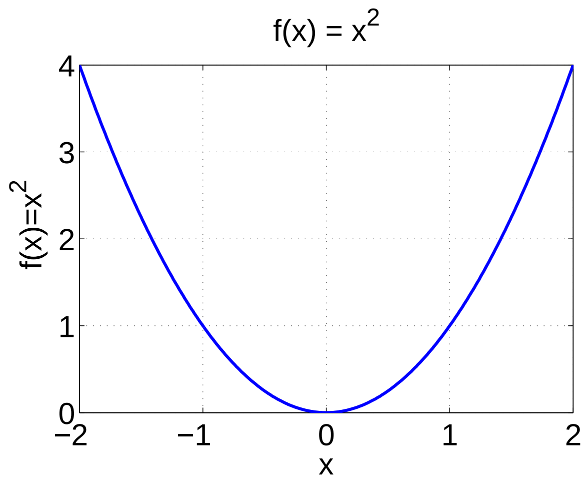
Given n i.i.d. samples, $x_i \sim N(\mu, \sigma^2)$, $i = 1$ to n.

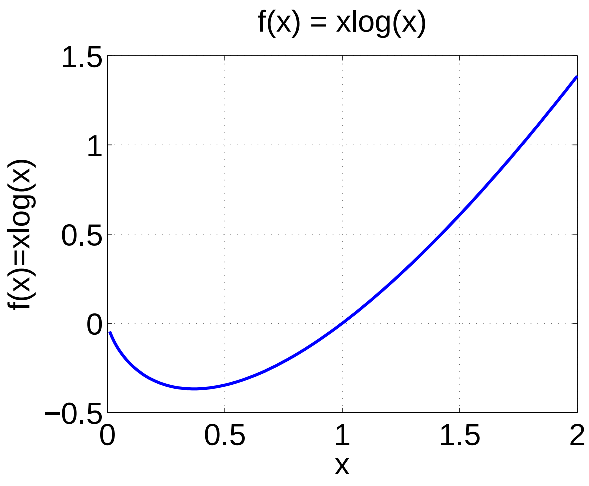
$$\begin{aligned} l\left(x_1, x_2, ..., x_n; \mu, \sigma^2\right) &= \sum_{i=1}^{n} \log f_X(x_i; \mu, \sigma^2) \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2 - \frac{1}{2} n \log(2\pi\sigma^2) \end{aligned}$$

$$\begin{aligned} \frac{\partial l}{\partial \mu} &= \frac{1}{2\sigma^2} 2 \sum_{i=1}^{n} (x_i - \mu) = 0 \Longrightarrow \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \\ \frac{\partial l}{\partial \sigma^2} &= \frac{1}{2\sigma^4} \sum_{i=1}^{n} (x_i - \mu)^2 - \frac{n}{2\sigma^2} = 0 \Longrightarrow \hat{\sigma^2} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2. \end{aligned}$$

Convex Functions

A function $f(x)$ is convex if the second derivative $f''(x) \geq 0$.





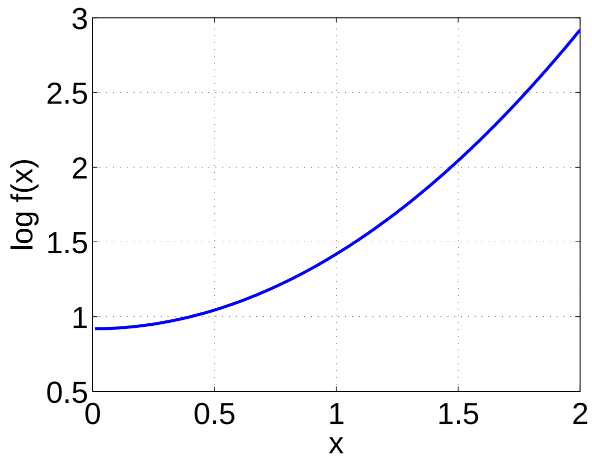
$f(x) = x^2 \Longrightarrow f'' = 2 > 0$, i.e., $f(x) = x^2$ is convex for all x.

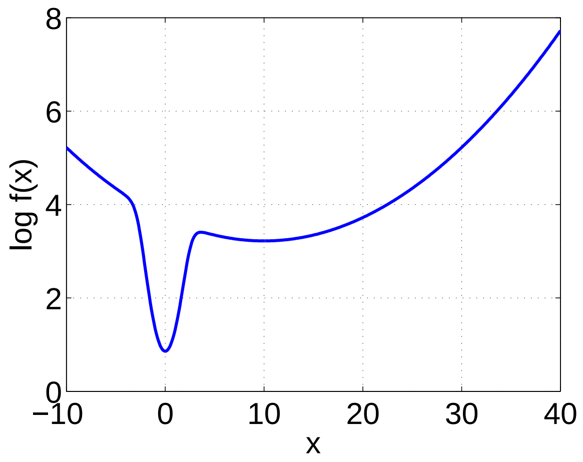
$f(x) = x \log x \Longrightarrow f'' = \frac{1}{x}$, i.e., $f(x) = x \log x$ is convex if $x > 0$.

Both are widely used in statistics and data mining as loss functions,

$\Longrightarrow$ computationally tractable algorithms: least square, logistic regression.

Left panel: $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ is -log convex, $\frac{\partial^2[-\log f(x)]}{\partial x^2} = 1 > 0.$



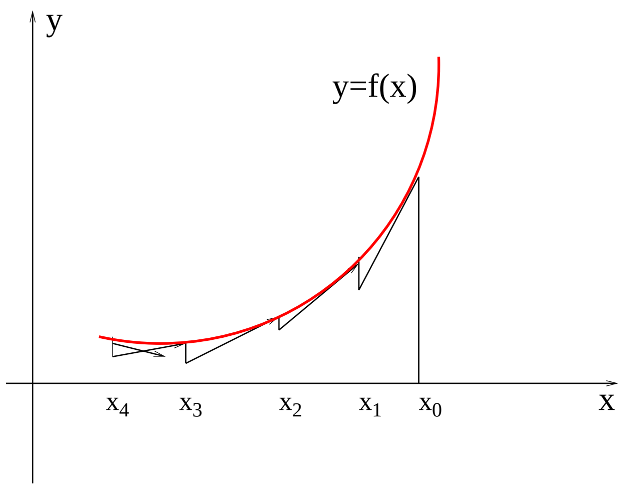


Right panel: a mixture of normals is not -log convex $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} + \frac{1}{\sqrt{2\pi}10} e^{-\frac{(x-10)^2}{200}}$

The mixture of normals is an extremely useful model in statistics.

In general, only a local minimum can be obtained.

Steepest Descent



Procedure:

Start with an initial guess x_0.

Compute $x_1 = x_0 - \Delta f'(x_0)$, where Δ is the step size.

Continue the process $x_{t+1} = x_t - \Delta f'(x_t)$.

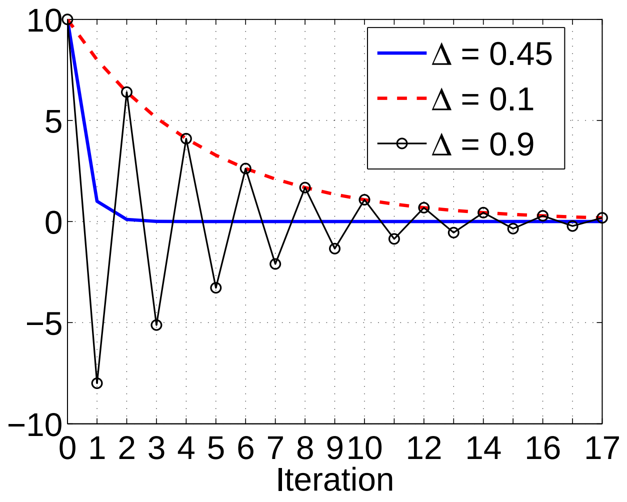
Until some criterion is met, e.g., $f(x_{t+1}) \approx f(x_t)$

The meaning of "steepest" is more clear in the two-dimensional situation.

An Example of Steepest Descent: $f(x) = x^2$

$f(x) = x^2$. The minimum is attained at $x = 0$, $f'(x) = 2x$.

The steepest descent iteration formula $x_{t+1} = x_t - \Delta f'(x_t) = x_t - 2\Delta x_t$.



Choosing the step size Δ is important (even when $f(x)$ is convex).

Too small $\Delta \Longrightarrow$ slow convergence, i.e., many iterations,

Too large $\Delta \Longrightarrow$ oscillations, i.e., also many iterations.

Steepest Descent in Practice

Steepest descent is one of the most widely techniques in real world

- It is extremely simple
- It only requires knowing the first derivative
- It is numerically stable (for above reasons)
- For real applications, it is often affordable to use very small Δ
- In machine learning, Δ is often called learning rate
- It is used in Neural Nets and Gradient Boosting (MART)

Newton's Method

Recall the goal is to find the x^* to minimize $f(x)$.

If $f(x)$ is convex, it is equivalent to finding the x^* such that $f'(x^*) = 0$.

Let $h(x) = f'(x)$. Take Taylor expansion about the optimum solution x^*:

$$h(x^*) = h(x) + (x^* - x)h'(x) + \text{“negligible” higher order terms}$$

Because $h(x^*) = f'(x^*) = 0$, we know approximately

$$0 \approx h(x) + (x^* - x)h'(x) \Longrightarrow x^* \approx x - \frac{h(x)}{h'(x)} = x - \frac{f'(x)}{f''(x)}$$

The procedure of Newton's Method

Start with an initial guess x_0

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

Repeat $x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}$

Until some stopping criterion is reached, e.g., $x_{t+1} \approx x_t$.

An example: $f(x) = (x-c)^2$. $\qquad f'(x) = 2(x-c)$, $f''(x) = 2$.

