Data 401
Gradient Descent

Dennis Sun

October 12, 2016

1. Review
2. Gradient Descent
3. Stochastic Gradient Descent

1 Review

2 Gradient Descent

3 Stochastic Gradient Descent

Review of Linear Regression

- Linear regression chooses $\boldsymbol{\beta}$ to minimize

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2$$

Review of Linear Regression

- Linear regression chooses $\boldsymbol{\beta}$ to minimize

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2$$

- We took the derivatives, set them equal to 0, and found that

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}.$$

Review of Linear Regression

- Linear regression chooses $\boldsymbol{\beta}$ to minimize

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2$$

- We took the derivatives, set them equal to 0, and found that

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}.$$

- Of course we do not actually compute $(X^T X)^{-1}$, but we compute $\hat{\boldsymbol{\beta}}$ by solving the linear system:

$$(X^T X)\hat{\boldsymbol{\beta}} = X^T \mathbf{y}.$$

Review of Linear Regression

- Linear regression chooses $\boldsymbol{\beta}$ to minimize

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2$$

- We took the derivatives, set them equal to 0, and found that

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}.$$

- Of course we do not actually compute $(X^T X)^{-1}$, but we compute $\hat{\boldsymbol{\beta}}$ by solving the linear system:

$$(X^T X)\hat{\boldsymbol{\beta}} = X^T \mathbf{y}.$$

- This is a system of p equations with p unknowns. What is the complexity?

Review of Linear Regression

- Linear regression chooses $\boldsymbol{\beta}$ to minimize

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2$$

- We took the derivatives, set them equal to 0, and found that

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}.$$

- Of course we do not actually compute $(X^T X)^{-1}$, but we compute $\hat{\boldsymbol{\beta}}$ by solving the linear system:

$$(X^T X)\hat{\boldsymbol{\beta}} = X^T \mathbf{y}.$$

- This is a system of p equations with p unknowns. What is the complexity? **Answer:** $O(p^3)$.

Today

Efficient ways to fit linear regression to massive data sets where p is large.

1 Review

2 Gradient Descent

3 Stochastic Gradient Descent

Numerical Optimization

We want to find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

Numerical Optimization

We want to find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

Consider the following *iterative* approach:

Numerical Optimization

We want to find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

Consider the following *iterative* approach:

1. Start with a random guess of $\boldsymbol{\beta}$. Call it $\boldsymbol{\beta}^{(0)}$.

Numerical Optimization

We want to find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

Consider the following *iterative* approach:

1. Start with a random guess of $\boldsymbol{\beta}$. Call it $\boldsymbol{\beta}^{(0)}$.
2. Move $\boldsymbol{\beta}$ in the direction that will decrease L the most to get a new guess $\boldsymbol{\beta}^{(1)}$.

Numerical Optimization

We want to find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}))^2.$$

Consider the following *iterative* approach:

1. Start with a random guess of $\boldsymbol{\beta}$. Call it $\boldsymbol{\beta}^{(0)}$.
2. Move $\boldsymbol{\beta}$ in the direction that will decrease L the most to get a new guess $\boldsymbol{\beta}^{(1)}$.
3. From $\boldsymbol{\beta}^{(1)}$, there will be a new direction that decreases L the most. Move in that direction to get a new guess $\boldsymbol{\beta}^{(2)}$.

Numerical Optimization

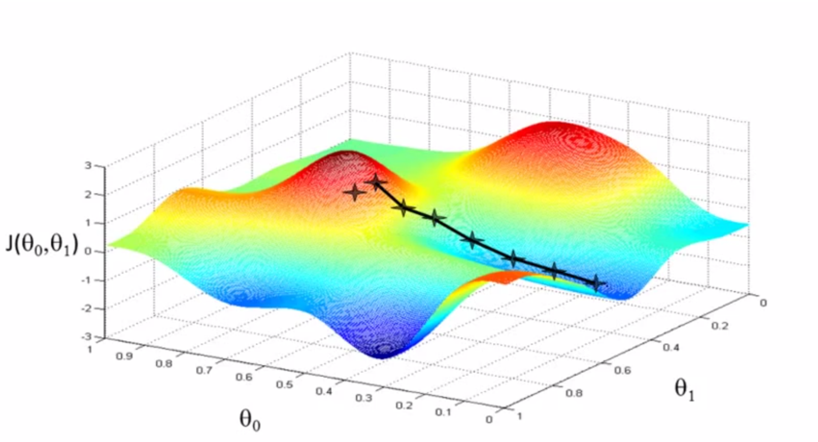
We want to find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

Consider the following *iterative* approach:

1. Start with a random guess of $\boldsymbol{\beta}$. Call it $\boldsymbol{\beta}^{(0)}$.
2. Move $\boldsymbol{\beta}$ in the direction that will decrease L the most to get a new guess $\boldsymbol{\beta}^{(1)}$.
3. From $\boldsymbol{\beta}^{(1)}$, there will be a new direction that decreases L the most. Move in that direction to get a new guess $\boldsymbol{\beta}^{(2)}$.
4. Repeat until it is no longer possible to decrease the value of L.

