Neurons and the brain

Tuesday, 19 September 2023 20:49

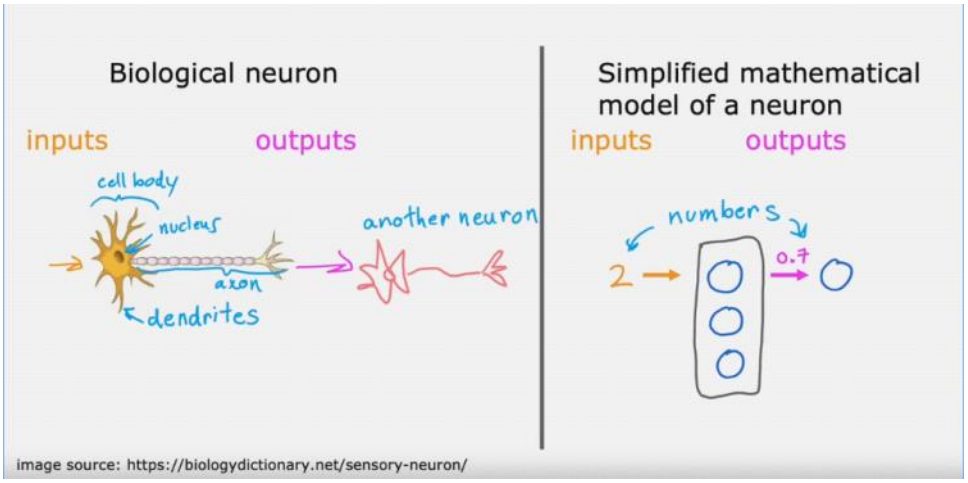
The original goal of neural networks was to mimic the brain

Neural networks were used originally for speech they have advanced from there to things like language processing (NLP)

Today's neural networks have almost nothing to do with how the brain works, just the motivation was from the brain.

Mathematical model of a neuron:

An input is given to a group of neurons to undertake a task, then there output is given to more neurons



Neural networks are now possible thanks to the amount of data now available

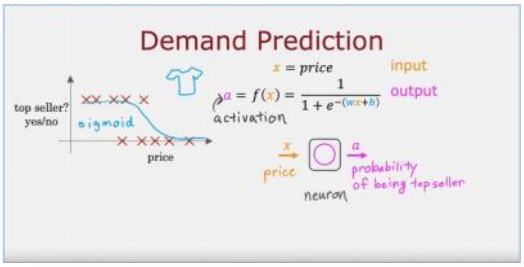
These Neural networks are great if you have a lot of data to use

The uptick in GPU power allowed Neural Networks to be made

Demand Prediction

Wednesday, 20 September 2023 07:02

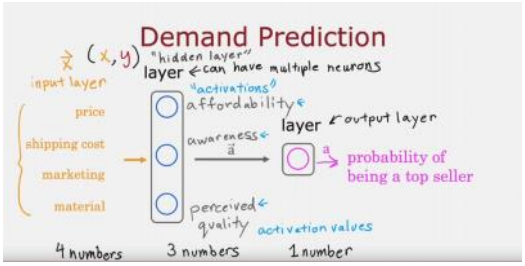
Using Logistic Regression as a neural network:



A = activation is a neurons output

Neuron contains our logistic regression algorithm, so building a neural network means wiring loads of these together

Neural network demand prediction:



Each of these neurons is logistic regression (for this example)

A group of neurons is called a layer

The Input vector is given to first layer of neurons (models)

All layers passed to are "hidden layers" as a "a" vector (activation values)

We cannot decide what is produced by hidden layers nor what weighting different neurons it gives it features

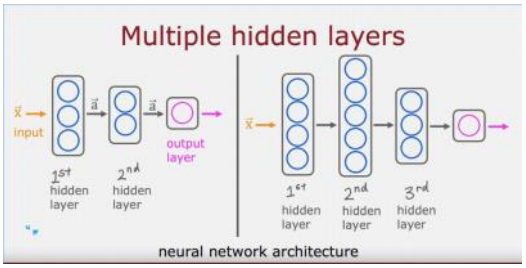
They produce activation values (a vector)

Which is sent from layer to layer (remember we don't know what the different features / inputs are)

We can only decide the amount of layers and neurons

Each neuron learns via gradient descent to forfill its objectives

Although what features are in hidden layers can't be decided, the amount is.



The amount of layers can also be decided

Left 3 activation values in vector-> 2 Vector activation values in vector -> 1 output

So the amount of data used as features for the next layer can be decided

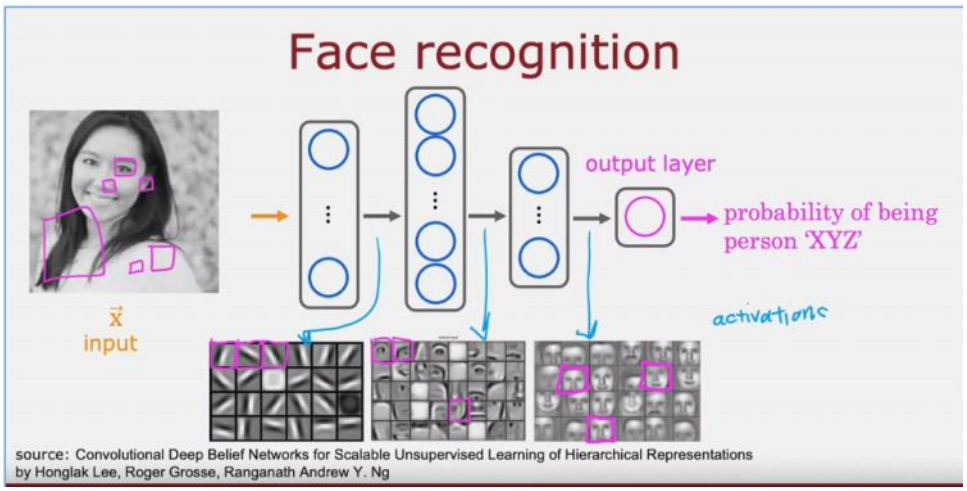
We cannot decide what features will be produced or what weights it gives its inputs by each hidden layer

Example: Recognizing Images

Wednesday, 20 September 2023 08:01

An image is stored in the computer as a matrix of intensity values

There are a lot of these values depending on image size



Each layer does a different job to advance the next layer

First layer looks for lines, second for nose, third for face shape

The neural network will learn what's best for each layer

Each neuron learns its objective to help forfill the layers objective

Then because its logistic regression, you get probability at the end

We can't decide what each hidden layer weights features with nor can we decide what they produce

Because we don't decide and the input or output of each layer, it can find the best way to detect a face

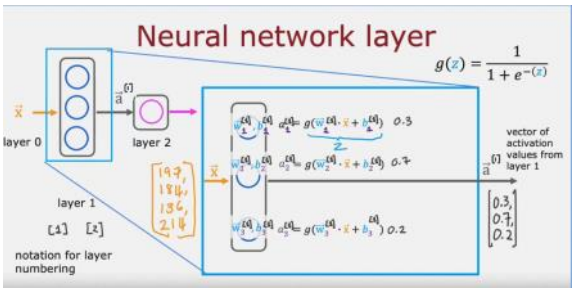
Neural Network Layer (Thresholding)

Wednesday, 20 September 2023 08:44

Square brackets means a layer in a neural network

Threshold are applied after your output layer

Layer 1:



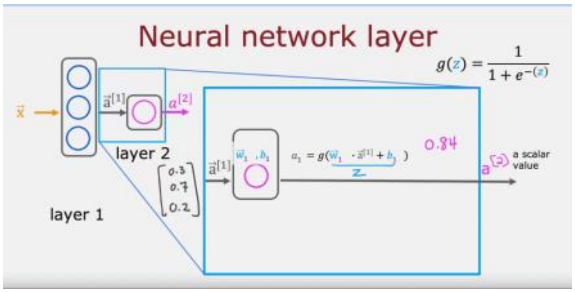
Each Neuron within each layer fulfils an objective when taught from gradient descent

Remember, each layer contains a model (logistic regression etc.)

The output of "a"[1] (activation values from layer one) becomes the input for layer 2

Layer 2:

Output from layer 1 becomes input for layer 2

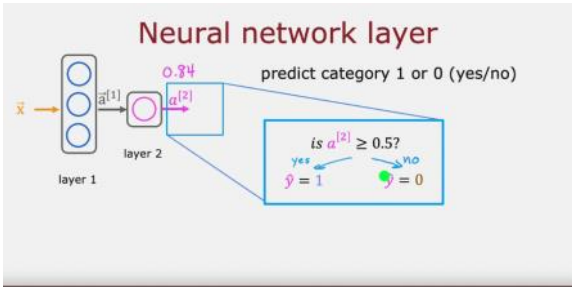


This takes in "a" 1 which the activation values from layer 1

It produces "a"2 activation values

Output layer:

As part of your output layer you can install thresholding



Summary:

Every layer inputs a vector of numbers and applies a model to it.

and passes its results from layer to layer to make predictions

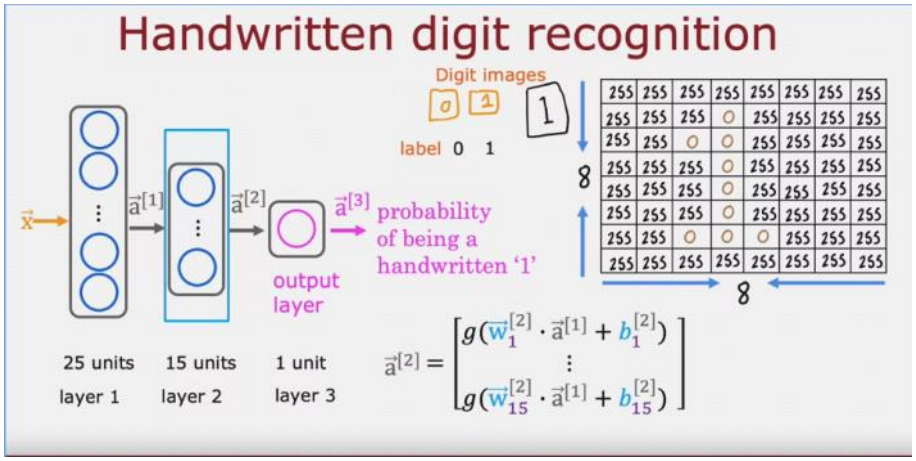
Thresholds are applied after output layer

Inference: making predictions (forward propagation) (Units)

Tuesday, 26 September 2023 06:20

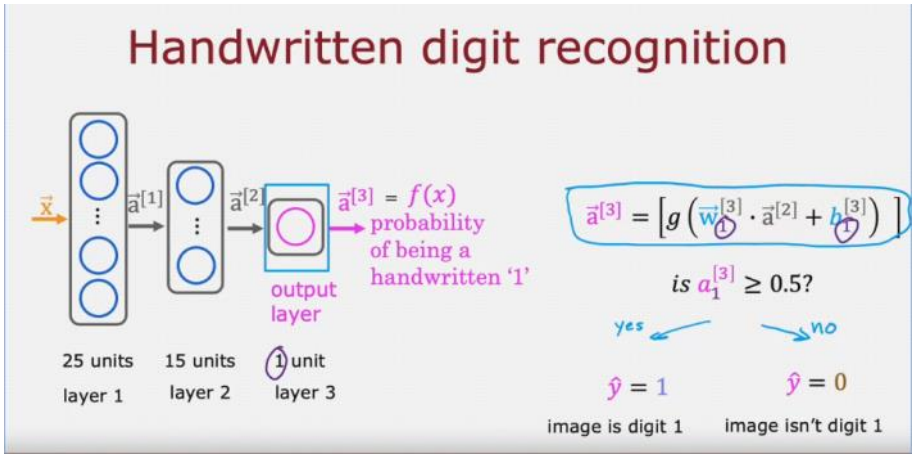
Inference is when you use a trained model

A unit is a neuron



Layer two has 15 units so its computes go to w15 & b15

This is called forward propagation because you compute from left to right through the network using the previous layers activation values



F(x) is your neural networks output

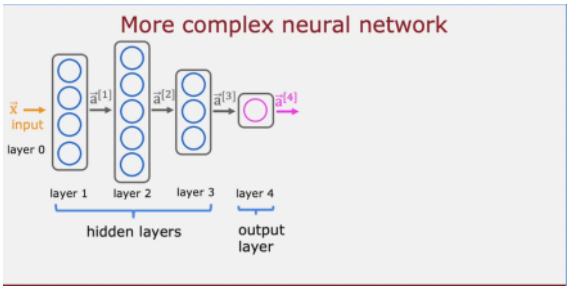
With forward propagation, the number of neurons should decrease as you get closer to output layer (as above)

More Complex Neural Networks (L is for)

Wednesday, 20 September 2023 09:39

Each layer takes in the previous layers activation values

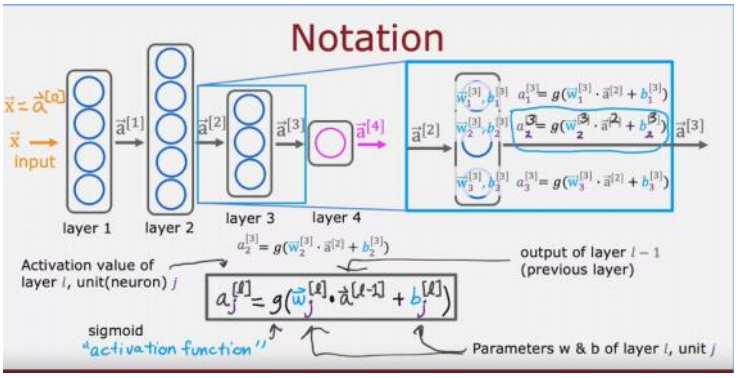
G = activation function (your learning algorithm of choice)



The activation values (output) of each layer of a neural network as shown as "a"[1-4]

Its "a" and then whatever layer 1-4

Notation:



Each layer takes in the previous layers activation values

A[number] = activation values taken in or given out

$g(\vec{w}_2^{[3]}$

Bottom right is which neuron in layer (neuron 2)

Top right is which layer overall (layer 3 in this case)

$\vec{a}^{[2]}$

This is two because its the activation values from previous layer

Notation Exampled:

$$a_2^{[3]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]})$$

G = sigmoid function (logistic regression) or **Activation function**

$(\vec{w}_2^{[3]}$

Layer 3, neuron 2

$\vec{a}^{[2]}$

Activation values from pervious layer (2)

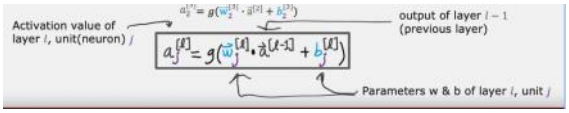
$b_2^{[3]}$

Bias for layer 3, neuron 2

When building neural networks:

L = layer

J = neuron



G = activation function (your learning algorithm of choice)

Neurons & Layers (Lab)

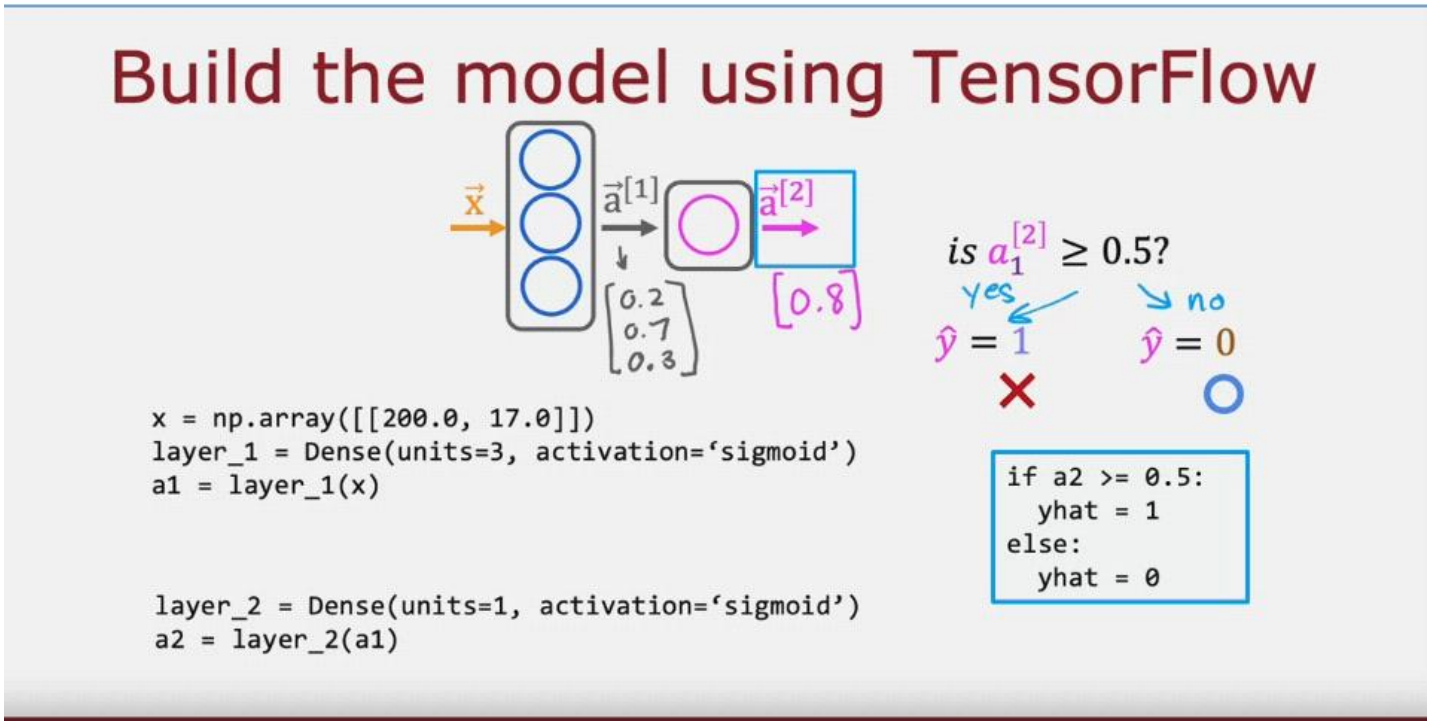
Tuesday, 26 September 2023 06:53

C2_W1_Lab
01_Neuro...

Neurons and Layer Labs

Inference In Code

Tuesday, 26 September 2023 07:12



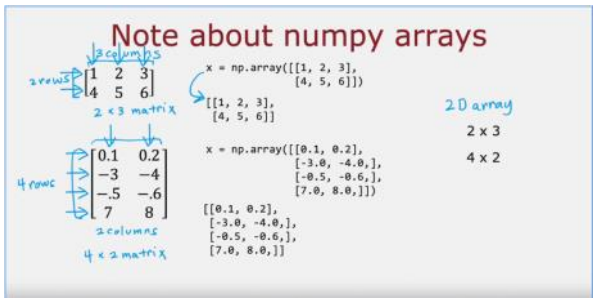
Second layer uses the output of first layer code etc - Forward pagination

Units are the amount of neurons

Then you can threshold after output layer

Data in TensorFlow (how tensorflow stores data)

Tuesday, 26 September 2023 07:38



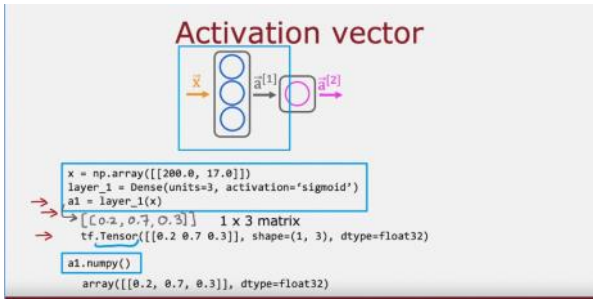
A matrix is a 2d array of data

Matrix can be lots of different sizes (4x2 etc)

Row Vector:

Column Vector:

TesnorFlow wants you to use Matrix's (2d arrays), because its meant to handle large datasets



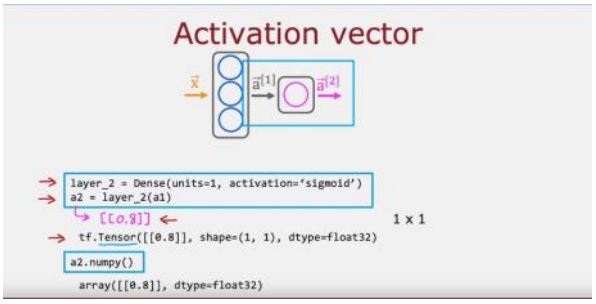
A1 is activation values for layer 1

2 row 1 column vector is produced by A1

Numpy and tensor arrays are different

A1.numpy() converts tensor matrix to numpy arrys

Layer 2 example:

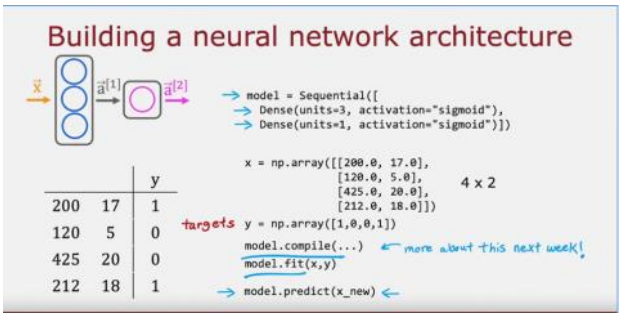


TensorFlow likes to convert to its own format called a Tensor

Building a neural network

Tuesday, 26 September 2023 08:20

Neural network code explained:




```
model = Sequential([
  Dense(units=3, activation="sigmoid"),
  Dense(units=1, activation="sigmoid")])
```

Creates a model sequence

```
x = np.array([[200.0, 17.0],
              [120.0, 5.0],
              [425.0, 20.0],
              [212.0, 18.0]])      4 x 2
targets y = np.array([1,0,0,1])
```

Your training data in matrix

```
model.compile(...)  more about this next week!
```

Compiles your model

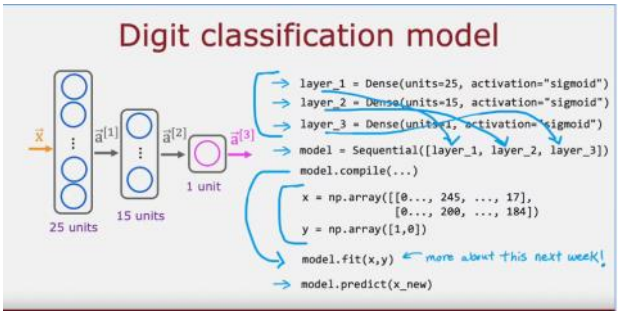
```
model.fit(x,y)
```

Trains with training data

```
model.predict(x_new)
```

Does forward propagation, feeds values from previous layer to next (left to right)

Digit Classification model example:



Simple Neural Network (Lab)

Tuesday, 26 September 2023 08:41

Normalization is feature scaling

Epochs controls how many times gradient decent sh ould cover entire database

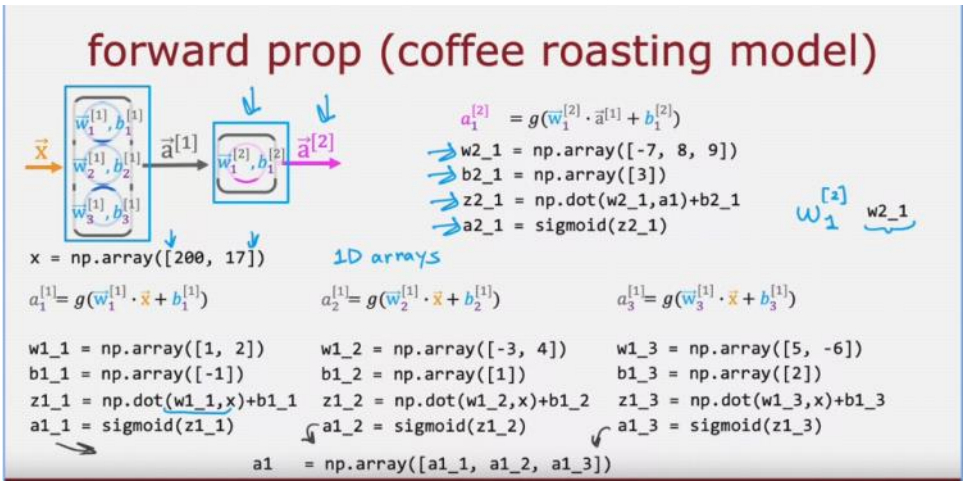
C2_W1_Lab 02_Coffe...

