NUS Fintech Society

Machine Learning
Linear Regression

Content

- Intro to Machine Learning
- Supervised Learning: Linear Regression
- Gradient Descent
- Normalisation
- Regularisation
- Intro to k-NN

Intro to Machine Learning

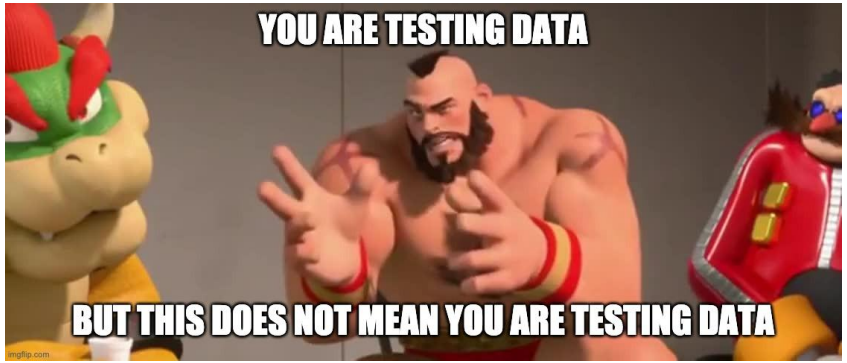
What is Machine Learning?

- A computer program said to learn from experience *E* with respect to task *T* and performance measure *P* if its performance at task *T*, as measured by *P*, increases with experience *E*.
- Examples
  - An algorithm that predicts housing prices based on factors such as floor size (T) is fed with data comprising of recent house sale transactions (E), leading to an improvement in accuracy of prediction (P).
  - An algorithm that finds patterns in data comprising of customer spending. The algorithm could segment the customers into segments based on age and spending. This would help businesses formulate targeted advertising campaigns to focus on each group rather than engaging in inefficient “one-size fits all” marketing.
- Machine Learning is of two types
  - Supervised Learning
  - Unsupervised Learning

Supervised vs Unsupervised Learning

- Supervised learning: Learning that involves finding a mapping for select inputs to provide an output. The algorithm is essentially fed with the "right answer".
  - Housing price prediction
  - Can be used for stock-price prediction
  - Many algorithms → We will look at *linear regression* today
- Unsupervised learning: Learning that involves finding "clusters" of data-points from a large volume of data → not covered today
  - Finding patterns in data about tumours
  - Finding patterns in data about consumers
  - Can be used for stock price prediction

Training vs Testing Data

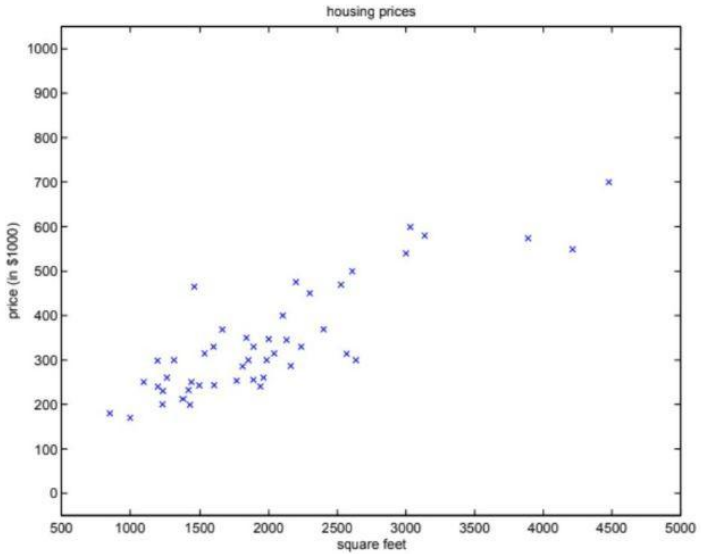
- Training Data
  - The data we learn from
- Testing Data
  - The data upon which we evaluate our learning
- Validation Data
  - 

Explore!

Linear Regression

Linear Regression

- Objective: Predict the housing prices (Target Variable) using input variables (e.g. Floor Size)



Linear Regression

- Question: What exactly do we need to have in order to be able to “predict”?
  - Predict: given a new, unseen input, tell us an output that we think is most likely
  - Hence, we need to identify the relationship between inputs and outputs.
  - In a more mathematical sense, we are given (x, y) where x is input and y is output, and find a function f such that f(x) = y
  - This is what is meant by “learning a mapping”
  - In ML parlance it’s usually called “model” instead of function
- Sounds easy enough?

