# Lec 11: Regularized Learning

Ailin Zhang

# Agenda

- Newton-Raphson Method (application to logistic regression)
- Summary: Classical Learning
- Regularized Learning Roadmap
- Overfitting
- Ridge Regression
- Lasso Regression

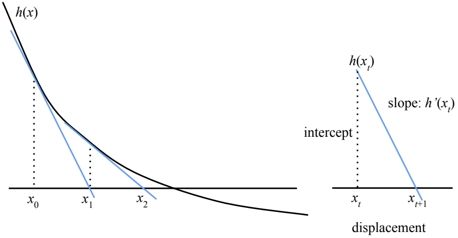
# Newton-Raphson

Suppose we want to solve $h(x) = 0$. At $x_t$, we take the first order Taylor expansion

$$h(x) \doteq h(x_t) + h'(x_t)(x - x_t).$$

Each iteration, we find the root of the linear surrogate function

$$x_{t+1} = x_t - \frac{h(x_t)}{h'(x_t)}.$$



# Newton-Raphson

Suppose we want to find the mode of $f(x)$, we can solve $f'(x) = 0$.

Using Newton-Raphson, we have

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

Each iteration maximizes a quadratic approximation to the original function at $x_t$,

$$f(x) \doteq f(x_t) + f'(x_t)(x - x_t) + \frac{1}{2} f''(x_t)(x - x_t)^2.$$

$f''(x_t)$ is the curvature of $f$ at $x_t$.

- If the curvature is big, the step size should be small.
- If the curvature is small, the step size can be made larger.

# Newton Raphson

If the variable is a vector $x = (x_1, x_2, \cdots, x_n)^\top$, let

$$f'(x) = \left(\frac{\partial f}{\partial x_i}\right)_{n\times 1} \quad f''(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)_{n\times n},$$

$f''(x_t)$ is called the Hessian matrix, we have

$$x_{t+1} = x_t - f''(x_t)^{-1} f'(x_t).$$

$f''(x_t)$ tells us the local shape of $f$ around $x_t$.

$f''(x_t)^{-1} f'(x_t)$ gives us better direction than $f'(x_t)$. The Newton-Raphson is a second order algorithm.

# Newton-Raphson vs Gradient Descent

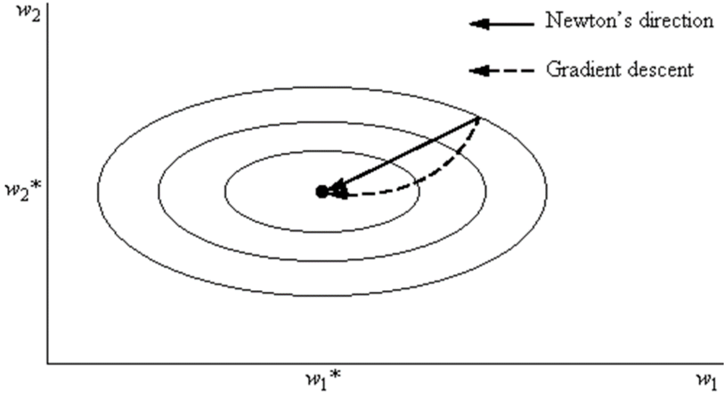



**Figure 2:** Newton-Raphson gives better direction

# Newton-Raphson is IRLS for logistic regression

For maximum likelihood estimate of $\beta$ in logistic regression, the second derivative of the log likelihood function

$$l''(\beta) = -\sum_{i=1}^{n} p_i(1-p_i)X_iX_i^\top.$$

We can update $\beta$ by

$$\beta^{(t+1)} = \beta^{(t)} + l''(\beta^{(t)})^{-1}l'(\beta^{(t)}).$$

$$\beta^{(t+1)} = \left(\sum_{i=1}^{n} w_iX_iX_i^\top\right)^{-1}\left(\sum_{i=1}^{n} w_iX_i\hat{y}_i\right)$$

