[POL3 8500] Linear regression and “big data” optimization: gradient descent

Professor Jason Anastasopoulos
ljanastas@uga.edu

University of Georgia

January 26, 2017
For today...

- Linear regression model and least squares.
- Parameter estimation – normal equations.
- Parameter estimation – gradient descent and stochastic gradient descent.
Introduction to linear regression

\[ E(Y|X) = \theta_0 + \theta_1 X_1 + \cdots + \theta_n X_n + \epsilon \]

■ Linear regression model assumes that the regression function is linear in terms of the inputs $X_1, \cdots, X_n$. 
Introduction to linear regression

\[ Y = \theta_0 + \sum_{i=1}^{n} \theta_i X_i + \epsilon \]

\[ \epsilon \sim N(0, 1) \]

In other words the output \( Y \) can be thought of as a linear function of the inputs \( X \)
Introduction to linear regression

\[
\% \text{ Republican}_s^g = \theta_0 + \sum_{i=1}^{n} \theta_i X_i + \epsilon
\]

In political science or other social science disciplines, you might encounter linear regression in models that attempt to predict elections.
Why learn about linear regression?

- It’s uncommon to see linear regression applied in the machine learning context **BUT**...
- many nonlinear techniques (support vector machines, neural networks) are simply generalizations of linear regression.
Linear regression and least squares

\[ Y = f(X) + \epsilon \]
\[ f(X) = \theta_0 + \sum_{i=1}^{n} \theta_i X_i \]

- Recall from last time that the goal in machine learning is to find a function \( f(\cdot) \) that does the best job of predicting outcomes.
- A **linear regression model** is a parametric model in which we assume that the inputs/predictors \( X \) are a linear function of \( Y \).
Linear regression and least squares

\[ Y = f(X) + \epsilon \]

\[ f(X) = \theta_0 + \sum_{i=1}^{n} \theta_i X_i \]

- In linear regression models the output \( Y \) is usually a continuous outcome and the inputs \( X \) can be quantitative or qualitative (dummy variables etc).
Linear regression and least squares

\[
\% \text{Rep.}_t^g = \theta_0 + \theta_1 \% \text{Rep.}_{t-1}^p \\
+ \theta_2 \text{Rep. Governor}_{t-1} + \theta_3 \text{Region} + \epsilon
\]

\% \text{Rep.}_t^g – is the Republican vote share for the president in the general election.

\text{Rep. Governor}_{t-1} – is a dummy variable indicating whether the state had a Republican governor in \( t – 1 \).

etc...
Linear regression and least squares

\[ Y = \theta_0 + \theta_1 X_1 + \theta_2 X_1^2 + \theta_3 X_1^3 + \cdots \]

- X’s can also be basis expansions, leading to an \( n^{th} \) order polynomial.
- What’s important however is that the model is linear in the \( \theta \)'s.
Linear regression and least squares

Training data: \((x_1, y_1), \ldots, (x_N, y_N)\)

\[ x_i = (x_{i1}, \ldots, x_{ip})^T \]

Consider some training data with \(N\) observations and \(p\) features.
Goal is to find $\theta$’s which minimize loss or cost function.

Goal is to estimate: $\theta = (\theta_0, \cdots, \theta_p)^T$

such that: $\text{arg min} \ J(\theta)$

Our goal is to find a set of $\theta$’s that minimize the cost function $J(\theta)$ or residual sum of squares (RSS) in the language of regression analysis.
Goal is to find $\theta$’s which minimize loss or cost function

Where...

$$J(\theta) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

$$= \sum_{i=1}^{N} \left( y_i - \theta_0 - \sum_{j=1}^{p} x_{ij} \theta_j \right)^2$$
Goal is to find $\theta$’s which minimize loss or cost function

Where...

$$J(\theta) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

$$= \sum_{i=1}^{N} \left( y_i - \theta_0 - \sum_{j=1}^{p} x_{ij}\theta_j \right)^2$$