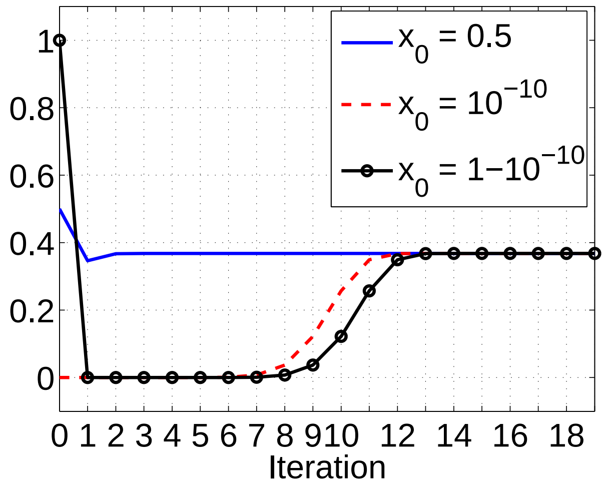
$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} \Longrightarrow x_1 = x_0 - \frac{2(x_0-c)}{2} = c$$

But we already know that $x = c$ minimizes $f(x) = (x-c)^2$.

Newton's method may find the minimum solution using only one step.

An Example of Newton's Method: $f(x) = x\log x$

$$f'(x) = \log x + 1, \qquad f''(x) = \frac{1}{x}. \qquad x_{t+1} = x_t - \frac{\log x_t + 1}{1/x_t}$$



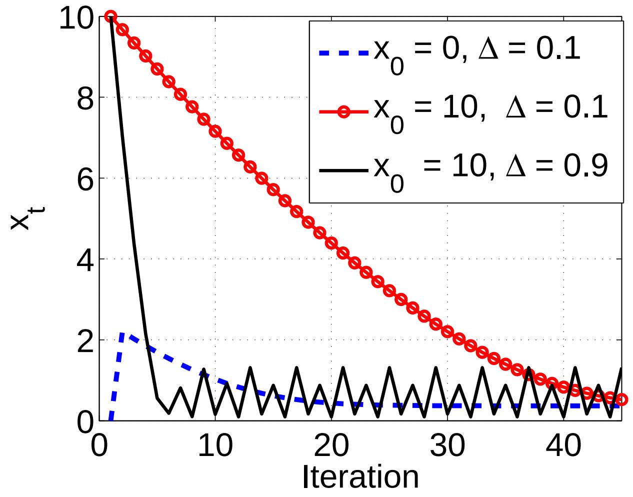
When x_0 is close to optimum solution, the convergence is very fast

When x_0 is far from the optimum, the convergence is slow initially

When x_0 is badly chosen, no convergence. This example requires $0 < x_0 < 1$.

$\boxed{\textbf{Steepest Descent for } f(x) = x \log x}$

$$f'(x) = \log x + 1, \qquad x_{t+1} = x_t - \Delta(\log x_t + 1)$$



Regardless of x_0, convergence is guaranteed if $f(x)$ is convex.

May be oscillating if step size Δ is too large

Convergence is slow near the optimum solution.

$\boxed{\textbf{Error Analysis}}$

To assess the quality of the estimator $\hat{\theta}$ of θ, it is common to use bias, variance, and MSE (mean square error):

$$\textbf{Bias}: \ E(\hat{\theta}) - \theta$$

$$\textbf{Var}: \ E\left(\hat{\theta} - E(\hat{\theta})\right)^2 = E(\hat{\theta}^2) - E^2(\hat{\theta})$$

$$\textbf{MSE}: \ E\left(\hat{\theta} - \theta\right)^2 = Var + Bias^2$$

The last equality is known as the bias variance trade-off. For unbiased estimators, it is desirable to have smaller variance as possible. As the sample size increases, the MLE (under certain conditions) becomes unbiased and achieves the smallest variance. Therefore, the MLE is often a desirable estimator. However, in some cases, biased estimators may achieve smaller MSE than the MLE.

The Expectations and Variances of Common Distributions

The derivations of variances are not required in this course. Nevertheless, it is useful to know the expectations and variances of common distributions.

- **Binomial**: $X \sim binomail(n, p)$, $E(X) = np, Var(X) = np(1-p)$.
- **Normal**: $X \sim N(\mu, \sigma^2)$, $E(X) = \mu, Var(X) = \sigma^2$.
- **Chi-square**: $X \sim \chi^2(k)$, $E(X) = k, Var(X) = 2k$.
- **Exponential**: $X \sim exp(\lambda)$, $E(X) = \frac{1}{\lambda}, Var(X) = \frac{1}{\lambda^2}$.
- **Poisson**: $X \sim Pois(\lambda)$, $E(X) = \lambda, Var(\lambda)$.

Multinomial Distribution

The multinomial is a natural extension to the binomial distribution.

Consider c cells and denote the observations by $(n_1, n_2, ..., n_c)$, which follow a c-cell multinomial distribution with the underlying probabilities $(\pi_1, \pi_2, ..., \pi_c)$ (with $\sum_{i=1}^{c} \pi_i = 1$). Denote $n = \sum_{i=1}^{c} n_i$. We write

$$(n_1, n_2, ..., n_c) \sim Multinomial\,(n, \pi_1, \pi_2, ..., \pi_c)$$

The expectations are (for $i = 1$ to c and $i \neq j$)

$$E\,(n_i) = n\pi_i, \quad Var\,(n_i) = n\pi_i(1 - \pi_i), \quad Cov\,(n_i n_j) = -n\pi_i\pi_j.$$

Note that the cells are negatively correlated (why?).

Logistic Regression

Logistic regression is one of the most widely used statistical tools for predicting cateogrical outcomes.

General setup for binary logistic regression

n observations: $\{x_i, y_i\}, i = 1$ to n. x_i can be a vector.

$y_i \in \{0, 1\}$. For example, “1” = “YES” and “0” = “NO”.

Define

$$p(x_i) = \mathbf{Pr}\left(y_i = 1|x_i\right) = \pi(x_i)$$
$$i.e., \ \mathbf{Pr}\left(y_i = 0|x_i\right) = 1 - p(x_i).$$

The major assumption of logistic regression

$$\log \frac{p(x_i)}{1 - p(x_i)} = \beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p} = \sum_{j=0}^{p} \beta_j x_{i,j}.$$

Here, we treat $x_{i,0} = 1$. We can also use vector notation to write

$$\log \frac{p(x_i; \beta)}{1 - p(x_i; \beta)} = x_i \beta.$$

Here, we view x_i as a row-vector and β as a column-vector.

The model in vector notation

$$p(x_i;\beta) = \frac{e^{x_i\beta}}{1+e^{x_i\beta}}, \qquad 1 - p(x_i;\beta) = \frac{1}{1+e^{x_i\beta}},$$

Log likelihood for the ith observation:

$$\begin{aligned} l_i(\beta|x_i) =&(1-y_i)\log\left[1-p(x_i;\beta)\right] + y_i \log p(x_i;\beta) \\ =& \begin{cases} \log p(x_i;\beta) & \text{if } y_i = 1 \\ \log\left[1-p(x_i;\beta)\right] & \text{if } y_i = 0 \end{cases} \end{aligned}$$

To understand this, consider binomial with only one sample $binomial(1, p(x_i))$ (i.e., Bernouli). When $y_i = 1$, the log likelihood is $\log p(x_i)$ and when $y_i = 0$, the log likelihood is $\log\left(1-p(x_i)\right)$. These two formulas can be written into one.

Joint log likelihood for n observations:

$$l(\beta|x_1, ..., x_n) = \sum_{i=1}^{n} l_i(\beta|x_i)$$

$$= \sum_{i=1}^{n} (1 - y_i) \log \left[1 - p(x_i; \beta)\right] + y_i \log p(x_i; \beta)$$

$$= \sum_{i=1}^{n} y_i \log \frac{p(x_i; \beta)}{1 - p(x_i; \beta)} + \log \left[1 - p(x_i; \beta)\right]$$

$$= \sum_{i=1}^{n} y_i x_i \beta - \log \left(1 + e^{x_i \beta}\right)$$

The remaining task is to solve the optimization problem by MLE.

Logistic Regression with Only One Variable

Basic assumption

$$\text{logit}(\pi(x_i)) = \log \frac{p(x_i;\beta)}{1 - p(x_i;\beta)} = \beta_0 + \beta_1 x_i$$

Joint Log likelihood

$$l(\beta|x_1, ..., x_n) = \sum_{i=1}^{n} \left[y_i(\beta_0 + x_i\beta_1) - \log\left(1 + e^{\beta_0 + x_i\beta_1}\right)\right]$$

Next, we solve the optimization problem for maximizing the joint likelihood, given the data.

First derivatives

$$\frac{\partial l(\beta)}{\partial \beta_0} = \sum_{i=1}^{n} y_i - p(x_i), \qquad \frac{\partial l(\beta)}{\partial \beta_1} = \sum_{i=1}^{n} x_i \left(y_i - p(x_i)\right),$$

Second derivatives

$$\frac{\partial^2 l(\beta)}{\partial \beta_0^2} = -\sum_{i=1}^{n} p(x_i)\left(1 - p(x_i)\right),$$

$$\frac{\partial^2 l(\beta)}{\partial \beta_1^2} = -\sum_{i=1}^{n} x_i^2 p(x_i)\left(1 - p(x_i)\right),$$

$$\frac{\partial^2 l(\beta)}{\partial \beta_0 \beta_1} = -\sum_{i=1}^{n} x_i p(x_i)\left(1 - p(x_i)\right)$$

Solve the MLE by Newton’s Method or steepest descent (two-dim problem).

