Introduction

15 November 2020 01:21 PM

Genera definition

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E

-Applications

Email classifying
Medical diagnostics
Recommendation Systems
Automation in robotics

Types of ML algorithms

-Supervised Learning

The computer is programmed to learn from specific dataset and a given model
Eg: we first state the that a particular variable depend on other variables say linearly then the constants of that linear equation are to be learned by the computer.
Basically we fit a curve to the given model.
The data set already contains the labels (answers)
Predicts on examples other than dataset

-Unsupervised learning

It is used mainly for clustering population in different groups
The dataset is unlabelled (no answers given)
predicts on the dataset

Linear Regression

18 November 2020 12:06 PM

Linear regression

- -one variable
  - We are required to produce a function that gives one variable output given a single input
  - We assume a "hypothesis function" as y= $\theta_1 + \theta_2$X
  - The we basically fit the parameters θ_1 & θ_2 using the given data set by minimizing the squared error function "cost function".
- -multi
  - We are required to produce a function that gives one variable output given a multi variable input.
  - We assume a "hypothesis function" as y= $\theta_1 + \theta_2 X_2 + \theta_3 X_3 + \theta_4 X_4 + \theta_5 X_5 + \theta_6 X_6 + ...$
  - The we basically fit the parameters $\theta_1, \theta_2 ......$ using the given data set by minimizing the squared error function "cost function".

Minimizing Cost Function

- -Gradient descent
  - An algorithm to compute the minima of a function, required that there is only one global minima also called batch gradient descent because Each step of gradient descent uses all the training examples.
- -pseudo code

```
Repeat until convergence{
    For all j{
        $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_1, \theta_{2...})$
    }
}
}
```

$$\frac{\partial}{\partial \theta_j} J(\theta_1, \theta_{2...}) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

- -Learning rate α
  - If too small gradient descent is slow
  - If too big gradient descent may not converge or even may diverge
  - In order to make sure that our choice is correct we may plot a graph of cost function vs no of iteration
    - -if J(θ) decreses quickly and saturates its ok
    - -if J(θ) blows up or osscilates α *is high*

Multivariable linear regression matrix implementation

$h_\theta(x) = \theta_1 + \theta_2 X_2 + \theta_3 X_3 + \theta_4 X_4 + \theta_5 X_5 + \theta_6 X_6 + ...$

$= \theta^T X$

Where $\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ *and* $X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ *and* $x_1 = 1$

X is also called feature matrix

Also $y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$ *and* $X =$

Update rule becomes

$\theta = \theta - \left(\frac{\alpha}{m}\right) * (X^T * \left(\left(X * \theta - y\right)\right))$ *in matrix notation*

$J(\theta) = \frac{1}{2m} sum((X\theta\text{-}y).\text{^}2)$

Feature scaling

If the different features we use are not scaled to a common limit we end up with a very complex cost function which would take gradient descent more time to come to a minima.

Following types of scaling may be used.

-Andrews idea

Divide the features by the highest among them it will make every feature in the range [-1,1]

-mean normalization

Subtract a feature f by its mean the divide the feature by its range

$feature\ (f) = \frac{f - mean\ of\ all\ f}{largest\ of\ all\ f\ -\ smallest\ of\ all\ f}$

Sometimes slandered

.. deviations are used in place of range

Polynomial regression

- If a linear hypothesis does not fit our data we may choose any mathematical operation eg square, roots, or any thing.
- We may chose $h_\theta(x) = \theta_1 + \theta_2 x_2^2 + \theta_3\sqrt{x_3} +$.......
- The we would make new feature as say $x_4 = x_2^2$ *and* $x_5 = \sqrt{x_3}$ *and the the equation becomes*
- $h_\theta(x) = \theta_1 + \theta_2 X_4 + \theta_3 X_5 +$.......
- The we proceed as linear regression.

Normal equation

Upon solving for the minima of J(θ) we get the following result

$\theta = (X^T X)^{-1} X^T y$ *here the inverse is a psudo inverse*

Comparison table

Gradient descent	Normal equation
Need to choose α	no need to choose α
Needs many iterations	no need to iterate
	need to compute $(X^T X)^{-1}$
works well even when n is large	slow if any n is large

Logistic Regression

18 November 2020 12:20 PM

Definition

logistic regression basically deals with classification problem.

