Introduction to Deep Learning

Module 3 – Classification and Cost Function

Presented By: Istvan Lengyel

Course Overview

- Linear Classification
- Hypothesis expanded
- Loss/Cost Function
  - Mean Squared Error
  - Mean Absolute Error
  - Categorical Cross-Entropy
  - Sparse Categorical Cross-Entropy
  - Binary Cross-Entropy

Classification

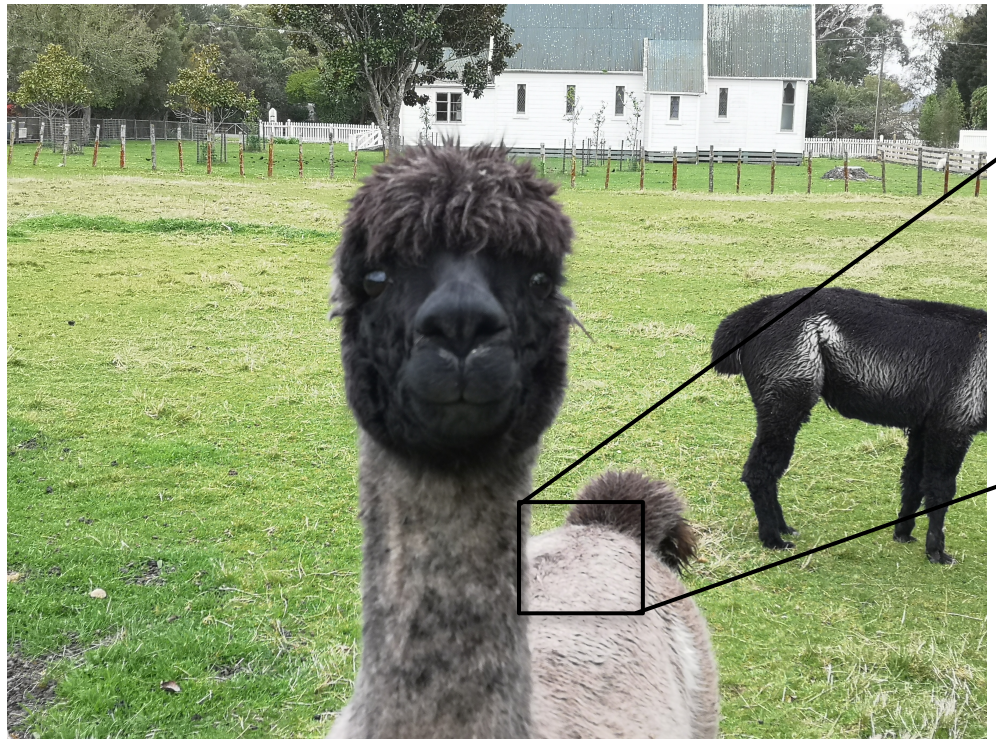
Image Classification

- Image classification is an essential component of Computer Vision
- Given a set of labels {dog, car, cat, alpaca,…}

Alpaca

Problem: Semantic Gap

- What we see are images, which is an easy task for us to classify.
- However, what the computer sees, is a large matrix of numbers.

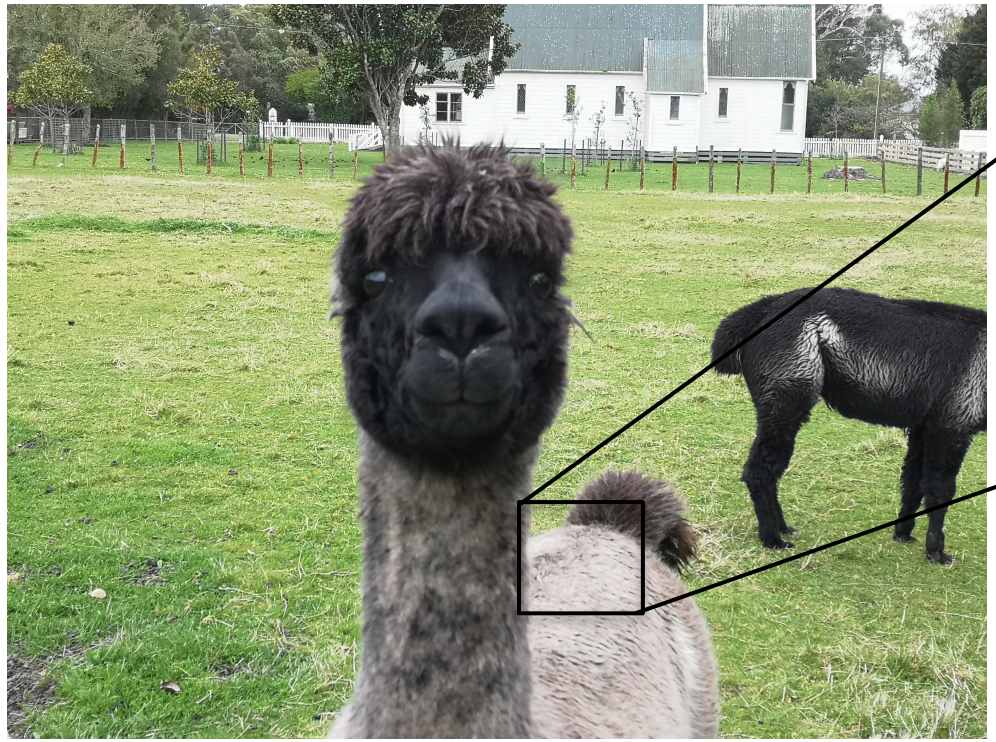


253	252	240	235	235
240	240	235	230	224
157	154	200	200	215
145	150	198	176	202
201	196	198	204	210