In words, we want to choose $\theta$’s which minimize the squared difference between the observed and predicted values.
In the single feature case we need to solve:

\[
\frac{\partial J}{\partial \theta_0} = 0 \\
\frac{\partial J}{\partial \theta_1} = 0
\]
Fitting a line with slope $\theta_1$ and y-intercept $\theta_0$
Starting with $\theta_0$ we have:

$$\sum \frac{\partial J}{\partial \theta_0} (y_i - \theta_0 - \theta_1 x_1)^2 = 0$$

$$\Rightarrow \theta_0 = \bar{y} - \theta_1 \bar{x}$$

(Proof on the board)
Is $\theta_0$ a minimum?

$$\frac{\partial^2 J}{\partial \theta_0^2} = 2N > 0$$

- Second derivative test guarantees that $\theta_0$ is a local minimum when we set the first derivative to 0 and solve for $\theta_0$. 
For $\theta_1$ we have:

$$\theta_1 = \frac{\sum y_i \sum x_i - N \sum x_i y_i}{(\sum x_i)^2 - N \sum x_i}$$

Proof will be part of problem set 1.
When we have more features we fit a plane or hyperplane to the data.
What if we have multiple parameters?

\[ J(\theta) = (y - X\theta)^T(y - X\theta) \]

\[ X \in \mathbb{R}^{nxp}, \theta \in \mathbb{R}^{nx1}, y \in \mathbb{R}^{nx1} \]
Need to find...

\[ \frac{\partial J}{\partial \theta} = 0 \]
\[ \frac{\partial J}{\partial \theta} = -2X^T(y - X\hat{\beta}) \]
\[ \hat{\theta} = (X^TX)^{-1}X^Ty \]

Proof will be part of problem set 1.
Interpretation and prediction

\[ \hat{\theta} = (X^T X)^{-1} X^T y \]

- Once we estimate \( \theta \) we might be interested both in *interpretations* of the parameters of the model and *prediction* using the model.
Interpretation

- Which features/feature sets do the best job of predicting $y$ and why?
Interpretation

To do so we need to:

1. Learn about the variance of $\theta$ and;
2. Make certain distributional and functional assumptions about the data and the errors.
Variance of $\theta$

- It is useful to learn about the variance of $\theta$ because once we know this, we can conduct hypothesis tests for the parameters.
Assumptions for inference

- Need to assume that the linear model is indeed appropriate.
- That the error term has a mean of 0 and variance of 1.
- That the $\hat{\theta}$ follow a multivariate normal distribution with a mean equal to the true $\theta$ and variance equal to the estimated variance.
Hypothesis testing

For each parameter, we might be interested in testing whether it contributes anything to the model (whether the true value is 0).

To accomplish this we can do hypothesis testing with the $t$ distribution or the normal distribution.
Prediction

- More often, we are interested in prediction and interpretability.
Prediction

- Remember, adding more features will always reduce the training error but will also reduce both interpretability AND tend to increase test error due to overfitting.
Model and variable selection

- In order to strike a good balance between the two, we can use the $F - test$ for two competing models or;
- Subset selection algorithms to choose which group of predictors should be included and excluded from the model.
F-Test

- Two models, one simple and one more complex.
- F-test will tell you whether the more complex model is an improvement over the simple one.
F-Test Example

Model 1: \( \text{Rep}_s^g = \theta_0 + \theta_1 \text{Rep}_s^p + \theta_2 \text{Pop}_s + \epsilon \)
Model 0: \( \text{Rep}_s^g = \theta_0 + \theta_1 \text{Rep}_s^p + \epsilon \)

- Two models predicting Republican presidential vote share.
- Model 1 includes primary vote share and population of the state.
- Model 0 includes on the primary vote share.
- Which model should we use?
F-Test Example

\[ H_0 : \text{Model 0 is better} \]

\[
F = \frac{(J(\theta)_0 - J(\theta)_1)/(p_1 - p_0)}{J(\theta)_1/(N - p_1 - 1)}
\]

\[ p_1 = \text{parameters in Model 1.} \]
\[ p_2 = \text{parameters in Model 2.} \]
Reject $H_0$ if...