hypothesis function

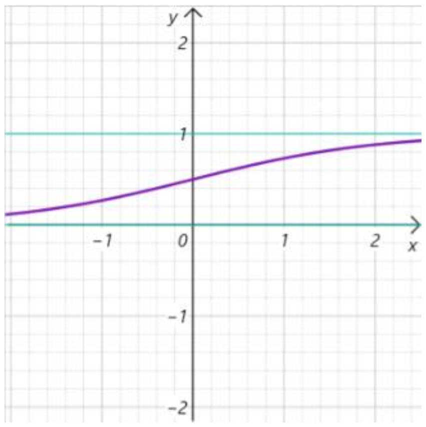
we need a hypothesis function hey outputs only one or zero to make this possible we wrap around another function to the hypothesis of linear regression

$h_\theta(x) = g(\theta^T X)$

The function g is sigmoid function Or logistic function given by

$g(x) = \frac{1}{1+e^{-x}}$

Graph of g(x)



$h_\theta(x) = g(\theta^T X) = \frac{1}{1+e^{-\theta^T X}}$

Interpretation of hypothesis function

The hypothesis function outputs values between zero and one which is to be interpreted as the probability that the classification is true or one

Decision boundary

decision boundaries line or curve on the training data set that differentiates between zero and one classification.

The term $(\theta^T X)$ inside the argument of Sigma*oid* function is the given decision boundary.

Logistic regression cost function

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y)\log\left(1 - h_\theta(x^{(i)})\right)\right] where\ (i)\ means\ the\ i^{th}\ data\ set$$

Minimizing Cost function

here the cost function is minimized the same way as in linear regression

-pseudo code

```
Repeat until convergence{
    For all j{
        $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_1, \theta_2 ...)$
    }
}
```

$\frac{\partial}{\partial \theta_j} J(\theta_1,\theta_2 ...) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

-in vector notation

*$J(\theta)$ = -(1/m)*sum((y.*log(sigmoid(X*θ)))+((1-y).*log(1-sigmoid(X*θ))))*

```
=-(1/m)*sum((y.*log(sigmoid(X*theta))+((1-y).*log(1-sigmoid(X*theta)))
```

X=design matrix

Gradient vector

```
grad=(1/m)*((sigmoid(X*theta)-y)'*X)'+(lambda/m)*theta;
grad(1)=((1/m)*sum((sigmoid(X*theta)-y).*X(:,1)));
```

Multiclass classifications

If there are 3 groups to be classified then will run 3 separate logistic regression algorithms which is also called a one versus all approach for each data 2 of the algorithm will give a negative result and one of them will give a positive result

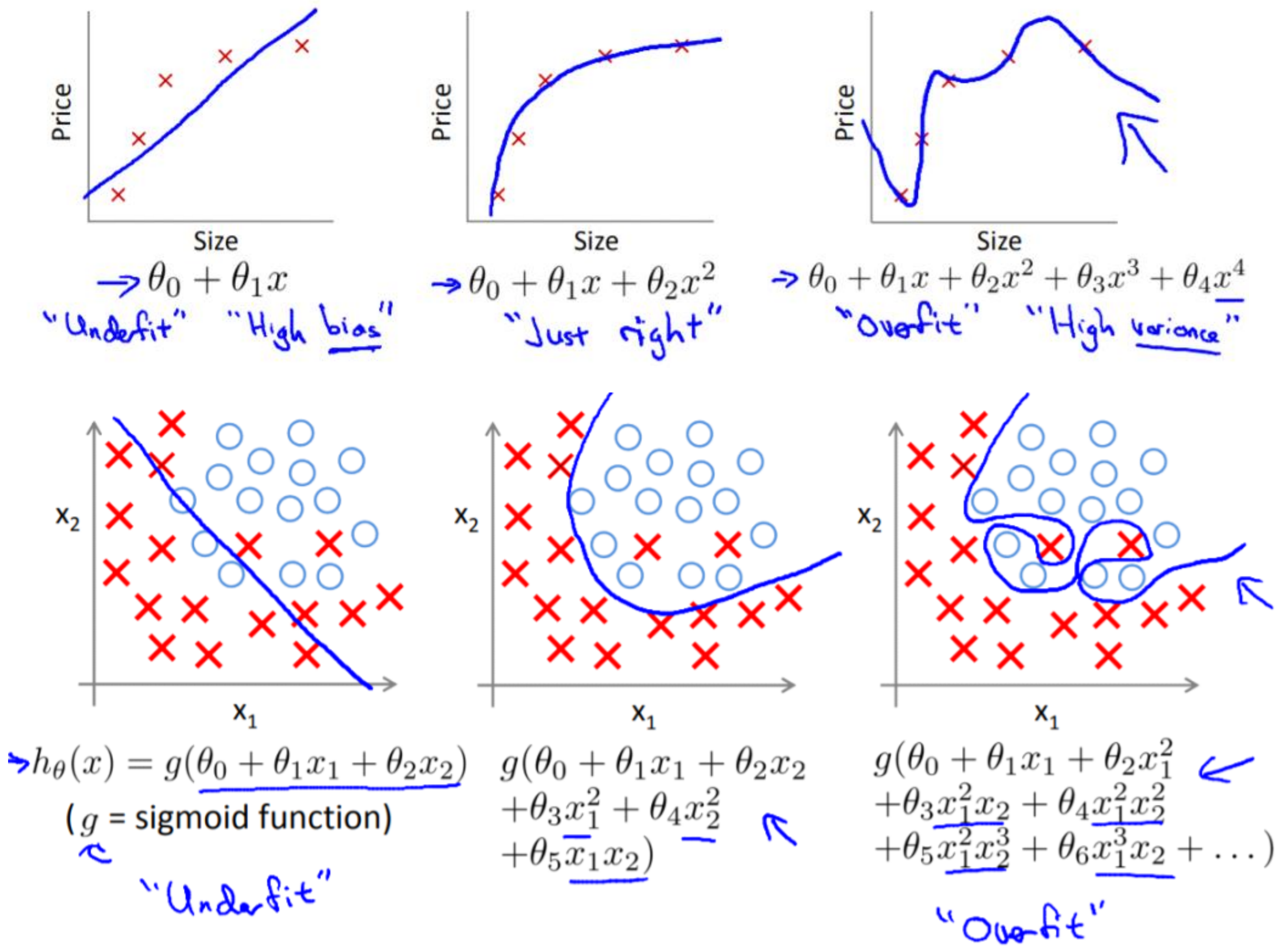
Or the one whose probability is the highest is the result of the classifier