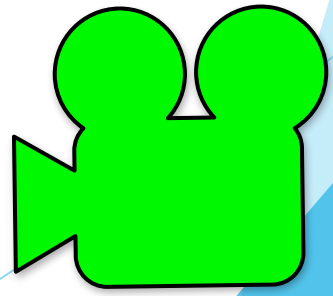
- An Image is a value between [0,255] represented as a large matrix
- The image of my Alpaca Black Pearl is 3648 x 2736 @ 3 channels for RGB this equates out to 29.9M

Challenges: Viewpoint

- When a camera's viewpoint changes our Matrix values change.
- This poses a challenge as Black Pearl is still Black Pearl, with differing Matrix values.



253	252	240	235	235
240	240	235	230	224
157	154	200	200	215
145	150	198	176	202
201	196	198	204	210



Challenges: Lighting

- Another challenge is lighting.
- Here we have two identical images, with lighting lowered and a with lighting raised. It's difficult to make out Black Pearl in one of the images and difficult to see the clouds in the second image. We tend to loose a number of features in both images.

Challenges: Deformation

- Another challenge is deformation.
- Here we have my cat Sakima, he loves to stretch and lounge about.
- In this image it's difficult for a computer to work out that this is a cat, let alone being Sakima.
- For us it would be trivial. **Note: The AI software on my Huawei P20 Pro cellphone knows that the second image is a cat.**

Challenges: Occlusion

- Another challenge is occlusion.
- Here we have two images of Sakima covered and in socks.
- Much of the image is of socks with portions of Sakima being viewable.
- We are able to work it out that the image is of a cat and it is Sakima, however this is a challenge for computers.

Challenges: Background Clutter

- Another challenge is Background Clutter.
- Here we have an image of Sakima blending in with a tree and leaves.
- We are able to work it out that the image is of a cat and it is Sakima, however this is also a challenge for computers.

Challenges: Interclass Variation

- Another challenge is Interclass Variation.
- Here we have an images of Alpacas young and old and of various colours.
- We are able to work out that the images are of alpacas regardless of size and colour, again this is also a challenge for computers.

Challenges: Image Classifier

- In Python if I wanted to sort through some random numbers there is a function for this **sort()**
- However, for image classification returning a label for an image, this would be a challenge. There are plenty of Deep Learning frameworks that can be used. However, creating a functions that reads and image and returns a label, requires a bit of work, with varying results. There is no way to hard-code the algorithm to recognise an alpaca or any other class.

```
numbers = [6,3,9,1,5,4,8,7,2,10,18,16,14,17,13,19,12,15,11]
numbers.sort()
print(numbers)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

```
def classify(image):
    #read image with magic
    return label
```

A Data Driven Approach

1. Collect a dataset of images and labels.
2. Use Machine Learning to train a classifier
3. evaluate classifier on new data