Let $w_i = p_i(1-p_i)$

# Newton-Raphson is IRLS for logistic regression

we can rewrite the update equation as

$$\begin{aligned}
\beta^{(t+1)} &= \beta^{(t)} + \left[\sum_{i=1}^{n} p_i(1-p_i)X_iX_i^\top\right]^{-1} (y_i - p_i)X_i \\
&= \left(\sum_{i=1}^{n} w_iX_iX_i^\top\right)^{-1} \left[\sum_{i=1}^{n} w_iX_iX_i^\top\beta^{(t)} + (y_i - p_i)X_i\right] \\
&= \left(\sum_{i=1}^{n} w_iX_iX_i^\top\right)^{-1} \left[\sum_{i=1}^{n} w_iX_i\left(X_i^\top\beta^{(t)} + \frac{y_i - p_i}{w_i}\right)\right].
\end{aligned}$$

Let $\bar{y}_i = X_i^\top\beta^{(t)} + \dfrac{y_i - p_i}{w_i}, \tilde{X}_i = X_i\sqrt{w_i}, \tilde{y}_i = \bar{y}_i\sqrt{w_i}$, we can rewrite the equation above as follows:

$$\beta^{(t+1)} = \left(\sum_{i=1}^{n} w_iX_iX_i^\top\right)^{-1} \left(\sum_{i=1}^{n} w_iX_i\bar{y}_i\right)$$

# IRLS Revisited

$$\beta^{(t+1)} = \left(\sum_{i=1}^{n} w_i X_i X_i^\top\right)^{-1} \left(\sum_{i=1}^{n} w_i X_i \bar{y}_i\right)$$
$$= \left(\sum_{i=1}^{n} \tilde{X}_i \tilde{X}_i^\top\right)^{-1} \left(\sum_{i=1}^{n} \tilde{X}_i \tilde{y}_i\right).$$

Consider the flow:

$$\beta^{(t)} \to \eta_i = X_i^\top \beta^{(t)} \to p_i = \sigma(\eta_i) \to w_i = p_i(1-p_i) \to \bar{y}_i = \eta_i + \frac{y_i - p_i}{w_i}$$

$$\to \tilde{X}_i = X_i\sqrt{w_i}, \tilde{y}_i = \bar{y}_i\sqrt{w_i} \to \beta^{(t+1)}.$$

# Classical learning

- Linear Regression
- Gauss-Jordan Elimination
- Sweep Operator
- QR Decomposition
- Orthogonal Matrix
- Householder Reflections
- Linear Regression by QR
- Eigen Decomposition and Diagonalization
- Power Method
- Gram-Schimdt
- Principle Component Analysis and Singular Value Decomposition
- Logistic Regression
- Gradient Ascent
- Iterated Reweighted Least Squares (IRLS)
- Newton-Raphson

# Roadmap for Regularized Learning

- Ridge regression
- Lasso regression
- Coordinate descent
- Spline regression
- Least angle regression
- Stagewise regression for epsilon learning
- Bayesian regression
- Perceptron
- SVM
- Adaboost

Note: We will have midterm after regularized learning! (Nov.10)

# Overfitting

For the simplest regression model

| obs | $\mathbf{X}_{n\times 1}$ | $\mathbf{Y}_{n\times 1}$ |
|---|---|---|
| 1 | $X_1$ | $y_1$ |
| 2 | $X_2$ | $y_2$ |
| ... | | |
| $n$ | $X_n$ | $y_n$ |

# Overfitting

For the simplest regression model

| obs | $\mathbf{X}_{n\times 1}$ | $\mathbf{Y}_{n\times 1}$ |
|---|---|---|
| 1 | $X_1$ | $y_1$ |
| 2 | $X_2$ | $y_2$ |
| ... | | |
| $n$ | $X_n$ | $y_n$ |

Model: $y_i = X_i\beta + \epsilon_i$, where $\epsilon_i \sim N(0, \sigma^2)$

$\hat{\beta_{LS}} = \frac{\sum_{i=1}^n X_i y_i}{\sum_{i=1}^n X_i^2}$ ; if $y_i = \epsilon_i$, $\hat{\beta_{LS}} = \frac{\sum_{i=1}^n X_i \epsilon_i}{\sum_{i=1}^n X_i^2}$

Overfitting: model absorbs noise.