Linear Regression

- New Question: What mapping to learn?
  - So many functions we can use:

$$y = mx + c \qquad y = A\sin(kx) + c$$
$$y = ax^2 + bx + c \qquad y = Ae^{kx}$$

- Answer: Trial and Error (with some heuristics)
  - Guess a good model
  - Optimize that model to find the best coefficients
  - See how badly it predicts the output we have

Linear Regression

- New Question: What mapping to learn?
  - So many functions we can use:

$$y = mx + c \qquad y = A\sin(kx) + c$$
$$y = ax^2 + bx + c \qquad y = Ae^{kx}$$

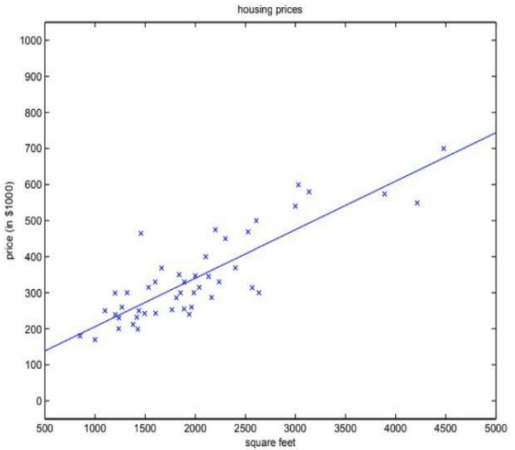
- Answer: Trial and Error (with some heuristics)
  - Guess a good model
  - Optimize that model to find the best coefficients
  - See how badly it predicts the output we have

Linear Regression

- Approach:
  - Guess a good function (Linear seems good here)
  - Optimize that function to find the best coefficients
- What is “best” coefficients ?
  - Either it minimizes some sense of error, or
  - it maximizes some sense of success
- We pick the first approach, because error is easy to quantify (difference between prediction and actual output)

Linear Regression

- Root Mean Square (RMS) Error:
  - Sum of the square of the distances between the actual points and the prediction/model on the graph
  - Distance is squared to ensure that under and over estimation are both equally weighted.
  - Whether the perpendicular distance to the line should be taken or the vertical distance is a subject of debate.
    For the sake of simplicity, we take the vertical distance.



Linear Regression

Approach:

- Guess a good function
- Optimize that function to find the best coefficients

Advantages	Disadvantages
Simple	What if we pick a function that is not good: either too complicated or too simple for the data (overfitting or underfitting problem)
Understandable	What if there are a significant number of outliers?

Bias vs Variance

- Bias error: error from erroneous assumptions in the learning algorithm.
  - Like learning the wrong topic for an exam
  - Underfitting is a problem due to high bias error → assumption is that the relationship to be modelled is simpler than it actually is
- Variance error: error from sensitivity to small fluctuations in the training data
  - Like memorizing answers according to the question AND question number
  - Overfitting is a problem due to high variance error → instead of modelling just the relationship, useless stuff (random noise) is also modelled
- There is thus a need to balance between bias and variance (Bias-Variance Trade off)

Linear Regression

- Approach:
  - Guess a good function (Linear seems good here)
  - Optimize that function to find the best coefficients (Methods to do so include gradient descent, normal
- What if we pick $y = Ae^{kx}$ instead as the function? Is it still linear regression?
- Answer: Yes (left as exercise as to why)
  - What if we pick $y = A\sin(kx)$ instead as the function? Is it still linear regression?
  - What are some common functions that we can use for regression?

Linear Regression

- Case study: We are looking at the stocks of a particular ETF, let's say alternative energy sources. Say we are looking to purchase these stocks. Say we have monthly data of revenues, along with data with regards to humidity, wind speeds, sunlight, oil production, oil prices, and a host of other features. We have a dependent variable, and several independent variables. This can be modelled into a regression problem.