$\boxed{\textbf{Logistic Regression without Intercept } (\beta_0 = 0)}$

The simplified model

$$\text{logit}(\pi(x_i)) = \log \frac{p(x_i)}{1 - p(x_i)} = \beta x_i$$

Equivalently,

$$p(x_i) = \frac{e^{\beta x_i}}{1 + e^{\beta x_i}} = \pi(x_i), \qquad 1 - p(x_i) = \frac{1}{1 + e^{\beta x_i}},$$

Joint log likelihood for n observations:

$$l(\beta | x_1, ..., x_n) = \sum_{i=1}^{n} x_i y_i \beta - \log\left(1 + e^{\beta x_i}\right)$$

First derivative

$$l'(\beta) = \sum_{i=1}^{n} x_i \left(y_i - p(x_i)\right),$$

Second derivative

$$l''(\beta) = -\sum_{i=1}^{n} x_i^2 p(x_i)\left(1 - p(x_i)\right),$$

Newton’s Method updating formula

$$\beta_{t+1} = \beta_t - \frac{l'(\beta_t)}{l''(\beta_t)}$$

Steepest descent (in fact ascent) updating formula

$$\beta_{t+1} = \beta_t + \Delta l'(\beta_t)$$

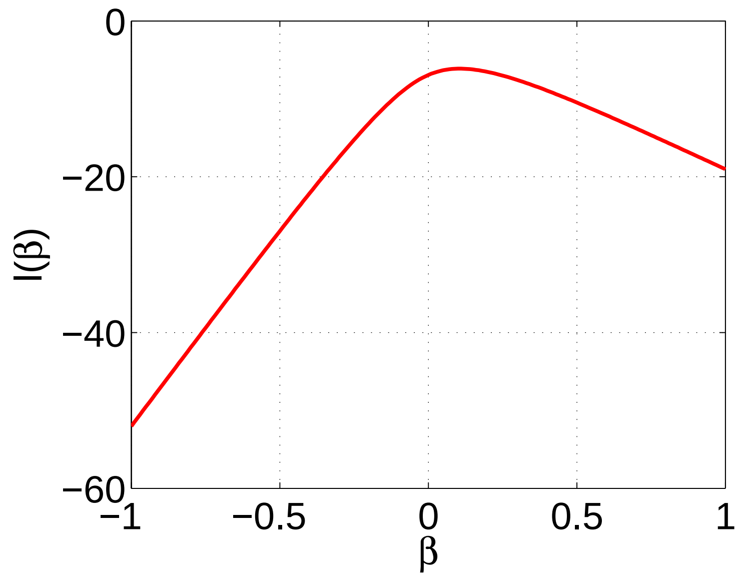
A Numerical Example of Logistic Regression

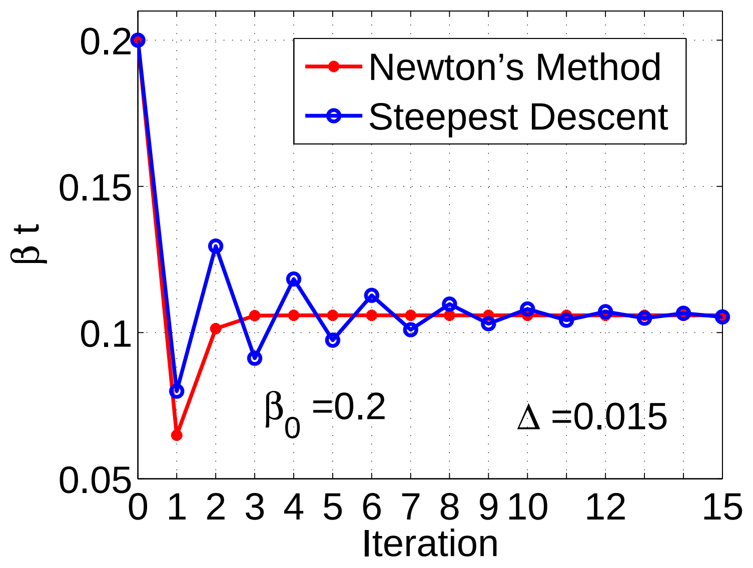
Data

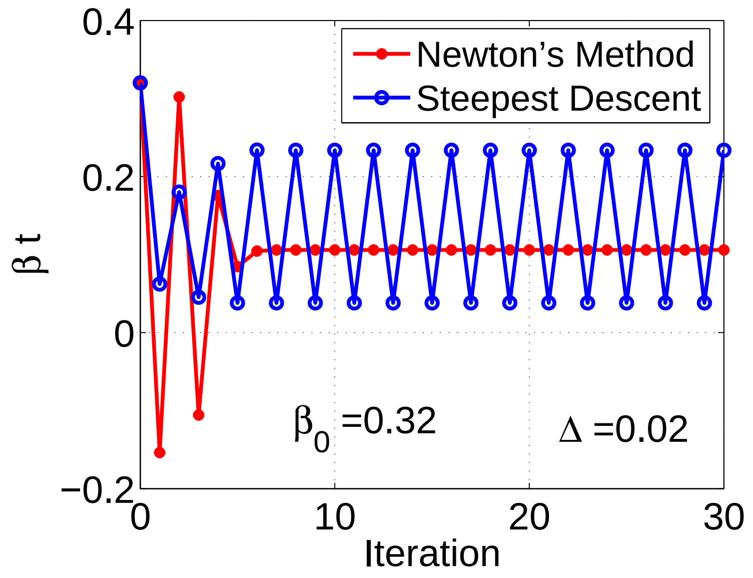
$x = \{8, 14, -7, 6, 5, 6, -5, 1, 0, -17\}$

$y = \{1, 1, 0, 0, 1, 0, 1, 0, 0, 0\}$

Log likelihood function



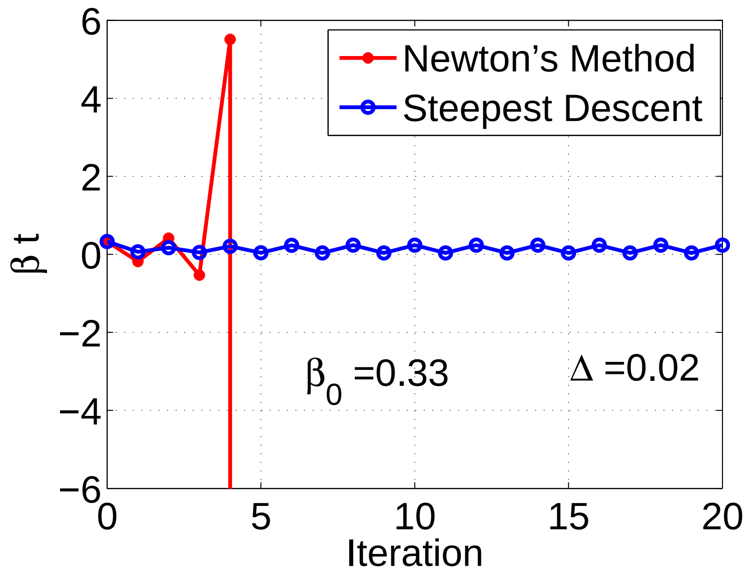


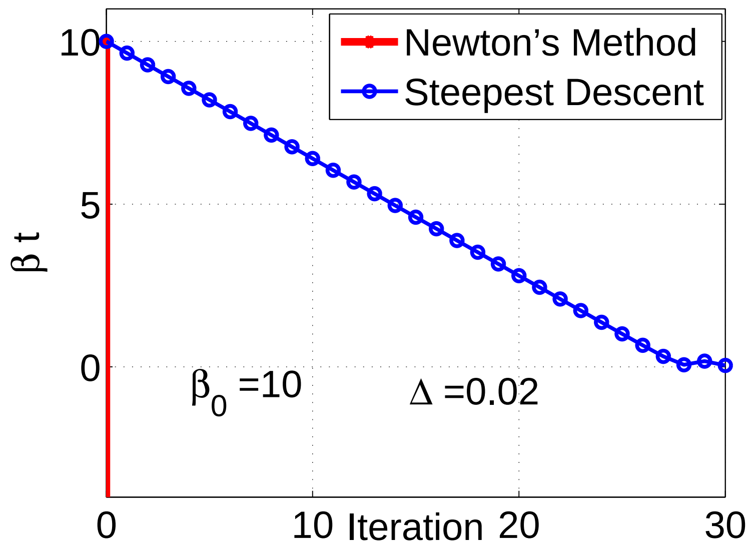


Steepest descent is quite sensitive to the step size Δ.

Too large Δ leads to oscillation.

Here, β_0 means the initial value of β.





Newton's Method is sensitive to the starting point β_0. May not converge at all.

The starting point (mostly) only affects computing time of steepest descent.

In general, with multiple variables, we need to use the matrix formulation, which in fact is easier to implement in matlab.

Newon's Method for Logistic Regression with β_0 and β_1

Analogous to the one variable case, the Newton's update formula is

$$\beta^{\mathbf{new}} = \beta^{\mathbf{old}} - \left[\left(\frac{\partial^{\mathbf{2}} \mathbf{l}(\beta)}{\partial \beta \partial \beta^{\mathsf{T}}} \right)^{-\mathbf{1}} \frac{\partial \mathbf{l}(\beta)}{\partial \beta} \right]_{\beta^{\mathbf{old}}}$$

where $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$,

$$\frac{\partial \mathbf{l}(\beta)}{\partial \beta} =$$

$$\begin{bmatrix} \sum_{i=1}^{n} y_i - p(x_i) \\ \sum_{i=1}^{n} x_i \left(y_i - p(x_i) \right) \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \\ 1 & x_n \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} y_1 - p(x_1) \\ y_2 - p(x_2) \\ \dots \\ y_n - p(x_n) \end{bmatrix} = \mathbf{X}^{\mathsf{T}} (\mathbf{y} - \mathbf{p})$$

$$
\begin{aligned}
&\left(\frac{\partial^{\mathbf{2}}\mathbf{l}(\beta)}{\partial\beta\partial\beta^{\mathsf{T}}}\right)\\
&=\begin{bmatrix} -\sum_{i=1}^{n} p(x_i)\left(1-p(x_i)\right) & -\sum_{i=1}^{n} x_i p(x_i)\left(1-p(x_i)\right) \\ -\sum_{i=1}^{n} x_i p(x_i)\left(1-p(x_i)\right) & -\sum_{i=1}^{n} x_i^2 p(x_i)\left(1-p(x_i)\right) \end{bmatrix}\\
&=-\mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X}
\end{aligned}
$$

$$
\mathbf{W}=\begin{bmatrix} p(x_1)(1-p(x_1)) & 0 & 0... & 0 \\ 0 & p(x_2)(1-p(x_2)) & 0... & 0 \\ ... & & & \\ 0 & 0 & 0... & p(x_n)(1-p(x_n)) \end{bmatrix}
$$

Multivariate Logistic Regression Solution in Matrix Form

Newton' update formula

$$\beta^{new} = \beta^{old} - \left[\left(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^{\mathsf{T}}} \right)^{-1} \frac{\partial l(\beta)}{\partial \beta} \right]_{\beta^{old}}$$

where, in a matrix form

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{n} x_i \left(y_i - p(x_i; \beta) \right) = \mathbf{X}^{\mathbf{T}} (\mathbf{y} - \mathbf{p})$$

$$\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^{\mathsf{T}}} = - \sum_{i=1}^{n} x_i^{\mathsf{T}} x_i p(x_i; \beta) \left(1 - p(x_i; \beta) \right) = -\mathbf{X}^{\mathbf{T}} \mathbf{W} \mathbf{X}$$

We can write the update formula in a matrix form

$$\beta^{\mathbf{new}} = \left[\mathbf{X}^{\mathbf{T}} \mathbf{W} \mathbf{X} \right]^{-\mathbf{1}} \mathbf{X}^{\mathbf{T}} \mathbf{W} \mathbf{z},$$

$$\mathbf{z} = \mathbf{X} \beta^{\mathbf{old}} + \mathbf{W}^{-\mathbf{1}} (\mathbf{y} - \mathbf{p})$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & ... & x_{1,p} \\ 1 & x_{2,1} & x_{2,2} & ... & x_{2,p} \\ ... & & & & \\ 1 & x_{n,1} & x_{n,2} & ... & x_{n,p} \end{bmatrix} \in \mathbb{R}^{n \times (p+1)}$$

$$\mathbf{W} = \begin{bmatrix} p_1(1-p_1) & 0 & 0 & ... & 0 \\ 0 & p_2(1-p_2) & 0 & ... & 0 \\ ... & & & & \\ 0 & 0 & 0 & ... & p_n(1-p_n) \end{bmatrix} \in \mathbb{R}^{n \times n}$$

where $p_i = p(x_i; \beta^{old})$.

