Introduction to Machine Learning and Tensorflow

Manu Mathew Thomas
Creative Coding Lab
What is Machine Learning?

A class of algorithms which learns to perform a specific task based on sample data and without any explicit instruction.

Example: Classifying whether an email is a spam or not.
What is Machine Learning?

A class of algorithms which learns to perform a specific task based on sample data and without any explicit instruction.

Example: Classifying whether an email is a spam or not

Traditional Algorithms

```plaintext
if mail contains "...
{

}
else if sender == ""
{

}
else if ...
{

}
```
What is Machine Learning?

A class of algorithms which learns to performs a specific task based on sample data and without any explicit instruction.

Example: Classifying whether an email is a spam or not

Traditional Algorithms

```c
if mail contains "...
{
}
else if sender == ""
{
}
else if ...
{
}
```
What is Machine Learning?

A class of algorithms which learns to perform a specific task based on sample data and without any explicit instruction.

Example: Classifying whether an email is a spam or not.

Traditional Algorithms

```
if mail contains "...
{
}
else if sender == "" 
{
}
else if ...
{
}
```

Learning Algorithms

```

ML Model
```

Spam
Types of Machine Learning

Supervised learning - find patterns and insights from a labelled dataset

Regression

- Linear regression
- Support Vector
- Decision tree
- Random Forest
- Neural Networks

Classification

- K Nearest Neighbor
- Naive Bayes
- Support Vector Machine
- Decision tree
- Random Forest
- Neural Networks
Types of Machine Learning

Unsupervised learning - find patterns and insights from an unlabelled dataset

Clustering
- K Mean
- K Nearest Neighbor
- SVD
- Neural Networks

Feature Extraction
- Principal Component Analysis
- Gaussian Mixture Model
- Hidden Markov Model
- Neural Networks
Types of Machine Learning

Reinforcement learning - agent does an action to increase the reward
Neural Networks

NNs are a collection of small processing units (neurons) arranged in a hierarchical fashion.

Each processing unit is a combination of a linear function followed by a nonlinear function.
Neural Networks

NNs are a collection of small processing units (neurons) arranged in a hierarchical fashion.

Each processing unit is a combination of a linear function followed by a nonlinear function.
Neural Networks

NNs are a collection of small processing units (neurons) arranged in a hierarchical fashion.

Each processing units is a combination of a linear function followed by a nonlinear function.

**Linear Function:** $Wx + b$

$Wx + b$ is the equation for a straight line ($y = Mx + c$) where $M$ is the slope and $c$ is the $y$-intercept.

$Wx + b$ is preferred because:

- Straight lines are useful to model decision boundaries.
- It’s easier to work with.
Neural Networks

NNs are a collection of small processing units (neurons) arranged in a hierarchical fashion.

Each processing units is a combination of a linear function followed by a **nonlinear function**

- Non-linear function or Activation function: $\sigma(Wx + b)$
- $\sigma$ activates neuron based on the output of the linear function
- Creates non-linearity in the network
Neural Networks

NNs are a collection of small processing units (neurons) arranged in a hierarchical fashion.

Each processing units is a combination of a linear function followed by a nonlinear function.
Let’s build an image classifier

Images are just numbers (usually between 0 and 255)

We scale the images in the range [0,1] as part of feature scaling
Let’s build an image classifier

Fully Connected Network (FCN) - Each neuron in one layer is connected to every neuron in the next layer

- No spatial information
- Need a large number of neuron (parameters)
- Increases computations and memory footprint
- Not commonly used anymore
Let’s build an image classifier

Convolutional Neural Network - Each neuron is connected to a local region neurons of previous layer

- Looks at spatial information
- Reuse neurons (less # parameters)
- Lower computations and memory footprint
Let’s build an image classifier

Convolution in image processing

Kernel slides over the image and computes new pixel value as a sum/avg of element-wise multiplication
Let’s build an image classifier