Feature 1	Feature 2	Feature 3	Revenue
x^1_1	x^1_2	x^1_3	y^1
x^2_1	x^2_2	x^2_3	y^2
x^3_1	x^3_2	x^3_3	y^3
x^4_1	x^4_2	x^4_3	y^4

Linear Regression

- What the equation might look like: -
  - $y^i = \Theta_0 + \Theta_1 \cdot x_1^i + \Theta_2 \cdot x_2^i + \Theta_3 \cdot x_3^i$
  - $y^i = \Theta_0 + \Theta_1 \cdot (x_1^i)^2 + \Theta_2 \cdot (x_2^i)^2 + \Theta_3 \cdot (x_3^i)^2$
  - $y^i = \Theta_0 + \Theta_1 \cdot (x_1^i)^2 \cdot x_2^i + \Theta_2 \cdot (x_2^i)^3 + \Theta_3 \cdot (x_3^i)^{\frac{1}{2}}$
  - $y^i = \Theta_0 + \Theta_1 \cdot (x_1^i)^2 \cdot x_2^i + \Theta_2 \cdot \cos(x_2^i)^3 + \Theta_3 \cdot \sin(x_3^i)^{\frac{1}{2}}$
- There are two issues here
  - Which model do we choose?
  - How do we find the optimal numbers of theta?
- Root Mean Square (RMS)

Cost Function

- The model we will use
  - $h_\Theta(x^i) = \Theta_0 + \Theta_1 \cdot x_1^i + \Theta_2 \cdot x_2^i + \Theta_3 \cdot x_3^i$
- The error term for each “prediction” for a certain value of theta
  - $\delta^i = y^i - h_\Theta(x^i)$
- Cost function: The cost function sums the squared error of the and averages the cost out
  - $J(\Theta) = \sum_{i=1}^{n} (y^i - (h_\Theta(x^i))^2$
  - $J(\Theta) = \frac{1}{2n} \sum_{i=1}^{n} (y^i - (h_\Theta(x^i))^2$

Cost Function

- Cost Function
  - $J(\Theta) = \frac{1}{2n}\sum_{i=1}^{n}(y^i - (h_\Theta(x^i))^2$
- The goal is to minimise the cost function, which can be achieved using calculus.
  - $\Theta_j = \Theta_j - \alpha \cdot \frac{\partial}{\partial \Theta_j} J(\Theta)$
  - $\Theta_j = \Theta_j - \alpha \cdot \frac{\partial}{\partial \Theta_j} (\frac{1}{2n}\sum_{i=1}^{n}(y^i - (h_\Theta(x^i))^2)$
- Alpha represents the learning rate of the model. It is always positive.

Cost Function

- This updating must be done simultaneously, for all Θ_j for all values of j in range 1 to m, with m being the number of features we have in our model.
- $$\Theta_0 = \Theta_0 - \alpha \cdot \frac{\partial}{\partial \Theta_0}(\frac{1}{2n}\sum_{i=1}^{n}(y^i - (h_\Theta(x^i))^2)$$

  $$\Theta_1 = \Theta_1 - \alpha \cdot \frac{\partial}{\partial \Theta_1}(\frac{1}{2n}\sum_{i=1}^{n}(y^i - (h_\Theta(x^i))^2)$$

  $$\Theta_2 = \Theta_2 - \alpha \cdot \frac{\partial}{\partial \Theta_2}(\frac{1}{2n}\sum_{i=1}^{n}(y^i - (h_\Theta(x^i))^2)$$

  $$\Theta_3 = \Theta_3 - \alpha \cdot \frac{\partial}{\partial \Theta_3}(\frac{1}{2n}\sum_{i=1}^{n}(y^i - (h_\Theta(x^i))^2)$$