Derivation

$$\begin{aligned}
\beta^{\mathbf{new}} &= \beta^{\mathbf{old}} - \left[\left(\frac{\partial^2 l(\beta)}{\partial\beta\partial\beta^{\mathsf{T}}} \right)^{-1} \frac{\partial l(\beta)}{\partial\beta} \right]_{\beta^{old}} \\
&= \beta^{\mathbf{old}} + \left[\mathbf{X^T W X}\right]^{-\mathbf{1}} \mathbf{X^T}(\mathbf{y} - \mathbf{p}) \\
&= \left[\mathbf{X^T W X}\right]^{-\mathbf{1}} \left[\mathbf{X^T W X}\right] \beta^{\mathbf{old}} + \left[\mathbf{X^T W X}\right]^{-\mathbf{1}} \mathbf{X^T}(\mathbf{y} - \mathbf{p}) \\
&= \left[\mathbf{X^T W X}\right]^{-\mathbf{1}} \mathbf{X^T W} \left(\mathbf{X}\beta^{\mathbf{old}} + \mathbf{W}^{-\mathbf{1}}(\mathbf{y} - \mathbf{p})\right) \\
&= \left[\mathbf{X^T W X}\right]^{-\mathbf{1}} \mathbf{X^T W z}
\end{aligned}$$

Note that $\left[\mathbf{X^T W X}\right]^{-\mathbf{1}} \mathbf{X^T W z}$ looks a lot like (weighted) least square.

Two major practical issues:

- The inverse may not (usually does not) exist, especially with large datasets.
- Newton update steps may be too agressive and lead to divergence.

Fitting Logistic Regression with a Learning Rate

At time t, update each coefficient vector:

$$\beta^{\mathbf{t}} = \beta^{(\mathbf{t-1})} + \nu \left[\mathbf{X^T W X}\right]^{-\mathbf{1}} \mathbf{X^T}(\mathbf{y} - \mathbf{p})\Big|_{\mathbf{t-1}}$$

where

$$\mathbf{W} = \text{diag}\left[p_i(1 - p_i)\right]_{i=1}^{n}$$

The magic parameter ν can be viewed as the learning rate to help make sure that the procedure converges. Practically, it is often set to be $\nu = 0.1$.

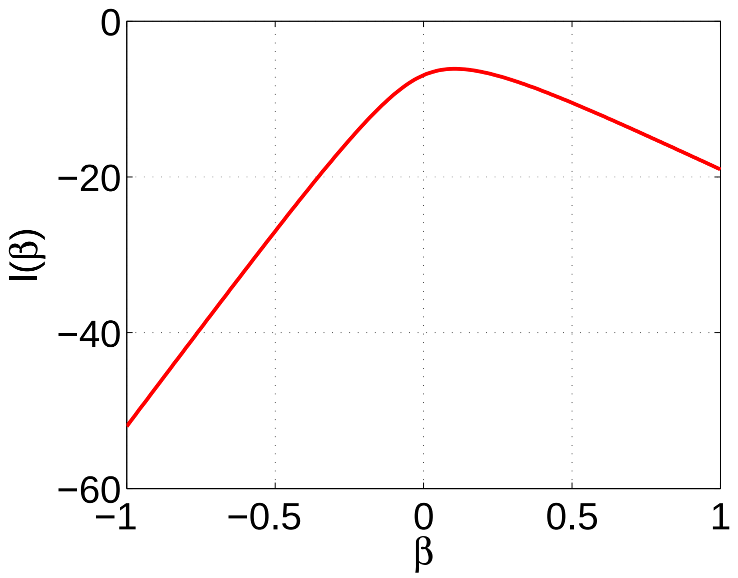
Revisit The Simple Example with Only One β

Data

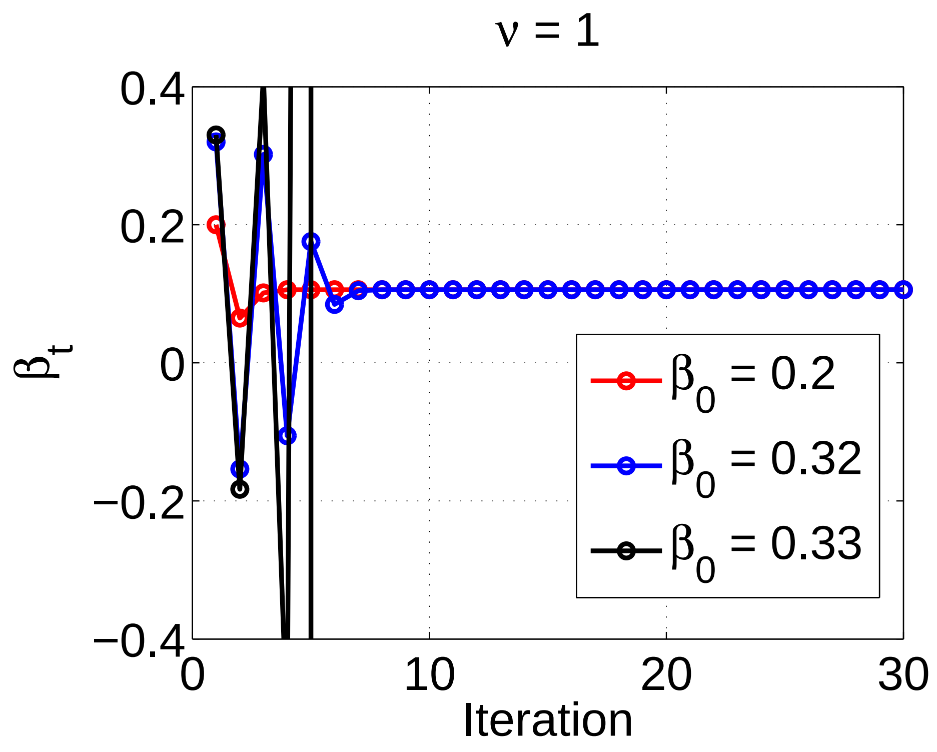
$x = \{8, 14, -7, 6, 5, 6, -5, 1, 0, -17\}$

$y = \{1, 1, 0, 0, 1, 0, 1, 0, 0, 0\}$

Log likelihood function

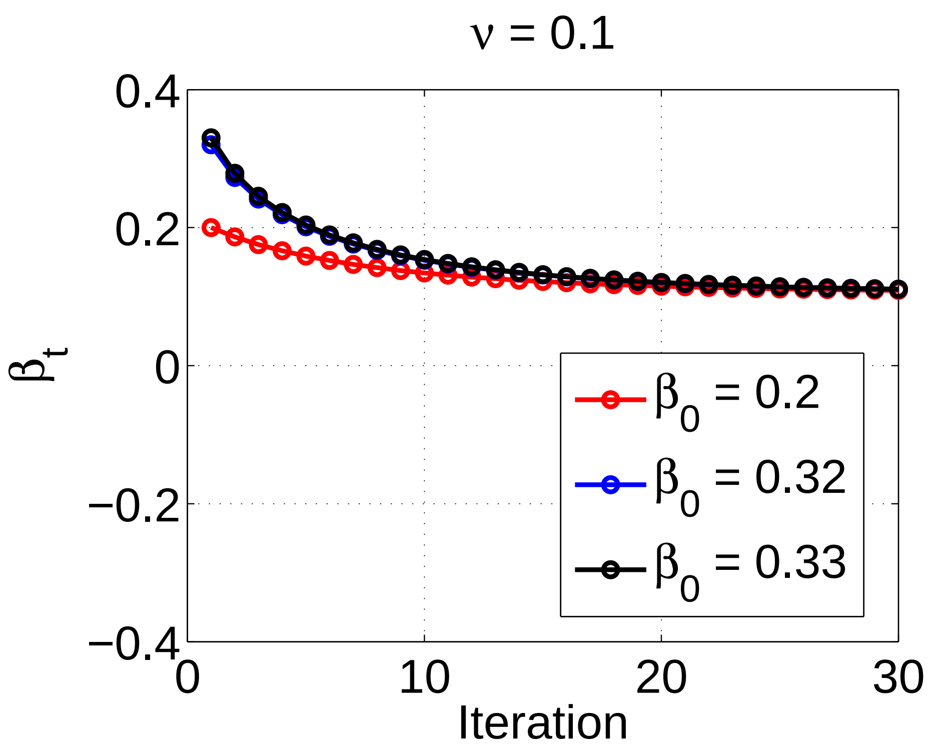


Newton’s Method with Learning Rate $\nu = 1$



When initial $\beta_0 = 0.32$, the method coverges. When $\beta_0 = 0.33$, it does not converge.

Newton's Method with Learning Rate $\nu = 0.1$



Fitting Logistic Regression With Regularization

The almost correct update formula:

$$\beta^{\mathbf{t}} = \beta^{(\mathbf{t-1})} + \nu \left[\mathbf{X^T W X} + \lambda \mathbf{I}\right]^{-\mathbf{1}} \mathbf{X^T}(\mathbf{y} - \mathbf{p})\Big|_{\mathbf{t-1}}$$

Adding the regularization parameter λ usually improves the numerical stability and some times may even result in better test errors.