There is always a certain amount of overfitting in $\hat{\beta}$ , even if true $\beta = 0$

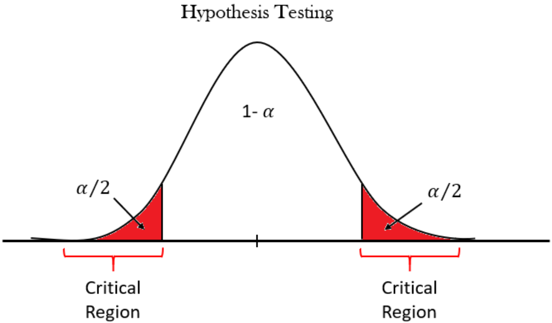
# Ways to avoid overfitting

- Hypothesis test
- Regularization (ML treatment)

# Hypothesis test for linear regression

$H_0 : \beta = 0$ Null hypothesis: simper model

$H_\alpha : \beta \neq 0$ Alternative hypothesis: more complex




Type I Error: $\alpha$, usually taken as 5%

p value: how extreme $\hat{\beta_{LS}}$ is relative to $H_0$

# Hypothesis test for logistic regression

$H_0 : \beta = 0 \rightarrow \eta_i = X_i\beta$

$p_i = \dfrac{e^{\eta_i}}{1 + e^{\eta_i}} = \frac{1}{2} \rightarrow y_i \sim Bernoulli(\frac{1}{2})$ It's a coin flip event.

Noise in the logistic regression (classification) is an over interpretation of a coin flipping as opposed to Gaussian noise observed in linear regression.

# Regularization, ML treatment

Ridge regression

The ridge regression estimates $\beta$ by

$$\hat{\beta}_\lambda = \arg\min_{\beta} \left[ \|\mathbf{Y} - \mathbf{X}\beta\|_{\ell_2}^2 + \lambda\|\beta\|_{\ell_2}^2 \right] = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{Y}$$

for a tuning parameter $\lambda > 0$. The $\lambda\|\beta\|_{\ell_2}^2$ term is a penalty or regularization term.

The resulting estimator is called the shrinkage estimator.

The computation can be accomplished by the sweep operator.

# R code

```
myRidge <- function(X, Y, lambda)

  n = dim(X)[1]
  p = dim(X)[2]
  Z = cbind(rep(1, n), X, Y)
  A = t(Z) %*% Z
  D = diag(rep(lambda, p+2))
  D[1, 1] = 0
  D[p+2, p+2] = 0
  A = A + D
  S = mySweep(A, p+1)
  beta = S[1:(p+1), p+2]
  return(beta)
```

# Lasso Regression

The Lasso regression estimate $\beta$ by

$$\hat{\beta}_\lambda = \arg\min_\beta \left[ \frac{1}{2} \|\mathbf{Y} - \mathbf{X}\beta\|_{\ell_2}^2 + \lambda \|\beta\|_{\ell_1} \right],$$

where $\|\beta\|_{\ell_1} = \sum_{j=1}^p |\beta_j|$.

Lasso stands for **l**east **a**bsolute **s**hrinkage and **s**election **o**perator.