Regularization

18 November 2020 01:57 PM

Problem

The following figure explains the problem


Price
Size
$\theta_0 + \theta_1 x$
"Underfit" "High bias"
$\theta_0 + \theta_1 x + \theta_2 x^2$
"Just right"
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
"Overfit" "High variance"
x_1
x_2
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
(g = sigmoid function)
"Underfit"
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$
"Overfit"

In both these figures we have a problem of underfitting or overfitting

in the problem of underfitting we have too few features describing our hypothesis Sir function cannot separate the data set

in the problem of overfitting we have a large number of features which makes the error nearly zero in the training data set but fails to generalize

Solution

Reduce the number of features

use regularization

Regularization

We add an extra term in the cost function $\lambda \sum_{j=1}^{m} \theta_j^2$ *Now the new cost function is*

Previous + $\frac{1}{2m} \lambda \sum_{j=1}^{m} \theta_j^2$

Now if you set a proper Lambda and minimize this function

this will give a value of Theta also minimising the sum of the squares of all the Theta.

Neural Networks

20 November 2020 09:17 AM

Need of another ml algorithm

For a complex decision boundary in case of logistic regression and with large number of input features The quadratic term sums up to O(n^2) $for\ cubic\ terms\ O\left(n^3\right)$

generally ml problem have a lot of input features so the previous algorithms does not work well as time as well as space complexity would be too high.

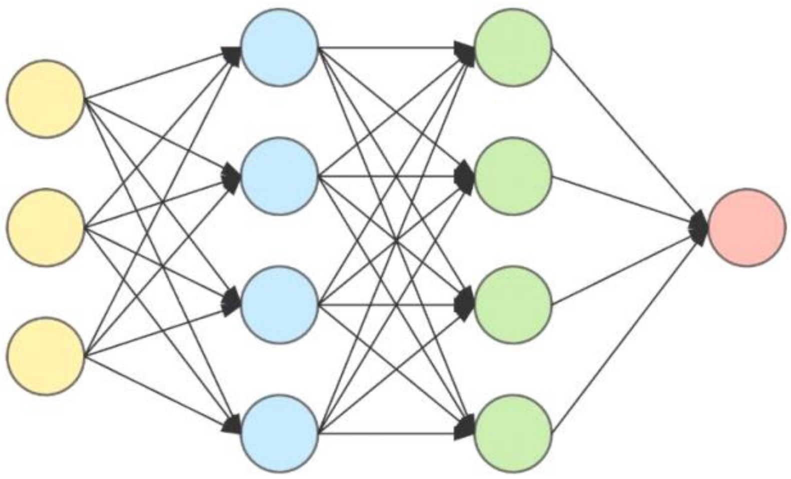
Representing the model of neural networks

A single neuron is represented by a logistic regression unit.

