Linear regression multiple variables

Let us say that you have data of this kind

SquareFt	Bedrooms	Bathrooms	Price
1000	3	2	$350,000
3000	5	5	$765,000
800	1	1	$320,000
2100	4	3	$540,000
2200	3	4	$520,000

Notations

- n number of features (3)
- Input data $\boldsymbol{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots x_n^{(i)}]$
  - The features of the i-th training example
- Output data y
- Training set size m
- This is a **regression** problem - we are trying to predict a floating point number.

The hypothesis function

- It was:

$$f(x, \boldsymbol{\theta}) = \theta_1 x + \theta_0$$

- Now it will be, for $\boldsymbol{x} = [x_1, x_2, \ldots x_n]$

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \ldots \theta_n x_n$$

- For the convenience of notation, we can say $x_0 = 1$

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{x}$$

Gradient descent in multiple variables

- Wait, we already did this: previously $\boldsymbol{\theta}$ had two components
- Now it has $n + 1$ components. Nothing else changes.

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{\partial \mathcal{L}(\theta_0, \ldots, \theta_n)}{\partial \theta_0}$$

$$\theta_1 \leftarrow \theta_1 - \alpha \frac{\partial \mathcal{L}(\theta_0, \ldots, \theta_n)}{\partial \theta_1}$$

$$\theta_2 \leftarrow \theta_2 - \alpha \frac{\partial \mathcal{L}(\theta_0, \ldots, \theta_n)}{\partial \theta_2}$$

$$\ldots$$

Tips and tricks

Feature scaling and mean normalization

- Bedrooms change on the range of 1-5
- Square feet change on the range of 800 - 5000
- All the surfaces will be elongated, which makes gradient descent either unstable or slow.
- Solution:
    - **Feature scaling**: get every feature to the range of approximately $-1 \leq x_i \leq 1$
    - **Mean normalization**: get the feature to approximately zero mean
      $x_i \leftarrow x_i - \mu_i$
    - Do not apply it to $x_0 = 1$
- Remember the transformations for the test data!!!

Learning rate

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial \mathcal{L}(\theta_0, \ldots, \theta_n)}{\partial \theta_i}$$

- How to choose α?

How do you know it is done?

- Visual inspection of the learning curve: plot $\mathcal{L}$ wrt iterations or epochs
  - See if it flattens out
- Automatic convergence test: declare convergence if $\mathcal{L}$ decreases by less than 10^-03 (or something) in one iteration

Tweaking α

- For a sufficiently small α, the loss should decrease at every iteration
- But if α is too small, it will be slow to converge
- If it α is too big, it will be unstable, diverge, or cycle

Feature engineering, polynomial regression etc.

- **Linear** regression, assumes that there is a linear relationship between the **features** x_i **and parameters** θ_i
- But in many applications, the relationship between the regression output and input features is not linear
  - House width x_1
  - House length x_2
- But usually, the price is linearly proportional to area, so we create $x_3 = x_1 x_2$
- Similarly we can create features by squaring, taking square root, taking the log, taking the exponential etc.

Polynomial regression

Take

$$f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_n x^n$$

Rest is like linear regression.

Solving linear regression with a normal equation

- If your loss function is least squares, then you can find the optimal $\boldsymbol{\theta}$ analytically:

$$\boldsymbol{\theta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- $\boldsymbol{X}$ and $\boldsymbol{y}$ are all the training data stuck on top of each other
- This is one line in numpy.
- Why do we bother with gradient descent?

Gradient descent vs normal equation

- Normal equation
  - No need to choose α. Analytical. One shot.
  - You need to compute $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ an $n \times n$ matrix, with n number of features. It is an $O(n^3)$ operation
  - Only works for the least squares loss.
- Gradient descent
  - You need to choose α. Iterative.
  - Works for almost every loss.
  - Works with large n

What if $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ gives an error?

- If $\boldsymbol{X}^T\boldsymbol{X}$ is not invertible, it means that the features are linearly dependent
- Eg. you used
  - heated area
  - unheated area
  - total area