```
def train(images, labels):
    #more magic
    return model
```

```
def predict(model, testImages):
    #more magic
    return testlabel
```

dog

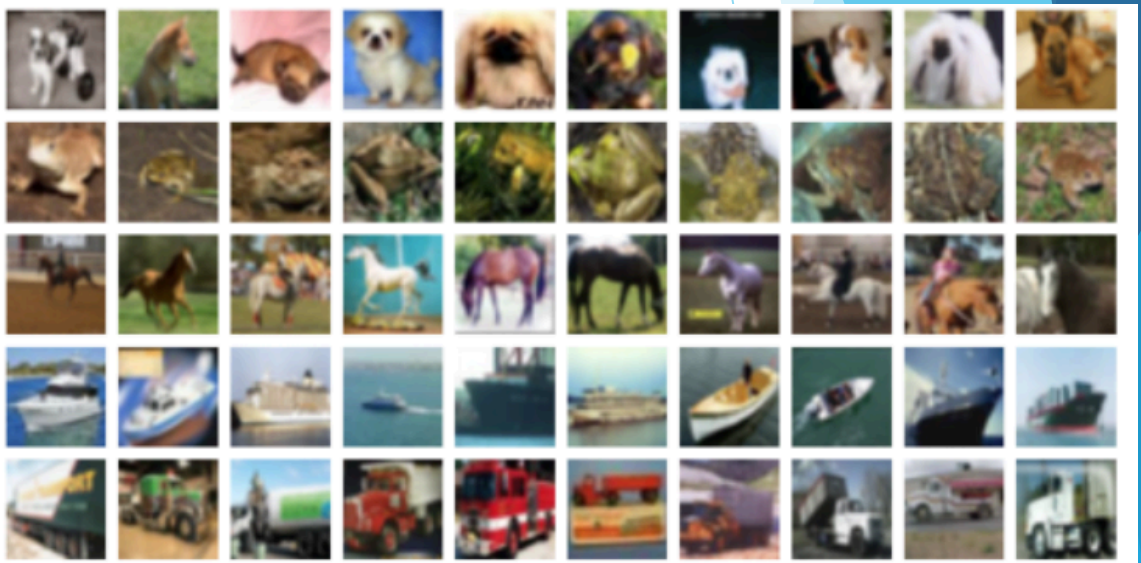
frog

horse

ship

truck

Example Training Set



Parametric Approach

$$f(x,W) = Wx + b$$

3072x1 (x)
10x1 (f(x,W))
10x3072 (W)
10x1 (b)

Image

Array of 32 x 32 x 3 = 3072

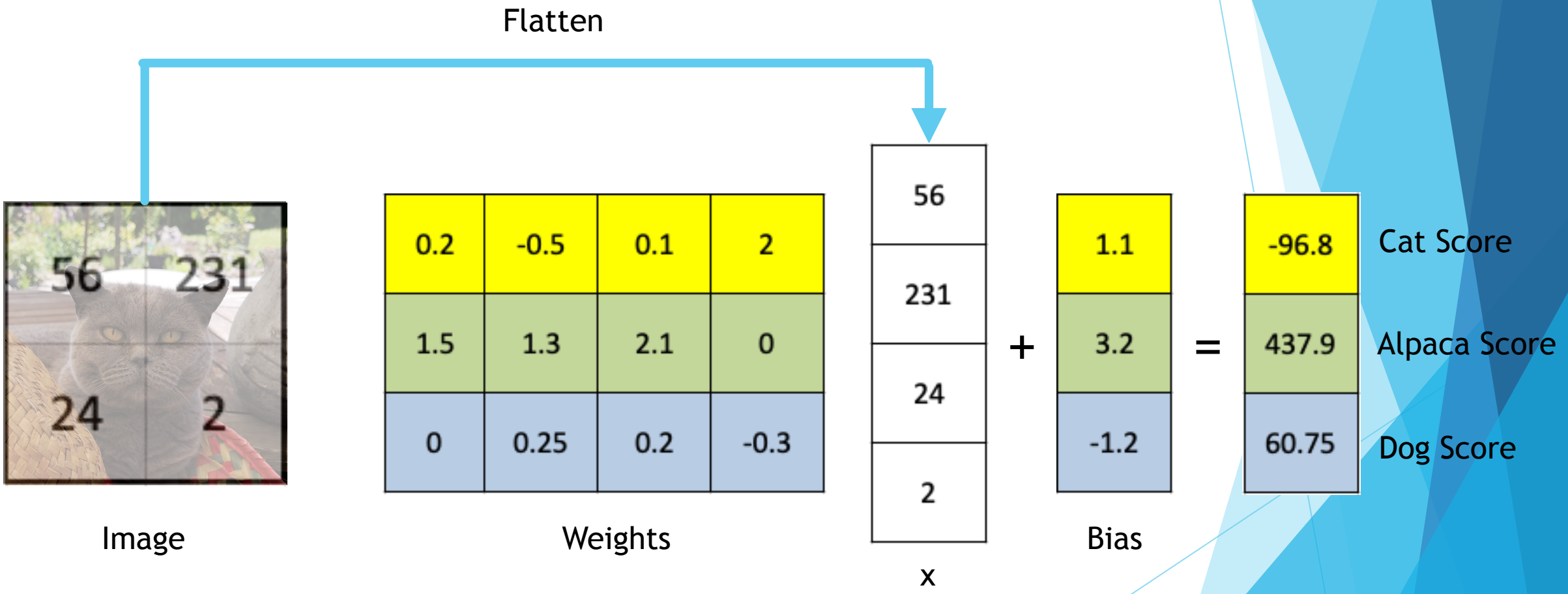
Input Data

f(x,W)

10 number giving class scores

Parameters or Weights
or sometimes called Theta

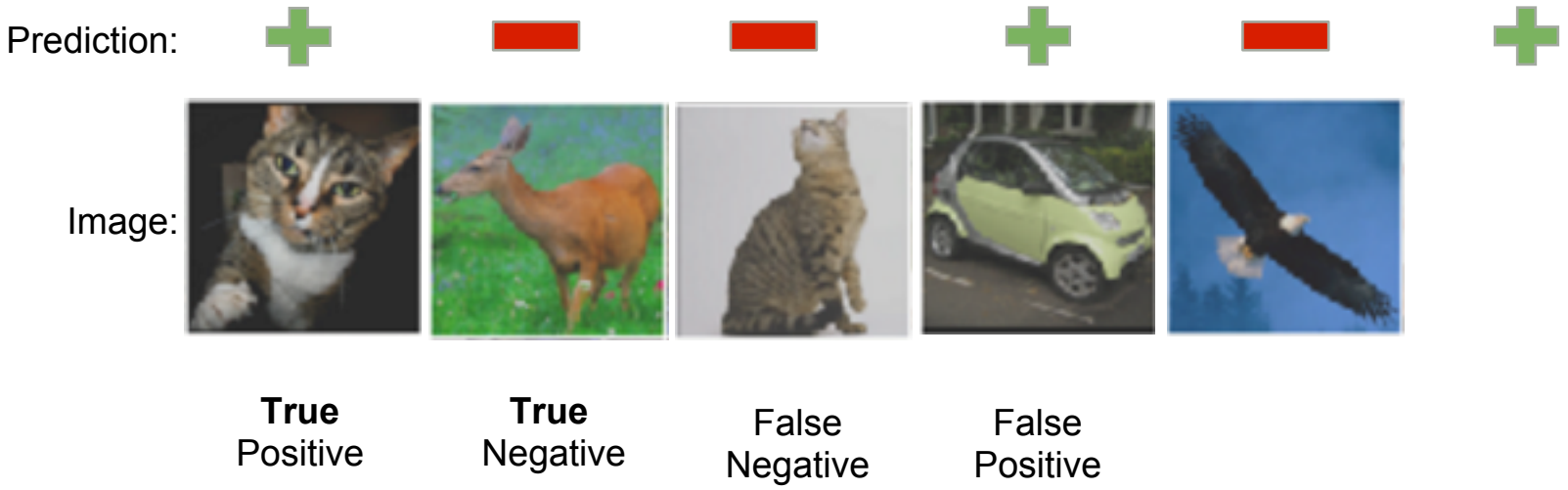
Parametric Approach



Measuring Success for Classification

- True Positive: Correctly identified as relevant
- True Negative: Correctly identified as not relevant
- False Positive: Incorrectly labeled as relevant
- False Negative: Incorrectly labeled as not relevant

Example: Identify cats

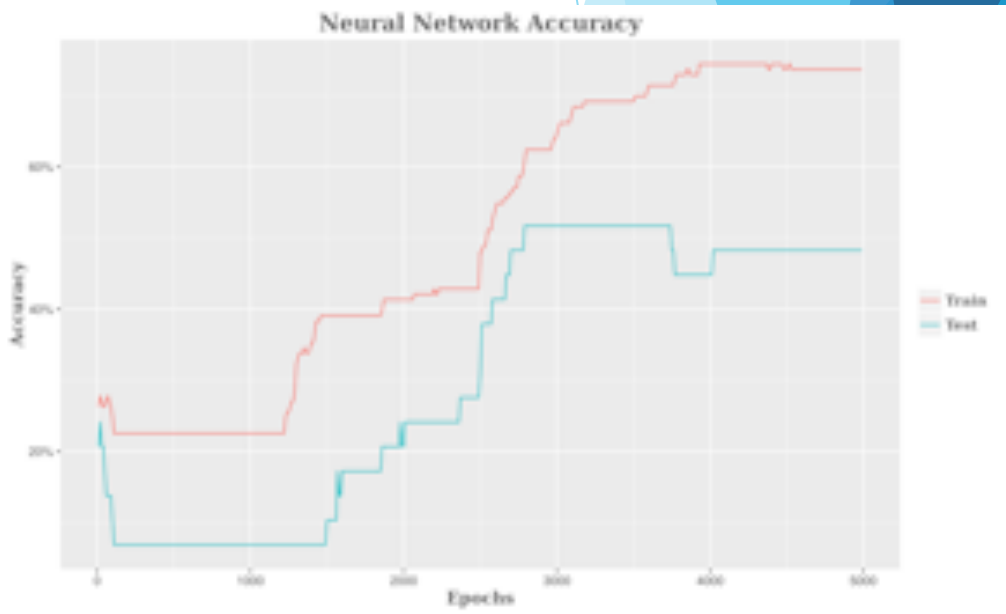


Precision, Recall and Accuracy

- Precision
  - Percentage of positive labels that are correct
    - Precision = (# of **True** Positives) **/** (# of **True** positives **+** # of **False** Positives)