A network has multiple such units hey in which a set of inputs is directed to the first layer of units

hey the output of the first layer of units is directed to the inputs of the second layer of units hey and so on until the output of the second last layer of units is directed to the input of the last layer which contains a single unit and outputs the final result.

Pictorial representation



here the blue as well as the green layers are called hidden layers and each small circle represents a logistic regression unit hey also Called a neuron or simply a unit.

Notations

$a_i^{(j)} = '\ activation\ '$ *in unit I of layer j.*

$\Theta^{\ j} = matrix\ of\ weights\ or\ parameter\ as\ called\ in\ logistic\ regression\ controling\ function\ mapping\ from\ layer\ j\ to\ layer\ j+1$.

$\Theta_{ij}^{(k)} = the\ jth\ parameter\ of\ ith\ unit\ of\ kth\ layer$

$\Theta^{\ j} = \begin{bmatrix} \Theta_{10}^{(j)} & \cdots & \Theta_{1n}^{(j)} \\ \vdots & \ddots & \vdots \\ \Theta_{m0}^{(j)} & \cdots & \Theta_{mn}^{(j)} \end{bmatrix}$ $stack\ all\ the\ parameters\ of\ unit\ 1\ in\ jth\ layer\ as\ row\ matrix\ then\ of\ unit\ 2\ and\ so\ on.$

Inputs in the first layer (orange circles) goes to second layer then the output of the second layer he becomes the input to the 3rd layer and so on.

usually at every layer one extra unit is given call the bias unit which is always turned on or the values are always positive "in terms of logistic regression".

First layer hello call the input layer and the last one is called the output layer , all the layers in between are called the hidden layers.

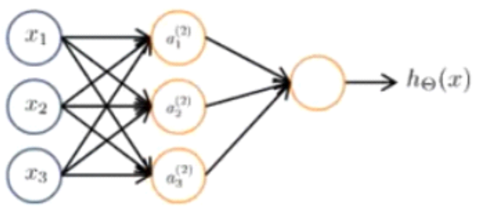
The way a neural network is connected it's called architecture of that neural network.

Working of a neural network

In the simple case of a logistic regression hey we had certain raw features and certain complex features which we calculated and using them and model you were able to output the probability of activation.

in case of a neural network we feed in the network with certain raw features and expect the network to calculate the hey complex features it needed itself that's what the hidden layers are doing the hidden layers are calculating features from the previous layer and as we proceed from a lower layer to a higher layer more complex features are calculated.

Calculation of different activation



$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_\Theta(x) = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

The features in the input layer are also called the activations of first layer
Implementing the above calculations vectorially

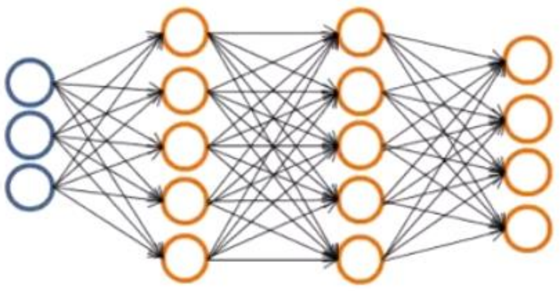
Input features = $\begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix} = a^1$

$$a^{(j+1)} = \begin{bmatrix} a_0^{(j+1)} \\ \vdots \\ a_k^{(j+1)} \end{bmatrix} = g\left(\begin{bmatrix} \Theta_{10}^{(j)} & \cdots & \Theta_{1n}^{(j)} \\ \vdots & \ddots & \vdots \\ \Theta_{k0}^{(j)} & \cdots & \Theta_{kn}^{(j)} \end{bmatrix} \begin{bmatrix} a_0^{(j)} \\ \vdots \\ a_n^{(j)} \end{bmatrix}\right) = g\left(\Theta^{\,j} a^{(j)}\right)$$

$z^{(j+1)} = \Theta^{\,j} a^{(j)} \ldots .\, a\ notation$

The subscripts k, m, n etc. are used here loosely

Multiclass classification with neural networks



$$h_\Theta(x) \in \mathbb{R}^4$$

We use this kind of a neural network with 4 outputs to classify things in four classes if the classification is positive for the first one the first value would be one and all the other values would be zero and so on
So the input maybe some kind of pictures and output is a 4 x 1 vector.