C2_W1_Lab 02_Coffe...

Forward prop in single layer

Tuesday, 26 September 2023 09:52

Non-Tensorflow version:



W2_1 means layer 2, neuron 1's weight etc (underscore means subscript)

Hard coding each neuron for forward prop

General implementation of forward propagation (weight matrix)

Wednesday, 27 September 2023 14:43

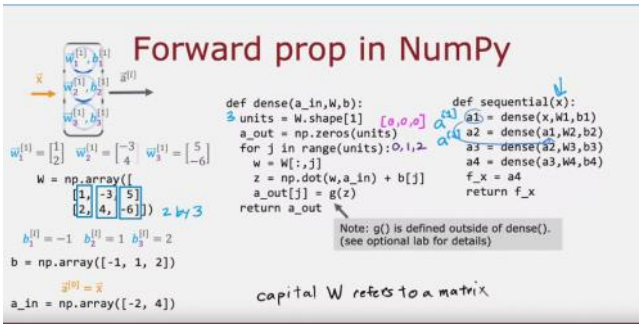
Forward Propagation is to compute your neural network from left to right using the previous layers activation values

Dense function:

Single layer of neural network

Inputs activation values from previous layer then produces next layer activiation values

---




```
W = np.array([
      [1, -3, 5]
      [2, 4, -6]]) 2 by 3
```

W should always be capital for matrix

W is loaded with fake data but this is the structure of the matrix

Each column in matrix is the weight vector for a neuron (first column is W1_1 etc)

```
3 units = W.shape[1]
```

Equals 3 because of shape of matrix -shape of w just gives us this information

```
a_out = np.zeros(units)
```

Makes a 3 zero array like units (w.shape)

```
for j in range(units):0,1,2
```

For loop runs for amount of units

```
  w = W[:,j]
```

Gets the weight column for each unit from matrix (first time 1,2, second time -3,4 etc)

```
z = np.dot(w,a_in) + b[j]
```

Calculates Z for sigmoid function (logistic regression)

```
a_out[j] = g(z)
```

Does activation function

Forward Propagation structure:

```
def sequential(x):
[1] a1 = dense(x,W1,b1)
[2] a2 = dense(a1,W2,b2)
    a3 = dense(a2,W3,b3)
    a4 = dense(a3,W4,b4)
    f_x = a4
    return f_x
```

Left to right computing

Taking in previous layers activiation values

Simple Neural Network (lab)

Thursday, 28 September 2023 06:28

C2_W1_Lab
03_Coffe...

C2_W1_Lab
03_Coffe...

Speculations On Artificial General Intelligence

Thursday, 28 September 2023 06:39

ANI - Artificial Narrow Intelligence models that do specific task, this has the most progress by far

AGI - Artificial General Intelligence, models that do many tasks, little progress here

Practice Lab: Neural Networks for Handwritten Digit Recognition, Binary

Thursday, 28 September 2023 09:57

C2_W1_Assignment

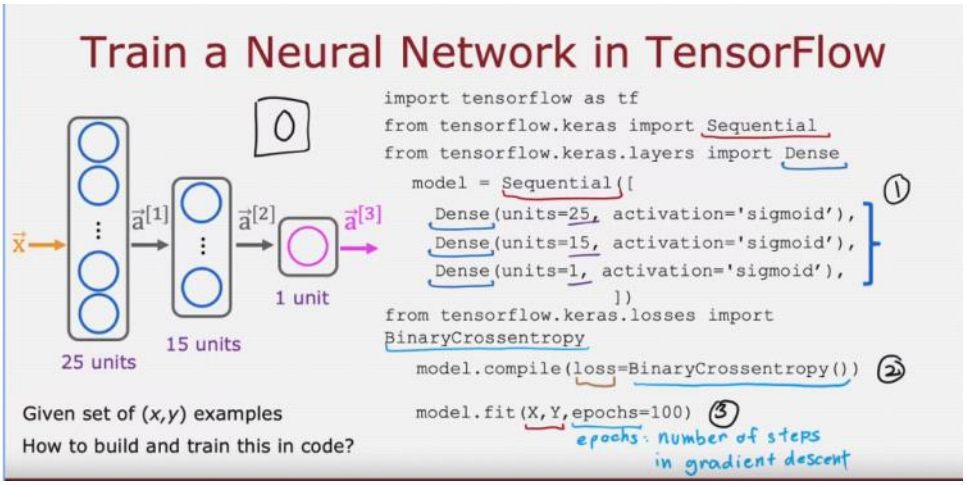
C2_W1_Assignment

Full of really good notes explaining everything

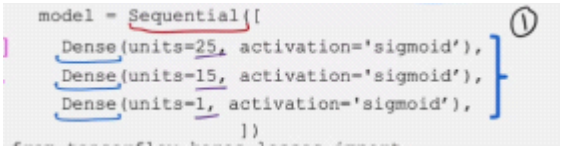
TensorFlow Implementation (Epochs)

Friday, 29 September 2023 06:26

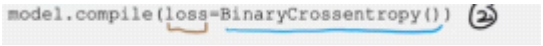
TensorFlow Neural Network + Training:



Epochs is how many steps you take in gradient descent



This is your model with forward pagination



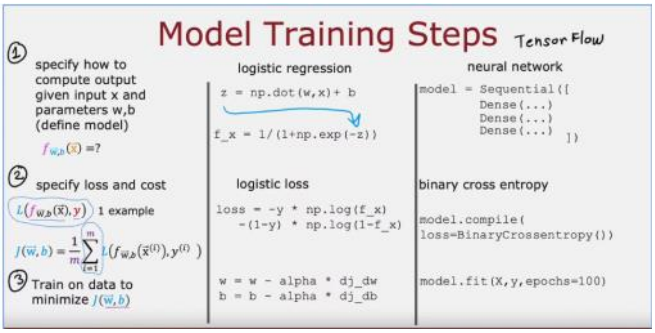
Your loss function is your training function

Training the model running 100 steps of gradient descent

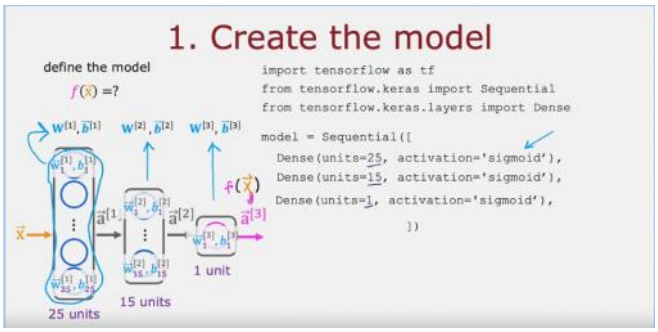
Training Details + Generally making a model

Friday, 29 September 2023 06:38

Neural Network Training:



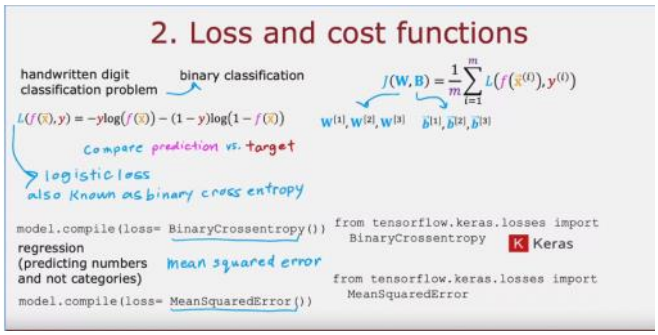
Create Model (1):



Sequential is forward prop

Dense defines each layer along with activation function

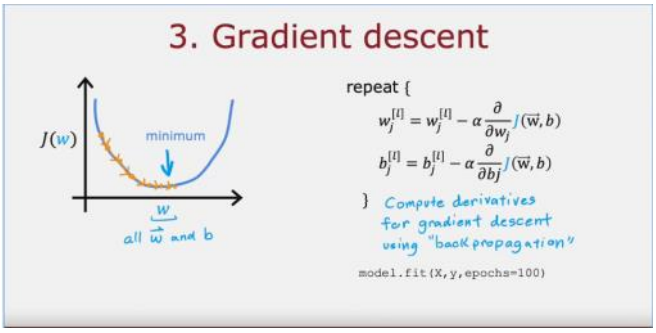
Loss and cost functions (2):



Loss defines your loss function and TensorFlow turns it into cost function

Google for more cost functions named weirdly

Gradient Descent(3):

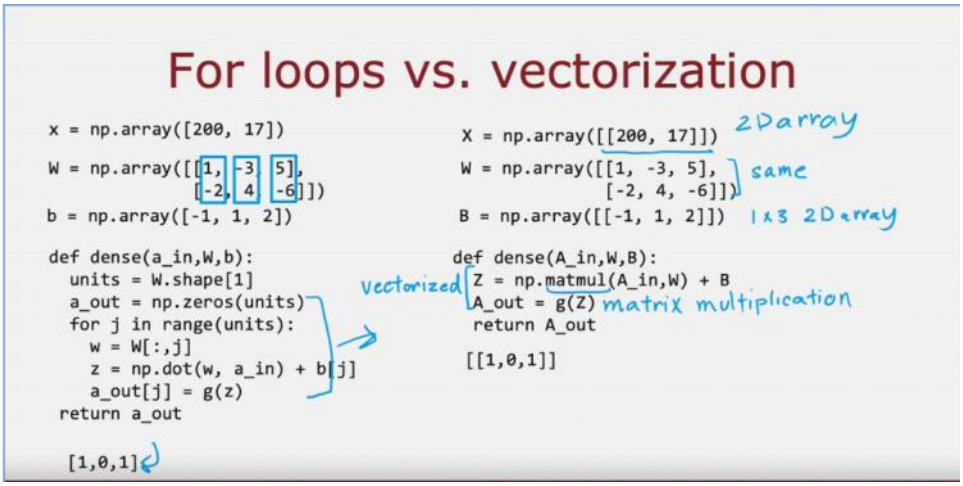


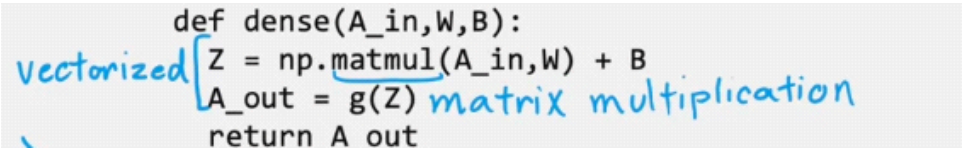
Model.fit() does gradient descent and back propagation

How Neural Networks are Implemented Efficiently (matmul)

Thursday, 28 September 2023 06:51

Forward Propagation With Vectors:





matmul - Times Activation values and Weights together because they are both matrix (vectorization)

Matrix multiplication

Thursday, 28 September 2023 07:07

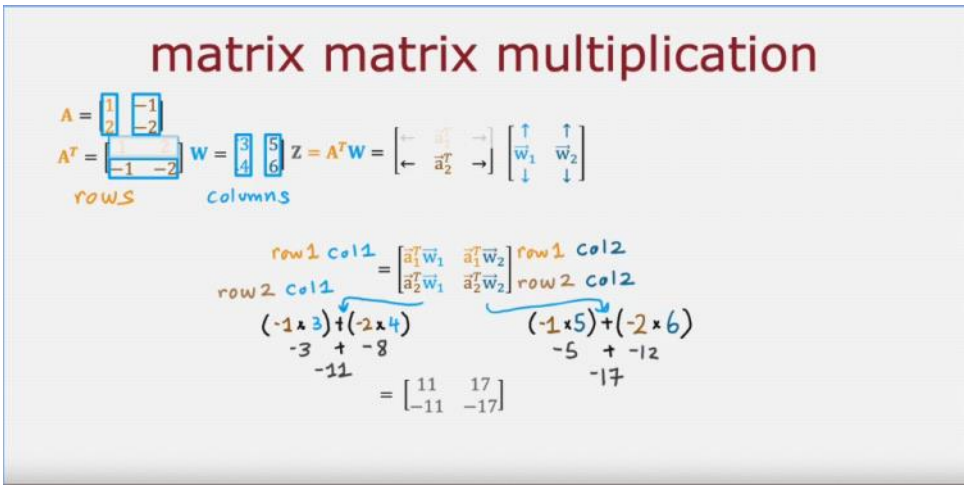
Matrix multiplication works by transposing data and multiplying it by columns

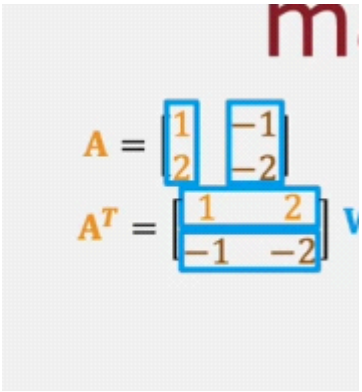
Transposing a matrix:

Lying a matrix out flat into rows from columns

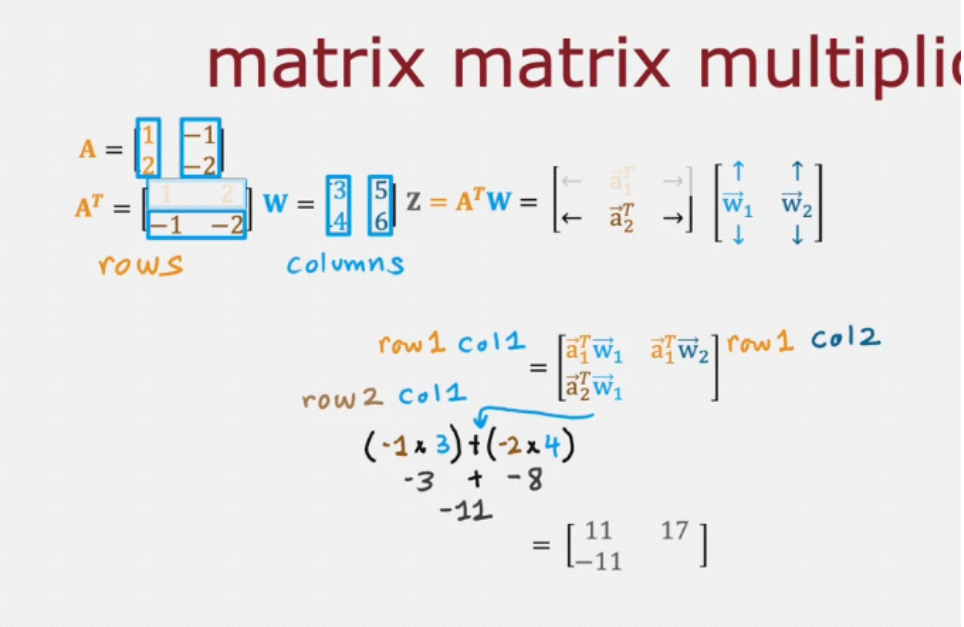
You must do transposing to do matrix multiplication

Matrix Matrix Multiplication:





Transposed A (flatten out into rows

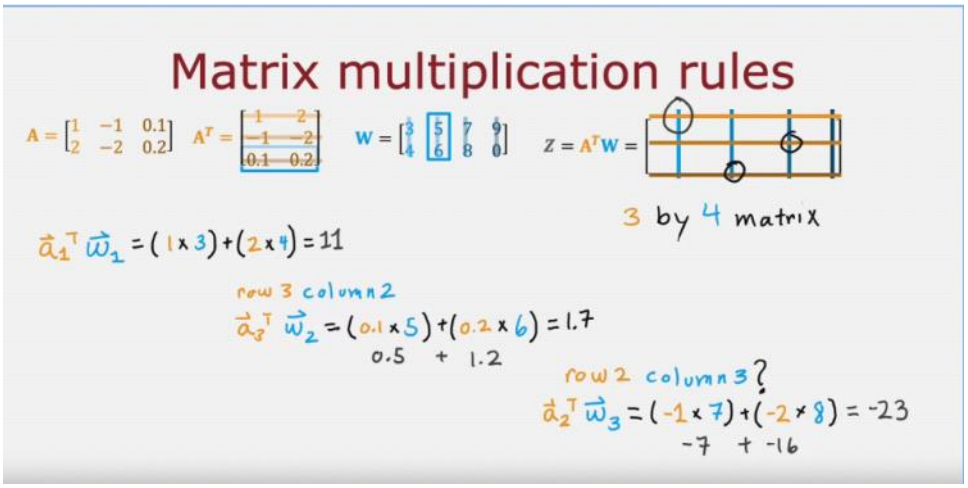


Each row of A (transpose A) is timed by both columns

Matrix Multiplication Rules

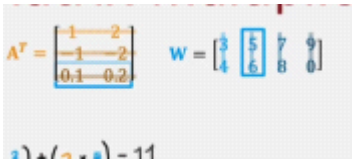
Thursday, 28 September 2023 07:35

Matrix Multiplication:



In matrix multiplication results, each row is a Activation value (transposed) matrix, each column is a different weighting matrix

When doing matrix multiplication:



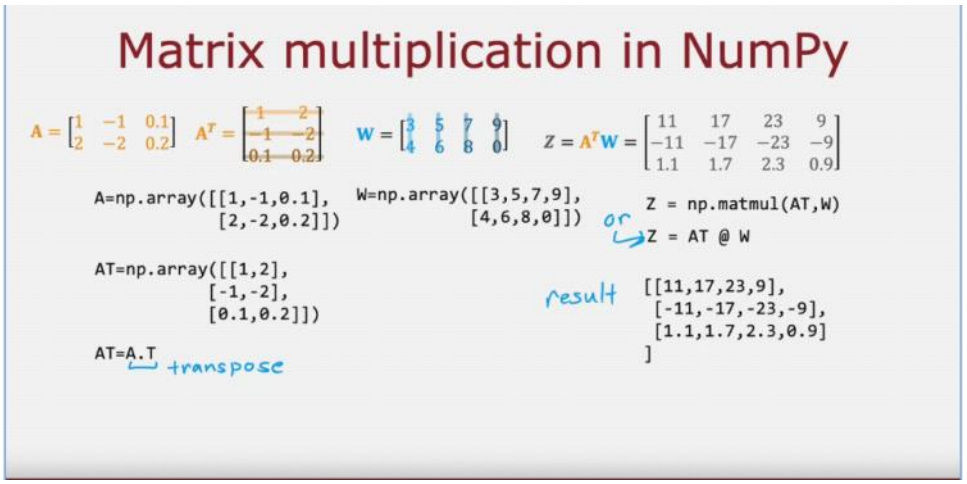
Your matrix must be the same LENGTH for dot product to work correctly

