Training Models: Black Box

Michael Claudius, Associate Professor, Roskilde with respect to Jens Peter Andersen

26-03-2021

Training models: Opening The Black Box

- **It is about understanding what goes on behind the stage when models are trained and fitted to labels!**

1. **Linear regression refreshed**
2. **Define the cost function**
3. **Minimize the cost function**
   - **Closed Form Solution (Normal Equation)**
   - **Gradient Descent Solutions: Batch GD, Stochastic GD (SGD), Mini-Batch GD**
4. **Regularize Linear Models to avoid vulnerabilities**
   - **Overfitting, Underfitting, Learning Curves, Early Stopping**

Linear model – refreshing cost-per-click

Hypothesis function in general:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots\dots\dots + \theta_n x_n$$

Where:

- $\hat{y}$ is the predicted value - e.g. We want to predict 'total clicks per day'
- n is the number of features - e.g. we may have only 1 feature, 'cost per click'
- x_i is the i^{th} feature value – e.g. x_1 may represent the value of a 'cost per click'
- θ_j is the j^{th} model parameter – e.g. only θ_0 and θ_1 are relevant with only one feature

Linear model – refreshing 50StartUps

Hypothesis function for linear model in general:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots\ldots\ldots + \theta_n x_n$$

Where:

- $\hat{y}$ the predicted value - e.g. Want to predict *"Profit"*
- n the number of features - e.g. 3 features, *"R&DCost", "Administration", "Marketing Spend"*
- x_i is the i^{th} feature value – e.g. X_1 represent the value of a "R&DCost",
- θ_j is the j^{th} model parameter – e.g. only θ_0, θ_1, θ_2, θ_3 are relevant when 3 features

OR on vectorized form

Equation 4-2. Linear Regression model prediction (vectorized form)

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

In this equation:

- $\boldsymbol{\theta}$ is the model's *parameter vector*, containing the bias term θ_0 and the feature weights θ_1 to θ_n.
- $\mathbf{x}$ is the instance's *feature vector*, containing x_0 to x_n, with x_0 always equal to 1.
- $\boldsymbol{\theta} \cdot \mathbf{x}$ is the dot product of the vectors $\boldsymbol{\theta}$ and $\mathbf{x}$, which is of course equal to $\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$.
- $h_{\boldsymbol{\theta}}$ is the hypothesis function, using the model parameters $\boldsymbol{\theta}$.

Linear model – refreshing 50StartUps

- **50StartUps Found**

- **intercept_ (θ_0)**
- **coef_ ($\theta_1, \theta_2, \theta_3$)**

```
In [46]:  #Get y-intersection (i.e. x=0)
          intersection=lin_reg.intercept_
          intersection

Out[46]:  array([48954.88930169])
```

```
In [49]:  #Get the weights
          lin_reg.coef_

Out[49]:  array([[ 0.80929046, -0.04210973,  0.03770244]])
```

- **Question is: How did LinearRegression find the weight values?**
- **Answer follows now!**

Define The Cost Function

- **Performance measure: Root Mean Square Error (RMSE)**
- **Need to a find a value of θ which minimize the RMSE**
- **Cost function: Mean Square Error (MSE), as it results in the same minimum and values as RMSE**

Equation 2-1. Root Mean Square Error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

Equation 4-3. MSE cost function for a Linear Regression model

$$\mathrm{MSE}\left(\mathbf{X}, h_{\boldsymbol{\theta}}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^{\intercal}\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

- **We are lucky. There are several solutions for this!**

Minimize The Cost Function MSE

Minimize the cost function solutions using

- **Closed Form Solution (Normal Equation)**
- **Gradient Descent Solution:**
  - **Batch GD,**
  - **Stochastic GD (SGD),**
  - **Mini-Batch GD**

- ***All solutions have advantages and disadvantages***

Closed Form Solution

The Normal Equation calculates the values of θ directly using some matrix manipulations and multiplications

Equation 4-4. Normal Equation

$$\hat{\boldsymbol{\theta}} = \left(\mathbf{X}^\intercal \mathbf{X}\right)^{-1} \ \mathbf{X}^\intercal \ \mathbf{y}$$

In this equation:

- $\hat{\boldsymbol{\theta}}$ is the value of $\boldsymbol{\theta}$ that minimizes the cost function.
- $\mathbf{y}$ is the vector of target values containing $y^{(1)}$ to $y^{(m)}$.

- **X^T Transpose matrix of X; i.e. mirror values along the diagonal:** https://en.wikipedia.org/wiki/Transpose

Closed Form: Evaluation

Typical complexity for closed form computations:

- Training the model with *m* feature instances is complexity $O(m)$ – proportional to number of instances m
- Training the model with *n* features is complexity $O(n^{(>2)})$ – proportional to quadratic number of features $n^{(2)}$

  - – e.g. increasing *n* by a factor 2 will increase processing resources needed by $2^2=4$
  - – e.g. increasing *n* by a factor 10 will increase processing resources needed by $10^2=100$! That's 100 times slower

  - Processing resources are time and memory

  - Advantage: Simple to compute
  - Disadvantage: Slow for high number of features (n > 10.000)
    - Examples: predicting on basis of the human genom

  - **We are lucky. Why? There is the Gradient Descent Algorithms for these cases.**
  - **BUT let us look at some code first.**