Lec Wed

Wednesday, October 24, 2018 4:02 PM

(Brief Review of Logistic Regression)

Today: Linear Regression

- We'll talk about least squares regression
- predicting a continuous outcome variable
  - eg shoe size from height, weight, and gender
  - eg stock price using its profit and other financial info
- we can measure "closeness" of prediction in different ways
  - each way will produce different models and errors

eg Housing Prices

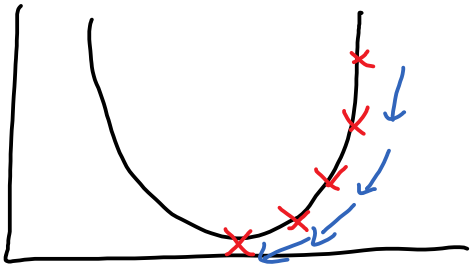
- features might include square feet
- could evaluate quality of prediction with
  - absolute difference | prediction - sale price |
  - squared difference (prediction - sale price)^2

our model looks like $y = w_1x_1 + w_2x_2 + \ldots$
where we have to train the weight vector **w**

What is the best weight vector?
We define the cost of a weight vector to be:

$$\frac{1}{2}\sum_{i=1}^{m}\left(y_i - w^T x_i\right)^2$$

How do we minimize the cost?
We'll use gradient descent
We start at a random point and ask for a direction (using the gradient of the curve). Then we take a step in that direction until we get to our destination (where the gradient is 0).



If the function is convex, we're guaranteed to find the global minimum (assuming we have a good step size).
If not, we might just get stuck in a local optimum.
How do we determine if the function is convex?
f(x) is convex if $f''(x) \geq 0$

Okay but we usually have more than 1 feature, meaning we have a multi-variate function
so we can use a Matrix of second-order derivatives (the Hessian)

If the Hessian is positive semi-definite $(H \geq 0\ ), then\ f\ is\ convex$
Ex:

1. $f(x) = x_1^2 + 2x_2^2$

$$\begin{matrix} 2 & 0 \\ 0 & 4 \end{matrix}$$

Gradient Descent Algorithm:

- Initialize weights
- compute gradient of cost
- shift values of weights in direction of gradient by step size/learning rate

Remarks:

- η is called the step size or learning rate - how far the update will go in the direction of the negative gradient
- with a good choice of learning rate, the iterative procedure converges to a stationary point where $\frac{\partial f}{\partial \theta} = 0$
- if learning rate is too large, we will overstep the minimum
- if learning rate is too small, we approach the minimum too slowly
- how to choose learning rate?
    - can use cross-validation, try different learning rates

Incremental/Stochastic Gradient Descent

- repeat for each example
    - pretend the entire training set is represented by this single example
    - use this example to calculate the gradient and update the model
- why does this work? we approximate the true gradient by a gradient at a single example at a time
- this will take more steps and more steps in random directions, but eventually we can get there
- analogous to perceptron algorithm

Batch Gradient Descent

- one update to the weight vector for every pass over the data