Convolution in image processing

Examples: http://setosa.io/ev/image-kernels/

Some other kernel examples

Unweighted 3x3 smoothing kernel

Weighted 3x3 smoothing kernel with Gaussian blur

Kernel to make image sharper

Intensified sharper image
Let’s build an image classifier

Padding

without padding, 5x5 -> 3x3

with padding, 5x5 -> 5x5
Let’s build an image classifier

Stride

Stride - 1

Stride - 2
Let's build an image classifier

Deconvolution
Let’s build an image classifier

Classifying a number using hand-made image kernels
Let’s build an image classifier

Classifying a number using hand-made image kernels

Image of the number 7

Matrix:

```
0 0 1
0 1 0
1 0 0
```
Let’s build an image classifier

Classifying a number using hand-made image kernels

![Image of a number 7]

```
0 1 0
1 0 0
0 1 0
```

* False
Let’s build an image classifier

Classifying a face using hand-made image kernels

Hard to build complex kernels
Let’s build an image classifier

Kernels/Filters are learnable parameters in a CNN
Let’s build an image classifier

Kernels/Filters are learnable parameters in a CNN
Let’s build an image classifier

CNNs for RGB images
Let’s build an image classifier

CNNs for RGB images
Let’s build an image classifier

Learned kernels/filters
Let’s build an image classifier

Feature maps - output of each convolution (linear function)
Let’s build an image classifier

Activation function (nonlinear function)
Let’s build an image classifier

Pooling/Downsampling

Pooling is good for extracting the most important features.

Reduces the no. of parameters (weights) in the next layer.

Another alternative is stride = 2 or 3.

Only non-negative values

Rectified Feature Map

Poolings

Max

Sum
Let’s build an image classifier

Our simple network

64 x 64 x 3
16 kernels
16
16
32
2

Dog

5 Layers are not deep enough (but CPU friendly)
Let’s build an image classifier

Our simple network - Training

64 x 64 x 3

16 kernels

16

16

32

Iteration 0

Output        Ground truth

0.75           0           Cat

0.25           1           Dog
Let's build an image classifier

Our simple network - Training

- Conv1: 64 x 64 x 3, 16 kernels
- Conv2: 16 kernels
- Conv3: 16 kernels
- FC1: 16 kernels
- FC2: 16 kernels

Iteration 0

Output: 0.75, 0.25  
Ground truth: Cat, Dog

\[
\text{min}(\text{distance}(\text{output }, \text{gt}))
\]
Let’s build an image classifier

Our simple network - Training

Cross entropy is a distance function (kind of) for probability distributions
Let’s build an image classifier

Our simple network - Training

64 x 64 x 3  16 kernels  16  16  32  2
Conv1  Conv2  Conv3  FC1  FC2

Iteration 1000

Output  Ground truth
0.05    0    Cat
0.95    1    Dog

min(distance(output ,gt))

Cross entropy is a distance function(kind of) for probability distributions
Let’s build an image classifier

Our simple network - inference

- **Conv1**: 64 x 64 x 3, 16 kernels
- **Conv2**: 16 kernels, 16
- **Conv3**: 16 kernels, 16
- **FC1**: 32
- **FC2**: 2

Output: Dog
Generative CNN - Autoencoder

Convolution + Bottleneck extracts the most significant features from the input to reconstruct the output.
Generative CNN - Autoencoder

L1 and L2 norms are used for computing pixel distance
Generative CNN - Autoencoder

L1 and L2 norms are used for computing pixel distance
Generative CNN - Autoencoder

L1 and L2 norms are used for computing pixel distance
Generative CNN - Autoencoder

L1 and L2 norms are used for computing pixel distance

min(distance(output ,gt))
Generative CNN - Autoencoder

Dataset not used in training
Changing the latent variable randomly will generate new images
Generative CNN - Variational Autoencoder

Mean Vector

Sampled Latent Variable

Standard Deviation Vector

Vector
Next time

Setup tensorflow environment

Building a simple image classifier in tensorflow

Maybe Gans?