There are also good statsitical interpretations.

Fitting Logistic Regression With Regularization

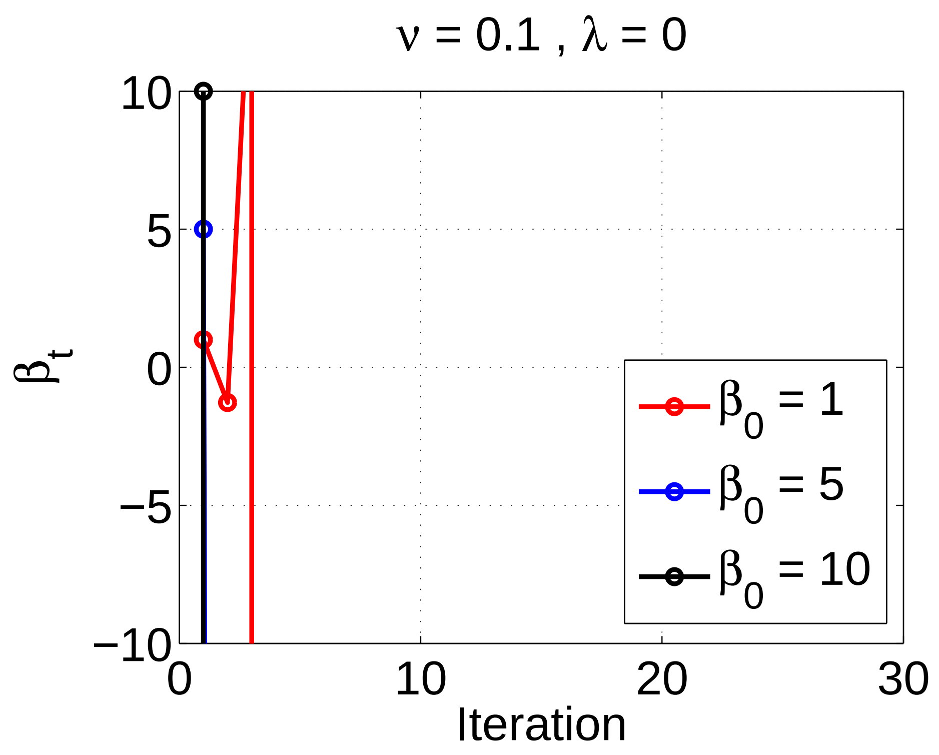
The update formula:

$$\beta^{\mathbf{t}} = \beta^{(\mathbf{t-1})} + \nu \left[\mathbf{X^T W X} + \lambda \mathbf{I}\right]^{-\mathbf{1}} \left[\mathbf{X^T}(\mathbf{y} - \mathbf{p}) - \lambda\beta\right]\Big|_{\mathbf{t-1}}$$

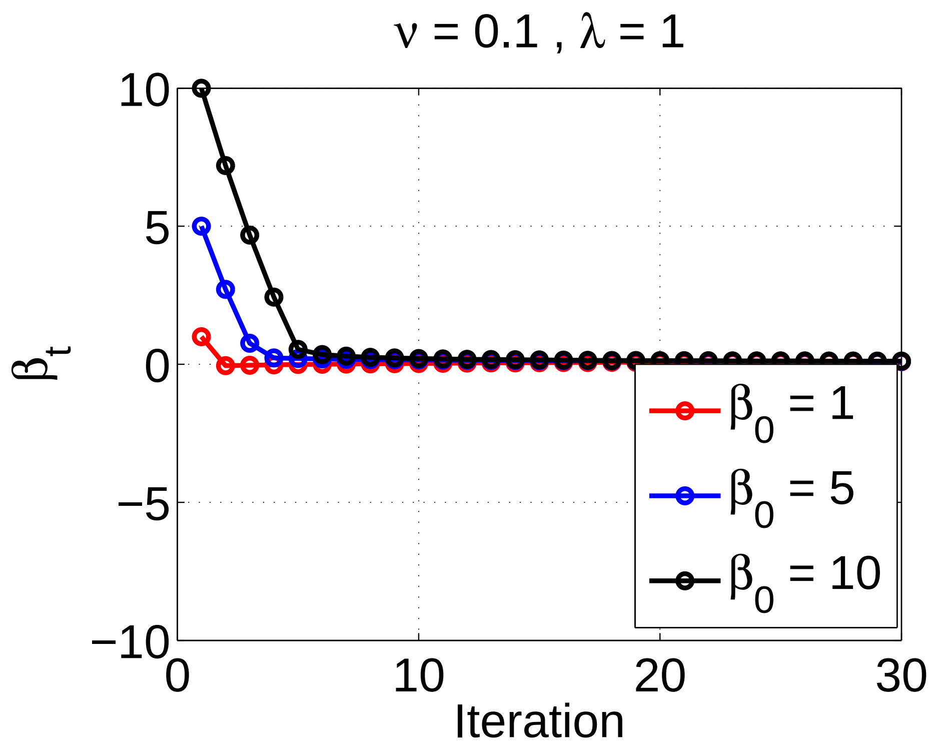
To understand the formula, consider the following modified (regularized) likelihood function:

$$l(\beta) = \sum_{i=1}^{n} \{y_i \log p_i + (1 - y_i)\log(1 - p_i)\} - \frac{\lambda}{2}\sum_{j=0}^{p} \beta_j^2$$

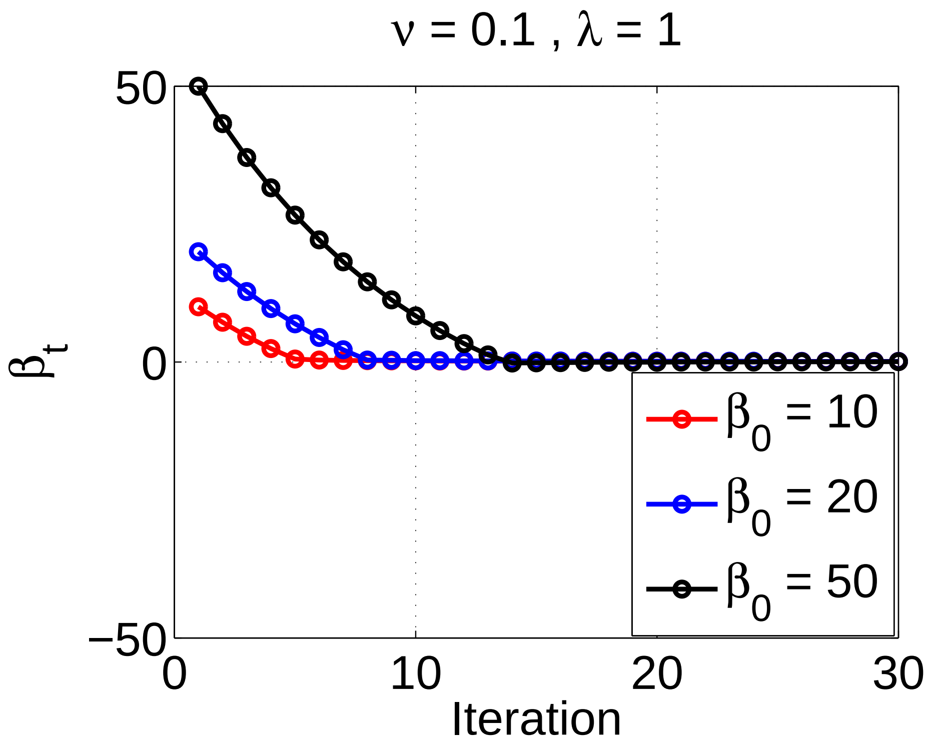
Newton's Method with No Regularization $\lambda = 0$ ($\nu = 0.1$)



Newton's Method with Regularization $\lambda = 1$ ($\nu = 0.1$)



Newton's Method with Regularization $\lambda = 1$ ($\nu = 0.1$)



Multi-Class Logistic Regression

Data: $\{x_i, y_i\}_{i=1}^{n}$, $x_i \in \mathbb{R}^{n\times p}$, $y_i \in \{0, 1, 2, ..., K-1\}$.

Probability model

$$p_{i,k} = \mathbf{Pr}\left\{y_i = k|x_i\right\}, \qquad k = 0, 1, ..., K-1,$$

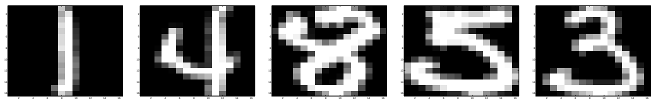
$$\sum_{k=0}^{K-1} p_k = 1, \qquad (\text{only } K-1 \text{ degrees of freedom}).$$

Label assignment

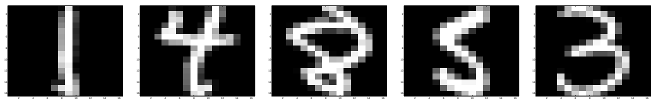
$$\hat{y}_i|x_i = \underset{k}{\text{argmax}}\ \hat{p}_{i,k}$$

Multi-Class Example: USPS ZipCode Recognition

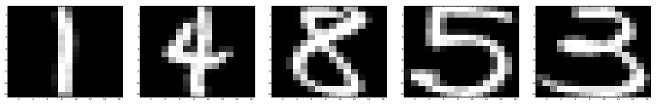
Person 1:



Person 2:



Person 3:



This task can be cast (simplified) as a $K = 10$-class classification problem.

Multinomial Logit Probability Model

$$p_{i,k} = \frac{e^{F_{i,k}}}{\sum_{s=0}^{K-1} e^{F_{i,s}}}$$

where $F_{i,k} = F_k(x_i)$ is the function to be learned from the data.