- Recall
  - Percentage of positive examples that are correctly labeled
    - Recall = (# of **True** positives) / (# of **True** positives + # of **False** negatives)
- Accuracy
  - Percentage of correct labels
    - Accuracy = (# of **True** positives + # of **True** negatives) / (# of Samples)

Training and Test Data

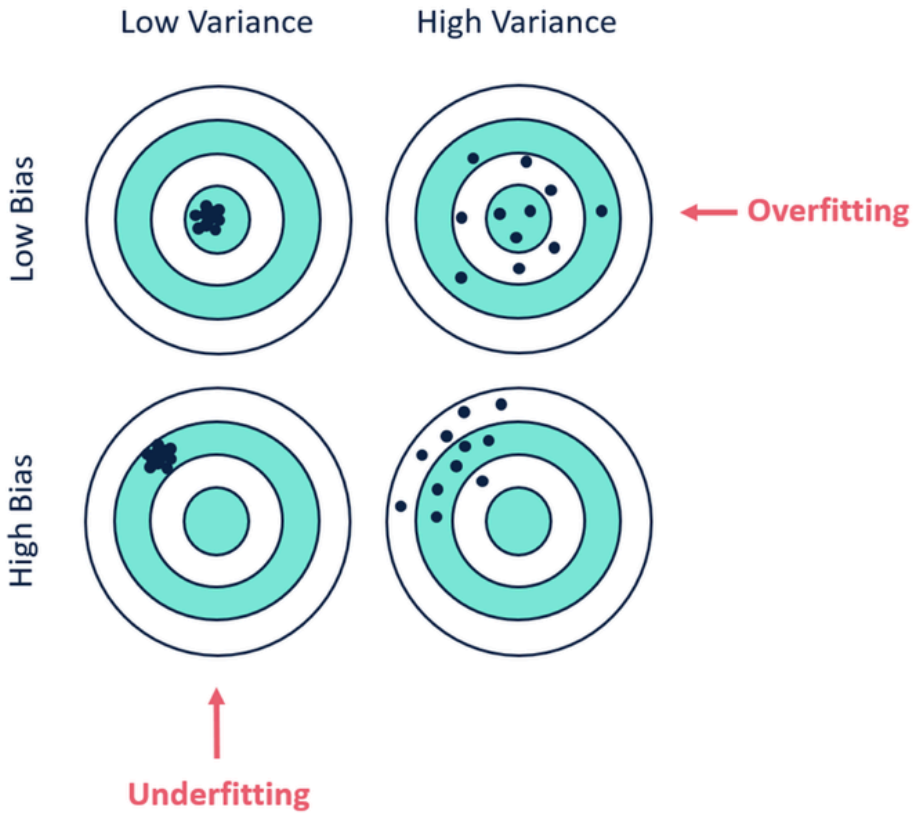
- Training Data
  - data used to learn a model
- Test Data
  - data used to assess the accuracy of model
- Overfitting
  - Model performs well on training data but poorly on test data



Bias and Variance

- Bias: expected difference between model's prediction and truth
- Variance: how much the model differs among training sets

- Model Scenarios
  - High Bias: Model makes inaccurate predictions on training data
  - High Variance: Model does not generalise to new datasets
  - Low Bias: Model makes accurate predictions on training data
  - Low Variance: Model generalises to new datasets



Bias and Variance

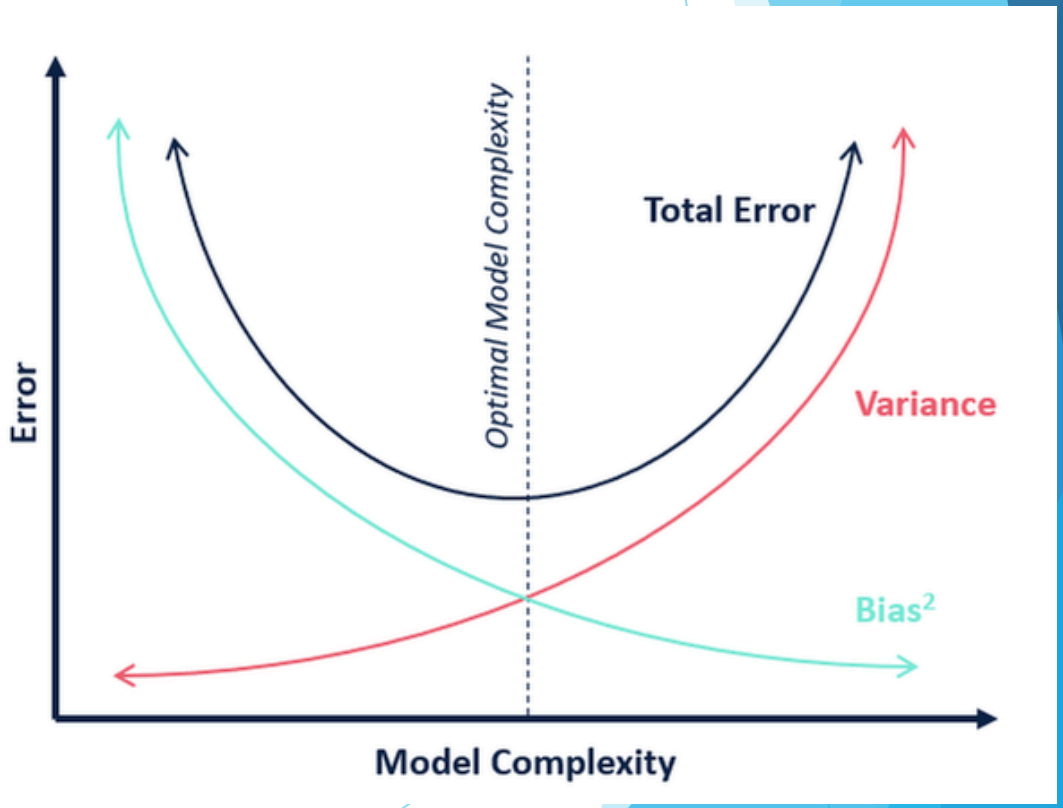
With **Bias** and **Variance** we are wanting the achieve the optimal balance between the two.

