Advanced Optimization

Logistic Regression Model

Logistic Regression

Optimization algorithm

Cost function $J(\theta)$. Want $\min_\theta J(\theta)$.

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial\theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Optimization algorithm

Cost function $J(\theta)$. Want $\min_\theta J(\theta)$.

Given θ, we have code that can compute

- $J(\theta)$
- $\boxed{\frac{\partial}{\partial\theta_j} J(\theta)}$ (for $j = 0, 1, \ldots, n$)

Gradient descent:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial\theta_j} J(\theta)}$$

$\}$

Andrew Ng

Optimization algorithm

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial\theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Optimization algorithms:

- Gradient descent

Optimization algorithm

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial\theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Andrew Ng

Optimization algorithm

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial\theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:

- More complex

Example:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

Example:

$\min_{\theta} J(\theta)$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$\theta_1 = 5, \theta_2 = 5.$

$$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

Example:

$\min_{\theta} J(\theta)$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$\theta_1 = 5, \theta_2 = 5.$

$$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

```
function [jVal, gradient]
              = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
             (theta(2)-5)^2;
  gradient = zeros(2,1);
  gradient(1) = 2*(theta(1)-5);
  gradient(2) = 2*(theta(2)-5);
```

Example:

$\min_{\theta} J(\theta)$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$\theta_1 = 5, \theta_2 = 5.$

$$\boxed{J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2}$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

```
function [jVal, gradient]
            = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
         (theta(2)-5)^2;
  gradient = zeros(2,1);
  gradient(1) = 2*(theta(1)-5);
  gradient(2) = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
      = fminunc(@costFunction, initialTheta, options);
```

$$\texttt{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

`function [jVal, gradient] = costFunction(theta)`

`jVal = [`code to compute $J(\theta)$`];`

`gradient(1) = [`code to compute $\frac{\partial}{\partial\theta_0}J(\theta)$`];`

`gradient(2) = [`code to compute $\frac{\partial}{\partial\theta_1}J(\theta)$`];`

$\vdots$

`gradient(n+1) = [`code to compute $\frac{\partial}{\partial\theta_n}J(\theta)$`];`

$$\texttt{theta} = \begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_n\end{bmatrix}$$

theta(1)
theta(2)
theta(n+1)

function [jVal, gradient] = costFunction(theta)

jVal = [code to compute $J(\theta)$];

gradient(1) = [code to compute $\frac{\partial}{\partial\theta_0}J(\theta)$];

gradient(2) = [code to compute $\frac{\partial}{\partial\theta_1}J(\theta)$];

$\vdots$

gradient(n+1) = [code to compute $\frac{\partial}{\partial\theta_n}J(\theta)$];

Andrew Ng

$$\texttt{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

theta(1)
theta(2)
theta(n+1)

$\boxed{\texttt{function [jVal, gradient] = costFunction(theta)}}$

jVal = [code to compute $J(\theta)$] ;

gradient(1) = [code to compute $\frac{\partial}{\partial\theta_0}J(\theta)$] ;

gradient(2) = [code to compute $\frac{\partial}{\partial\theta_1}J(\theta)$] ;

$\vdots$

gradient(n+1) = [code to compute $\frac{\partial}{\partial\theta_n}J(\theta)$] ;

Exercise

- Suppose you want to use an advanced optimization algorithm to minimize the cost function for logistic regression with parameters θ_0 and θ_1. You write the following code :

- $function\ [jVal, gradient] = \ costFunction(theta)$

  jVal = % code to compute J
- $gradient(1)\ =\ \ CODE\#1$ % $derivative\ for\ \ theta\_0$
- $gradient(1)\ =\ \ CODE\#1$ % $derivative\ for\ \ theta\_0$

Exercise

- What should CODE#1 and CODE#2 above compute?

What should CODE#1 and CODE#2 above compute?

- ○ CODE#1 and CODE#2 should compute $J(\theta)$.
- ○ CODE#1 should be θ_1 and CODE#2 should be θ_2.
- ○ CODE#1 should compute $\frac{1}{m}\sum_{i=1}^{m}\left[(h_\theta(x^{(i)}) - y^{(i)}).x_0^{(i)}\right] (= \frac{\partial}{\partial\theta_0}J(\theta))$,

  and CODE#2 should compute $\frac{1}{m}\sum_{i=1}^{m}\left[(h_\theta(x^{(i)}) - y^{(i)}).x_1^{(i)}\right] (= \frac{\partial}{\partial\theta_1}J(\theta))$
- ○ None of them.