Numerical Optimization



Gradient Descent

How do we find the direction that will decrease L the most?

Gradient Descent

How do we find the direction that will decrease L the most?

Remember from Calc IV that the **gradient** of L (∇L) points in the direction in which L is *increasing* the most.

Gradient Descent

How do we find the direction that will decrease L the most?

Remember from Calc IV that the **gradient** of L (∇L) points in the direction in which L is *increasing* the most.

$$\nabla L(\boldsymbol{\beta}) = \begin{pmatrix} \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \\ \vdots \\ \frac{\partial L}{\partial \beta_p} \end{pmatrix}.$$

Gradient Descent

How do we find the direction that will decrease L the most?

Remember from Calc IV that the **gradient** of L (∇L) points in the direction in which L is *increasing* the most.

$$\nabla L(\boldsymbol{\beta}) = \begin{pmatrix} \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \\ \vdots \\ \frac{\partial L}{\partial \beta_p} \end{pmatrix}.$$

So we move in the negative gradient direction.

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha \nabla L(\boldsymbol{\beta}^{(k)}),$$

where α (called the **learning rate**) determines how far we move in that direction.

Gradient Descent

How do we find the direction that will decrease L the most?

Remember from Calc IV that the **gradient** of L (∇L) points in the direction in which L is *increasing* the most.

$$\nabla L(\boldsymbol{\beta}) = \begin{pmatrix} \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \\ \vdots \\ \frac{\partial L}{\partial \beta_p} \end{pmatrix}.$$

So we move in the negative gradient direction.

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha \nabla L(\boldsymbol{\beta}^{(k)}),$$

where α (called the **learning rate**) determines how far we move in that direction. This is called **gradient descent** (or **steepest descent**).

Gradient Descent

How do we find the direction that will decrease L the most?

Remember from Calc IV that the **gradient** of L (∇L) points in the direction in which L is *increasing* the most.

$$\nabla L(\boldsymbol{\beta}) = \begin{pmatrix} \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \\ \vdots \\ \frac{\partial L}{\partial \beta_p} \end{pmatrix}.$$

So we move in the negative gradient direction.

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha \nabla L(\boldsymbol{\beta}^{(k)}),$$

where α (called the **learning rate**) determines how far we move in that direction. This is called **gradient descent** (or **steepest descent**).

What's the complexity of each iteration of gradient descent?

Gradient Descent

How do we find the direction that will decrease L the most?

Remember from Calc IV that the **gradient** of L (∇L) points in the direction in which L is *increasing* the most.

$$\nabla L(\boldsymbol{\beta}) = \begin{pmatrix} \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \\ \vdots \\ \frac{\partial L}{\partial \beta_p} \end{pmatrix}.$$

So we move in the negative gradient direction.

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha \nabla L(\boldsymbol{\beta}^{(k)}),$$

where α (called the **learning rate**) determines how far we move in that direction. This is called **gradient descent** (or **steepest descent**).

What's the complexity of each iteration of gradient descent?
Answer: $O(np)$.

Problems with Gradient Descent

Problems with Gradient Descent

- Each iteration is cheap, but how many iterations do we need to converge?

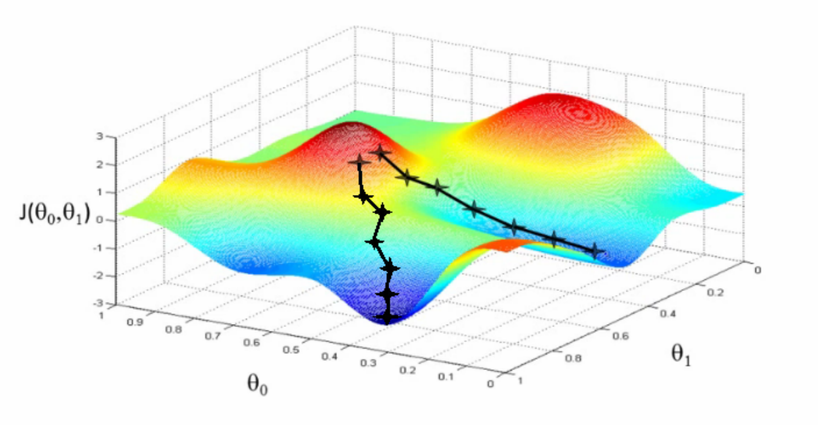
Problems with Gradient Descent

- Each iteration is cheap, but how many iterations do we need to converge?
- How do we choose the learning rate?