Solutions for underfitting

- using more complex model or model the fits the data
- better predictor variables(feature engineering)
- reduce the constraints applied to the model(regularization)

Solutions for overfitting

- selecting a model with few parameters or reducing the number of attributes
- gather more training data
- reduce noise in the training data(finding and correcting errors, removing outliers)



Loss/Cost Function

So if you recall we are trying to map x's to y's

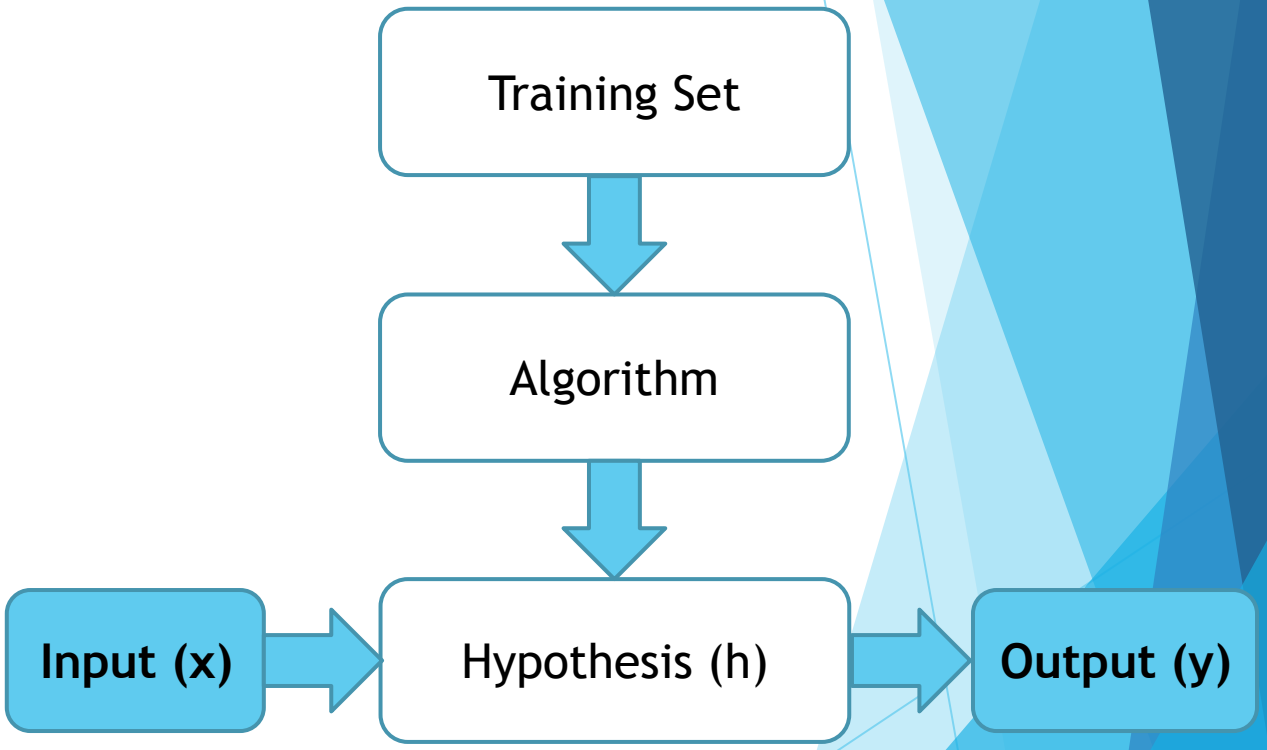
With supervised learning we start of with a training set.

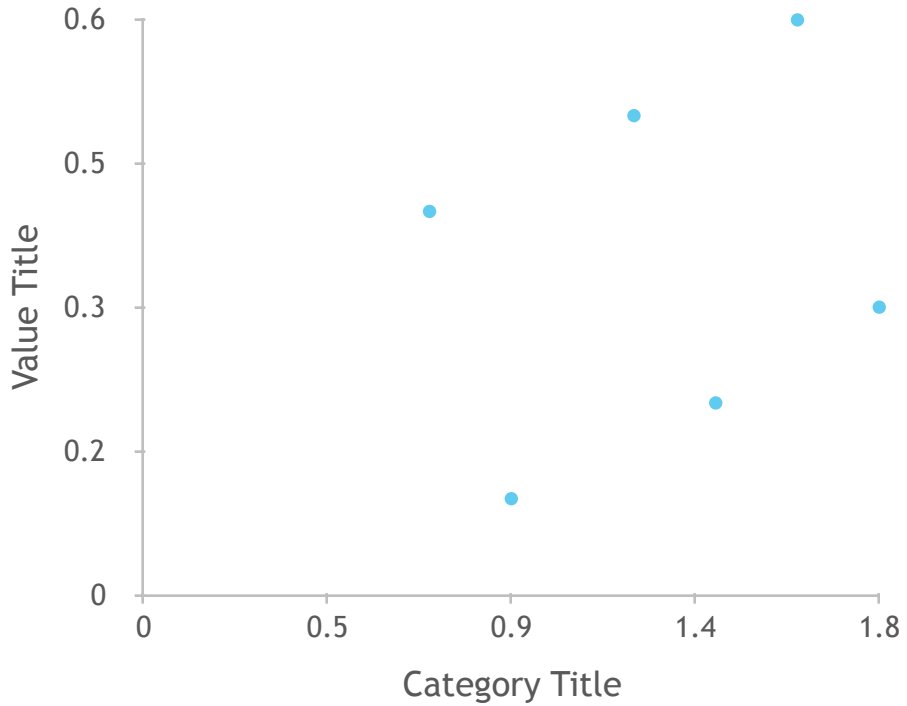
This training set then goes through an algorithm which outputs a hypothesis (denoted by a lowercase **h)**

The hypothesis takes in our input value **x** this is our square meters for a particular house

The Hypothesis then provides an output value **y** this is our predicted house value

So what we are doing is having our Hypothesis (h) map **x** to **y**



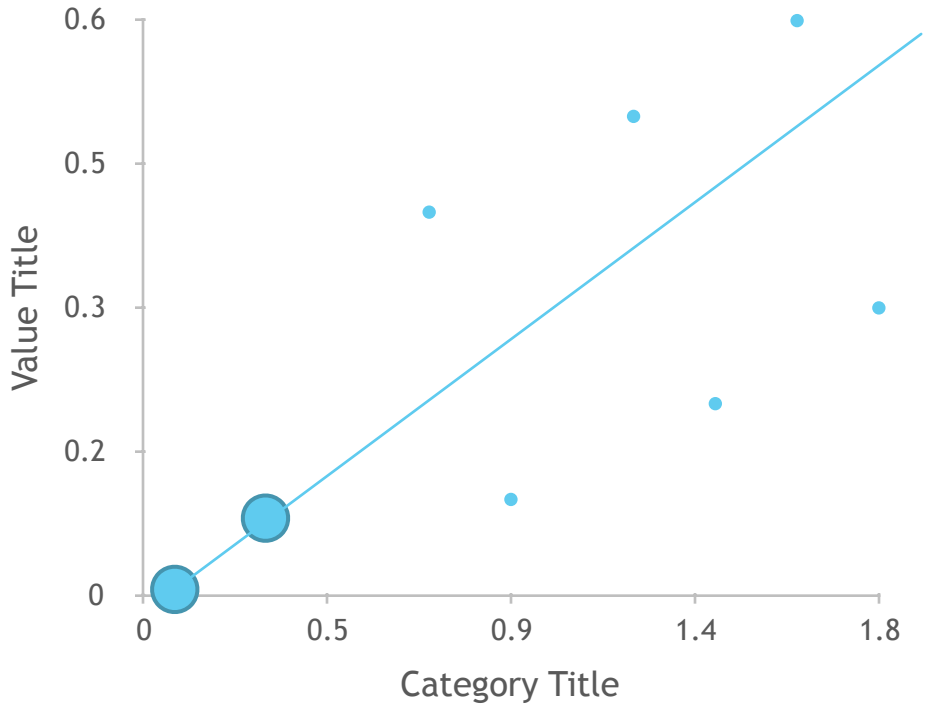