Cost Function

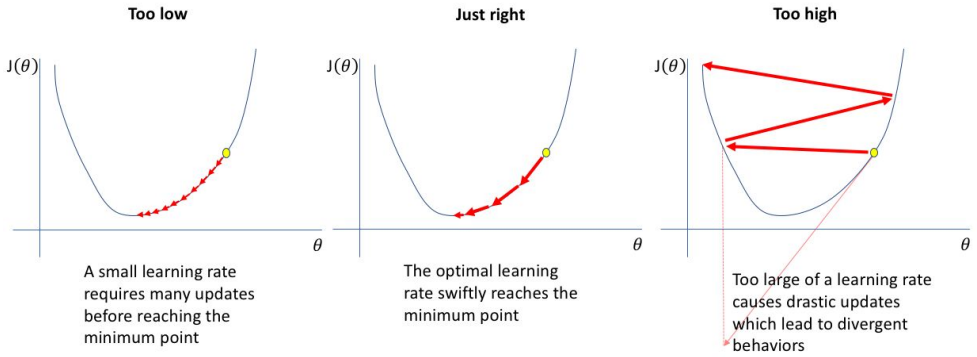
- $\Theta_j = \Theta_j - \alpha \cdot \frac{\partial}{\partial \Theta_j}(\frac{1}{2n}\sum_{k=1}^{n}(y^i - (h_\Theta(x^i))^2)$
- By using calculus we can show that the partial derivative with respect to Θ_j gives the above equation
  - $\Theta_j = \Theta_j - \alpha \cdot \frac{1}{n}\sum_{i=1}^{n}((y^i - h_\Theta(x^i)) \cdot x_j^i)$
- For x_0^i it the term for any value of i is 1.
- This iterative process of finding the optimal values of Θ_j is called gradient descent.

Gradient Descent

- What is “best” coefficients? The set of coefficients that minimises RMSE
- We can find this set of best coefficients by the *gradient descent algorithm*
  - Suppose you are on a mountain. You want to climb to the bottom of the mountain. But, the mist around you is so thick that you cannot see anything i.e. essentially blind. What will you do?
    1. Hold your hand out in front and touch the ground.
    2. Feel the slope of where you are now i.e. which direction seems to be pointing downwards
    3. Once you identify the direction that feels like it is going down, take a few steps in that direction
    4. Stop, and Repeat steps 1 to 3 again
    5. You have reached the bottom if, when you hold your hand out and feel the ground around you, everything feels flat i.e. no more direction that is pointing downwards.

Gradient Descent

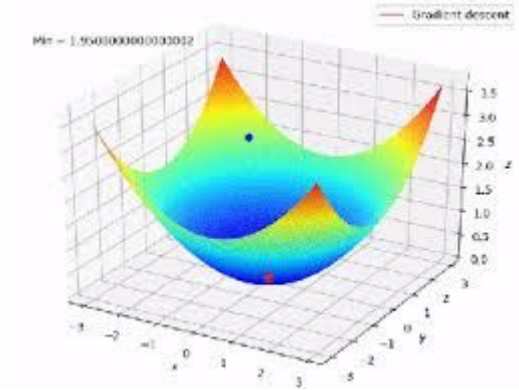
- So how does the gradient descent equation work?
- Recall that the manner in which we descent the “hill” is as follows
  - $\Theta_j = \Theta_j - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} ((y^i - h_\Theta(x^i)) \cdot x_j^i)$
- Recall that alpha is a positive constant known as the learning rate.

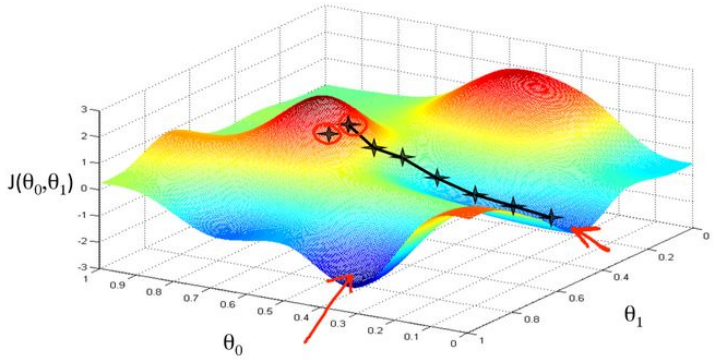


source: https://www.jeremyjordan.me/nn-learning-rate/

Visualisation of Gradient Descent

- So how does gradient descent look visually?
- We can only visualise gradient descent in 2 or 3 dimensions.
- Hence, let us assume that there is one feature x_1^i determining the output of some supervised learning problem.
- The equation of the hypothesis would be
  - $h_\Theta(x^i) = \Theta_0 \cdot (1) + \Theta_1 \cdot x_1^i$
  - $h_\Theta(x^i) = \Theta_0 \cdot x_0^i + \Theta_1 \cdot x_1^i$





Gradient Descent Factors

- The main factors that influence whether Gradient Descent ends with a local minima or a global minima are: -
  - Learning Rate
  - Starting Values of Parameters

Explore!