Results will always have the same number of rows as Activiation Values

Results will always have the same number of columns as W

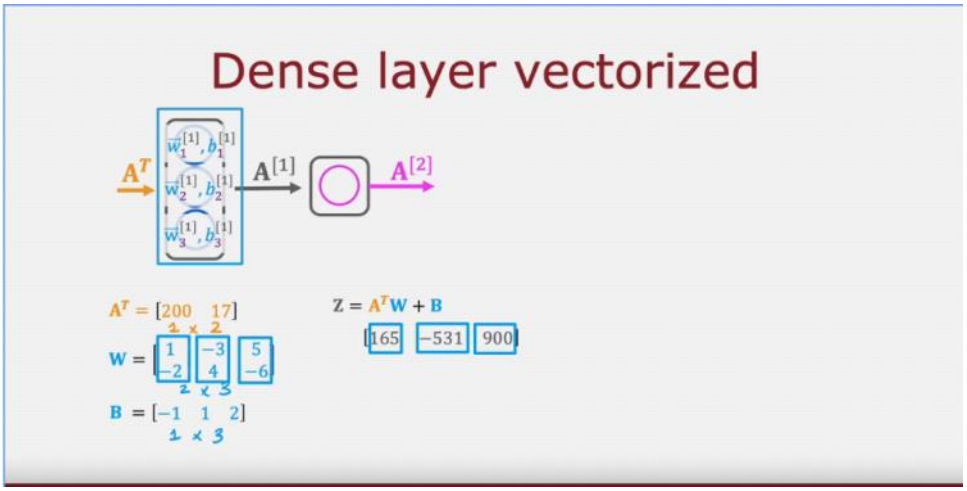
Matrix Multiplication Code

Thursday, 28 September 2023 08:09



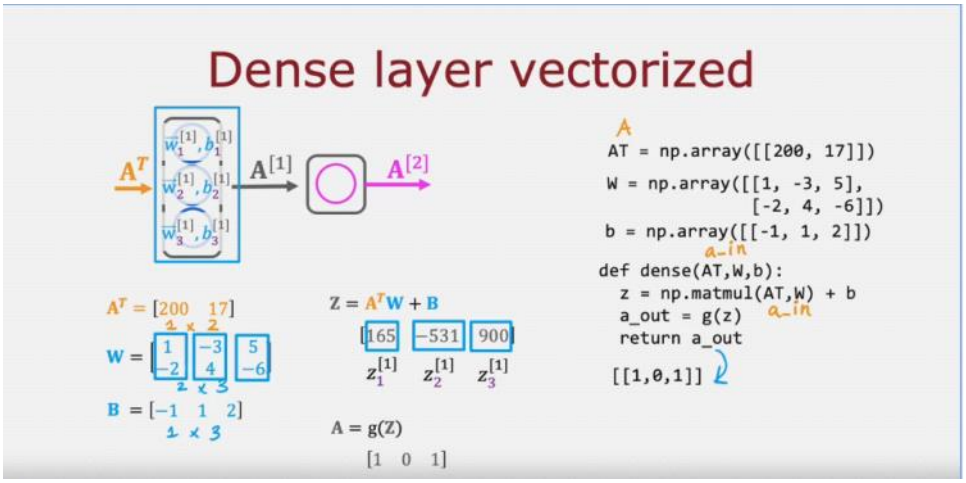
You must transpose your Activation Values before giving it to matmul

Dense Layer Vectorized:



Each activation value is multiplied by each column of W matrix to get Z

Vectorized Forward Prop:

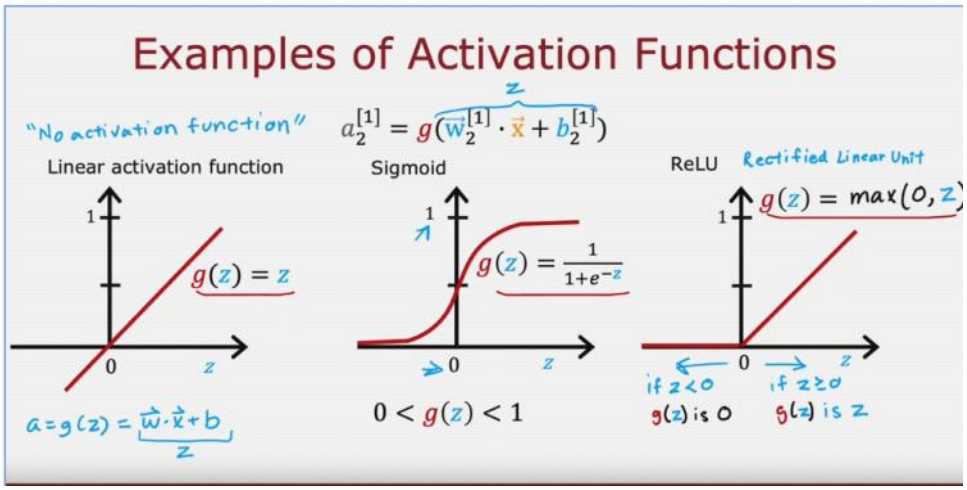


Using matmul to vectorize forward prop

Alternatives To The Sigmoid Activation

Friday, 29 September 2023 07:21

Choices for activation functions:



Linear Activation Function

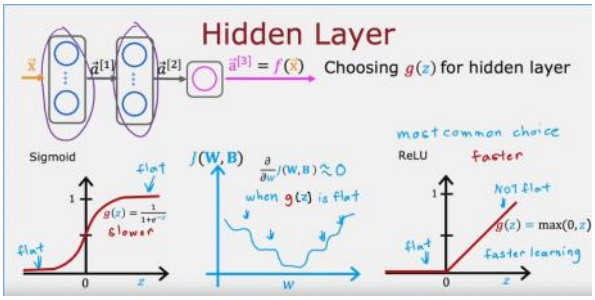
ReLU function

Sigmoid Function

Choosing Activation Functions

Friday, 29 September 2023 07:43

Hidden Layers Activation Functions:

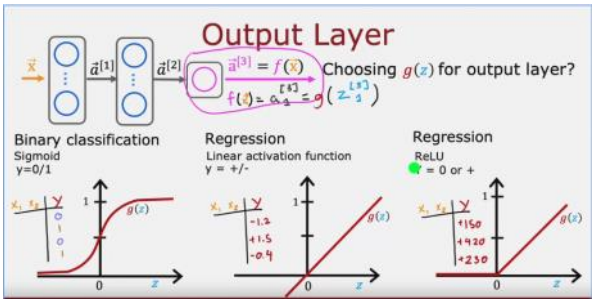


We used to use Sigmoid Functions, now we use ReLU most of the time (apart from output layer)

ReLU is faster than sigmoid when executing

ReLU less flat than sigmoid function, so much faster to train with gradient descent

You change your output activation function depending on requirements:

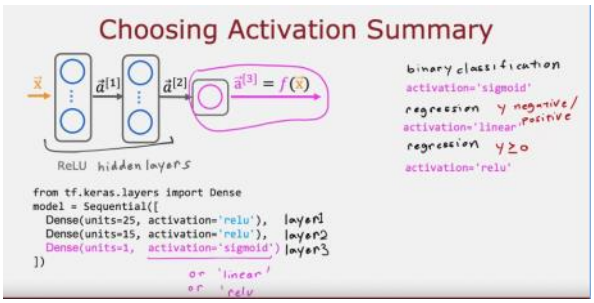


For binary classification - Use Sigmoid Function

For Regression (positive or negative output) - Use Linear Activation Function

For Regression (only positive output) - Use ReLU

Summary of Activation functions:



Output layers:

When classifying: use sigmoid

When doing positive or negative: use linear activation function

When doing only positive: use ReLU

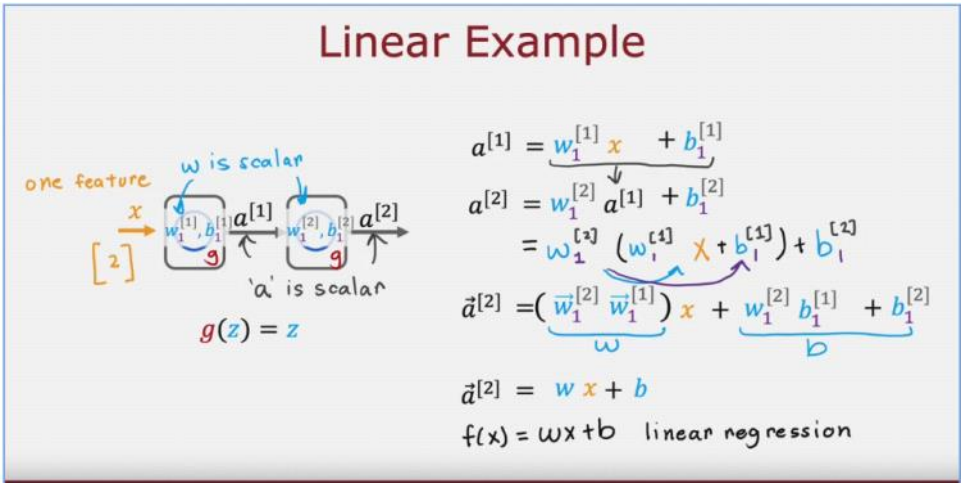
Hidden Layers before output:

Just use ReLU

Why Do We Need Activation Function?

Friday, 29 September 2023 08:18

There's no point making a neural network for linear activiation



A solely linear activation neural network cannot learn

Never use the linear activation function in hidden layers

ReLU Activation (Lab)

Saturday, 30 September 2023 11:46

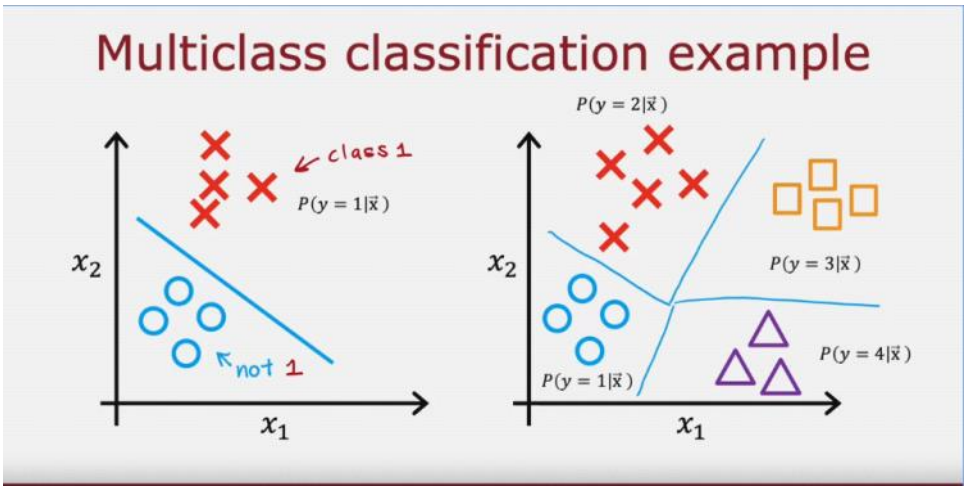
C2_W2_Relu

C2_W2_Relu - Jupyte...

Multiclass

Friday, 29 September 2023 09:01

Multiclass classification is when Y can be equal to lots of different classes



$P(y = 3|\vec{x})$

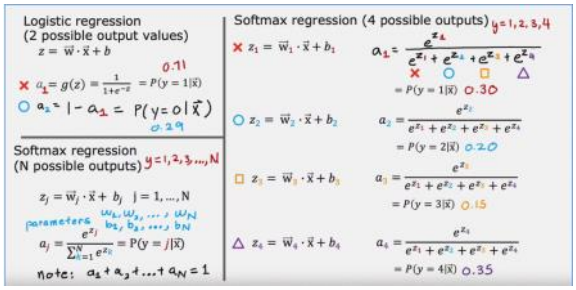
This means what is the chances Y = 3 given X etc

Softmax (Z probabilities)

Friday, 29 September 2023 09:11

Softmax regression is based on Linear Regression (sigmoid function)

With Softmax you should calculate Z (input same as Sigmoid) for each of its inputs



Basically, Softmax calculates the odds that an input belongs in each class

SoftMax formula:

$$z_j = \vec{w}_j \cdot \vec{x} + b_j \quad j = 1, \ldots, N$$

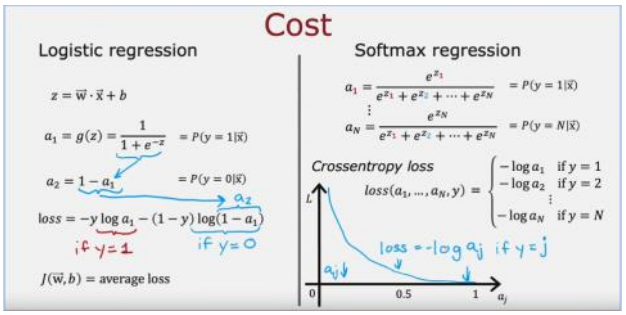
Calculate all your 'Z' inputs for each class first

$$a_j = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}} = P(y = j|\vec{x})$$

Then input them into the above softmax formula

E = numpy.exp()

Softmax Regression Cost:



Example:

If Y = 2 (example):

Then we get the log of a2

The closer to 2 our prediction was, the less loss

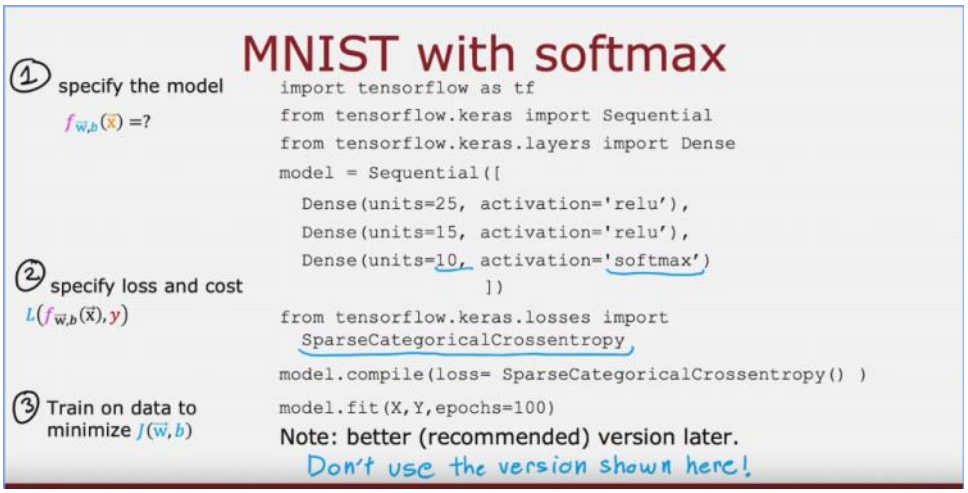
Softmax cost function calculates the loss between the correct category and one the model selected

Neural Network with SoftMax output

Saturday, 30 September 2023 12:13

Softmax calculates all the Z probabilities at once for all activation values

TesnorFlow with Softmax output (there's a better version coming:

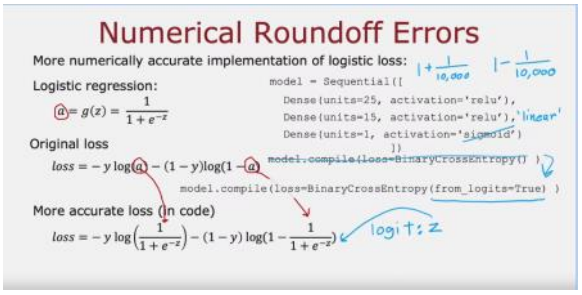


Improved implantation of Softmax (recommended) (Roundoff)

Saturday, 30 September 2023 12:27

When doing calculations on a computer you get round off error because computer doesn't have unlimited memory

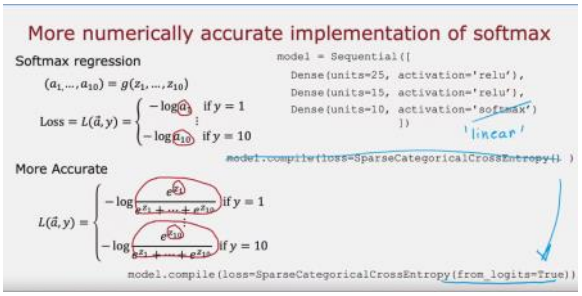
Example is Logistic Regression:



Stopping roundoff error during cost function:

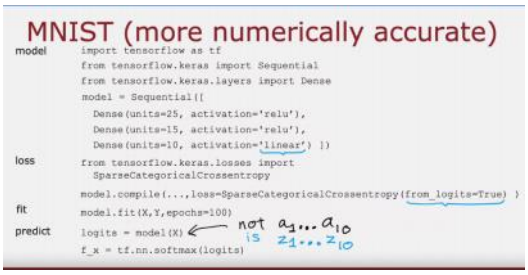
Allows TensorFlow to calculate Z as intermediate value reducing roundoff error

Example for Softmax:



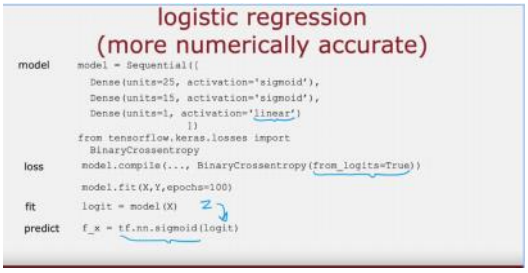
(from_logits=True) Stops roundoff errors due to small numbers with Z

Softmax Full Model:



Activation Function Changed ^

Logistic Regression:

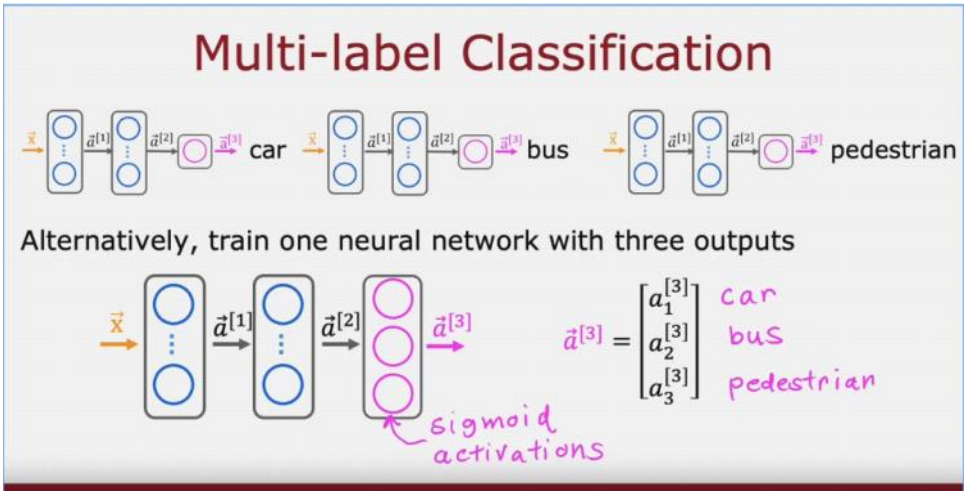


Activation Function Changed ^

Classification with multiple outputs

Tuesday, 3 October 2023 06:37

Muti label classification is when Y could fit multiple categories

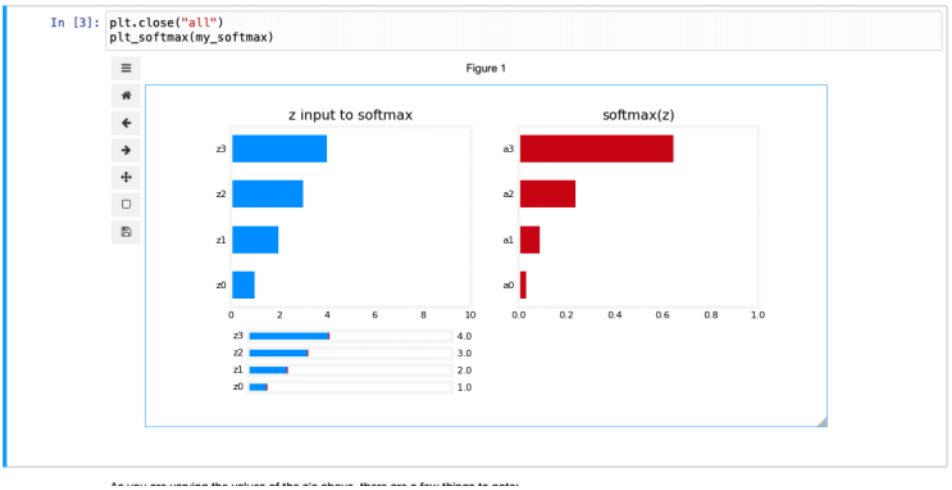


Use a sigmoid function for binary classification of output layer

3 neurons in output layer means 3 outputs overall

Softmax (Lab + Notes)

Tuesday, 3 October 2023 07:04



In Softmax classification your total probabilities must add to 1

C2_W2_Sof
tMax - Ju...

C2_W2_Sof
tMax

Mutliclass (Lab)

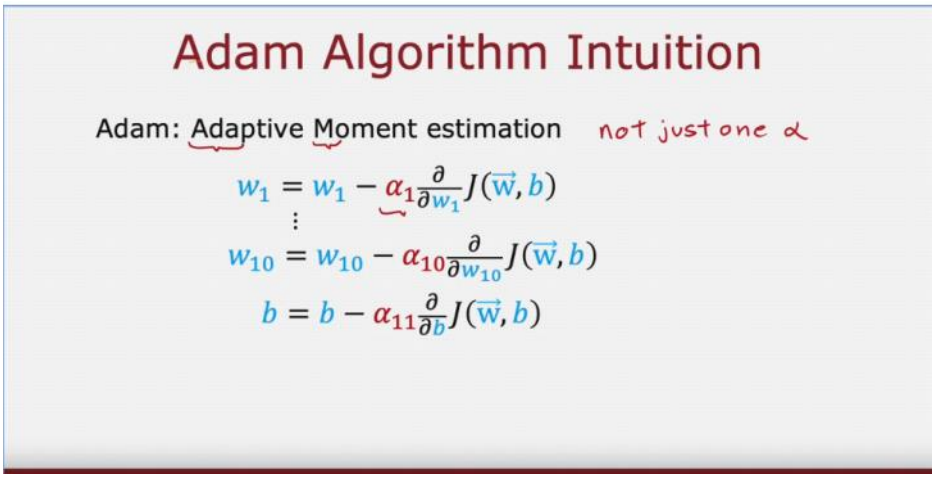
Tuesday, 3 October 2023 07:27

C2_W2_Multiclass_TF

C2_W2_Multiclass_T...