How do we represent this

$h_{\theta(x)} = \theta_0 + \theta_{1x}$ or $h(x) = \theta_0 + \theta_{1x}$

θ_i's are parameters

θ_{1x}



How do we represent this

$h_{\theta(x)} = \theta_0 + \theta_{1x}$ or $h(x) = \theta_0 + \theta_{1x}$

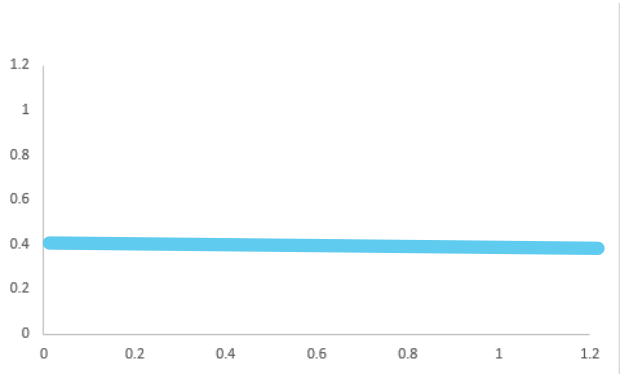
θ_i's are parameters

θ_{1x}

Lets say $\theta_0 = 0$

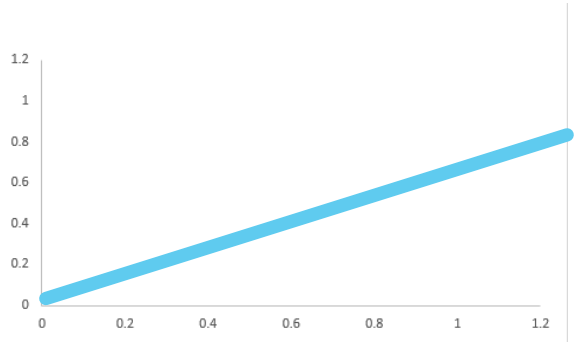
And $\theta_1 = 0.1$

$$h_{\theta(x)} = \theta_0 + \theta_1 x$$



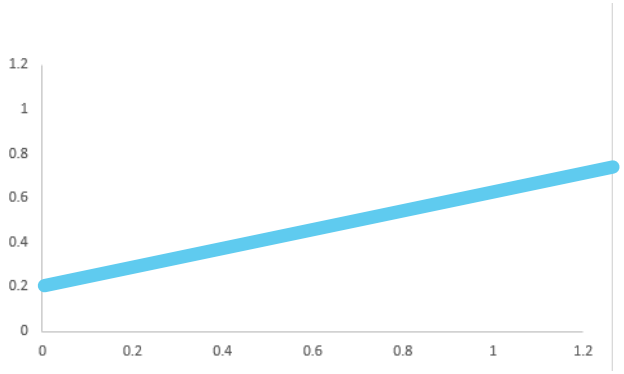
$\theta_0 = 0.4$

$\theta_1 = 0$



$\theta_0 = 0$

$\theta_1 = 0.2$



$\theta_0 = 0.2$

$\theta_1 = 0.4$

Cost Function

Cost Function

When it comes to training we want to minimise the training **Cost**, we will use J(θ) to represent our cost. The lower the training cost, the more accurate our results will be.

There are many ways to calculate our cost we will make use of the Mean-Squared Error (MSE) formula for our regression problem.

The formula is presented to the side.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

m	The number of training examples
$x^{(i)}$	The input vector for the i^{th} training example
$y^{(i)}$	The class label for the i^{th} training example
Θ	The chosen parameter values or "weights" (Θ_0, Θ_1, Θ_2)
$h_\theta\left(x^{(i)}\right)$	The algorithm's prediction for the i^{th} training example using the parameters Θ.

Cost Function

When it comes to training we want to minimise our **Cost function**, in the formula to the side J(θ) represents our cost. The lower the training cost, the more accurate our results will be. The higher the cost the poorer our training results will be.

There are many ways to calculate our cost we will make use of the Mean-Squared Error (MSE) formula for our regression problem.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

m	The number of training examples
$x^{(i)}$	The input vector for the i^{th} training example
$y^{(i)}$	The class label for the i^{th} training example
Θ	The chosen parameter values or "weights" (Θ_0, Θ_1, Θ_2)
$h_\theta\left(x^{(i)}\right)$	The algorithm's prediction for the i^{th} training example using the parameters Θ.

Python Example Implementing Loss Function in Keras