\[ P(X > F_{p_1-p_0,N-p_1-1}) > 0.05 \]

$p_1 = \text{parameters in Model 1.}$

$p_2 = \text{parameters in Model 2.}$
Often times in machine learning problems, the matrix $X^TX$ is very sparse in the case of linear regression or;

In the case of non–linear models (neural networks etc), there is no closed form solution available to estimate the parameters.
In these cases iterative methods such as gradient descent or stochastic gradient descent can estimate model parameters much more quickly than normal equations.
Gradient descent

\[
\theta : \theta - \eta \nabla_\theta J(\theta)
\]

\[
\nabla_\theta J(\theta) = \left[ \frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}, \ldots, \frac{\partial J}{\partial \theta_p} \right]
\]

- Gradient descent is an algorithm which starts with an initial guess for the \(\theta\)'s, calculates the cost function and updates \(\theta\) in the direction of the gradient.
- \(-\nabla_\theta J(\theta)\) is the direction of *steepest descent* of the cost function \(J(\theta)\).
- \(\eta\) is the *step size*.

**Professor Jason Anastasopoulos**  
[janas@uga.edu]  
**University of Georgia**

[POLS 8500] Linear regression and “big data” optimization: gradient descent
Thus $-\eta \nabla_{\theta} J(\theta)$ is taking a step of size $\eta$ down in the direction of steepest descent.
For univariate regression

repeat while (\| \eta \nabla J(\theta) \| > \epsilon) 
{
\theta_j := \theta_j - \eta \frac{\partial J}{\partial \theta_j} J(\theta_0, \theta_1)
}

Professor Jason Anastasopoulos ljanastas@uga.edu
University of Georgia

[POLS 8500] Linear regression and “big data” optimization: gradient descent
For univariate regression

repeat while ($\|\eta \nabla J(\theta)\| > \epsilon$)
{
\[\theta_0 := \theta_0 - \eta \frac{\partial J}{\partial \theta_0} J(\theta_0, \theta_1)\]
\[\theta_1 := \theta_1 - \eta \frac{\partial J}{\partial \theta_1} J(\theta_0, \theta_1)\]
}
Gradient descent

For univariate regression

\[
\frac{\partial J}{\partial \theta_0} = \frac{1}{N} \sum_{i=1}^{N} (\theta_0 + \theta_1 x_i - y_i)
\]

\[
\frac{\partial J}{\partial \theta_1} = \frac{1}{N} \sum_{i=1}^{N} (\theta_0 + \theta_1 x_i - y_i) x_i
\]
For univariate regression

repeat while \( \| \eta \nabla J(\theta) \| > \epsilon \)
\{
\begin{align*}
\theta_0 & := \theta_0 - \eta \frac{1}{N} \sum_{i=1}^{N} (\theta_0 + \theta_1 x_i - y_i) \\
\theta_1 & := \theta_1 - \eta \frac{1}{N} \sum_{i=1}^{N} (\theta_0 + \theta_1 x_i - y_i) x_i
\end{align*}
\}
Because the cost function is convex it will have a global minimum
Gradient descent will fit the best least squares line
For multivariate regression

\[
\theta = [\theta_0, \cdots, \theta_p] \\
y = [y_1, \cdots, y_N] \\
X \in \mathbb{R}^{N \times (p+1)}
\]

\[
\text{repeat while } (||\eta\nabla J(\theta)|| > \epsilon) \{
\theta := \theta - \eta \nabla J(\theta)
\}
\]
For multivariate regression

\[ \nabla J(\theta) = \frac{1}{N} (y^T - \theta X^T)X \]

repeat while (\[|\eta \nabla J(\theta)|\] > \(\epsilon\))

\{ \[
\theta := \theta - \eta \frac{1}{N} (y^T - \theta X^T)X
\}
For next week...

- HW 1 will be out.
- Stochastic gradient descent.
- Linear model selection.
- Cross-validation.
- Regularization/shrinkage methods.