Advanced Optimization (Learning rate + bias)

Tuesday, 3 October 2023 07:29

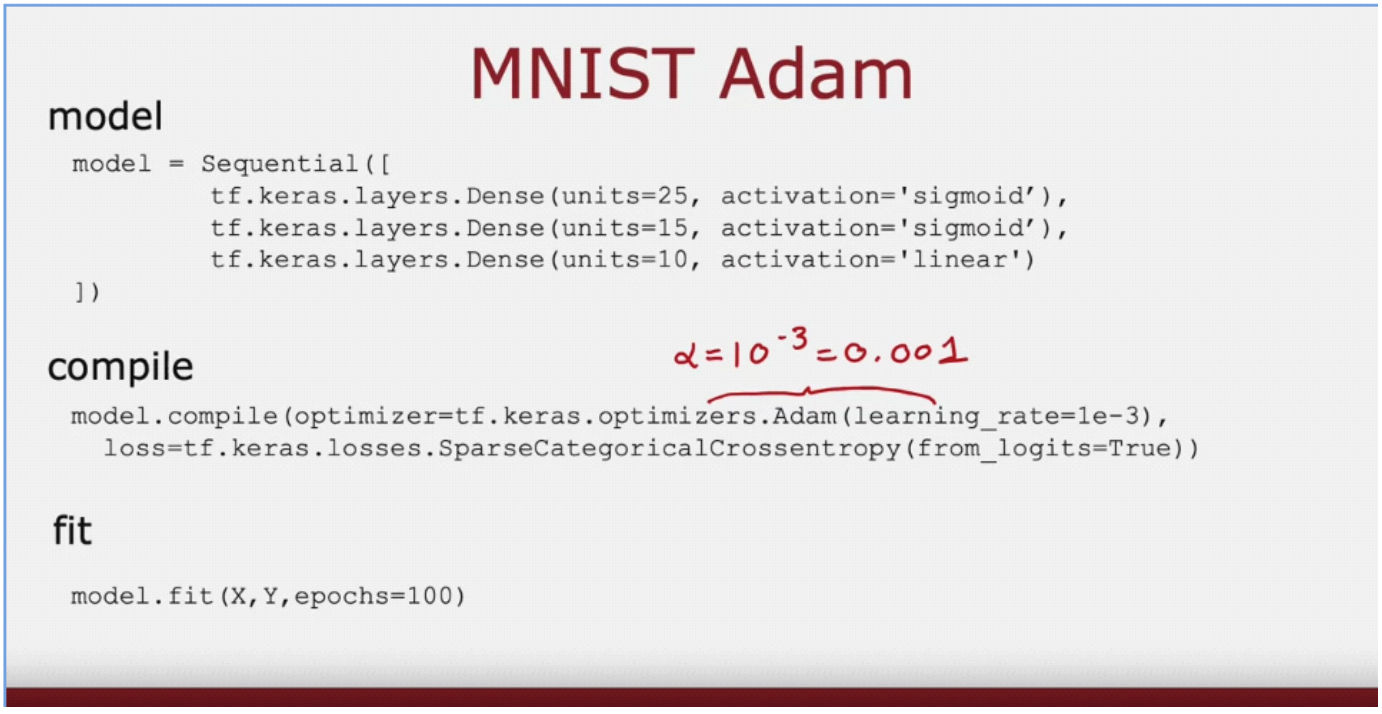


The Adam Algorithm gives each feature a separate learning rate including bias

If a parameter (feature) keeps moving in same direction, increase learning rate

If a parameter is "bouncing" decrease learning rate

Code for Adam Algorithm:



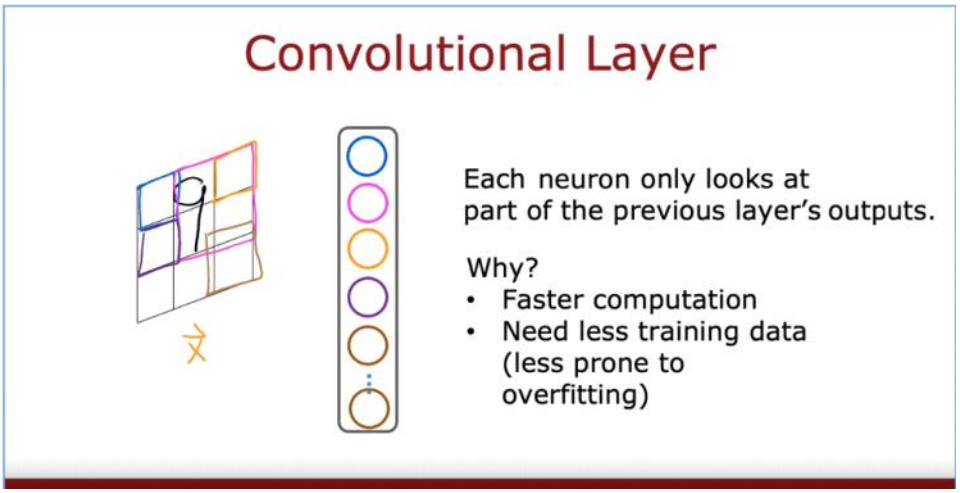
Above we set the initial learning rate

Additional Layer Types (CNN VS RNN)

Tuesday, 3 October 2023 08:11

RNN have the hidden layers who's inputs and outputs can't be controlled.

Convolutional Layer neurons can be limited to only use part of the input data



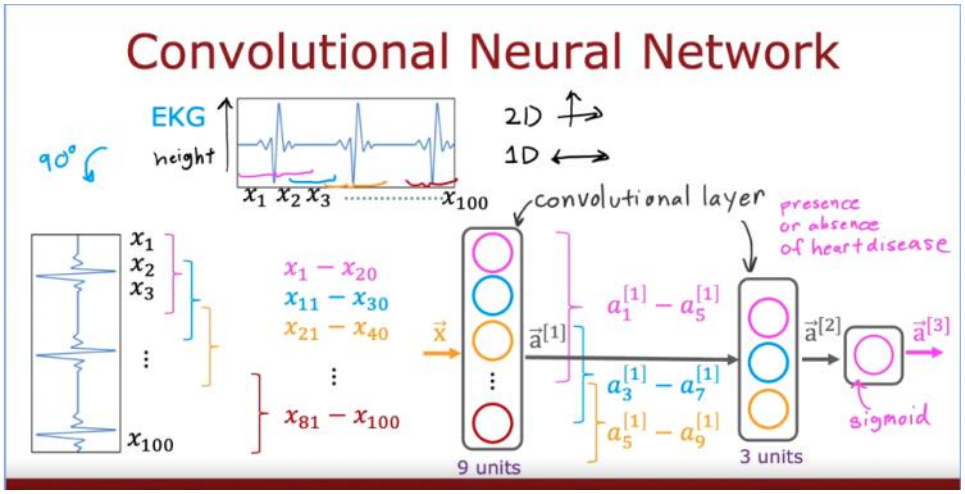
This means:

Faster computation

Less training data required

Less prone to overfitting

Convolutional Neural Network:



Gives us more choice to design, because we decide what areas we want each neuron to focus on

This is done with filters, we decide what pattern each neuron looks for.

Then each neuron gives a score (array of numbers) of how closely the image (example) makes the filter, going over the entire image

As you go deeper into the network the filters get more abstract, first layer might look for a door pattern, 5th layer filter might look for a house.

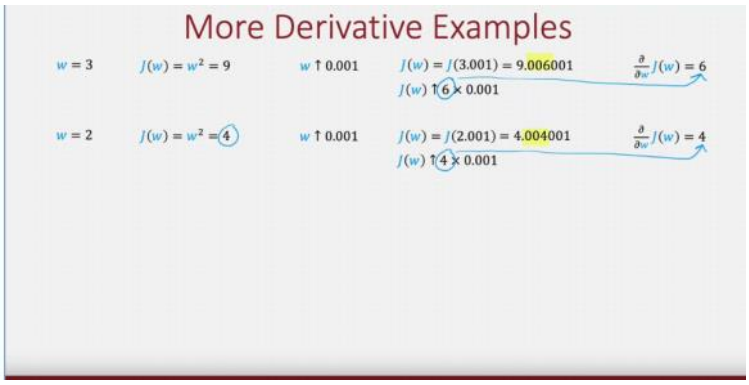
Make them good at looking for patterns.

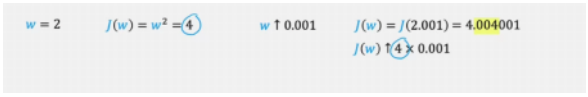
What is a derivative? (Epsilon)

Tuesday, 3 October 2023 09:10

A Derivative is a measure of the impact of changing the parameter on a functions ouput

Epsilon - a very small number we change our function value by to find derivative





Increasing W by:

Made J(W) go up by:

Making the derivative 4 because the impact of the parameter change is 4

Code for calculating Derivative:

Model Function output = J

Weight test = W

This code measures the difference

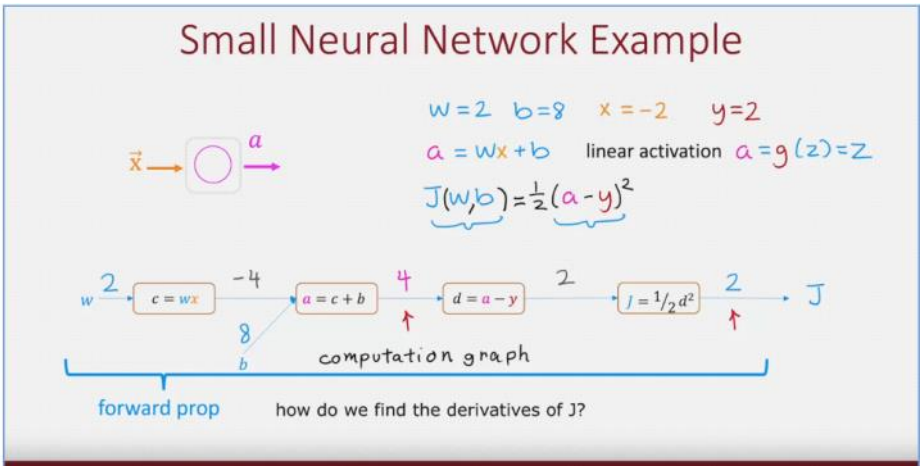
Ssss

Derivative means: How much does J(W)'s output change when W goes up by Epsilon

Computation Graph (back prop)

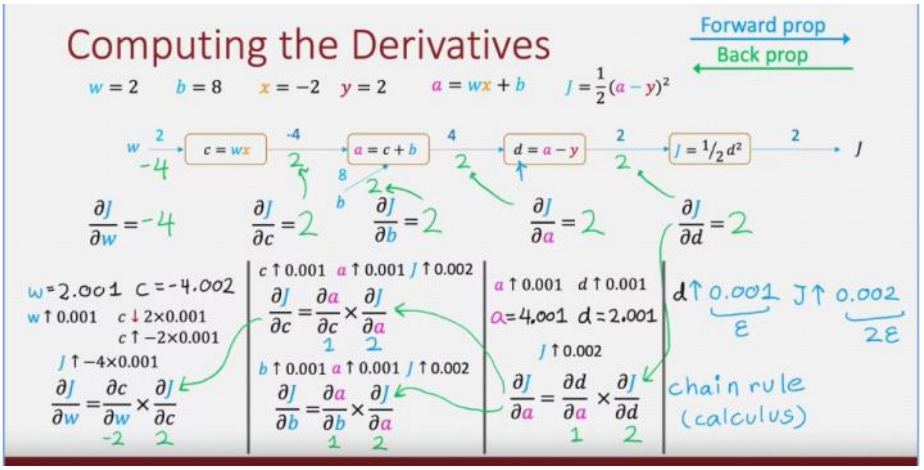
Friday, 6 October 2023 06:31

A computation graph is a set of nodes connected up (in the case of back prop, cost function result)



Back Prop:

Calculate all the different derivatives going backwards (right to left)



Calculate how much impact each calculation had on the next one (going right to left)

Back prop works by working out how much each layer contributed (derivative) the error produced by the cost function

Then we can correct each layer during training to minimize cost

Derivatives (Lab) + Info

Friday, 6 October 2023 06:53

C2_W2_Derivatives

C2_W2_Derivatives -...

Propagation Using A Computation Graph (Lab) + Info

Friday, 6 October 2023 06:58

C2_W2_Backprop

C2_W2_Backprop - Ju...

Lab

Thursday, 5 October 2023 16:30

For Softmax calculate Z dominator first always

```
### START CODE HERE ###
tf.keras.Input(shape=(400,)),
```

Above is required when creating a tensorflow model

> **Note:** It is also possible to add an input layer that specifies the input dimension of the first layer. For example:
> `tf.keras.Input(shape=(400,)),  #specify input shape`
> We will include that here to illuminate some model sizing.

Shape is the expected matrix size, 400 dimensional vectors

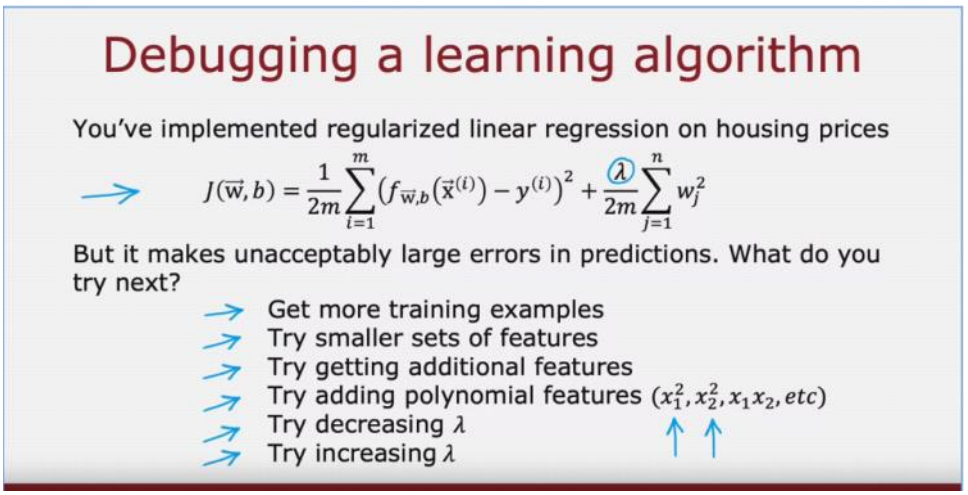
C2_W2_Assignment

C2_W2_Assignment

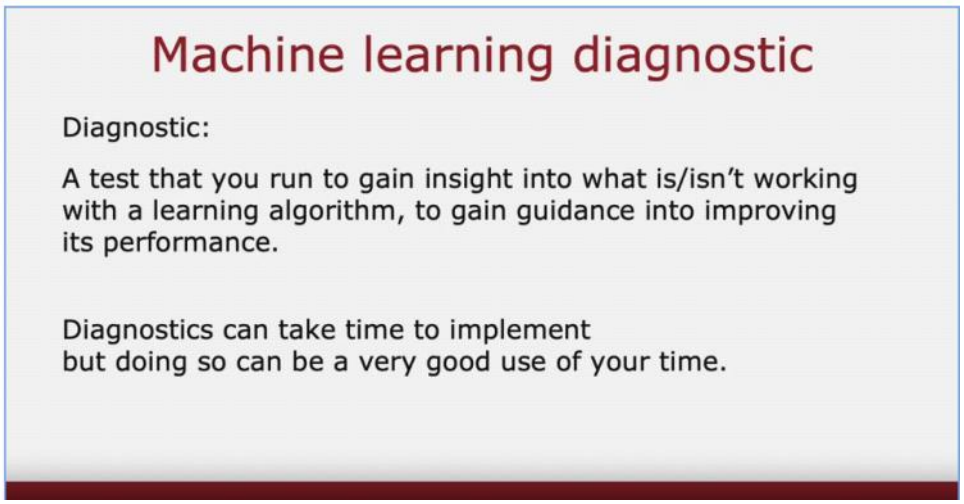
Deciding What To Try Next (Fixing)

Friday, 6 October 2023 07:06

There are lots of things you can do to fix a bad model:



Running a diagnostic test will tell you what's wrong

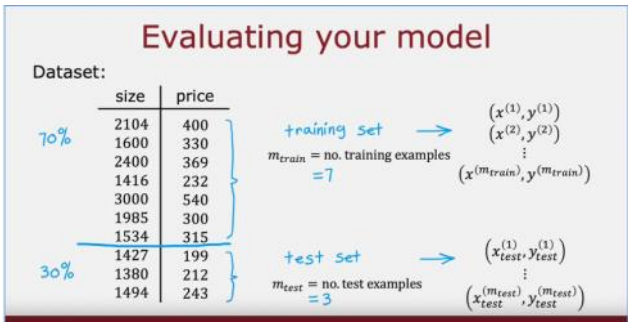


Evaluating a model

Friday, 6 October 2023 07:12

Various ways to evaluate your model:

Splitting your dataset:

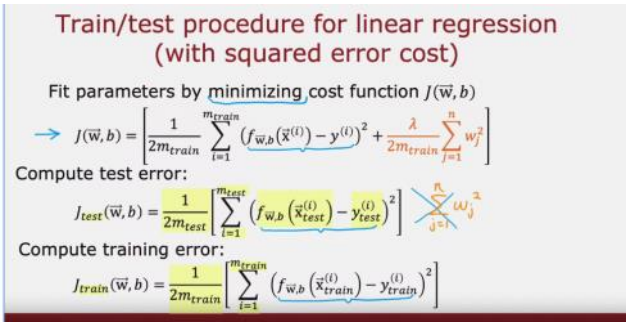


Training set - For teaching the model

Test Set - For testing the model

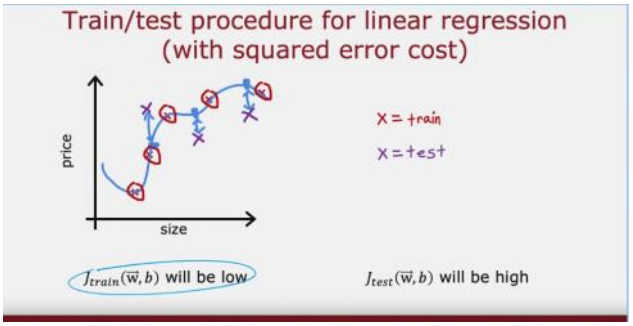
80/20 split

Calculating Test and Training Error (Linear Regression):



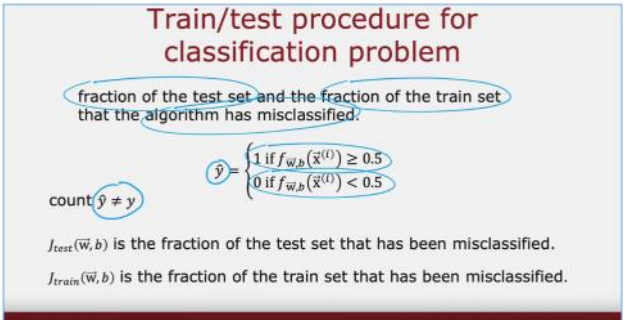
Test Error - Different between the correct answer and your predictions (much more accurate prediction for general error)

Training Error - Different between the training set and the correct answer on data its seen before



Evaluating a classification problem:

For classification you calculate the fraction of the test set that is mis-classified:



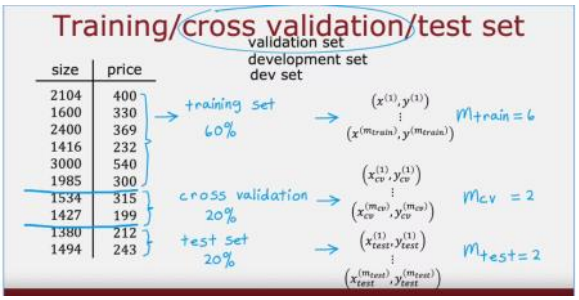
E.g. 61/200

Model Selection And Training / Cross Validation (choosing)

Friday, 6 October 2023 07:47

You can't just use the test error to see how good a model is, because its often lower / more optimistic then the real error rate

Cross Validation:



Split your dataset into 3

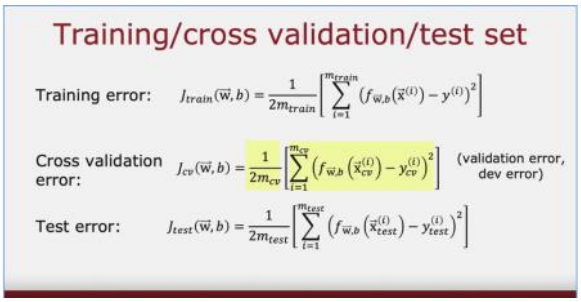
Training set - what you train your model on

Cross validation - for cross checking how valid your model is

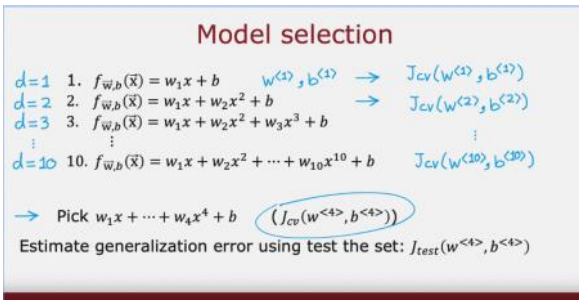
Test Set - For testing your model on new data

Evaluating a model:

Calculate:

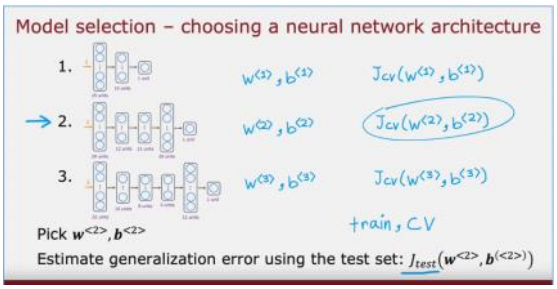


Pick the model with the LOWEST Cross Validation error (Jcv)



Above we are choosing what model to use given there polynomials by there Jcv / Cross Validation Error

Cross validation Error can be used to pick network design as well once model is chosen



Pick the one with the lowest (JCV)

For all Decisions use Cross Validation Error & Training set

Once you have picked a model, use a test set

This makes your test set fair and not optimistic

Model Evolution And Selection (Lab)

Friday, 6 October 2023 17:33

C2W3_Lab_
01_Mode...

C2W3_Lab_
01_Mode...