Linear logistic regression: $F_{i,k} = F_k(x_i) = x_i\beta_k$

Note that, $\beta_k = [\beta_{k,0}, \beta_{k,1}, ..., \beta_{k,p}]^\mathsf{T}$

Multinomial Maximum Likelihood

Mutlinomial likelihood: Suppose $y_i = k$,

$$Lik \propto p_{i,0}^{0} \times ... \times p_{i,k}^{1} \times ... \times p_{i,K-1}^{0} = p_{i,k}$$

log likelihood:

$$l_i = \log p_{i,k}, \qquad \text{if } y_i = k$$

Total log-likelihood in a double summation form:

$$l(\beta) = \sum_{i=1}^{n} l_i = \sum_{i=1}^{n} \left\{ \sum_{k=0}^{K-1} r_{i,k} \log p_{i,k} \right\}$$

$$r_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases}$$

Derivatives of Multi-Class Log-likelihood

The first derivative:

$$\frac{\partial l_i}{\partial F_{i,k}} = \left(r_{i,k} - p_{i,k}\right)$$

Proof:

$$\frac{\partial p_{i,k}}{\partial F_{i,k}} = \frac{\left[\sum_{s=0}^{K-1} e^{F_{i,s}}\right] e^{F_{i,k}} - e^{2F_{i,k}}}{\left[\sum_{s=0}^{K-1} e^{F_{i,s}}\right]^2} = p_{i,k}\left(1 - p_{i,k}\right)$$

$$\frac{\partial p_{i,k}}{\partial F_{i,t}} = \frac{-e^{F_{i,k}} e^{F_{i,t}}}{\left[\sum_{s=0}^{K-1} e^{F_{i,s}}\right]^2} = -p_{i,k}p_{i,t}$$

$$\frac{\partial l_i}{\partial F_{i,k}} = \sum_{s=0}^{K-1} r_{i,s} \frac{1}{p_{i,s}} \frac{\partial p_{i,s}}{\partial F_{i,k}} = r_{i,k} \frac{1}{p_{i,k}} p_{i,k}(1 - p_{i,k}) + \sum_{s \neq k} r_{i,s} \frac{1}{p_{i,s}} \frac{\partial p_{i,s}}{\partial F_{i,k}}$$

$$= r_{i,k}(1 - p_{i,k}) - \sum_{s \neq k} r_{i,s} p_{i,k} = r_{i,k} - \sum_{s=0}^{K-1} r_{i,s} p_{i,k} = r_{i,k} - p_{i,k} \Box$$

The second derivatives:

$$\frac{\partial^2 l_i}{\partial F_{i,k}^2} = -p_{i,k}\left(1 - p_{i,k}\right),$$

$$\frac{\partial^2 l_i}{\partial F_{i,k} F_{i,s}} = -p_{i,k} p_{i,s}$$

Multi-class logistic regression can be fairly complicated. Here, we introduce a simpler approach, which does not seem to explicitly appear in common textbooks.

Conceptually, we fit K binary classification problems (one vs rest) at each iteration. That is, at each iteration, we update β_k seperately for each class. At the end of each iteration, we jointly update the probabilities $p_{i,k} = \frac{e^{x_i \beta_k}}{\sum_{s=0}^{K-1} e^{x_i \beta_s}}$.

A Simple Implementation for Multi-Class Logistic Regression

At time t, update each coefficient vector:

$$\beta_{\mathbf{k}}^{\mathbf{t}} = \beta_{\mathbf{k}}^{(\mathbf{t-1})} + \nu \left[\mathbf{X}^{\mathbf{T}}\mathbf{W}_{\mathbf{k}}\mathbf{X}\right]^{-\mathbf{1}} \mathbf{X}^{\mathbf{T}}(\mathbf{r}_{\mathbf{k}} - \mathbf{p}_{\mathbf{k}})\Big|_{\mathbf{t-1}}$$

where

$$\mathbf{r}_{\mathbf{k}} = \left[r_{1,k}, r_{2,k}, ..., r_{n,k}\right]^{\mathsf{T}}$$

$$\mathbf{p}_{\mathbf{k}} = \left[p_{1,k}, p_{2,k}, ..., p_{n,k}\right]^{\mathsf{T}}$$

$$\mathbf{W}_{\mathbf{k}} = \text{diag}\left[p_{i,k}(1 - p_{i,k})\right]_{i=1}^{n}$$

Then update $\mathbf{p}_{\mathbf{k}}, \mathbf{W}_{\mathbf{k}}$ for the next iteration.

Again, the magic parameter ν can be viewed as the learning rate to help make sure that the procedure converges. Practically, it is often set to be $\nu = 0.1$.

$\boxed{\text{Logistic Regression With } L_2 \text{ Regularization}}$

Total log-likelihood in a double summation form:

$$l(\beta) = \sum_{i=1}^{n} \left\{ \sum_{k=0}^{K-1} r_{i,k} \log p_{i,k} \right\} - \frac{\lambda}{2} \sum_{k=0}^{K-1} \sum_{j=0}^{d} \beta_{k,j}^2$$

$$r_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases}$$

Let $g(\beta) = \frac{\lambda}{2} \sum_{k=0}^{K-1} \sum_{j=0}^{d} \beta_{k,j}^2$, then

$$\frac{\partial g(\beta)}{\beta_{k,j}} = \beta_{k,j}\lambda, \qquad \frac{\partial^2 g(\beta)}{\beta_{k,j}^2} = \lambda$$

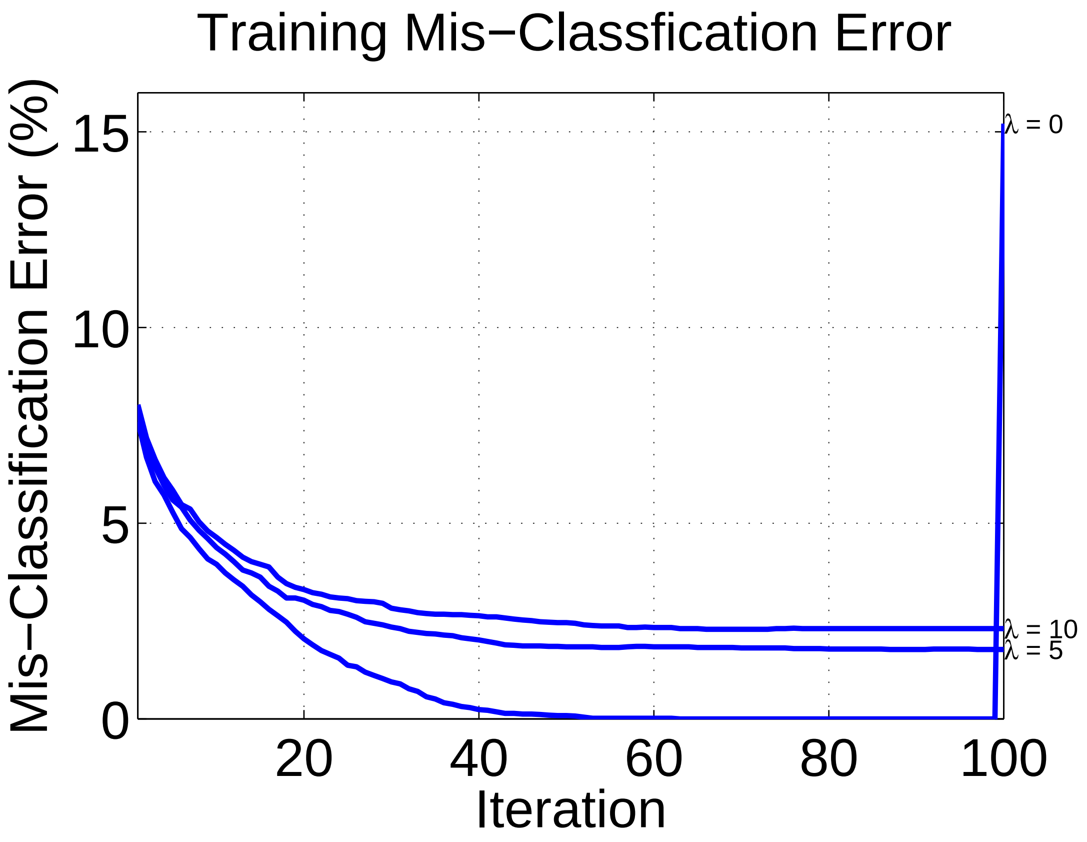
At time t, the updating formula becomes

$$\beta_{\mathbf{k}}^{\mathbf{t}} = \beta_{\mathbf{k}}^{(\mathbf{t-1})} + \nu \left[\mathbf{X}^{\mathbf{T}} \mathbf{W}_{\mathbf{k}} \mathbf{X} + \lambda \mathbf{I}\right]^{-1} \left[\mathbf{X}^{\mathbf{T}}(\mathbf{r}_{\mathbf{k}} - \mathbf{p}_{\mathbf{k}}) - \lambda \beta_{\mathbf{k}}\right]\Big|_{\mathbf{t-1}}$$

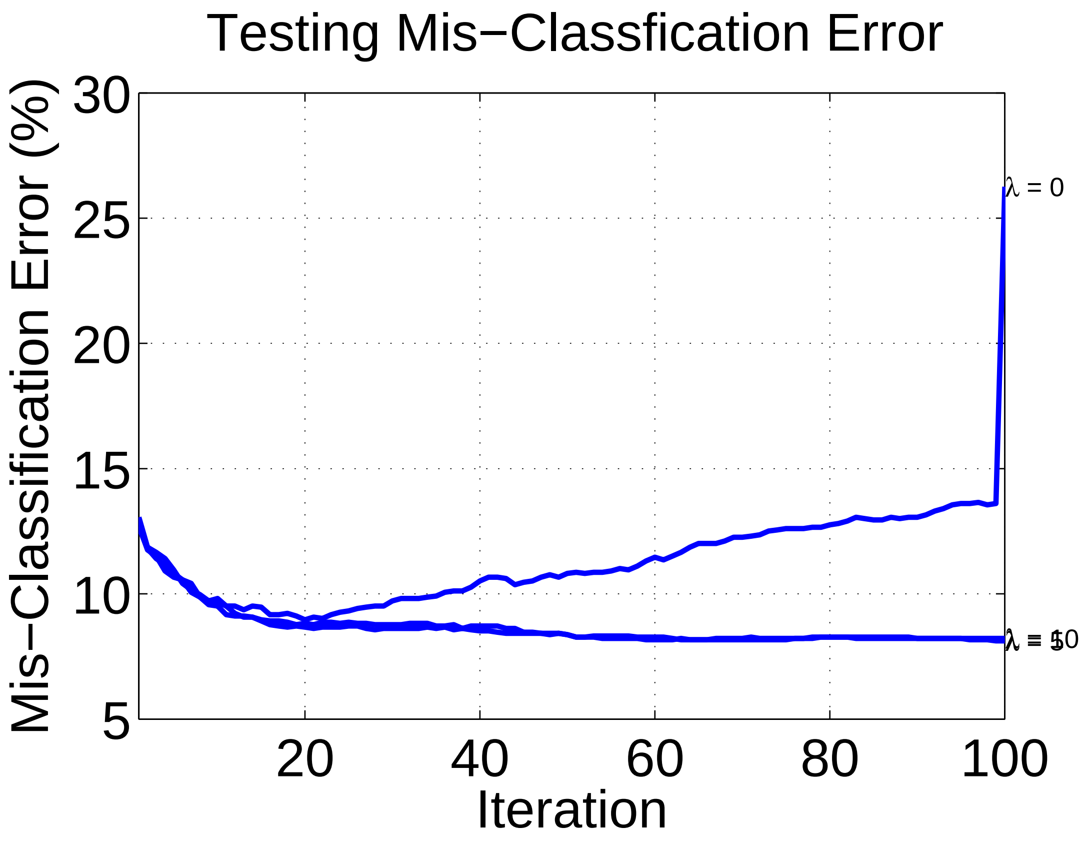
L_2 regularization sometimes improves the numerical stability and some times may even result in better test errors.