```
model.compile(loss='mean_squared_error', optimizer='sgd')
```

or

```
from keras import losses
model.compile(loss=losses.mean_squared_error, optimizer='sgd')
```

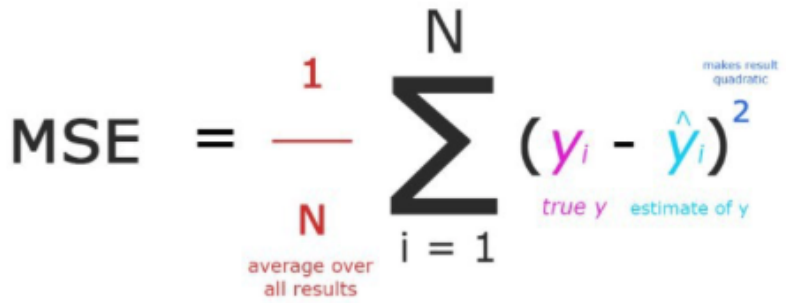
In this example **Stochastic Gradient Descent** has been selected as the Optimizer. More on Stochastic Gradient Descent in a later module. The loss being used is **MSE.**

Cost Function and Loss Function

- Both the cost function and loss function are the same. We are wanting to minimise them as much as we can.
- When we use Keras as a framework the loss function is one of the **two required parameters** that we need to use when we train our model. The second parameter is our **optimizer**.
- The choices we have for our loss function are extensive under **Keras**, a few examples include the following:
  - **mean_squared_error**
  - **mean_absolute_error**
  - **categorical_crossentropy**
  - **sparse_categorical_crossentropy**
  - **binary_crossentropy**
- The full list can be view on the following reference page https://keras.io/losses/ or a more define API reference from the Tensorflow site https://www.tensorflow.org/api_docs/python/tf/keras/losses
- When we are designing our **models** we often will try out both different loss functions with different optimiser to refine our **model**.

Mean Squared Error

- Mean Squared Error (MSE) is one of the most common regression cost function.
- The function is differentiable at every point of its domain and its convex, allowing it to be optimised using Stochastic Gradient Descent (SGD).
- There are drawbacks in using this cost function when outliers are present from our data point spread.
- Outliers can lead to an unacceptable correction when the distance between the prediction and the actual value is large, which can lead to the error being high.



Mean Absolute Error (MAE)

- Mean Absolute Error (MAE) is a loss function used for regression.
- The loss is the mean overseen data of the absolute difference between true and predicted values.
- MAE is not sensitive to Outliers and given several examples with the same input feature value, and the optimal predication will be their median target value. This should be compared with MSE, where the optimal prediction is the mean.
- A disadvantage MAE is that the gradient magnitude is not dependent on the error size. This leads to the gradient magnitude being large even when the error is small, which can lead to convergence problems.
- When to use MAE, when doing image reconstruction, MAE encourages less blurry images compared to MSE.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} |y_i - \hat{y}_i|$$

Categorical Cross-Entropy

- Categorical Cross-Entropy is the most diffused classification cost function adopted by logistics regression and used by the majority of neural architects.
- The cost function is convex and can be easily optimised using Stochastic Gradient Descent (SGD).
- Used for single label categorisation, this is when only one category is applicable for each data point. One example belongs to one class.
- An example, would be the MNIST problem where we have numbers from 1 to 9. We can use Categorical Cross-Entropy for the predicated value.

$$L(y,\hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * log(\hat{y}_{ij}))$$

Sparse Categorical Cross-Entropy

- Sparse Categorical Cross-Entropy is the very similar to Categorical Cross-Entropy.
- When to use Sparse Categorical Cross-Entropy, when classes are mutually exclusive, for example when each belongs exactly to one class.
- This allows us to conserve time and memory not having to use sum.
- Consider the following case of 10000 classes when they are mutually exclusive - just 1 log instead of summing up 10000 for each sample, just one integer instead of 10000 floats

Binary Cross-Entropy

- Also known as Sigmoid Cross-Entropy loss.
- It is a Sigmoid Activation plus a Cross-Enthropy
- Used for multi-label classification, where the insight of an element belonging to a certain class should not influence another class.
- As the name implies our problems are represented as a Yes/No (binary) decision.
- For Example, you might have a piece of music that can have more than one mood, for instance, it can be “happy” or “Energetic” at the same time. To solve this problem we can use Binary cross entropy

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$