We have closed form solution for $p = 1$, where $\mathbf{X}$ is an $n \times 1$ vector,

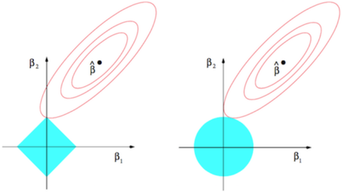
$$\hat{\beta}_\lambda = \begin{cases} (\langle \mathbf{Y}, \mathbf{X} \rangle - \lambda)/\|\mathbf{X}\|_{\ell_2}^2, & \text{if } \langle \mathbf{Y}, \mathbf{X} \rangle > \lambda; \\ (\langle \mathbf{Y}, \mathbf{X} \rangle + \lambda)/\|\mathbf{X}\|_{\ell_2}^2, & \text{if } \langle \mathbf{Y}, \mathbf{X} \rangle < -\lambda; \\ 0 & \text{if } |\langle \mathbf{Y}, \mathbf{X} \rangle| < \lambda. \end{cases}$$

We can write it as

$$\hat{\beta}_\lambda = \mathrm{sign}(\hat{\beta}) \max(0, |\hat{\beta}| - \lambda/\|\mathbf{X}\|_{\ell_2}^2),$$

Npte: There is no closed form solution for general $p$.

# Ridge vs Lasso




The primal form of Lasso: $\min \|\mathbf{Y} - \mathbf{X}\beta\|_{\ell_2}^2/2$ subject to $\|\beta\|_{\ell_1} \leq t$.

The dual form of Lasso : $\min \|\mathbf{Y} - \mathbf{X}\beta\|_{\ell_2}^2/2 + \lambda\|\beta\|_{\ell_1}$.

The primal form also reveals the sparsity inducing property of $\ell_1$ regularization in that the $\ell_1$ ball has low-dimensional corners, edges, and faces, but is still barely convex.

# Coordinate descent for Lasso solution path

Update one component at a time:

For multi-dimensional $\mathbf{X} = (\mathbf{X}_j, j = 1, ..., p)$, given the current values of $\beta = (\beta_j, j = 1, ..., p)$, let $\mathbf{R}_j = \mathbf{Y} - \sum_{k \neq j} \mathbf{X}_k \beta_k$, we can update $\beta_j = \mathrm{sign}(\hat{\beta}_j) \max(0, |\hat{\beta}_j| - \lambda / \|\mathbf{X}\|_{\ell_2}^2)$, where $\hat{\beta}_j = \langle \mathbf{R}_j, \mathbf{X}_j \rangle / \|\mathbf{X}_j\|_{\ell_2}^2$.

Solution path of Lasso:

Start from a big $\lambda$ so that all of the estimated $\beta_j$ are zeros. Then we gradually reduce $\lambda$. For each $\lambda$, we cycle through $j = 1, ..., p$ for coordinate descent until convergence, and then we lower $\lambda$. This gives us $\hat{\beta}(\lambda)$ for the whole range of $\lambda$.

The whole process is a forward selection process, which sequentially selects new variables and occasionally removes selected variables.

# R Code

```
n = 50;p = 200;s = 10;T = 10
lambda_all = (100:1)*10
L = length(lambda_all)

X = matrix(rnorm(n*p), nrow=n)
beta_true = matrix(rep(0, p), nrow = p)
beta_true[1:s] = 1:s
Y = X %*% beta_true + rnorm(n)
beta = matrix(rep(0, p), nrow = p)
beta_all = matrix(rep(0, p*L), nrow = p)
R = Y
ss = rep(0, p)
for (j in 1:p)
    ss[j] = sum(X[, j]^2)
err = rep(0, L)
for (l in 1:L)
{
    lambda = lambda_all[l]
    for (t in 1:T)
    {
      for (j in 1:p)
      {
        db = sum(R*X[, j])/ss[j]
        b = beta[j]+db
        b = sign(b)*max(0, abs(b)-lambda/ss[j])
        db = b - beta[j]
        R = R - X[, j]*db
        beta[j] = b
      }
    }
    beta_all[, l] = beta
    err[l] = sum((beta-beta_true)^2)
}
par(mfrow=c(1,2))
matplot(t(matrix(rep(1, p), nrow = 1)%*%abs(beta_all)), t(beta_all), type = 'l')
plot(lambda_all, err, type = 'l')
```