Gradient Descent

Analogy	Gradient Descent
Feel the slope of where you are now i.e. which direction seems to be pointing downwards	Get derivative of the function with respect to the parameters at the current parameter
Once you identify the direction that feels like it is going down, take a few steps in that direction	Update current parameters by subtracting it with a small multiple of the derivative

See Jupyter Notebook for more details on how this is actually done

Final notes about gradient descent

- Gradient descent can take on multiple types
  - Batch gradient descent
  - Stochastic gradient descent
  - Mini Batch gradient descent
- If you have a large number of features, you might find better ways of optimising your parameters.
  - Normal Equation: $\Theta = (X^T X)^{-1} X^T Y$
  - L-BFGS
  - Levenberg-Marquardt Algorithm (LMA)
- Keep in mind that your starting point, your learning rate, and your feature normalisation all determine how effective and efficient your gradient descent is.

Normalisation

What is Normalisation?

- The practice of scaling your features down or up to a certain range.
- What does it help with?
  - Reaching the global optimum faster.
  - Reduces the risk of being stuck in a local optimum.
- How do we normalise the data?
  - Subtract the mean or the min of the data from each of the data points.
  - Divide each data point by the *(max-min)* of the data points.
- What do we normalise the data to, exactly?
  - https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
  - https://towardsdatascience.com/clearly-explained-what-why-and-how-of-feature-scaling-normalization-standardization-e9207042d971

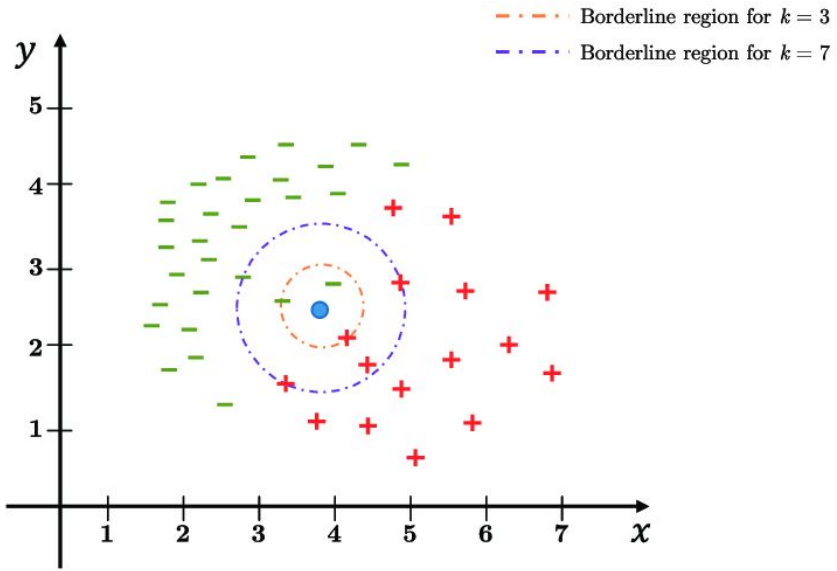
Regularization

What is Regularization?

- The practice of shrinking your coefficients by adding a term to your cost function.
- This term might look something like
  - $\sum_{j=1}^{n} \Theta_j$
- Goal: To minimise cost function. How do we achieve this?
- Reducing the size of Theta values!
- Why do we do this?
  - Ensure the model is not overfitted to the data
  - The amount by which we shrink the coefficients can be modulated using a parameter lamda λ

Explore!

k-NN Model


Borderline region for $k = 3$
Borderline region for $k = 7$
y
5
4
3
2
1
1
2
3
4
5
6
7
x

k-NN Model

- An interesting model that doesn't do very much 'learning' at all!
- The model almost "memorizes" data of training examples.
- Input data-point is measured for numerical similarity with training data-points.
- Exploration:
  - Where might k-NN fall short?
  - What do k-NN decision boundaries look like with different values of k?

Explore!

Summary

Summary

- Intro to Machine Learning:
  - Supervised Learning vs Unsupervised Learning
  - Linear Regression
- Optimisation of models: Gradient Descent Algorithm
- Normalisation

Conclusion

- We’re done for today, but…
- There are a few things we would like to say before we go.
- Today was more of a theoretical discussion about the fundamentals of regression and classification models. To really get into stock price predictions, you should read more about neural networks. These build on the concepts we discussed in this presentation, and provide for multiple layers of learning, and as a result, more accurate models.

Thank You!