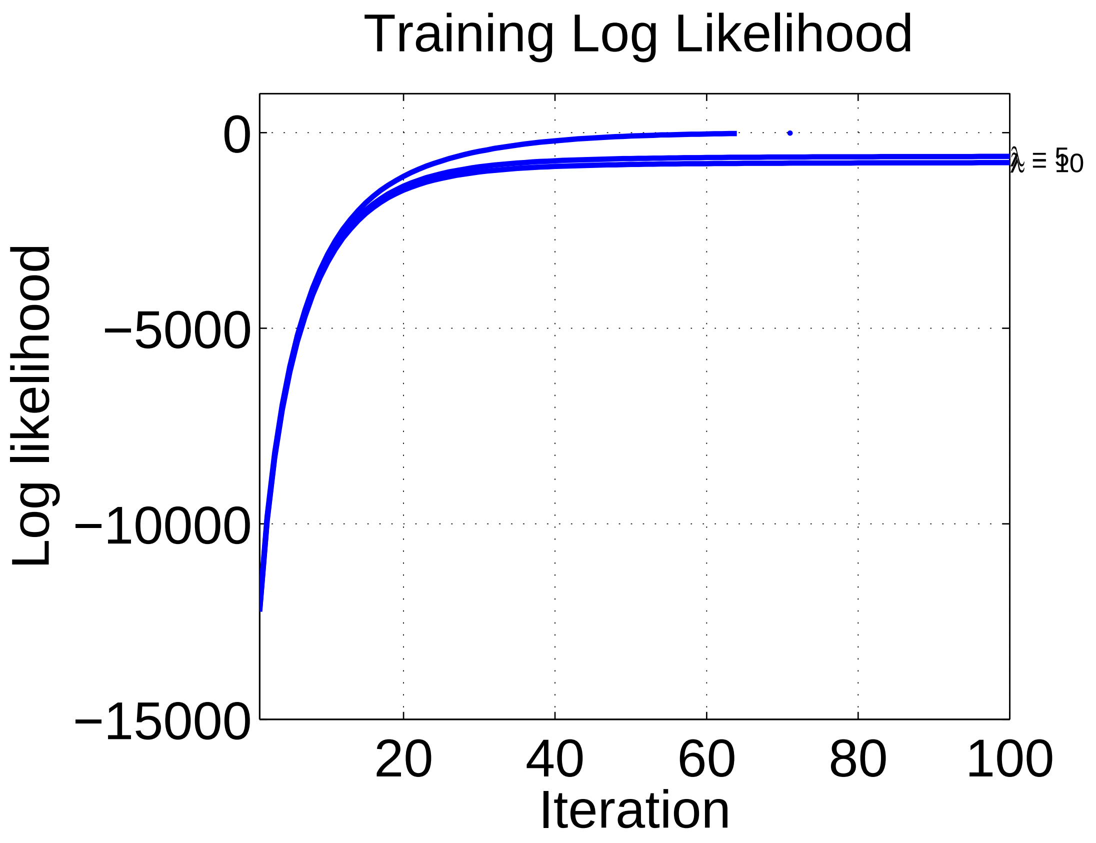
Logistic Regression Results on Zip Code Data

Zip code data: 7291 training examples in 256 dimensions. 2007 test examples.



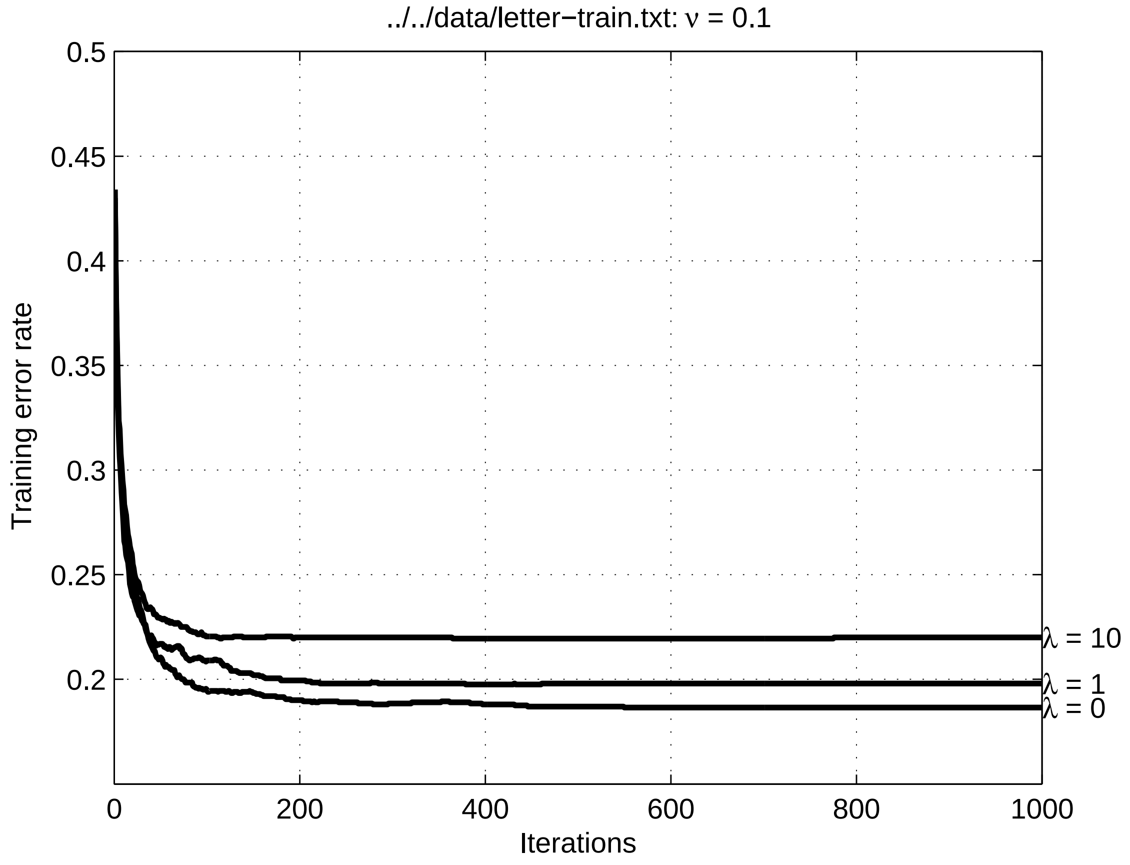
With no regularization ($\lambda = 0$), numerical problems may occur.

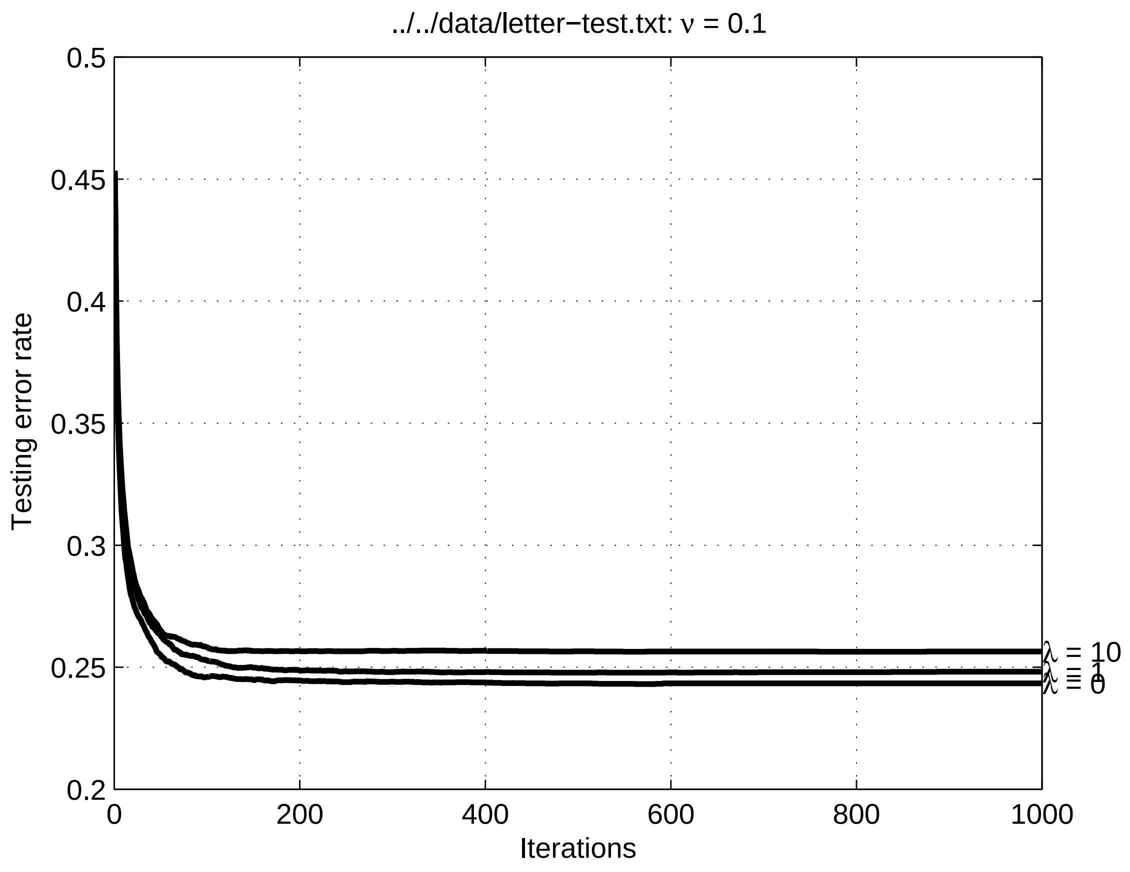

Testing Mis−Classfication Error
Mis−Classification Error (%)
30
25
20
15
10
5
20
40
60
80
100
Iteration
λ = 0
λ = 5
λ = 10