Cross validation techniques (10 fold, tuning parameters)

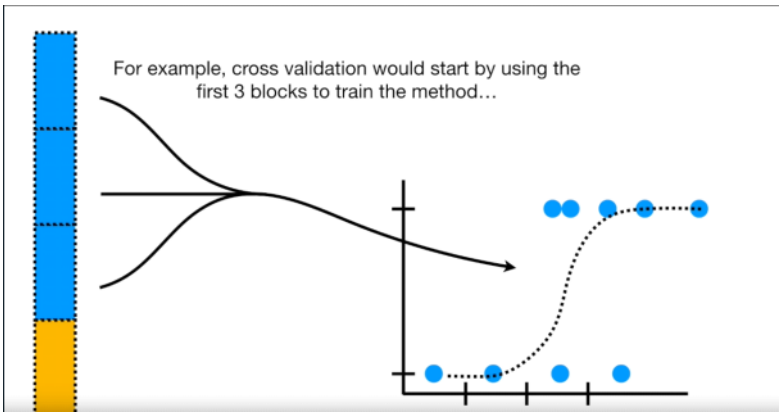
19 November 2024 20:45

Cross validation is a way of testing to see if your model will work well on new data.

Never we reuse your training data to test your model it won't work.

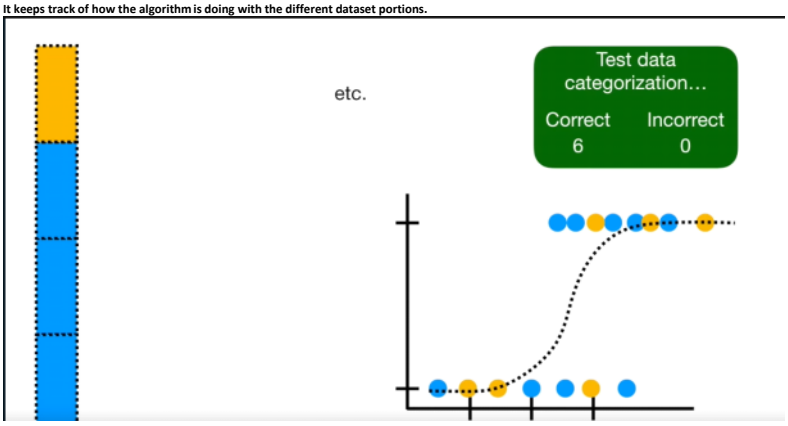
When deciding what data to use for testing you use cross validation.

Cross validation uses all the data one at a time, then summarizes the results at the end



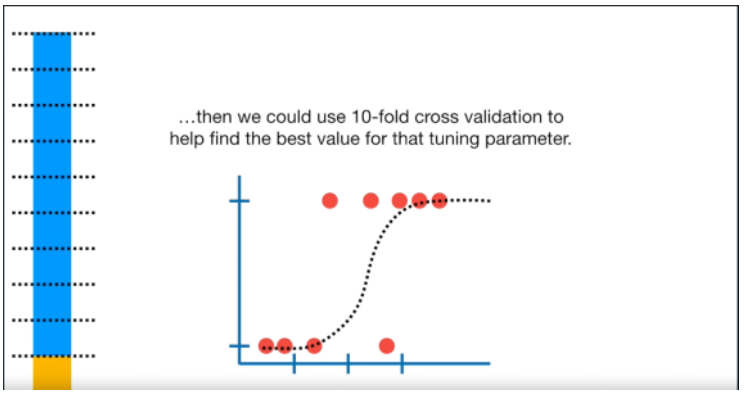
the data is divided into blocks, cross validations might start by using the first 75% for training and the last 25% (4 fold cross validation) for testing as above. Then it will keep changing which blocks it is using.

It keeps track of how the algorithm is doing with the different dataset portions.



Tuning Parameters:

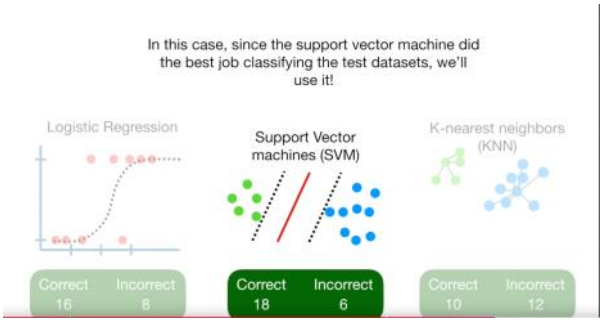
If you are using an algorithm that has parameters that aren't trained, you can use cross validation to compare a choice of values



We can use this to decide which algorithm and/or which configuration of model to use

The score from cross validation depends on how well the model does on all portions (switching testing and training blocks)

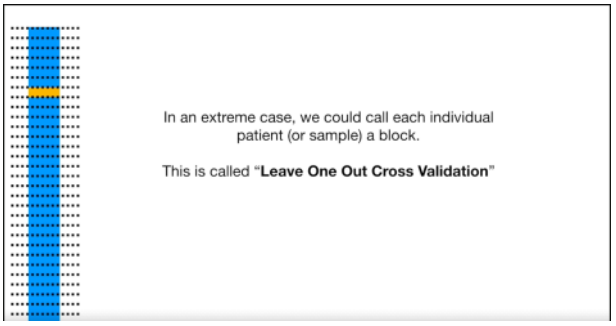
Example for picking a model:



How many blocks the dataset is divided into:

In example to the right, we are doing "Four Fold Cross validation" - Because we have 4 blocks

"Leave One Out Cross validation" - Each sample is tested individually



But most of the time you will use:

"Ten Fold Cross validation" - When the data is divided into 10 blocks

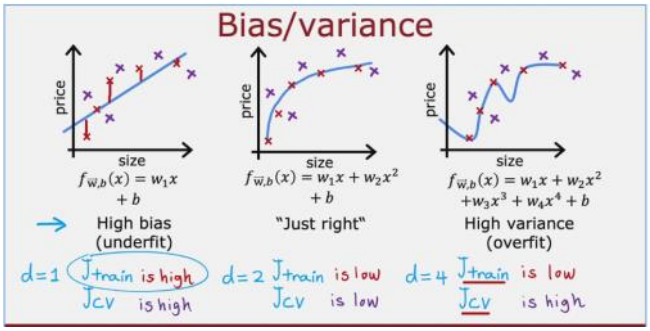
K-folds cross validation

19 November 2024 20:44

Diagnosing Bias and Variance (check overfit etc)

Friday, 6 October 2023 08:50

This is about working out if your model is under or over fit



Underfit:

Train Error: High

Cross Validation: High

"Just right":

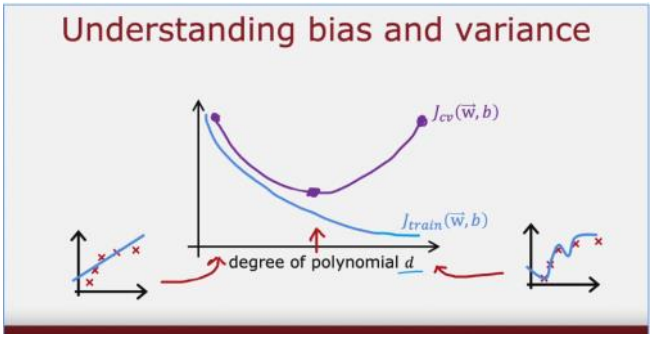
Train Error: Low

Cross Validation: Low

Overfit:

Train Error: Low

Cross Validation: High

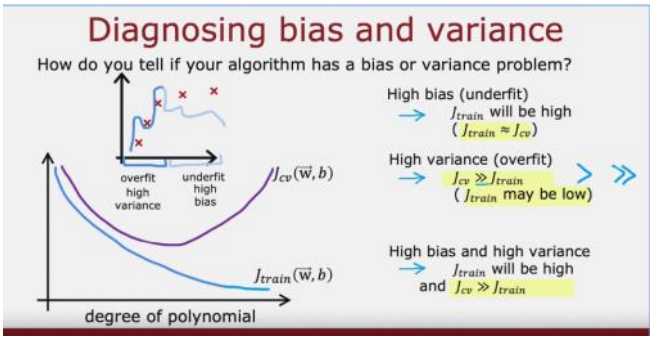


Jtrain: As model gets more polynomials it fits dataset better

Jcv: Fits better for a while, then gets overfit

Choose the point both lines are closest

Summary:



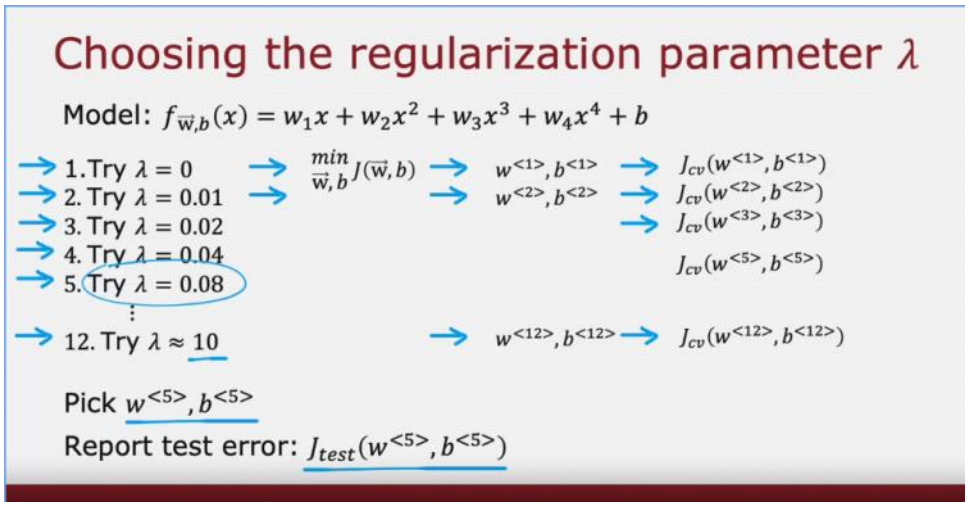
Regularization And Bias / Variance (regularization) (New Thing added)

Friday, 6 October 2023 09:28

To fix overfitting (High Variance) DO REGULARIZATION)

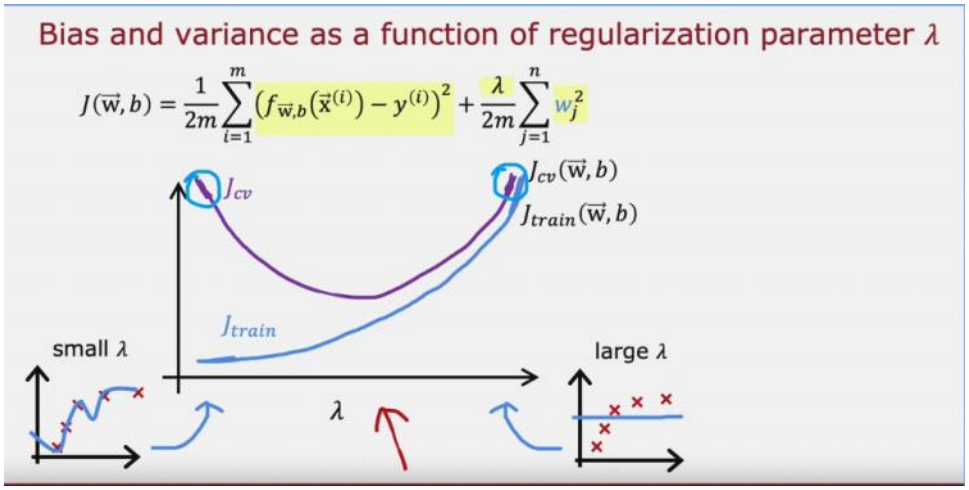
We know when our data is overfit because of training error and cross validation patterns (previous note)

Choosing the regularization parameter:



Try different regularization parameters and pick the one with the lowest cross validation

Then calculate the Test set error afterwards to confirm



As lambda (regularization) increases, Jtrain error will increase

Regularization can cause under and over fitting if too big or too small, that's why Jcv is the U shape

Pick the one where the lines are closest

Establishing a baseline level of performance (also high bias, high variance gaps)

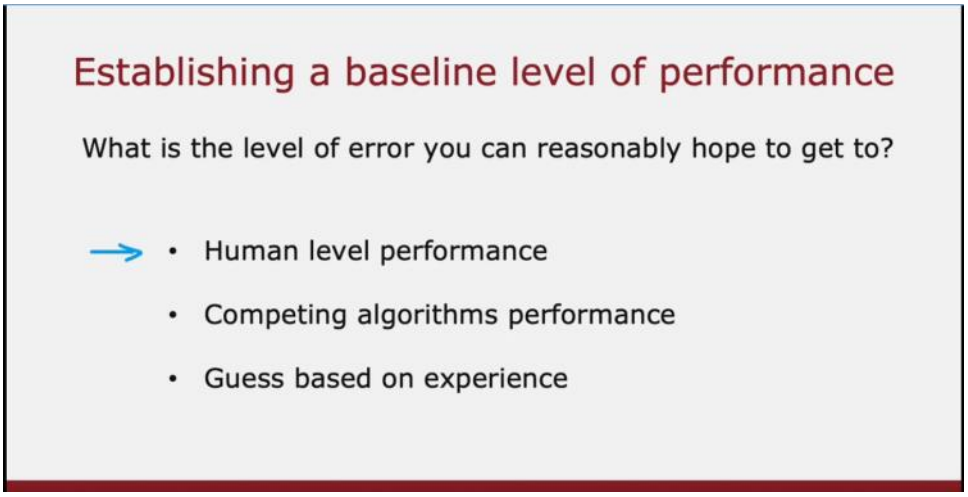
Monday, 9 October 2023 06:36

Side Note:

To summarise, A model with a high bias error underfits data and makes very simplistic assumptions on it. A model with a high variance error overfits the data and learns too much from it. A good model is where both Bias and Variance errors are balanced.

Performance baseline - what you can expect your model to achieve

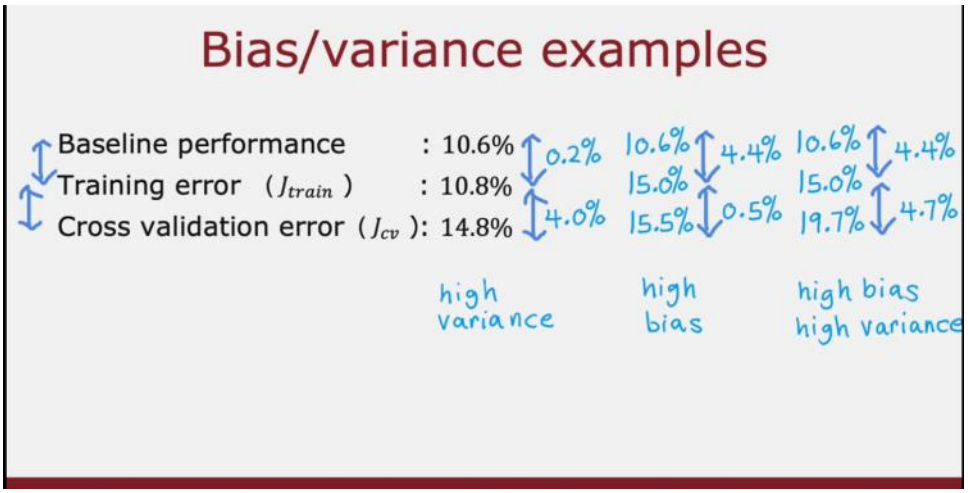
Types of baseline:



Human level performance - how well a human does this task

Competing algorithms performance - how well others models have done

Guess based on experience - given previous models you have made



High variance - overfit

High bias - underfit

If gap between baseline and training is smaller than gap between JCV and training - underfit

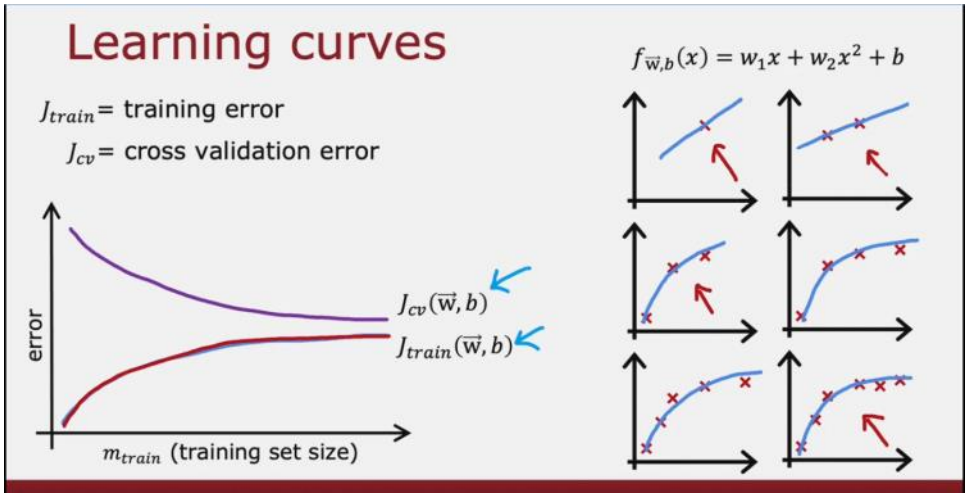
If JCV bigger than gap between baseline and training - overfit

If all 3 are large - high bias + high variance (fits part of the data)

Learning Curves (More data + Jtrain increase)

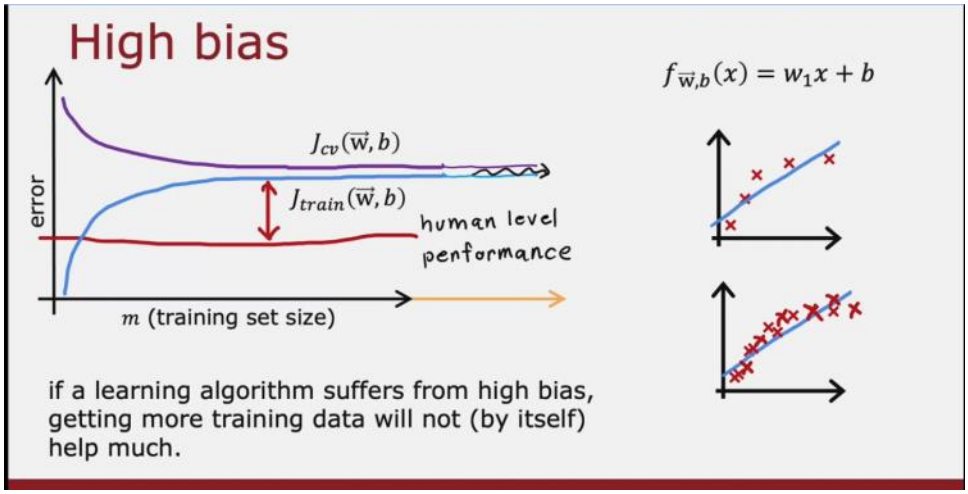
Monday, 9 October 2023 07:02

Learning Curves - Tell you how well your model model is doing given its experience



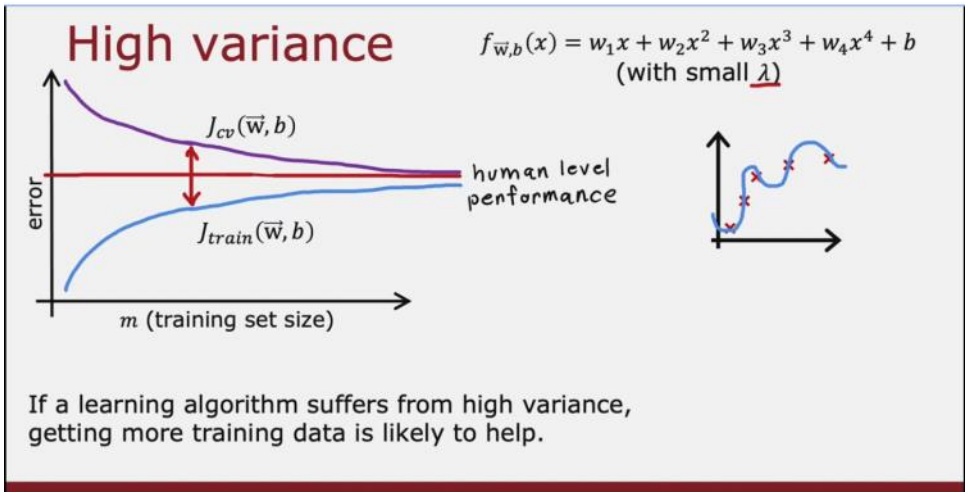
As you get more training data - your Jtrain gets bigger because harder to fit all the points (high variance)

High bias Learning Curve:



If your model is underfit (high bias) - getting more training data will not fix it alone

High Variance:

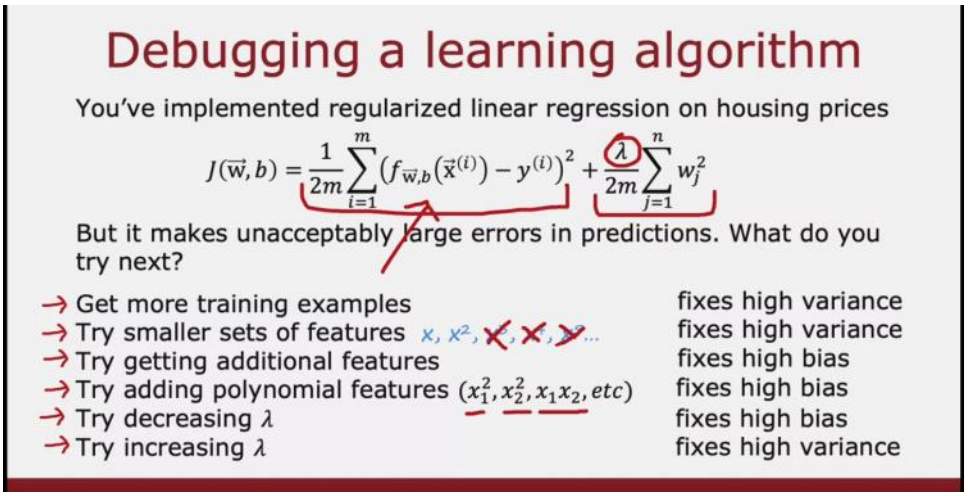


If your model has high variance (overfitting) - adding more data can help

Deciding What To Try Next Revisited (Removing data)