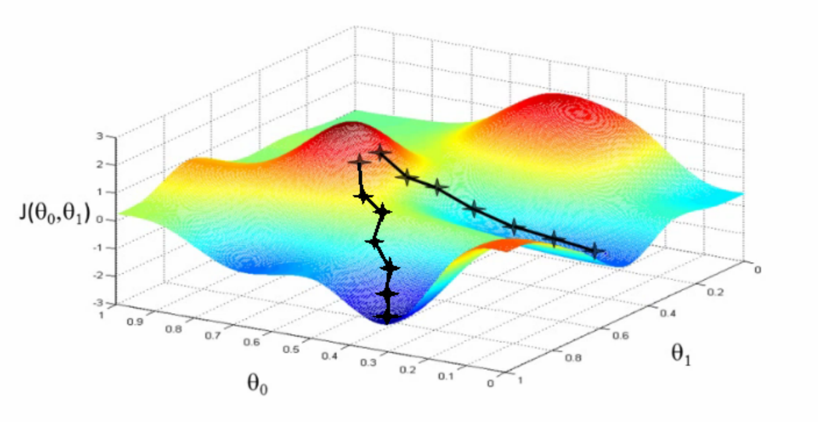
Problems with Gradient Descent

- Each iteration is cheap, but how many iterations do we need to converge?
- How do we choose the learning rate?
- Different starting points can give us different answers.

Problems with Gradient Descent



Problems with Gradient Descent



However, the objective function L for linear regression is **convex**, so it will only have different local minima. So gradient descent is guaranteed to converge to the minimizer (provided you choose α correctly).

The Gradient for Linear Regression

In-Class Exercise

The objective function for linear regression is

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{1i} + ... + \beta_p x_{pi}))^2.$$

Work out the gradient of L. Try to write your answer using linear algebra notation.

The Gradient for Linear Regression

In-Class Exercise

The objective function for linear regression is

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{1i} + ... + \beta_p x_{pi}))^2.$$

Work out the gradient of L. Try to write your answer using linear algebra notation.

$$\nabla L(\boldsymbol{\beta}) = -2X^T(\mathbf{y} - X\boldsymbol{\beta})$$

Implementing Gradient Descent for Linear Regression

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha \nabla L(\boldsymbol{\beta}^{(k)})$$
$$\nabla L(\boldsymbol{\beta}) = -2X^T(\mathbf{y} - X\boldsymbol{\beta})$$

In-Class Exercise

Open the notebook `Gradient Descent for Linear Regression.ipynb`. *Implement the function* `lm_gd`.

1. Review
2. Gradient Descent
3. Stochastic Gradient Descent

Streaming Data

The approach we described above is great for large static data, but what if data is streaming?

Streaming Data

The approach we described above is great for large static data, but what if data is streaming?

As we get more data, the objective function changes:

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

Streaming Data

The approach we described above is great for large static data, but what if data is streaming?

As we get more data, the objective function changes:

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

How do we update the coefficients $\boldsymbol{\beta}$?

Streaming Data

The approach we described above is great for large static data, but what if data is streaming?

As we get more data, the objective function changes:

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

How do we update the coefficients $\boldsymbol{\beta}$? (**Note:** Updates need to be very efficient because the velocity of data may be high!)

Main Idea

Write the gradient as

$$\nabla L(\boldsymbol{\beta}) = 2X^T(\mathbf{y} - X\boldsymbol{\beta})$$

Main Idea

Write the gradient as

$$
\begin{aligned}
\nabla L(\boldsymbol{\beta}) &= 2X^T(\mathbf{y} - X\boldsymbol{\beta}) \\
&= \sum_{i=1}^{n} 2(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))\mathbf{x}_i
\end{aligned}
$$

Main Idea

Write the gradient as

$$
\begin{aligned}
\nabla L(\boldsymbol{\beta}) &= 2X^T(\mathbf{y} - X\boldsymbol{\beta}) \\
&= \sum_{i=1}^{n} 2(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))\mathbf{x}_i \\
&= \sum_{i=1}^{n} \nabla L_i(\boldsymbol{\beta}).
\end{aligned}
$$

Main Idea

Write the gradient as

$$\begin{aligned}\nabla L(\boldsymbol{\beta}) &= 2X^T(\mathbf{y} - X\boldsymbol{\beta}) \\ &= \sum_{i=1}^{n} 2(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))\mathbf{x}_i \\ &= \sum_{i=1}^{n} \nabla L_i(\boldsymbol{\beta}).\end{aligned}$$

In other words, each observation has a contribution to the overall gradient.

Stochastic Gradient Descent

Move only in the direction of the gradient for the current observation:

$$\boldsymbol{\beta}^{(i+1)} = \boldsymbol{\beta}^{(i)} - \alpha \nabla L_i(\boldsymbol{\beta}^{(i)})$$

Stochastic Gradient Descent

Move only in the direction of the gradient for the current observation:

$$\boldsymbol{\beta}^{(i+1)} = \boldsymbol{\beta}^{(i)} - \alpha \nabla L_i(\boldsymbol{\beta}^{(i)})$$

∇L_i will be much noisier, but much less computationally intensive to calculate. (It's only $O(p)$.)

Implementing Gradient Descent for Linear Regression

$$\boldsymbol{\beta}^{(i+1)} = \boldsymbol{\beta}^{(i)} - \alpha \nabla L_i(\boldsymbol{\beta}^{(i)})$$
$$\nabla L_i(\boldsymbol{\beta}) = -2(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))$$

In-Class Exercise

In the notebook `Gradient Descent for Linear Regression.ipynb`, *implement the function* `lm_sgd`.