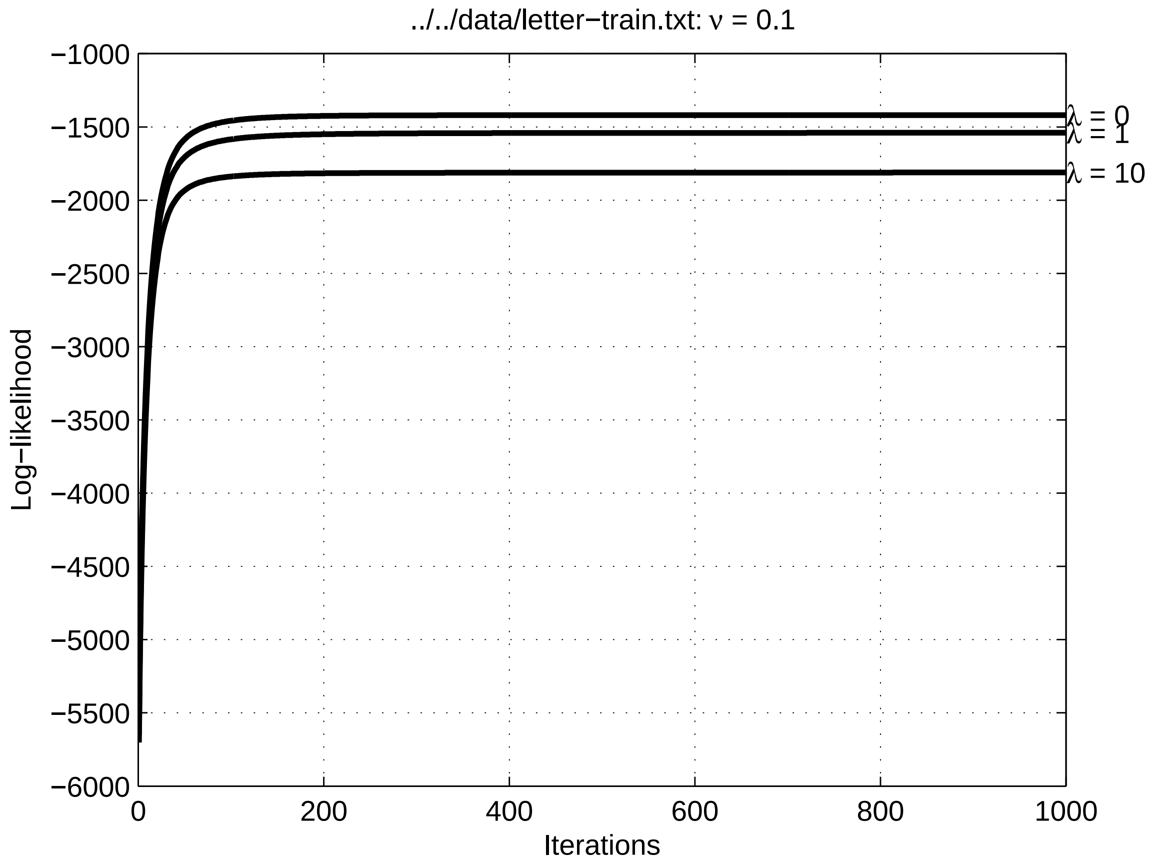

Training Log Likelihood
0
−5000
−10000
−15000
Log likelihood
20
40
60
80
100
Iteration
λ = 5
λ = 10

Another Example on Letter ($K = 26$) Recognition

Letter dataset: 2000 training samples in 16 dimensions. 18000 testing samples.




../../data/letter−test.txt: ν = 0.1
Testing error rate
Iterations
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0
200
400
600
800
1000
λ = 10
λ = 1
λ = 0


../../data/letter−train.txt: ν = 0.1
−1000
−1500
−2000
−2500
−3000
−3500
−4000
−4500
−5000
−5500
−6000
Log−likelihood
0
200
400
600
800
1000
Iterations
λ = 0
λ = 1
λ = 10

Simplifying Label Assignments

Recall **label assignment** in logistic regression:

$$\hat{y}_i | x_i = \underset{k}{\text{argmax}} \; \hat{p}_{i,k}$$

and the **probability model** of logistic regression:

$$p_{i,k} = \frac{e^{x_i \beta_k}}{\sum_{s=0}^{K-1} e^{x_i \beta_s}}$$

It is equivalent to assign labels directly by

$$\hat{y}_i | x_i = \underset{k}{\text{argmax}} \; x_i \hat{\beta}_k$$

This raises an interesting question: maybe we don't need a probability model for the purpose of classification? For example, a linear regression may be also good?

Linear Regression and Its Applications in Classification

Both linear regression and logistic regression are examples of

Generalized Linear Models (GLM).

We first review linear regression and then discuss how to use it for (multi-class) classification.

Review Linear Regression

Given data $\{x_i, y_i\}_{i=1}^n$, where x_i is a p-dimensional vector and y_i is a scalar (not limited to be categories).

We again construct the data matrix

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ 1 & x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \dots & & & & \\ 1 & x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

The data model is

$$\mathbf{y} = \mathbf{X} \times \beta$$

β (a vector of length $p+1$) is obtained by minimizing the mean square errors (equivalent to maximizing the joint likelihood under the normal distribution model).

Linear Regression Estimation by Least Square

The idea is to minimize the mean square errors

$$MSE(\beta) = \sum_{i=1}^{n} |y_i - x_i\beta|^2 = (\mathbf{Y} - \mathbf{X}\beta)^{\mathsf{T}} (\mathbf{Y} - \mathbf{X}\beta)$$

We can find the optimal β by setting the first derivative to be zero

$$\frac{\partial MSE(\beta)}{\beta} = \mathbf{X}^{\mathsf{T}} (\mathbf{Y} - \mathbf{X}\beta) = 0$$

$$\Longrightarrow \mathbf{X}^{\mathsf{T}}\mathbf{Y} = \mathbf{X}^{\mathsf{T}}\mathbf{X}\beta$$

$$\Longrightarrow \beta = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-\mathbf{1}}\mathbf{X}^{\mathsf{T}}\mathbf{Y}$$

Don’t worry much about how to do matrix derivatives. The trick is to view this simply as a scalar derivative but we need to manipulate the order (and add transposes) to get the dimensions correct.

Ridge Regression

Similar to l_2-regularized logistic regression, we can add a regularization parameter

$$\beta = (\mathbf{X}^\intercal \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\intercal \mathbf{Y}$$

which is known as **ridge regression**.

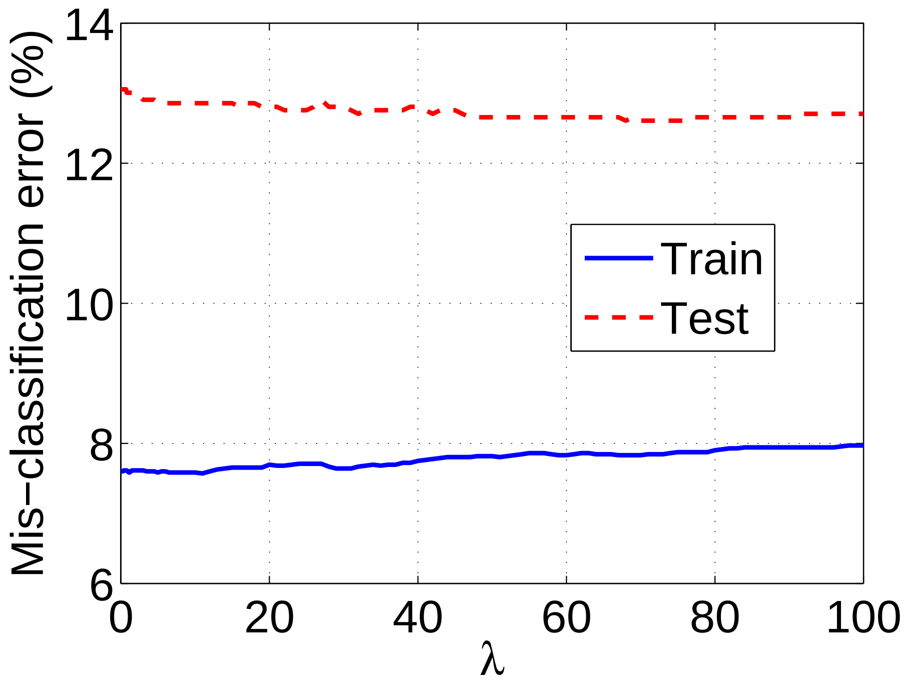
Adding regularization not only improves the numerical stability but also often increases the test accuracy.

Linear Regression for Classification

For binary classification, i.e., $y_i \in \{0, 1\}$, we can simply treat y_i as numerical response and fit a linear regression. To obtain the classification result, we can simply use $\hat{y} = 0.5$ as the classification threshold.

Multi-class classification (with K classes) is more interesting. We can use exactly the same trick as in multi-class logistic regression by first expanding the y_i into a vector of length K with only one entry being 1 and then fitting K binary linear regressions simultaneously and using the location of the maximum fitted value as the class label prediction.

Mis-Classification Errors on Zipcode Data



- This is essentially the first iteration of multi-class logistic regression. Clearly, the results are not as good as logistic regression with many iterations.
- Adding regularization (λ) slightly increases the training errors but decreases the testing errors at certain range.