Monday, 9 October 2023 07:40

Things to try:

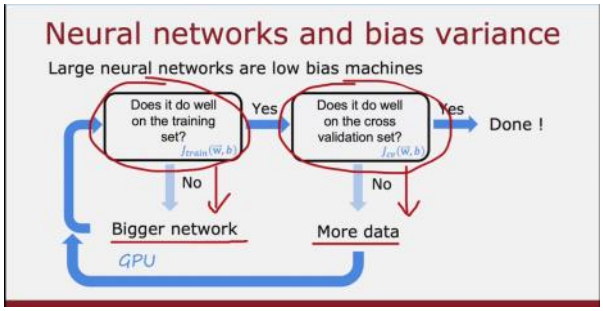


Don't remove training data to fix high bias

Bias / Variance and Neural Networks (New Line added)

Monday, 9 October 2023 08:02

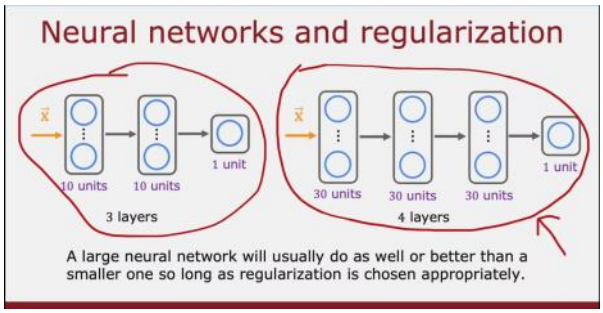
The "simple recipe":



Bigger network is good but expensive - high bias

More data is good but not always possible - high variance

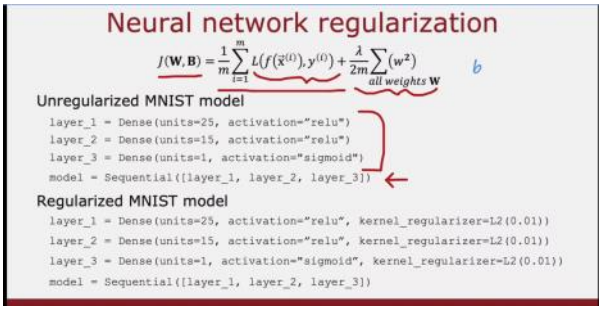
In large Neural Nets, you will be fighting variance (overfitting) mostly



As long as you choose the right neural network larger the better, apart from more expensive hardware wise

Larger Neural Network is better as long as you regularize correctly

Regularizing in TensorFlow:

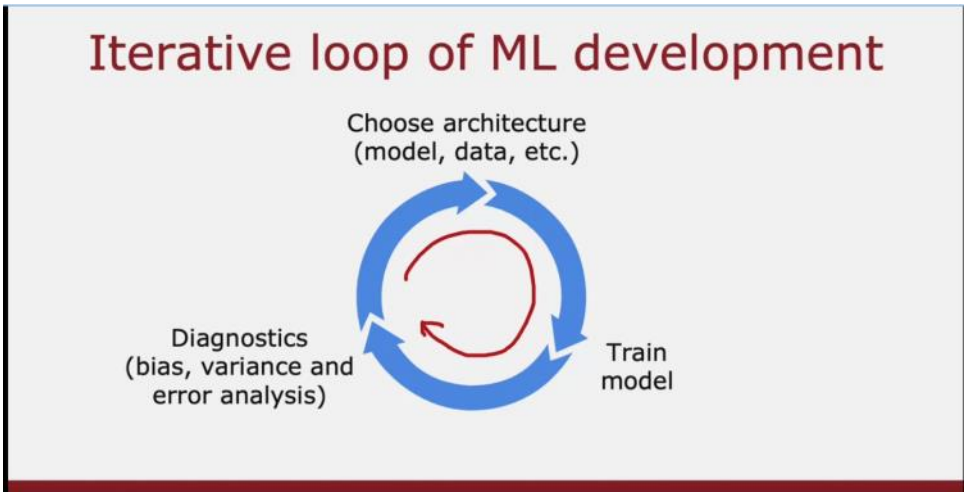


To fix overfitting (High Variance) DO REGULARIZATION)

Iterative loop of ML Development

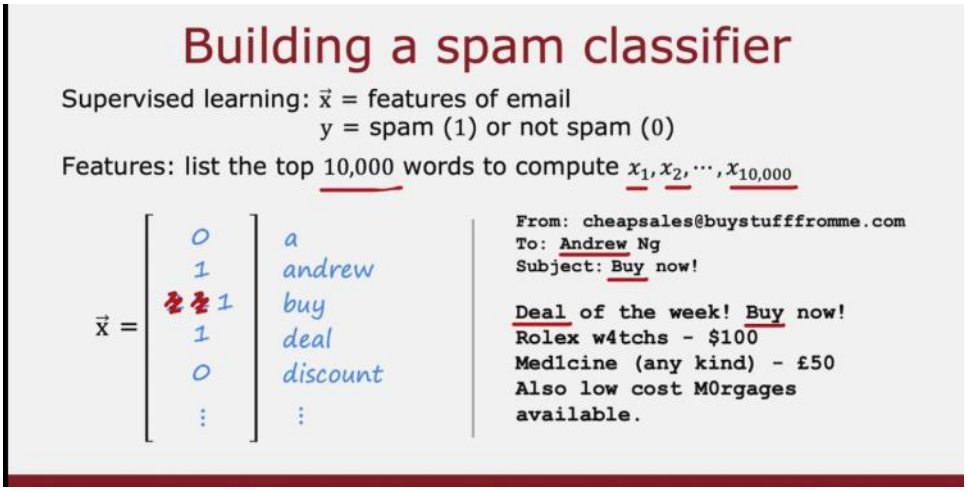
Monday, 9 October 2023 08:37

Development Loop:

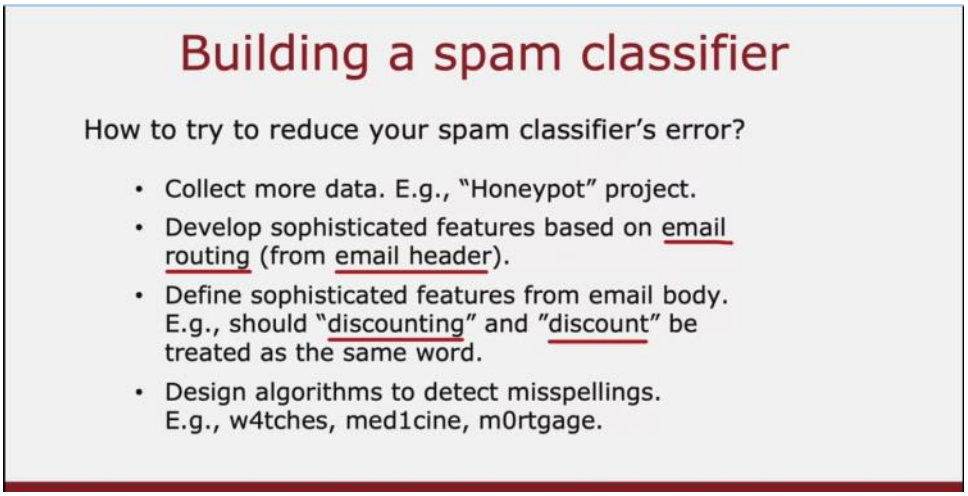


Keep going round this

Building Spam Classifier:



Training each feature…



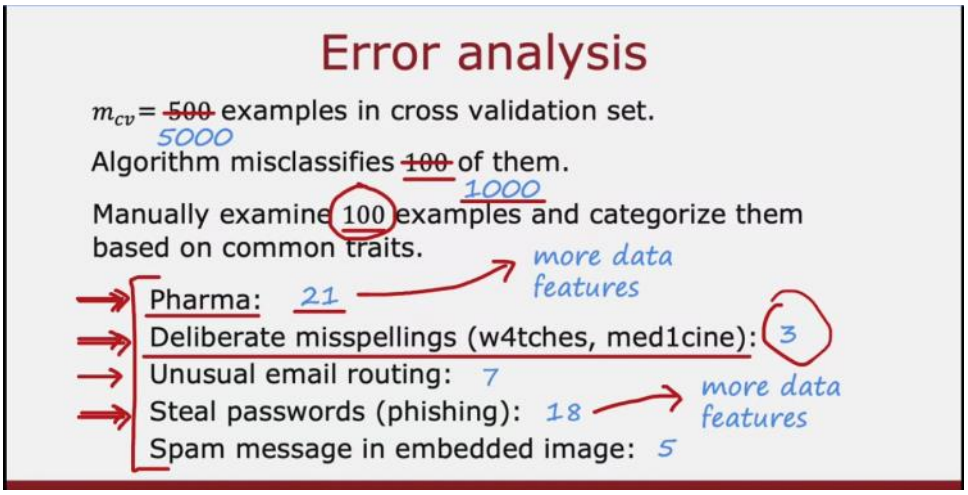
Options for improving (diagnosis)

Error Analysis

Monday, 9 October 2023 09:35

Bias and Variance is the most important thing for diagnosis, this is second.

Getting a sub-set of 100 misclassifications



Allows you to analyze where your problems might be

If one particular category is being misclassified, add more features of it

This allows you to try solutions

Adding Data (data augmentation, data systhesis, data driven)

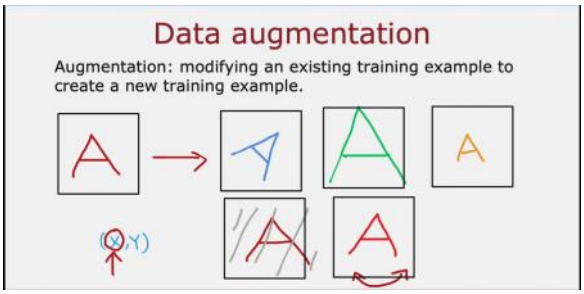
Wednesday, 18 October 2023 06:50

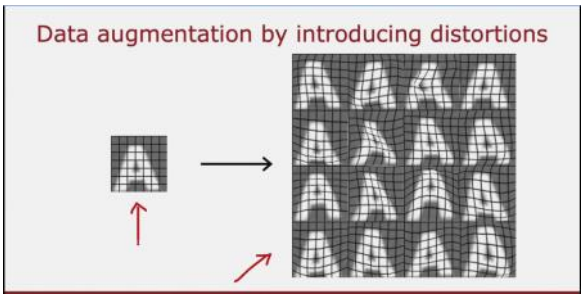
Adding more data of everything is good, but not always possible

So add data that Error Analysis has found would help

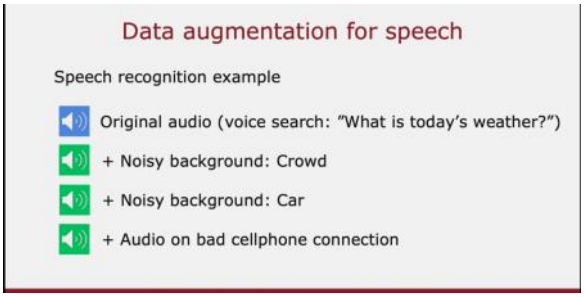
The more labelled data we add, the better

Data augmentation:

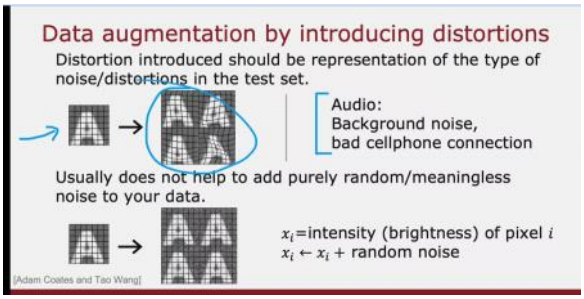




Adjusting existing data to make more examples

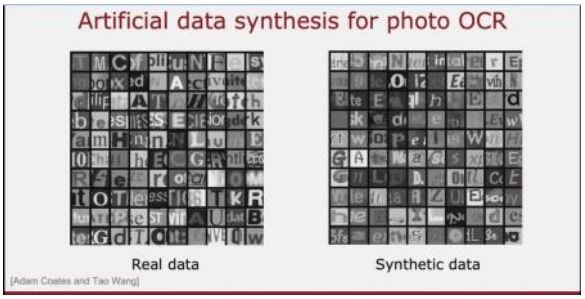


This can also be done with speech by adding in sounds



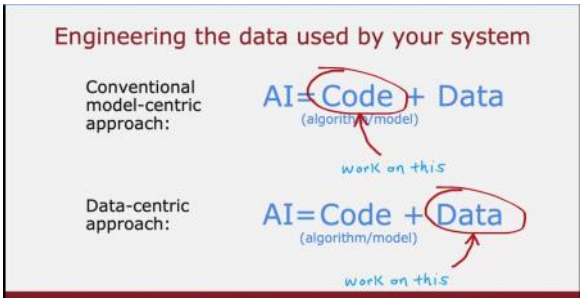
When doing data augmentation, your data should still be similar to original

Data Synthesis:



Data synthesis is about creating "fake" data that looks real, like using different fonts (screenshots) to train model

Data centric approach:



Engineering your data is as important as the code

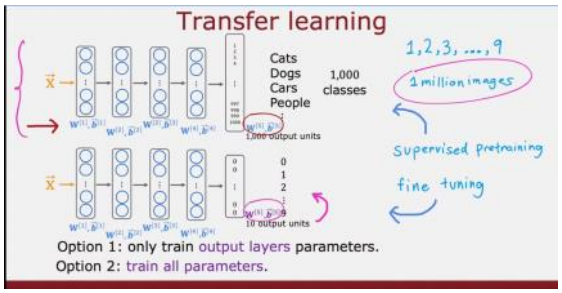
Transfer Learning: Using data from a different task (pre-train + fine tuning)

Wednesday, 18 October 2023 07:21

Transfer Learning:

Step 1 - Supervised pre-training - training an existing nueral net trained from an existing dataset on a new dataset on a related task with same input type, just select your output layer

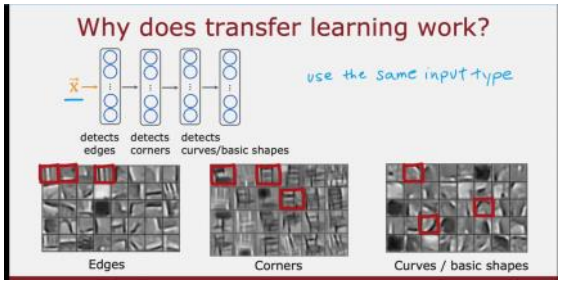
Step 2 - Fine tuning -Traning an existing model on a new dataset for a more specific task



When pre-training you make your own output layer to control what you want as output

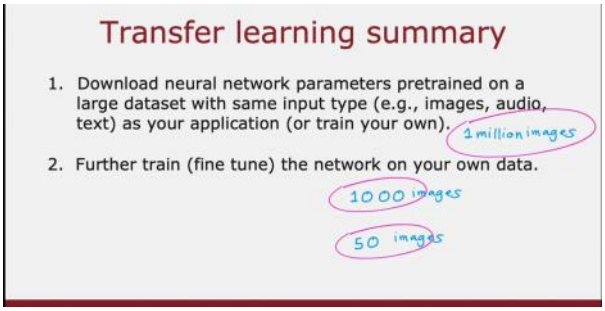
You can download pre-training models online

Sometimes pre-training models are available online



The reason pre-training works is because some of the stages, like finding edges, are very common across all models, so can be transferred

Summary:



GPT 3 etc was pretrained

Fine tuning

Transfer Learning: Using data from a different task (pre-train + fine tuning) 2

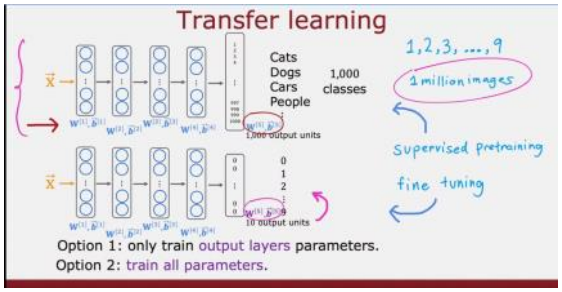
Wednesday, 18 October 2023 07:21

Transfer Learning:

Step 1 - Supervised pre-training - training an existing nueral net trained from an existing dataset on a new dataset on a related task just select a new output layer

Step 2 - Fine tuning -Traning an existing model on a new dataset for a more specific task - Can give noticeable improvements

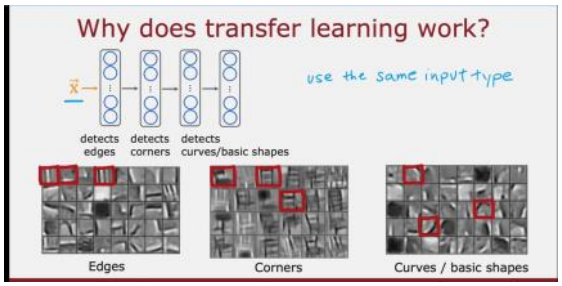
When doing transfer learning you 'freeze' the layers (part from output layer). Fine tuning is the process of unfeezing these are adapting them



When pre-training you make your own output layer to control what you want as output

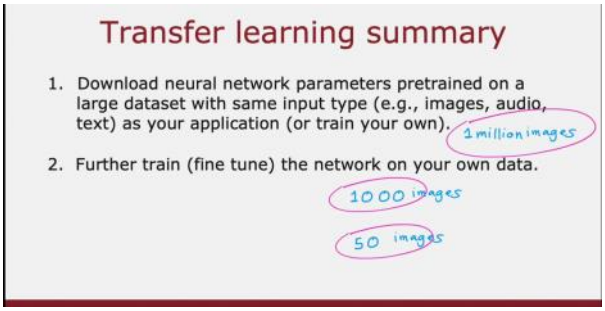
You can download pre-training models online

Sometimes pre-training models are available online



The reason pre-training works is because some of the stages, like finding edges, are very common across all models, so can be transferred

Summary:



GPT 3 etc was pretrained

Fine tuning:

Fine tuning is when you train all layers in the model, not just change the output layer.

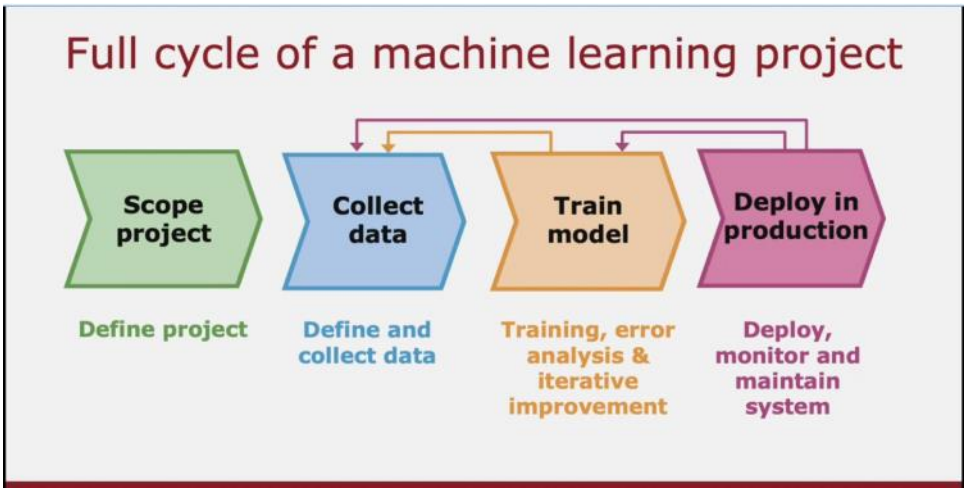
When doing transfer learning you 'freeze' all the layers of the model apart from output layer, ensuring they keep the same weights and bias's.

With fine tuning, once you have done some transfer learning, you unfreeze those layers. To allow them to 'tune' themselves to your new application.

This allows the model to learn the features of the new task while retaining useful knowledge from previous task.

