

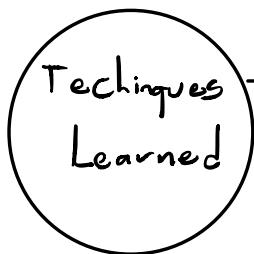
Data Science: Deep Learning

in Python (part 1)

SECTION 1: WELCOME

- Introduction and outline (video 1)
- This course → Fundamentals of Neural Networks
 - objective → Build a strong foundation for more advanced ideas such as CNN's or RNN's.
 - what is DL useful for
 - ↳ Like all ML → Making predictions
 - DL → Does it better
 - ↳ e.g. - Google's Alpha Go
 - ↳ Beat an AlphaGo champ!
 - Self-driving cars
 - Google Search
 - Stock prediction
 - Recognizing faces in an image
 - Predict outcome of next presidential election.
 - when it comes to ML algorithms → DL will give you the best performance.

Learning by example → Course project



Apply

- Engineer at e-commerce company

- Want to predict user actions based on data you've collected from your site.

- ↳ outline:
- Softmax → Going from binary to multi-class classification.
→ we will learn to discriminate between k number of things.
 - How to train a NN → Backpropagation
 - ↳ Extension of Logistic Regression
 - XOR/Donut problems → NN are capable of automatically discriminate features. ↳ No more feature engineering.
 - Building a NN → 1) Numpy → "By hand"
2) Tensorflow
 - ↳ "plug 'n play"
 - 3) Tensorflow playground →
Visualize what a NN is doing.

- More projects → Facial expression Recognition
- Where does this Course fit into your DL studies? (Video 2)
 - DL → Huge topic → Important to have this discussion.
↳ Good idea to understand previous ideas and how DL depends on these ideas.
- Before → Logistic Regression (Neuron) LoR
 - Linear Regression LR
 - ↳ Grandfather of ML Foundations
- Calculus & Probability Ideas that are the foundations of NN
 - Have some data
 - Fit a model to the data
 - The loss → Negative log-likelihood
 - Minimize loss via gradient descent (GD)
- After → Apply these ideas to more complex things
 - ↳ We may learn about more advanced libraries such as Keras, PyTorch, TensorFlow and Theano.

→ These libraries make writing complex NN
a lot easier.

→ Training takes a really LONG time.

→ GPU's improve Speed by orders of magnitude

Used by these modern DL libraries

Applied

- Specialized architectures (CNN's, RNN's, Recursive & Memory NN)
- Specialized problems (CV, NLP, RL)
↳ useful in a wide variety of applications

- This Course (DOESNT)
 - ↳ Abstraction approach? → No
 - ↳ Give me the API
 - so I don't have to worry about the details.

OK for narrow problems

- Problem arises → Need to do something different → Now stuck

!

Smallest change \rightarrow Huge obstacle

\hookrightarrow You won't understand what you are doing.

- Due to this, it is important to understand DL on more than a superficial level.

Any random coder can **copy** some KERAS Lines. \hookrightarrow If it is this easy \rightarrow why should they hire you?

This Course \rightarrow (Does)

- We cover the above details
- A bit more complex than LoR/LR
 - \rightarrow Not so specialized to require a Library.
 - \rightarrow Back propagation foundation \rightarrow How a NN learns
 - \hookrightarrow The basis of CNN's, RNN's and pretty much all NN's.
 - \rightarrow You will learn how DL really works.

Who this Course is NOT For

- Not for people without the necessary background knowledge.

- Not a "for dummies" course

- Those looking to learn about advanced architectures. → ~~CNN's~~

Learn about NN first!

• where to get the code (video 3)

- Github url:

<https://github.com/lazyprogrammer/machine-learning-examples>

L, o class folder: ann-class

o Project folder: ann-logistic-extra

o Extra repositories → Pay attention at the lectures

- In the terminal: (Do NOT copy)

git clone <url>

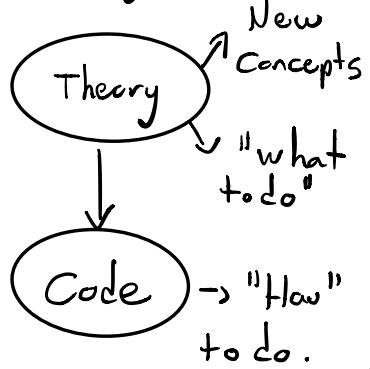
git pull → Do it often to ensure you have the latest version.

- Do NOT fork → You won't get updates and it may confuse other students.

- How to Succeed in this Course? (video 4)

Guideline #0 → write code!

Theory → Code



- You should be able to go from theory to code on your own.
- Good to start from scratch ↳ whenever you write the same code.
- Another practice opportunity → Modify your own code for your own data sets.
 - Changes are minimal regardless of the application.

- watch the lectures in order! 1, 2 ... 85
- All that we code comes from previous lectures.

- Guideline #1 →
- Make generous use of the QA forum.
 - Note some questions might require "research" work.

Great insight comes with the appropriate guidance.

- Guideline #2 → Meet the prerequisites
→ Some review will be provided

- Guideline #3 → Code! → Implement everything in code.

SECTION 2

REVIEW

- Review Section Introduction (Video 5)

Review Section → Quick review of the prerequisites needed to understand this course.

The "Neuron" → Major focus → Logistic Regression
Review A NN → Network of Neurons

\rightarrow ML \rightarrow Nothing but a geometry problem.

Mechanics of LoR { ① \rightarrow How to make predictions
② \rightarrow How to train a LoR Model.

① \rightarrow Must be able to make predictions before training.

What does Machine Learning do? (Video 6)

- $\text{ML} \rightarrow$ Nothing but a geometry problem.

Nothing but a "bunch" of dots in a grid \rightarrow It is what we do w/ this dots that makes ML what it is.

- Most basic ML problem \rightarrow we have an input in the x-axis and we would like to predict what's in the y-axis.

(LR) problem. \rightarrow Goal

Find the line that best fits the data

Line of best fit.

Writing the model

$$\hat{y} = \omega x + b, \quad x = \text{height}, \quad \hat{y} = \text{predicted height}$$

If we have more than 1 input: \rightarrow Real world problems!

$$\hat{y} = w_2 x_2 + w_1 x_1 + b = \boxed{\omega^T x + b}$$

Convention: $x = 0$ \rightarrow Denotes the size of ω & X

Classification vs. Regression

- Regression \rightarrow Tries to predict a continuous value. \rightarrow LR provides the groundwork necessary to progress to DL.
- Classification \rightarrow Tries to predict a category

e.g. \rightarrow Input = (Height, weight) \rightarrow Prediction

- 2 categories \rightarrow Binary classification \leftarrow whether or not the patient has high blood pressure (1) or not (0).
 \Downarrow

Tries to predict the label out of two different categories.

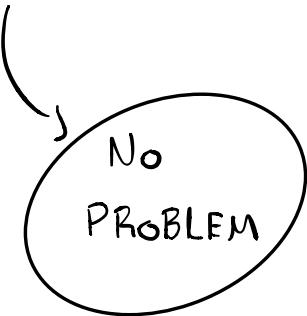
- LoR \rightarrow Linear model for classification \rightarrow uses a line or a plane for the model. \rightarrow Provides a boundary
 \leftarrow Separates (+) & (-) examples.

All data is the same

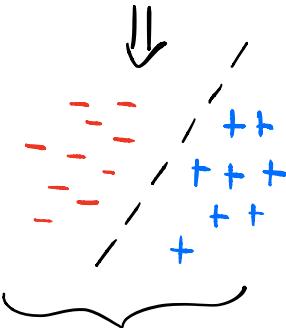
↳ The example above had $x_1 = \text{height}$
 $x_2 = \text{weight}$

and we were trying to predict $\hat{y} = \text{High blood pressure}$

- what if we switched the features $x_1 = \text{weight}$
 $x_2 = \text{height}$
- what if we have a completely different problem? $\rightarrow (\begin{matrix} \# \text{ hours studied} \\ \text{Midterm grade} \end{matrix}) \Rightarrow \begin{matrix} \text{Pass?} \\ \text{Final grade} \end{matrix}$



The semantics of the problem has changed, however, the mechanics of the problem hasn't.



ALL DATA IS THE SAME
No matter the nature of the problem.

Problem still is \rightarrow "we have this blue dots & red dots and we need to find the line that best separates the dots"

Summary

- ↳ ML is nothing but a geometry problem.
- You can plug in any data set from your own area of interest
 - Code doesn't change
 - Algorithms don't change.
- } ML works no matter where your data comes from.

Neuron Predictions (video 7)

LoR → Most important prerequisite of the Course.

↓
↳ A NN is just a network of Logistic units.

In LoR, we separate our categories with a line or a plane of the form:

$$w^T x + b = 0$$

Note that this equation will return a number but we need to predict a category.

How do we go from a continuous value to a category? → ENCODING

Making Predictions

- We encode our categories as 0 and 1. \rightarrow using a sigmoid function
- Sigmoid

$$\hookrightarrow \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\text{where: } 0 \leq \sigma(z) \leq 1$$

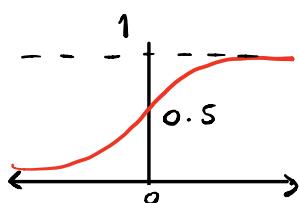
Now, consider:

$$P(y=1|x) = \sigma(\omega^T x + b),$$

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

Note that if $a \rightarrow +\infty \Rightarrow \sigma(a) = 1$

$a \rightarrow -\infty \Rightarrow \sigma(a) = 0$



\Rightarrow we typically use $a=0.5$ to make predictions:

If $\omega^T x + b < 0 \Rightarrow$ output < 0.5

$\omega^T x + b > 0 \Rightarrow$ output > 0.5



use the
round() function \rightarrow Thus:
in Python

Recall we can only predict 0 or 1

$$P(y=1|x) = 1 \text{ if } \omega^T x + b > 0$$

$$P(y=1|x) = 0 \text{ if } \omega^T x + b < 0$$

Python

Dotting vs. Looping

→ Slow

Loop method: (Accumulator)

```
result = 0  
for i in range(D):  
    result += a[i] * b[i]
```

$$a^T b = \sum_{i=1}^D a_i b_i$$

Numpy method: → optimal

a. dot(b)

• Calculating Multiple predictions

- Usually, we are given a set of samples X of shape $N \times D$

L, where: N: Number of samples

D: Number of features

- Another Convention → Lowercase letter for 1 - sample (A Vector)
→ uppercase letter for multiple samples (A matrix)

- One way to calculate multiple predictions:

`predictions = []`

`for i in range(N):` \rightarrow Not Ideal (For-loop)
 $p = \text{np.round}(\text{sigmoid}(w \cdot \text{dot}(x[i]) + b))$
`predictions.append(p)`

- Better:

`predictions = Sigmoid(X.dot(w) + b)`
 \downarrow
 $X \in \mathbb{R}^{N \times D}$ No for-loop

Note: $X \in \mathbb{R}^{N \times D}$
 $w \in \mathbb{R}^{D \times 1}$

$\Rightarrow Xw \in \mathbb{R}^{N \times 1}$ ($X \cdot \text{dot}(w)$)
 and $b \in \mathbb{R}^{N \times 1}$ $+ b$)

we add this scalar using
 Python Broadcasting.

The scalar is reshaped to size
 $N \times 1$ and added element-wise to
 all N elements of Xw .

ILs • Summary

- How to make predictions with a LR model.
- How to choose w ? \rightarrow Training the model!

Neuron Training (Video 8)

Review - Learning

- The "learning" part of Machine Learning simply means finding the weights of the model.

To DL analogies w/
the brain are only
analogies }
↳ Common analogy → The strength of
the connections b/w synapses in your
brain change when you form new memories.
A hypothesis ONLY! → • We Do NOT
know how

New research Suggests memories
could be stored in RNA, → could be even transferred
b/w organisms. } memories are formed.

• The cost/loss function → central component in Learning

↳ Perspective → ML is nothing but a probability problem.

→ we are solving a maximum likelihood problem

↳ e.g. we'd like to determine $p(\text{heads})$ from a series of flips:



- Flip it 1000 times and keep track of the results → Set up your likelihood function & minimize it wrt $p(\text{heads})$



$\xrightarrow{\text{closed form solution}}$ Likelihood = $p(\text{heads}) p(\text{heads}) \dots p(\text{tails}) p(\text{tails})$

$$p(\text{heads}) = \frac{\#\text{ heads}}{\text{total \# of coin tosses}}$$

- Binary Cross-entropy

↳ Negative log-likelihood

$$J = - \left\{ \sum_{n=1}^N t_n \log(y_n) + (1-t_n) \log(1-y_n) \right\}$$

↳ More conventions:

- N = Number of Samples

$n \rightarrow$ Index (other index variables $i, j, k \dots$)

* Be careful w/ the ambiguity of naming targets and predictions.

↳ Note we can also write our loss function as:

$$J = - \left\{ \sum_{n=1}^N y_n \log [P(y_n=1|x_n)] + (1-y_n) \log [1 - P(y_n=1|x_n)] \right\}$$

↳ Much more cumbersome, however, much more convenient to discuss how to get predictions.



- Training →
- No closed form solution to determine the MLP.
 - ↳ Must use numerical optimization methods like GD.

other methods:

- L-BFGS
 - Conjugate Gradient
- Not used as often in DL
- Using GD → we update our weights as follows:
- $$\omega \leftarrow \omega - \eta \nabla_{\omega} J$$
- $$b \leftarrow b - \eta \frac{\partial J}{\partial b}$$

↳ How to determine $\nabla_{\omega} J$ & $\frac{\partial J}{\partial b}$

$$\nabla_{\omega} J = X^T(Y - T) = \sum_{x_n=1}^N x_n (y_n - t_n)$$

$$\frac{\partial J}{\partial b} = \sum_{n=1}^N (y_n - t_n)$$

\hookrightarrow Note that: $w \in \mathbb{R}^D$ $y \in \mathbb{R}^N$
 $x_n \in \mathbb{R}^D$ $y_n, t_n \in \mathbb{R}$
 $X \in \mathbb{R}^{N \times D}$

For which: $\nabla_w J \in \mathbb{R}^D$ $\frac{\partial J}{\partial b} \in \mathbb{R}$
 \hookrightarrow $(X^T \underbrace{(y - T)}_{(N \times 1)})$
 $(D \times 1)$

\hookrightarrow In code:

$$\nabla_w J = X^T (y - T) \rightarrow \text{gradient_w} = X.T. \text{dot}(y - T)$$

$$\frac{\partial J}{\partial b} = \sum_{n=1}^N (y_n - t_n) \rightarrow \text{gradient_b} = \text{np.sum}(X - T, \text{axis}=0)$$

\downarrow

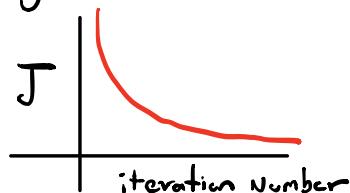
Now, we update our weights as follows:

for epoch in range(num_epochs)

$$w = w - \eta * X.T. \text{dot}(y - T)$$

$$b = b - \eta * \text{np.sum}(y - T, \text{axis}=0)$$

$\underbrace{\quad}_{\text{we do the above update procedure until}}$
 we observe a Convergence in our cost J :



Maximize or minimize?

- Maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood
- Semantically \rightarrow we refer to the objective as a cost or a loss \rightarrow Makes sense to minimize.

Regularization \rightarrow No longer MLP \hookrightarrow MAP (Maximum a posteriori)

$$\begin{aligned} J_{L1} &= J + \lambda_1 |w| \\ J_{L2} &= J + \lambda_2 |w|^2 \end{aligned} \quad \begin{cases} \text{Learning process} \\ \text{does not change} \end{cases}$$

L_1 (Encourages Sparsity)
 L_2 (Encourages Small weights)

one final note

\Downarrow In $J \rightarrow$ If target is 1, then $y = 1$ yields minimum loss $\rightarrow J = \log(1) = 0$

HOWEVER \rightarrow For $y = 1 \rightarrow w \rightarrow \infty$

Don't want \rightarrow we use regularization to prevent this.

- Deep Learning Readiness Test (Video 9)

- ① F
- ② Prediction output of Sigmoid function
- ③ C.E Loss function
- ④ weight update equation by GD
- ⑤ Gradient of $J \rightarrow$ calculates w.r.t to w "error"

- Review Section Summary (Video 10)

↳ * Course pattern: Prediction then training

* ML \rightarrow Geometry Problem
(ALL Data is the same)

SECTION 3

Preliminaries: From Neurons to Neural Networks

- Neural Networks with No Math (Video 11)

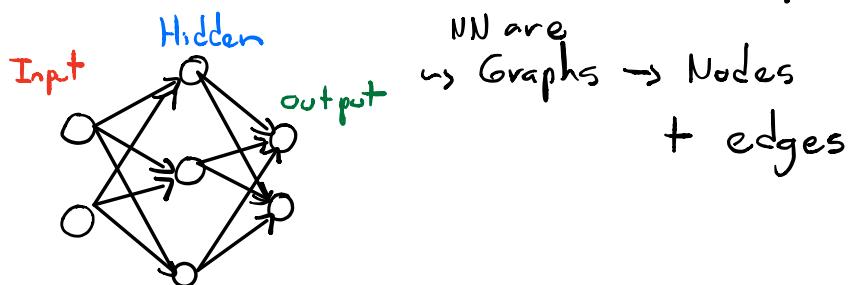
↳ NN w/ no math \rightarrow Lets obtain a high level intuition of NN
still required!

↳ NN → Another way to do Supervised learning.

↳ All models (LoR, KNN, NB, SVM, DT) have the same two functions.

(x, y) → train() → Learn model parameters from the data.
 x_test → predict() → Make accurate predictions using the
↓
 \hat{y} ↪ our predictions
params learned during training

NN Architecture ↪ NN are just networks of LoR (neurons) units. → Architecture is more complex.



- One or more hidden layers
- Every node is connected to every node in the next layer.
- Input layer → Features are fed into the input layer → Biology analogy

Rescaled by a weight



Hidden layer(s)

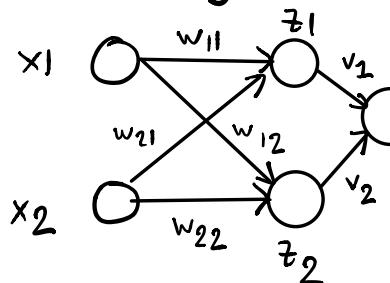
- Sensory nodes such as mechanoreceptors.
- Visual receptors in your eyes.

- Weights → Analogous to synapses in your brain. → stronger weights imply stronger weights from one node to another.
- ↙
- Signals → Travel from the input node, to the hidden layer and into the output.
- ↙
- Strong signal → Sent from neuron to neuron w/o much attenuation.
- The output aims for a target.

NN → Networks of Neurons \Rightarrow Layers of LoR units

Learning (Backpropagation)

↳ Consider the following NN:



Same pattern if more layers!

→ IN NN the learning part is called the back propagation

→ The error gets "propagated" backwards.

- ↳ o V depends on error at y
- o W depends on errors at z

A lot of modern
NN techniques are
based on back propagation
~~~~~  
CNN's, RNN's -----

} -> weights get updated  
based on this propagated  
error

## Introduction to the (video 12)

### E-commerce Course Project

#### Course Project Intro

- This project has business value
- It is important to take in real data that could be in the form of:
  - XLS spreadsheet
  - SQL table
  - Log Files

} we should be able to format them into a suitable for consumption by DL algorithms.

- focus -> Suppose you are a Data Scientist in an e-commerce store.

- Want to predict user actions in our website.

- Predict bounce -> Prompt

Ls Direct monetary impact:

||

We could suggest the user to take other action other than leaving the site

- ↓
- Discover areas of site which are weak.  $\rightarrow$  Could be improved  
 e.g. users begin checkout but not complete it  $\rightarrow$  why?
  - Ex.  $\rightarrow$  Mobile & user friendliness  $\rightarrow$  Improve friendliness of mobile app.
  - We can make data driven decisions  $\rightarrow$  use science to improve user experience.

## The e-commerce Data

- ↳ • Format: CSV file
- Header:
  - ↳ • Is\_mobile (0/1)
  - N\_products\_viewed  $\rightarrow$  (int  $\geq 0$ )  
 Viewed by the user during a session where the action was taken.
  - Visit\_duration  $\rightarrow$  (real  $\geq 0$ )
  - Is\_returning\_visitor (0/1)

- Time of day ( $0/1/2/3$ )

"Buckets"  $\rightsquigarrow$  • split into 4 categories  
 \* In ML  $\rightarrow$  Geometrical distances matter

- Assumptions?

users in the same bucket will behave similarly.

This is  $\rightarrow$  • user-actions  $\rightarrow$  our labels  
 what we will

$\underbrace{\text{predict}}$        $\underbrace{(\text{bounce} / \text{add-to-cart} / \text{begin-checkout} / \text{finish-checkout})}$

Classification problem  $\rightarrow$  Logistic class: Binary classification  
 $\qquad\qquad\qquad$  (Ignore last 2 classes)  
 $\rightarrow$  NN class: Multi-class classification  
 $\qquad\qquad\qquad$  Considers all labels

## • Data Pre-processing

$\hookrightarrow$  • We cannot feed categories in our LR/NN model  $\rightarrow$  Numerical vectors required

$\hookrightarrow$  Solution  $\rightarrow$  one-hot encoding:

| cat 1 | cat 2 | N |
|-------|-------|---|
| 1     | 0     | 0 |
| 0     | 1     | : |
| 0     | 1     | 0 |
| :     | :     | 0 |
| 0     | 0     | 1 |

## ↳ Dealing w/ binary categories:

- Is\_mobile & is\_returning\_visitor  
are also categories?

Yes, but we don't need 2 columns  
"off" and "on"

Just absorb the "off" into  
the bias term.

## ↳ Dealing with numerical columns

- Normalization ( $\mu = 0, \sigma = 0$ )  $z = \frac{x - \mu}{\sigma}$
- Simple way of treating numbers

Small range  $\rightarrow$  Sigmoid  $\sigma(a)$  has  
a more pronounced effect.

↳ e.g. Large values:

$$\sigma(1) \rightarrow \text{too }$$

NOT USEFUL  $\rightarrow$  No effect!

## Integration with the Course

- ↳ Each SL ML model has 2 tasks:

- ① Prediction  $\rightarrow$  Input  $\rightarrow$  Predict output
  - ② Training  $\rightarrow$  How to make the output accurate?  $\rightarrow$  Need to alter  
Corresponding section  
in the course  


**THEORY** + **IMPLEMENTATION**

↳ {

- ① How will we make predictions on our e-commerce data?
- ② How will we train our model on the e-commerce data?