Full Cycle Of A Machine Learning Project (Upgrade by use)

Wednesday, 18 October 2023 08:09



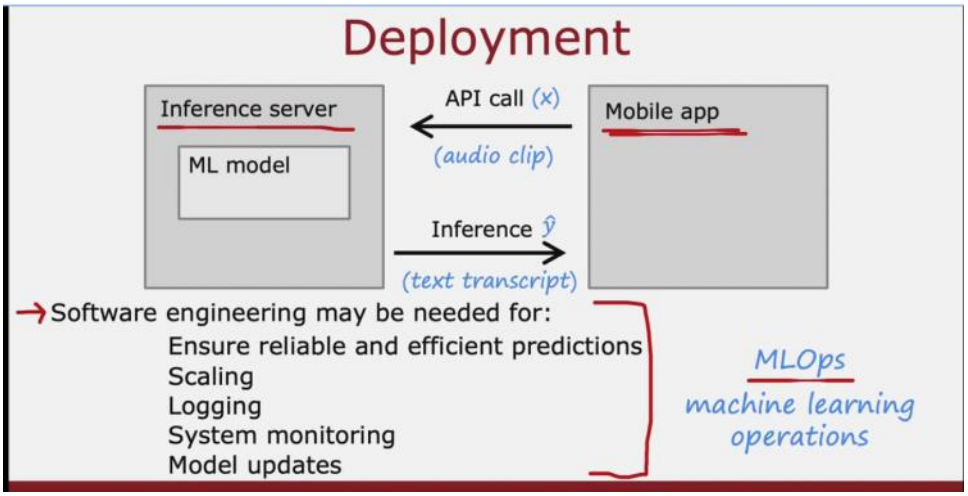
Scope Project - what you want to do

Collect Data - Label data and prepare it for model

Train Model - Train your model (including pre-training) in a loop until low cost

Deploy In Production - Deploy system and use data from it (if allowed) to prefect model

Deployment:



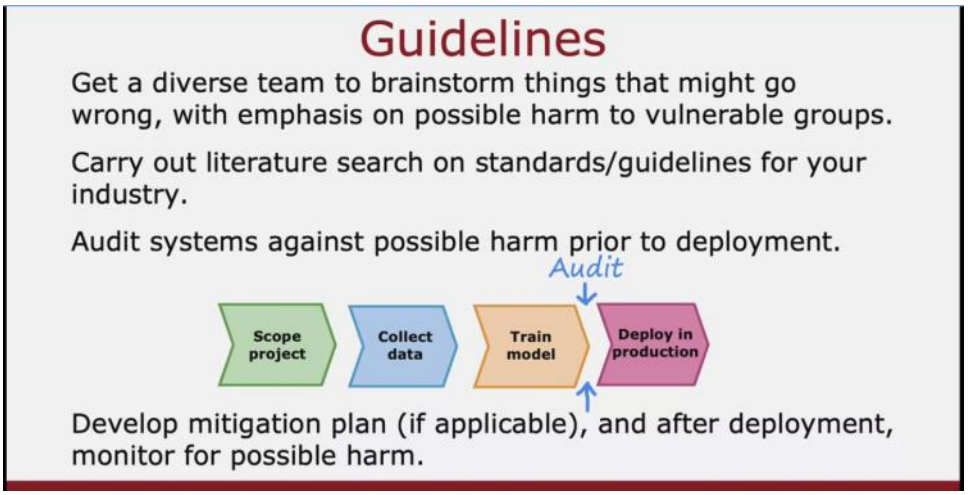
Inference Server - A REST API that utilizes your model

Model updates - taking data from usage attempts and training your model on it

MLOps - is to do all of this, Scaling, Logging, Model updates etc

Fairness, bias, and ethics

Wednesday, 18 October 2023 08:34



You should always carry out and Audit (checking for bias against a group etc) before deployment

Make a mitigation plan - A what if something goes wrong plan (software)

Transfer Learning & Fine tuning (v2)

21 November 2024 21:44

Transfer learning:

Transfer learning is the idea of faking features learned from one task and using them on a new, similar problem.

This is done because there will be common patterns between the problems as they are similar

Often done when you lack data for the new task

Workflow:

1. **Take the layers from the existing model**
2. **Freeze those layers to not destroy information**
3. **Added new trainable layers on top of the frozen ones. They will learn to turn old features into predictions for the data.**
4. **Train the new layers on your dataset**

Fine tuning:

Fine tuning is the process of unfeezing those pretrained layers and retraining them on the new data at a new low learning rate.

This can make noticeable improvements.

Error Metrics For Skewed Dataset (Precision & Recall)

Wednesday, 18 October 2023 08:50

When you have a dataset that isn't balanced very well, you can't use classification errors (train or test)

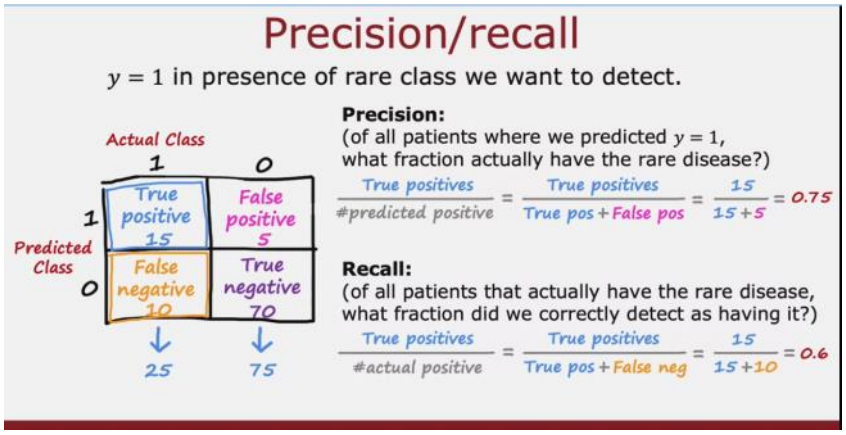
Precision and Recall - Allows you to know how accurate your model is when giving positive readings (Precision) & that the model is detecting most of them (Recall)

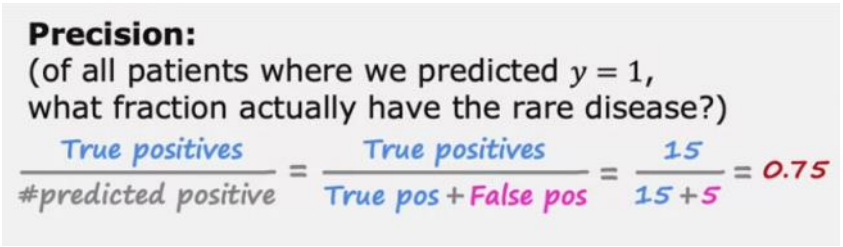
Precision - allows you to know how accurate your model is when giving positive readings

Recall - How good the model is at detecting the correct classifcation overall

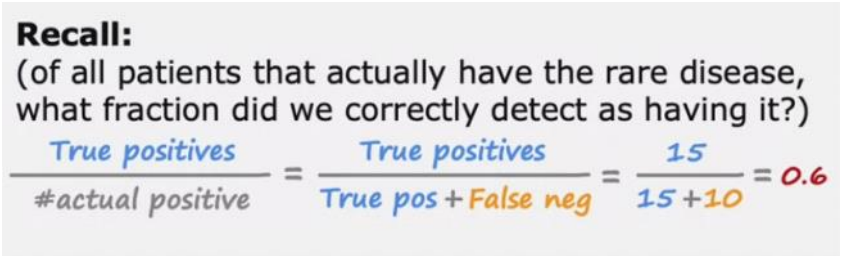
The higher Precision and Recall the better

Precision / Recall:





Precision - what fraction of the data is positive (y=1)

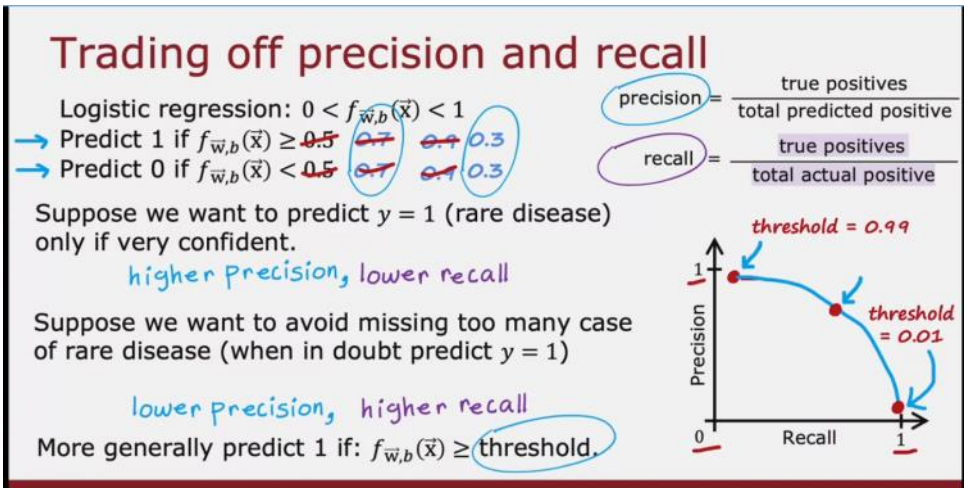


Recall - What fraction did our model correctly detect

Trading Off Precision and Recall (Thresholds, Confidence)

Wednesday, 18 October 2023 09:39

Pick the model with the highest Precision and Recall



The higher the threshold, higher the Precision.

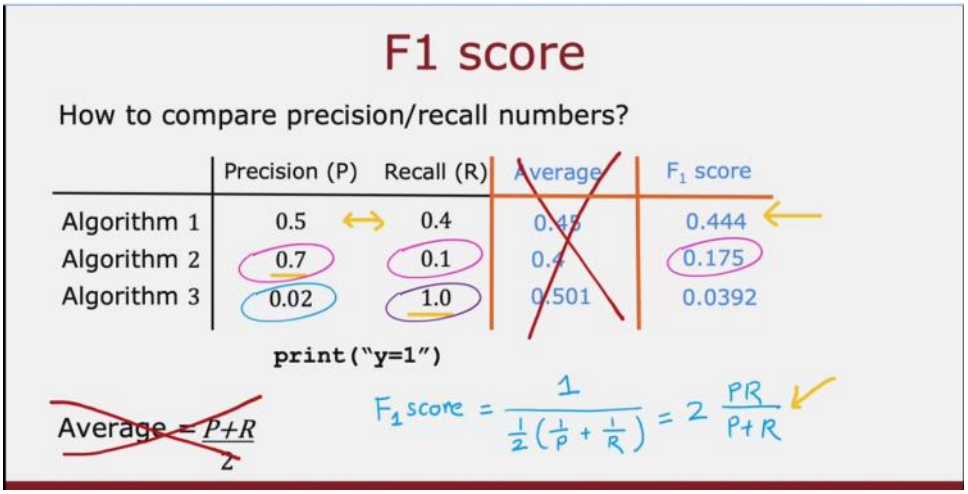
Precision is our confidence level when making predictions

The lower the threshold, the higher the Recall, because more examples are detected.

When both Precision and Recall are low, your model isn't very good.

F1 Score:

F1 Score works out how to trade off Precision and Recall



The higher the F1 Score the better the model (more balanced the Precision and Recall are)

Practice Lab - Advice For Applying Machine Learning

Sunday, 12 November 2023 20:01

Remember Regularization fixes overfitting (High Variance)

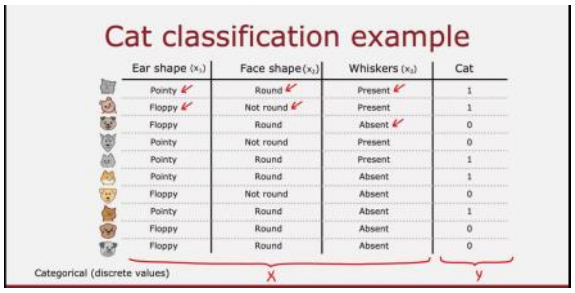
C2_W3_Assignment

C2_W3_Assignment - ...

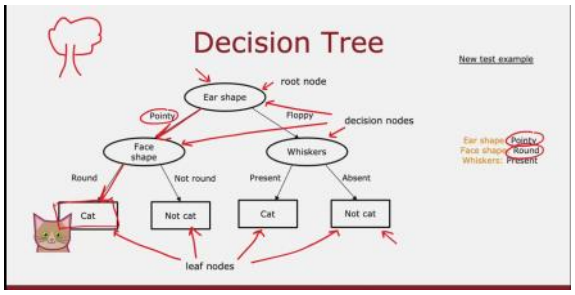
Decision Tree Model (Lowest…)

Wednesday, 15 November 2023 10:46

Cat Example:



This is binary classification

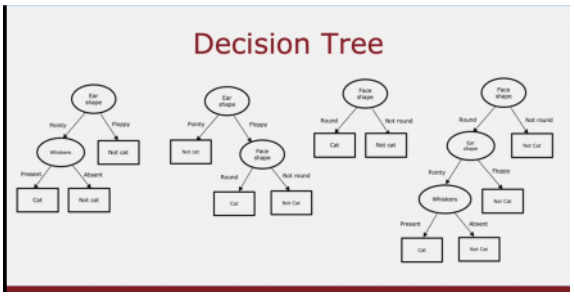


Decision Tree's start off at a root node

At each layer in the tree a Decision is made leading to a Decision Node

Then we reach our leaf nodes where a prediction is made

Other Tree's:

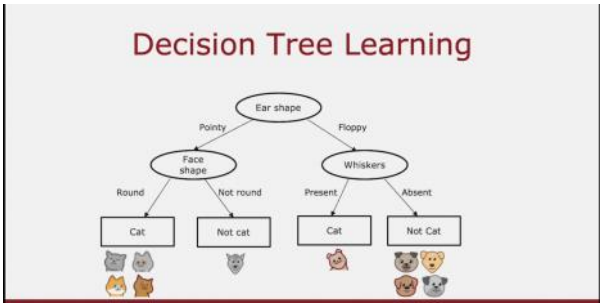


The job of a decision tree model is to pick a tree that gets the best JCV, Training error, Testing Error etc

Learning Process (max purity, stop splitting)

Wednesday, 15 November 2023 11:05

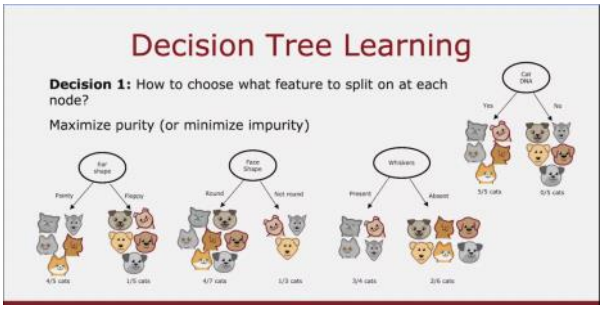
Below uses the example of the cat



The first step is to pick a root node feature (Lets say Ear shape of a cat)

At the next node, Decision node, We will pick another feature (Face shape)

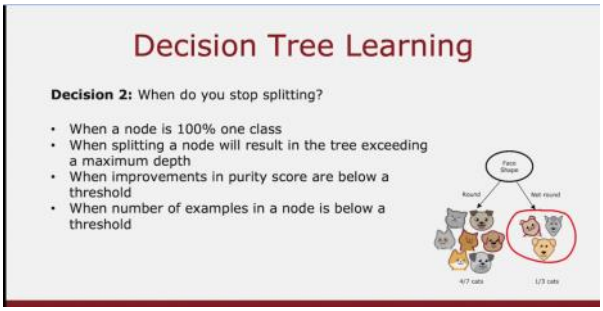
Once we are confident we have made our classification, create a leaf node (Cat, Not Cat)



Maximize Purity - How many of your dataset are the same class

When you reach high purity - Leaf Node (Make your prediction)

When to stop splitting (making choices):



When you reach 100% in one class

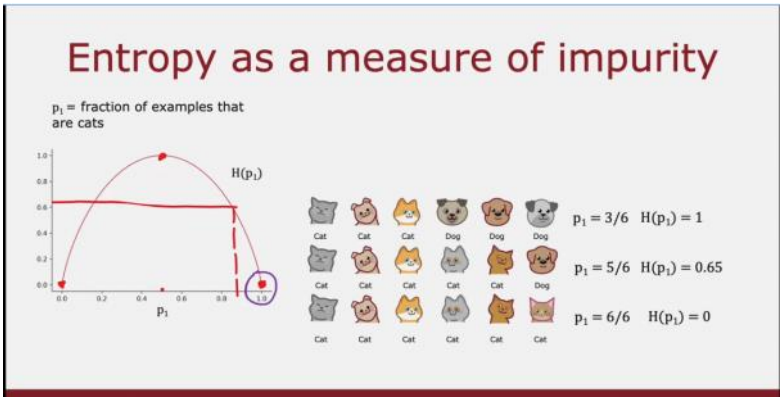
Or it reaches a depth limit (how many branches)

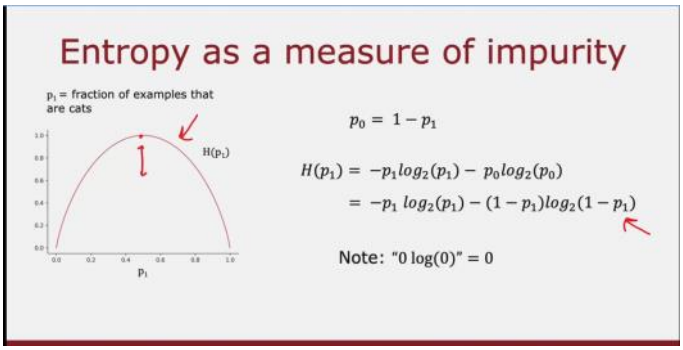
- **Keeping tree small helps limit overfitting**
- **Very small gains in purity**
- **makes it more manageable overall**
- **Number of examples at a node is below a threshold**

Measuring Purity (Entropy, Weighted Entropy…each branch)

Wednesday, 15 November 2023 11:37

Entropy as a measure of impurity:





Purity increases the more of your data is of the same class (cats etc)

On the X axis (bottom axis) is fraction of "same" classified data

On Y axis (Up axis) is the Entropy as measure of impurity

The lower the Entropy the better

W = Dataset fraction

H = Entropy

P = Purity

When choosing where to split, use a weighted entropy of each branch to make an average

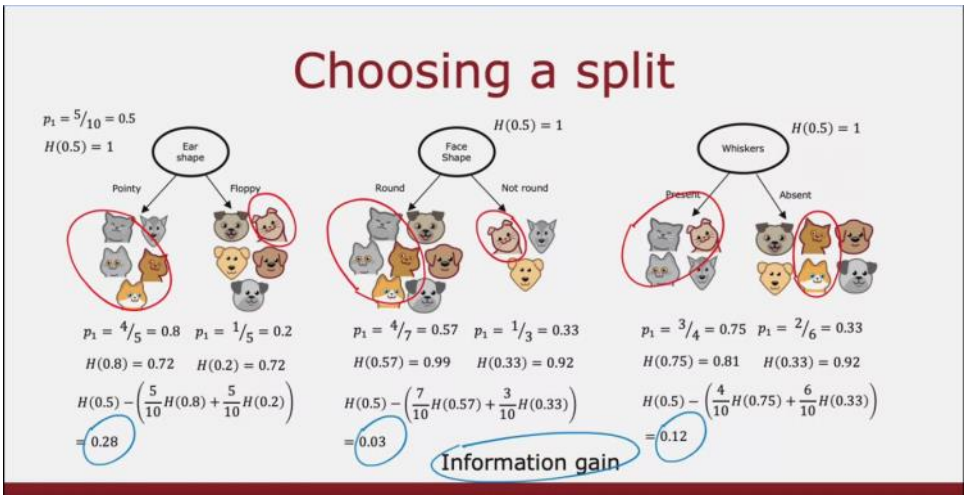
This will require you to calcluate variance

Choosing a split: Information Gain (+impacts…)

Monday, 20 November 2023 06:43

The size of the dataset impacts where you split

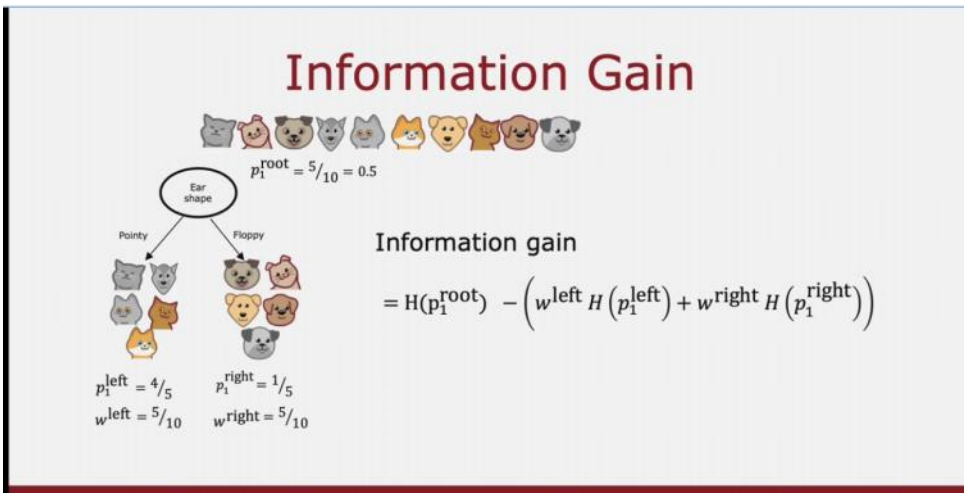
The larger the dataset with the greater the impurity the worse



Information Gain: Increase of purity given decision

Higher information Gain the better

Always choose the split with highest information Gain



W = Dataset fraction

H = Entropy

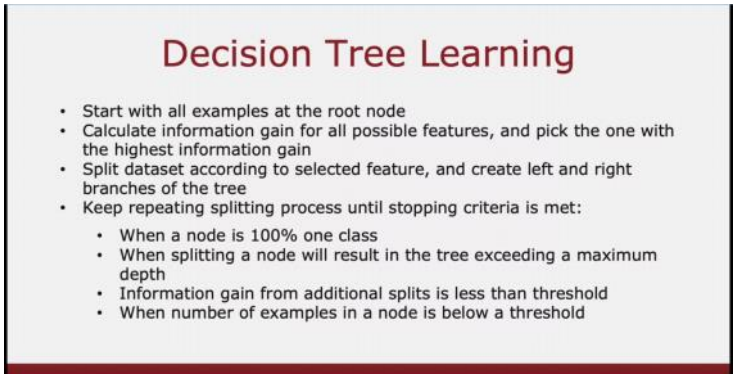
P = Purity

Root node purity -(dataset fractionLeft * Entropy scoresLeft + dataset fractionRight * Entropy scoresRight)

Putting It Together (info gain…building tree, max depth, Recursive…)

Monday, 20 November 2023 07:01

Decision Tree Process:



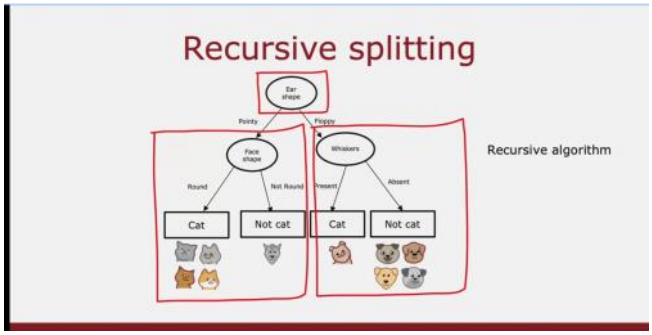
Start with all example at root node,

Caluclate information gain for all features, pick highest scoring one

Split dataset accordingly, creating left and right branches

Keep splitting (DOING ABOVE) until :

100% one class or depth limit



Recursive algorithm - Just means one tree leads to another

Choosing Max Depth:

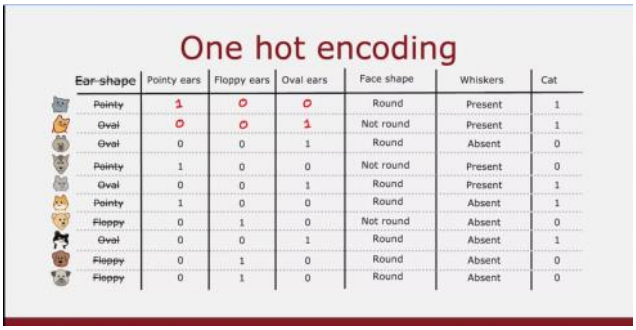
Higher the max depth, bigger the tree your willing to build

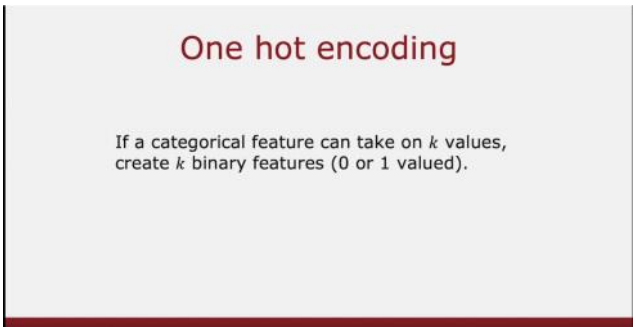
Like picking to use more polynomials, you increase chance of overfitting

Open source libraries can decide this for you quite well

Using One-Hot Encoding For Categorical Features (binary…)

Monday, 20 November 2023 07:19



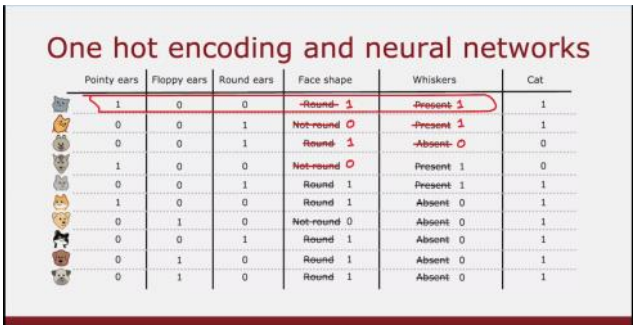


When we have multiple categories for a feature, turn it into K binary features

The feature that equals "1" is the "hot-one"

We don't need to adjust anything else to make this a decision tree

This approach can be used for "normal" neural networks with minor adjustments:



Switch all features to numbers

Doing this allows to feed to neural networks because they expect numbers as inputs

Lab: Decision Trees

Monday, 20 November 2023 10:11

C2_W4_Lab
_01_Deci...

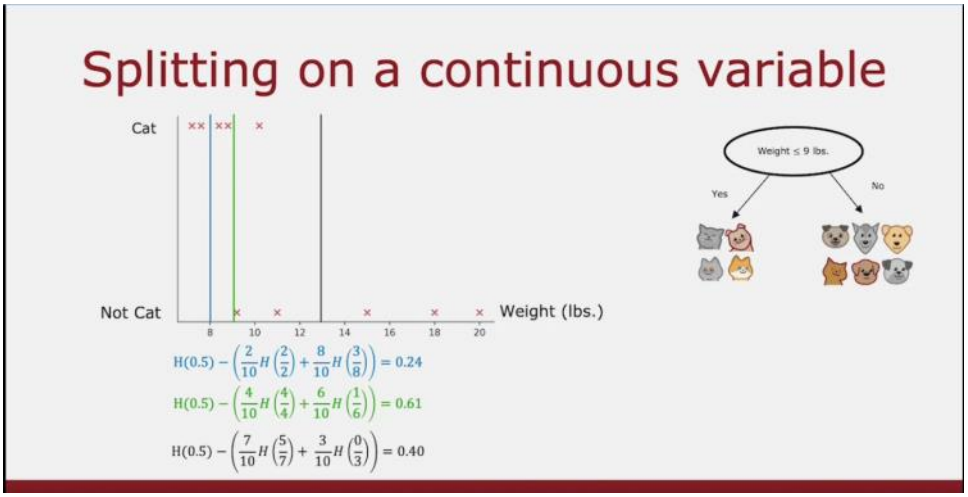
C2_W4_Lab
_01_Deci...

Continuous valued Features

Monday, 20 November 2023 07:50

Splitting on a Continuous Variable:

A continuous variable is a number not a classification but this is not regression because it still predicts a class overall



A good model will decide on a numerical threshold with the highest possible Information Gain

You still do the normal information gain calculation

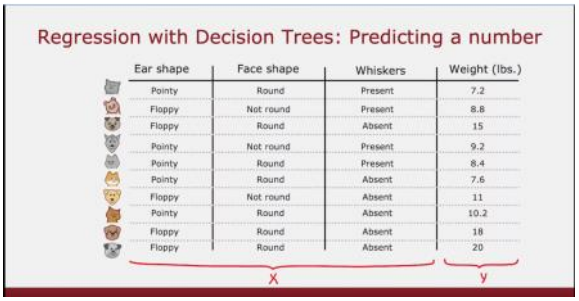
◉ Choose the 9 mid-points between the 10 examples as possible splits, and find the split that gives the highest information gain.

You can find the best information gain using midpoints

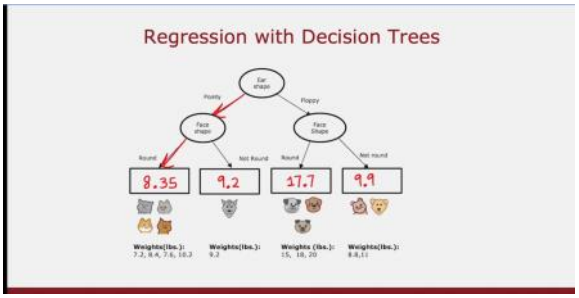
Regression Trees

Monday, 20 November 2023 08:05

Regression Trees predict numbers



Predicting an animals weight (y) as a number

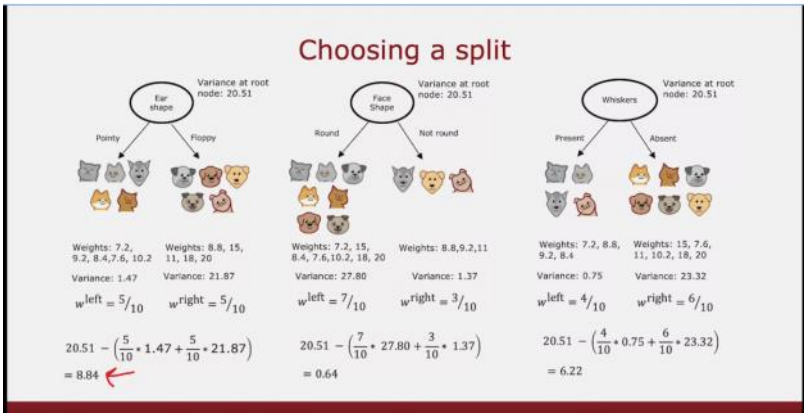


Regression trees take a numerical average of there leaf nodes to make predictions

Choosing a split for regression:

Variance is how much a set of number varies

The goal of our splits is to reduce variances of outputs (animal weights in this case)



As with classification tree, you must get weighted average (left and right)

You must also find root variance

Always pick greatest variance reduction (8.84 in this case)

Keep splitting until depth limit etc

Lab: Tree Ensembles

Tuesday, 21 November 2023 07:45

C2_W4_Lab
_02_Tree...

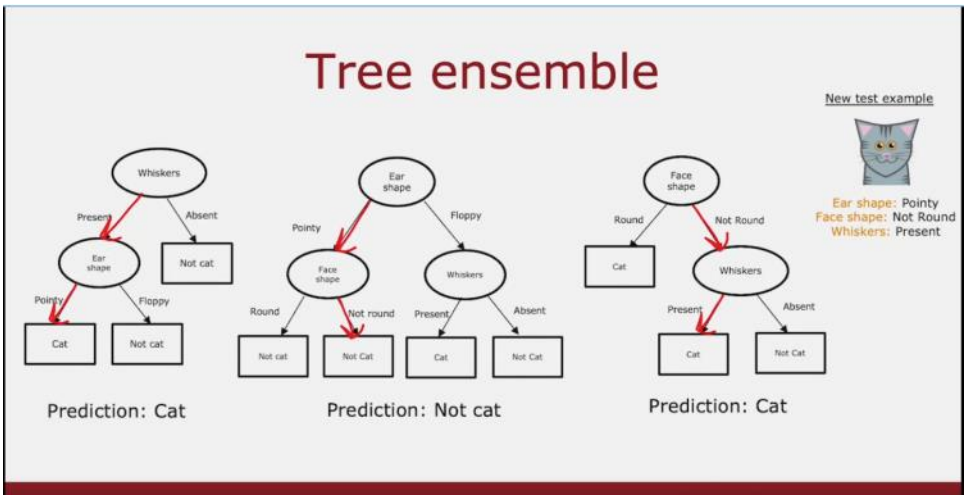
C2_W4_Lab
_02_Tree...

Using Multiple Decision Trees (sensitive, voting)

Monday, 20 November 2023 09:14

Decision Tree models are highly sensitive to changes in dataset

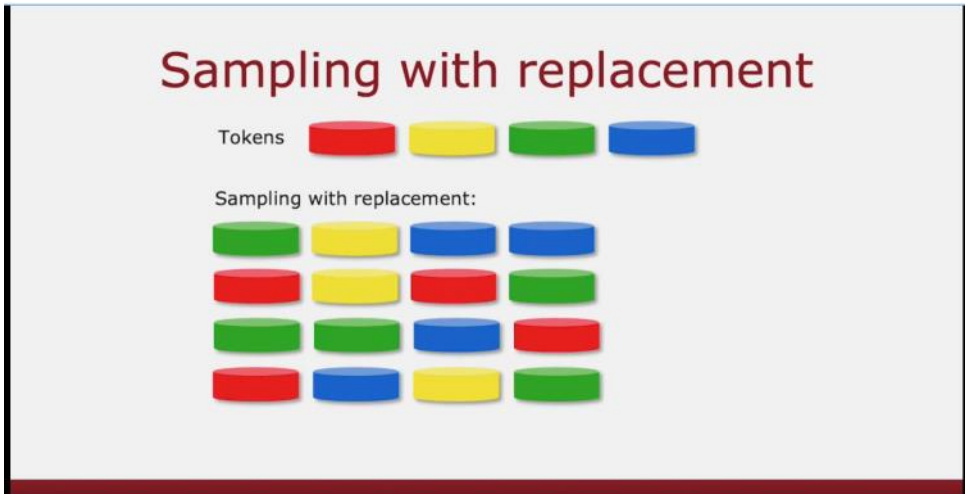
One change in data can lead to a completely different tree



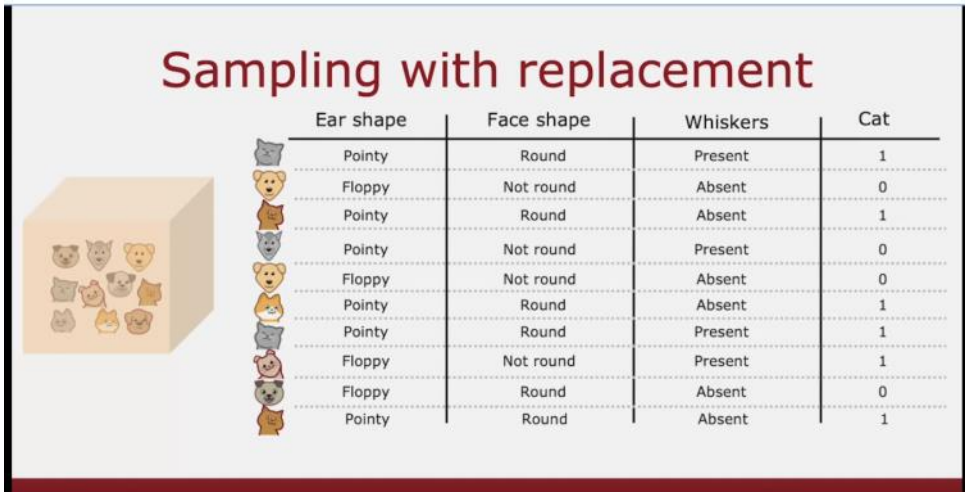
Having a series of trees and having them vote, it makes your model less sensitive to one tree

Sampling With Replacement (pick a token…)

Monday, 20 November 2023 09:26



Sampling With Replacement Means - You put back your "token" after randomly picking it



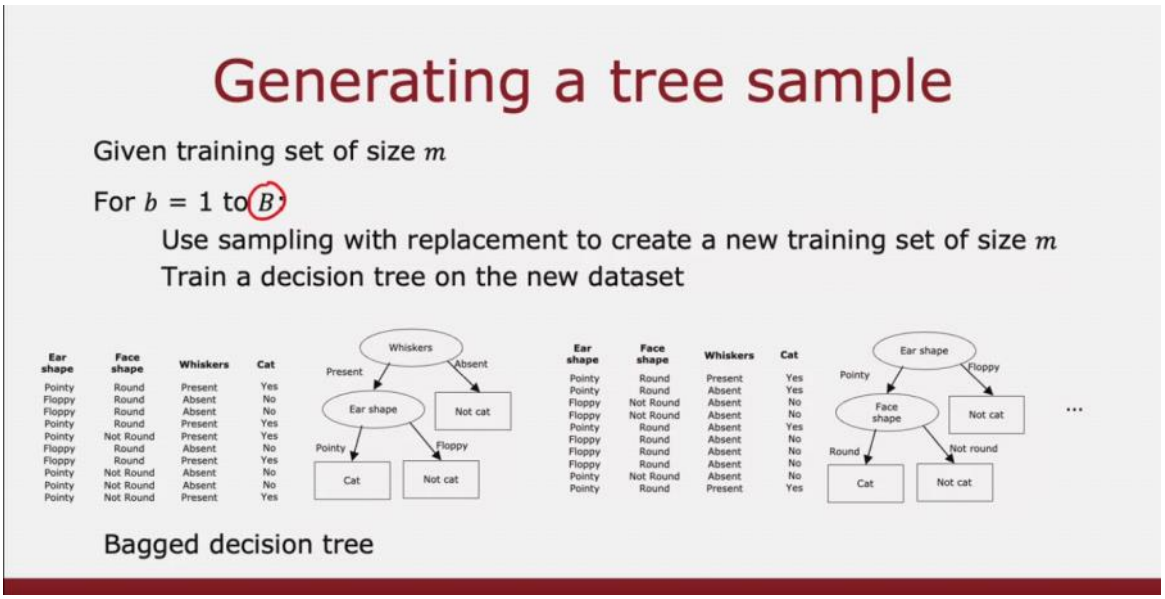
Using sampling with replacement you can make a new dataset to train a different tree

Due to replacement factor - some examples are the same (this is normal)

Random Forest Algorithm (using sampling…ensemble)

Monday, 20 November 2023 09:37

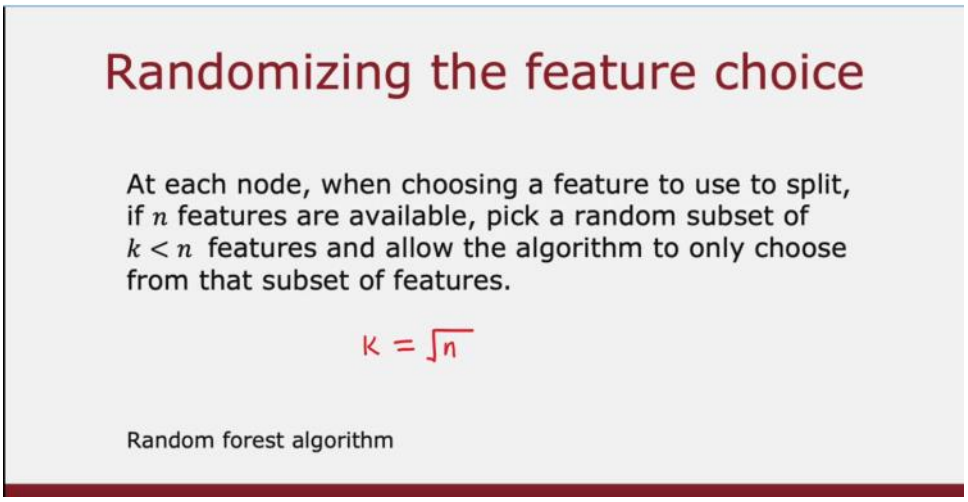
Generating a tree sample:



Using sampling with replacement you can make a dataset for an ensemble (series of) trees and have them vote

Increasing the amount of trees in the ensemble is always a good thing but does have decreasing returns

B = Recommended amount (64 to 128)



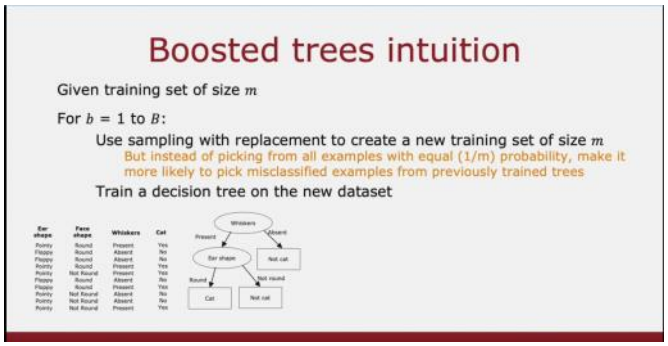
To make your trees more random (and better) in your ensemble (making them better), force it to pick from a subset of features (k) out of total (n)

This is only required for larger trees

XGBoost

Tuesday, 21 November 2023 06:08

Boosted Tree Intuition (XGBoost):



XGBoost in Code:

B = the size of our dataset segment

With Boosted Trees, we use sampling to build a dataset to train new trees, like before

But Boosted tree algorithm's are more likely to pick misclassified examples by previous trees in ensemble

XGBoost (extreme Gradient Boosting:

Open Source

Fast Efficient,

Good choice of default splitting criteria, start + stop

Built in regularization

Highly competitive algorithm for machine learning

When To Use Decision Trees

Tuesday, 21 November 2023 06:34

When to use Decision Trees & Tree ensembles:

Works well on tabular (structured) data

DO NOT use with unstructured data (images, audio etc)

Small Decision Tree's might be human readable

When to use Neural Networks:

Works well with all types of data (structured, unstructured)

May be slower than a decision tree

Works with transfer learning

Can be better paired with other multiple models compared with trees

Programming Assignment: Practice Lab: Decision Trees

Thursday, 23 November 2023 15:54

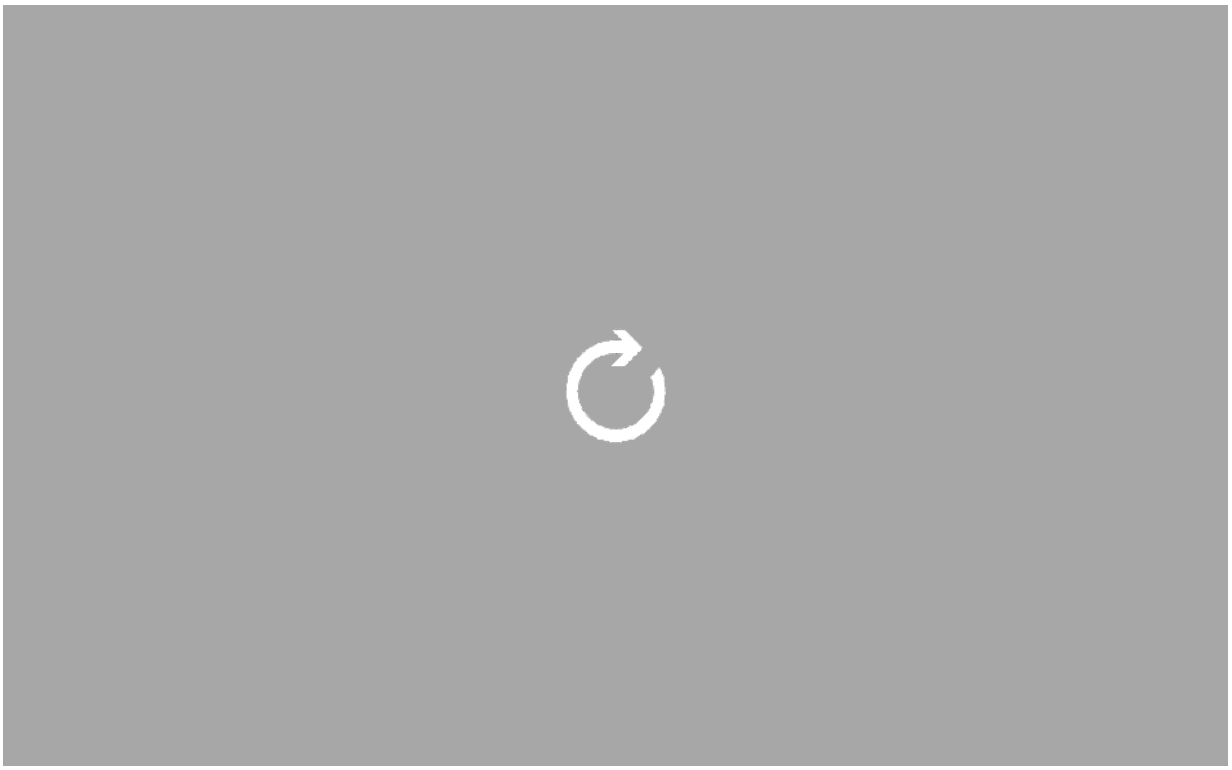
C2_W4_De
cision_Tr...

C2_W4_De
cision_Tr...

Video

Thursday, 23 November 2023 15:57

index.mp4



Losses functions for TensorFlow

Saturday, 30 September 2023 12:30

https://keras.io/api/losses/

Has the different loss functions for TensorFlow

Pandas

09 December 2024 23:55

https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf