Getting Started with ML

Mac Bellingrath
Principal Engineer, iOS at The Washington Post

Preface

This isn’t a traditional talk

Not a lot of code

We’ll talk about theory

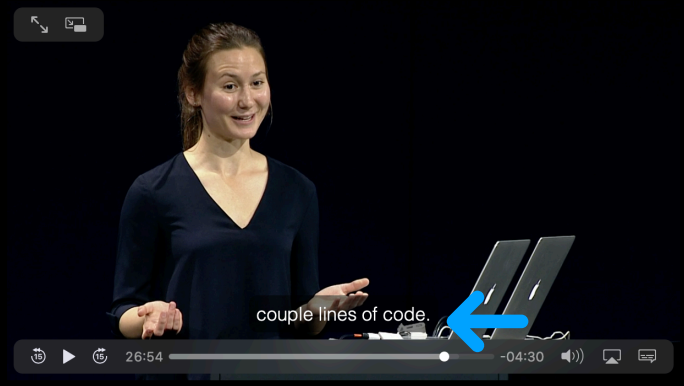
Code is less important than theory

My learning journey

WWDC 2017

Videos

Collections Topics All Videos



Overview Transcript

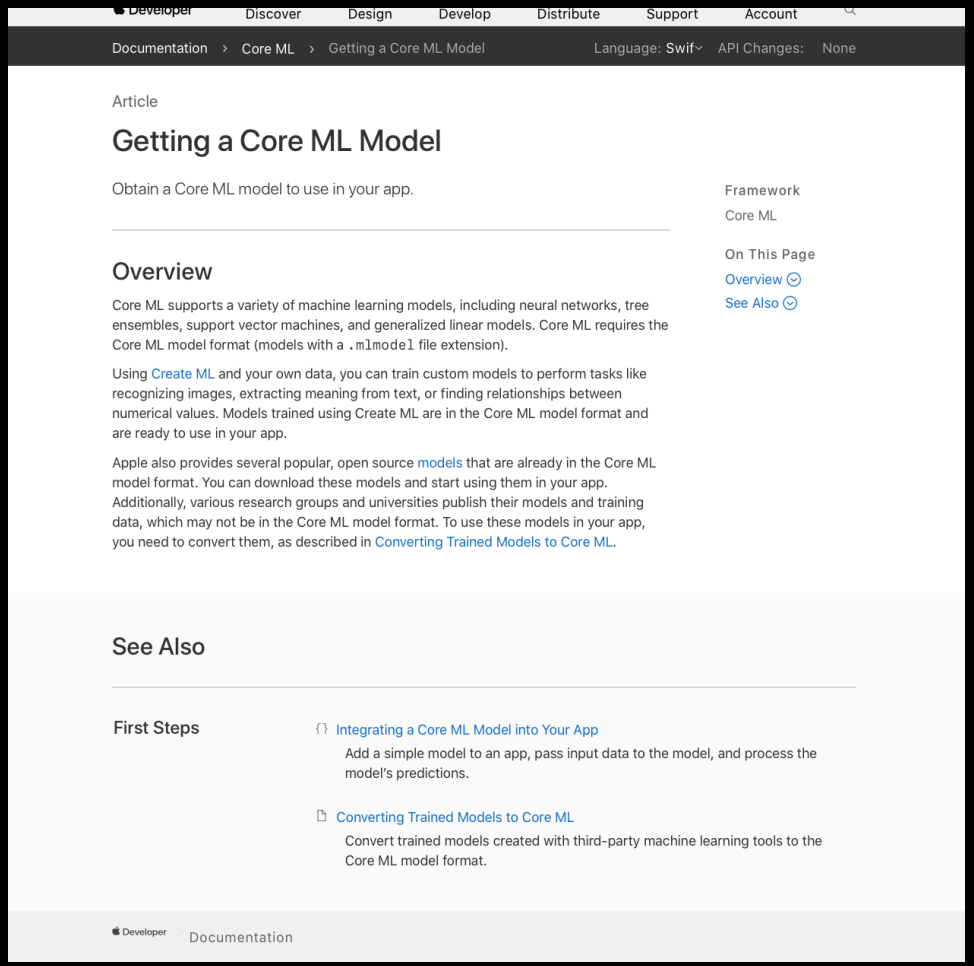
Introducing Core ML

Machine learning opens up opportunities for creating new and engaging experiences. Core ML is a new framework which you can use to easily integrate machine learning models into your app. See how Xcode and Core ML can help you make your app more intelligent with just a few lines of code.

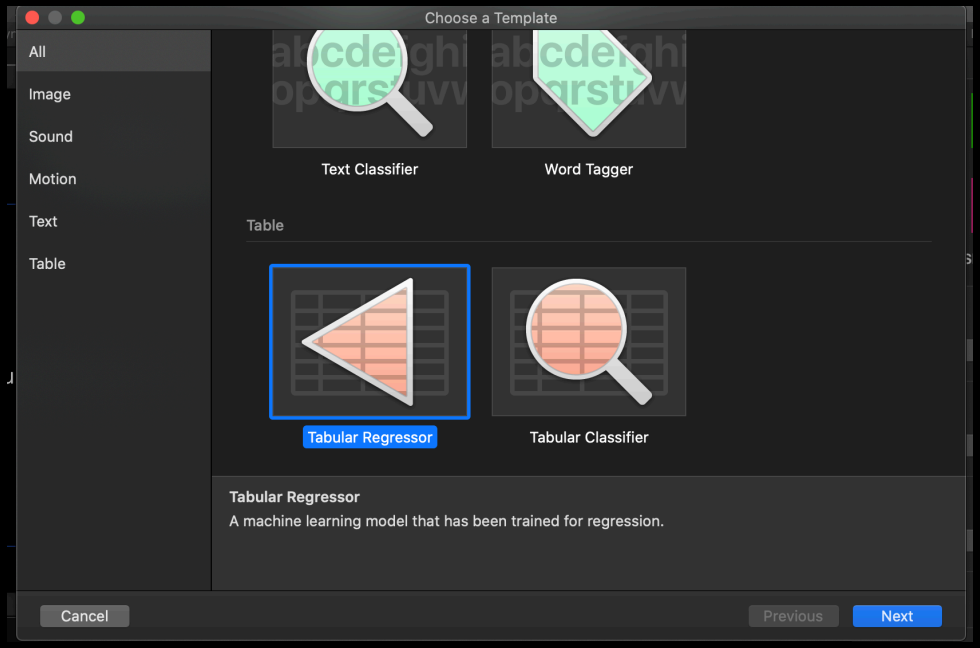
*be wary of statements like this

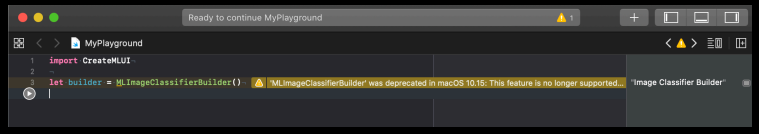
Drag your model into Xcode…

Where do I get a model?


Developer
Discover
Design
Develop
Distribute
Support
Account
Documentation › Core ML › Getting a Core ML Model
Language: Swif
API Changes: None
Article
Getting a Core ML Model
Obtain a Core ML model to use in your app.
Framework
Core ML
On This Page
Overview
See Also
Overview
Core ML supports a variety of machine learning models, including neural networks, tree ensembles, support vector machines, and generalized linear models. Core ML requires the Core ML model format (models with a .mlmodel file extension).
Using Create ML and your own data, you can train custom models to perform tasks like recognizing images, extracting meaning from text, or finding relationships between numerical values. Models trained using Create ML are in the Core ML model format and are ready to use in your app.
Apple also provides several popular, open source models that are already in the Core ML model format. You can download these models and start using them in your app. Additionally, various research groups and universities publish their models and training data, which may not be in the Core ML model format. To use these models in your app, you need to convert them, as described in Converting Trained Models to Core ML.
See Also
First Steps
Integrating a Core ML Model into Your App
Add a simple model to an app, pass input data to the model, and process the model's predictions.
Converting Trained Models to Core ML
Convert trained models created with third-party machine learning tools to the Core ML model format.
Developer
Documentation

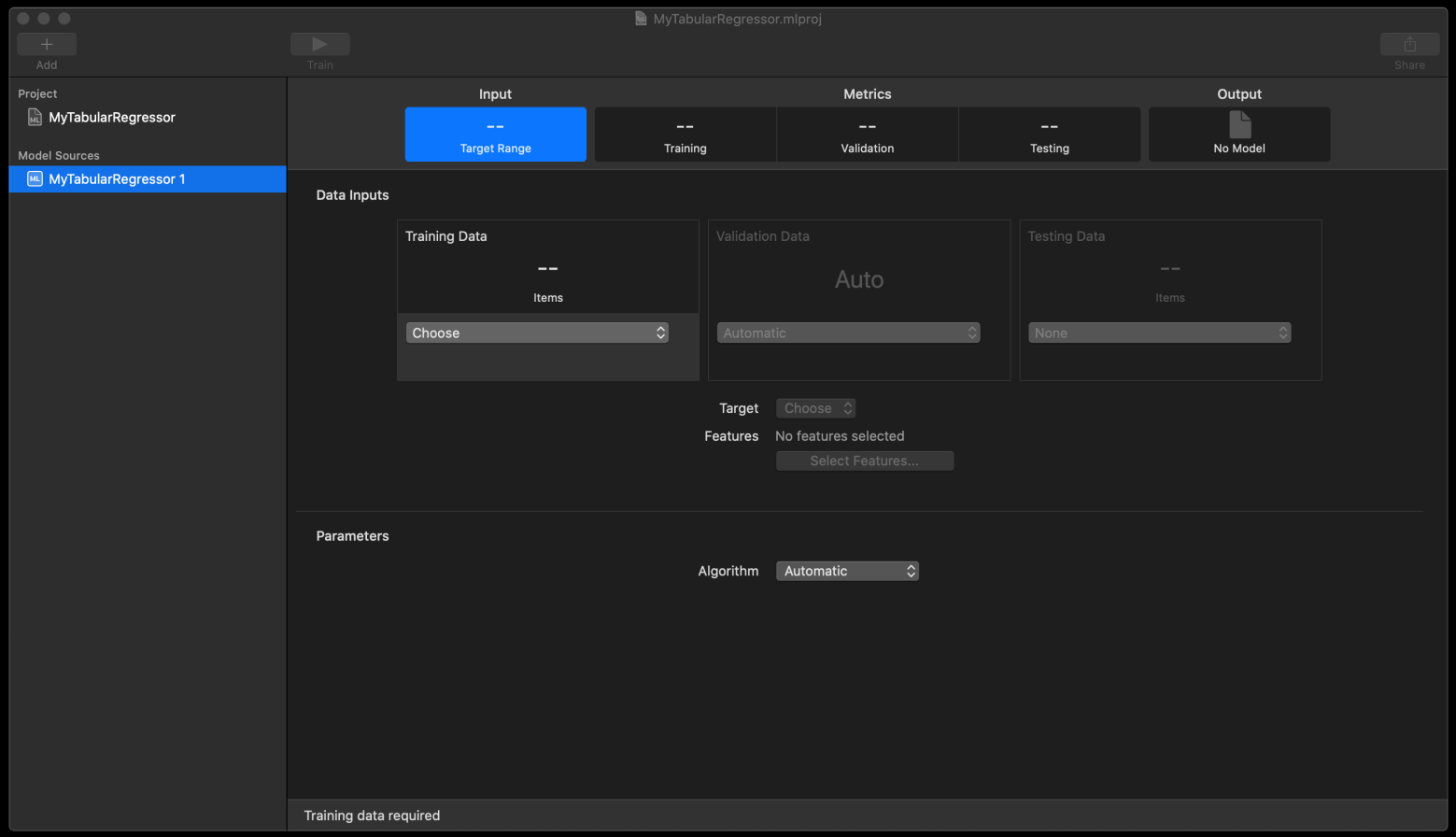
What about CreateML?





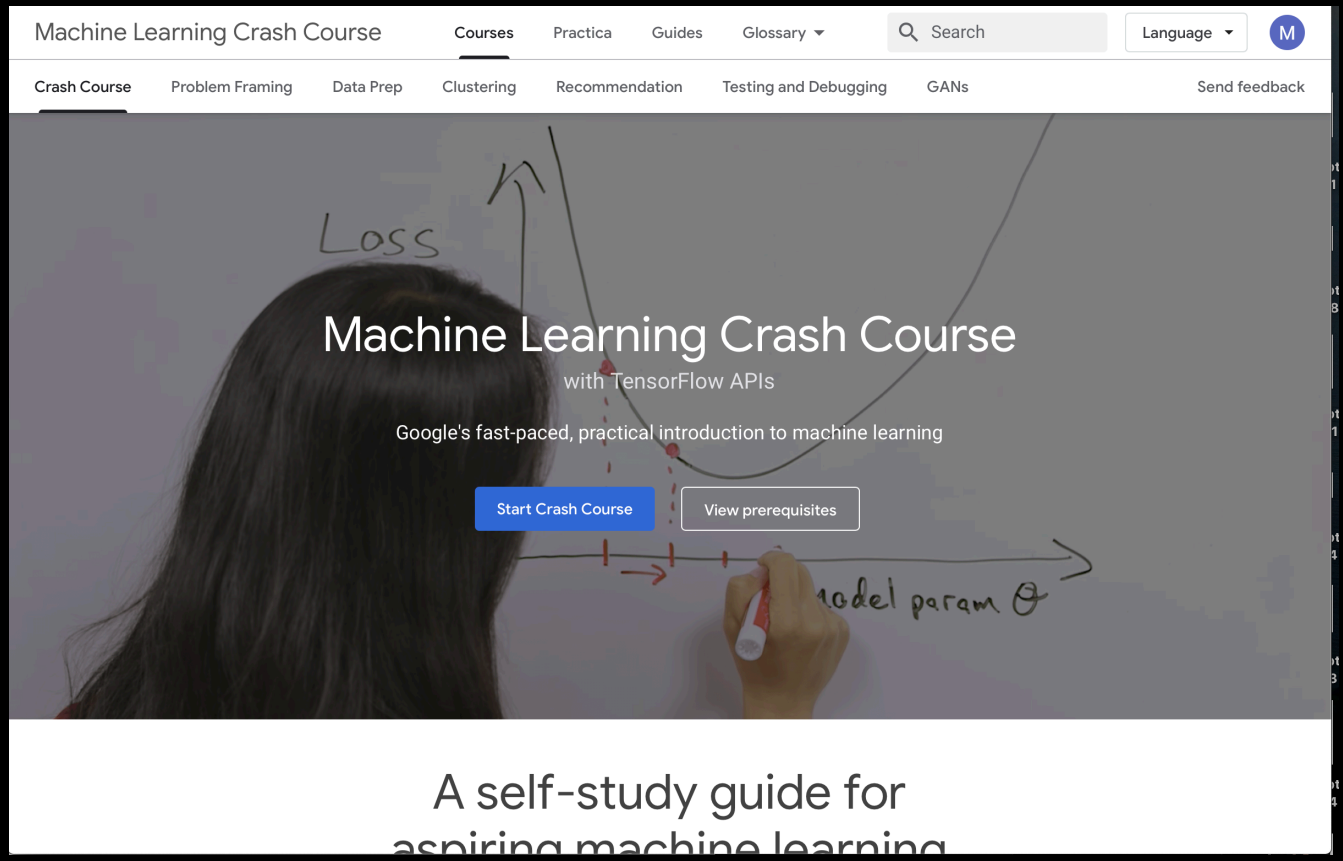
***RIP CreateMLUI**

Not really sure what to do here



I guess it’s not for me

2018


Machine Learning Crash Course
Courses
Practica
Guides
Glossary
Search
Language
M
Crash Course
Problem Framing
Data Prep
Clustering
Recommendation
Testing and Debugging
GANs
Send feedback
Loss
Machine Learning Crash Course
with TensorFlow APIs
Google's fast-paced, practical introduction to machine learning
Start Crash Course
View prerequisites
A self-study guide for

Until

Machine Learning Crash Course

Courses | Practica | Guides | Glossary | Search | Language | M

Crash Course | Problem Framing | Data Prep | Clustering | Recommendation | Testing and Debugging | GANs

ML Concepts

- Introduction to ML (3 min)
- Framing (15 min)
- Descending into ML (20 min)
- Reducing Loss (60 min)
  - Video Lecture
  - An Iterative Approach
  - Gradient Descent
  - Learning Rate
  - Optimizing Learning Rate
  - Stochastic Gradient Descent
  - Playground Exercise
  - Check Your Understanding
- First Steps with TF (60 min)
- Generalization (15 min)
- Training and Test Sets (25 min)
- Validation Set (40 min)

The gradient descent algorithm then calculates the gradient of the loss curve at the starti
3, the gradient of the loss is equal to the derivative (slope) of the curve, and tells you whic
"colder." When there are multiple weights, the **gradient** is a vector of partial derivatives wit

Click the plus icon to learn more about partial derivatives and gradients.

Note that a gradient is a vector, so it has both of the following characteristics:

- a direction
- a magnitude

The gradient always points in the direction of steepest increase in the loss function. The g
algorithm takes a step in the direction of the negative gradient in order to reduce loss as c

(negative) gradient

$$f(x, y) = e^{2y} \sin(x)$$

then:

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}(x, y), \frac{\partial f}{\partial y}(x, y) \right) = (e^{2y} \cos(x), 2e^{2y} \sin(x))$$

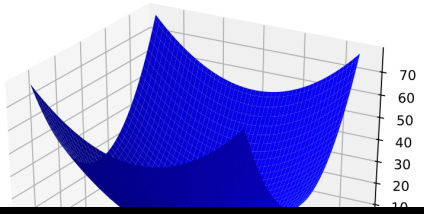
Note the following:

∇f	Points in the direction of greatest increase of the function.
$-\nabla f$	Points in the direction of greatest decrease of the function.

The number of dimensions in the vector is equal to the number of variables in the formula for f; in other words, the vector falls within the domain space of the function. For instance, the graph of the following function $f(x, y)$:

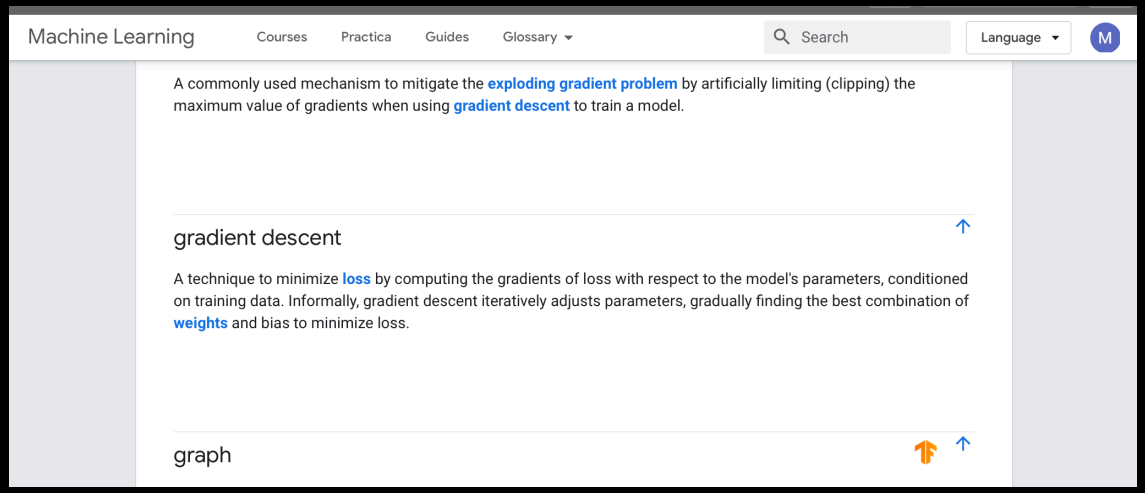
$$f(x, y) = 4 + (x - 2)^2 + 2y^2$$

when viewed in three dimensions with $z = f(x, y)$ looks like a valley with a minimum at $(2, 0, 4)$:



I guess it’s not for me

*shout out - the glossary is 👍



https://developers.google.com/machine-learning/glossary

2019

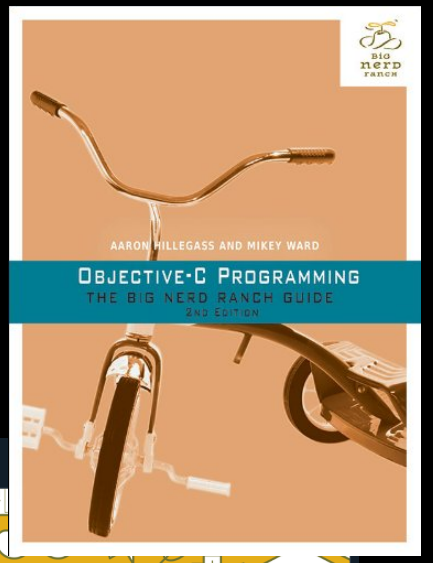
Tweet

Aaron Hillegass
@AaronHillegass

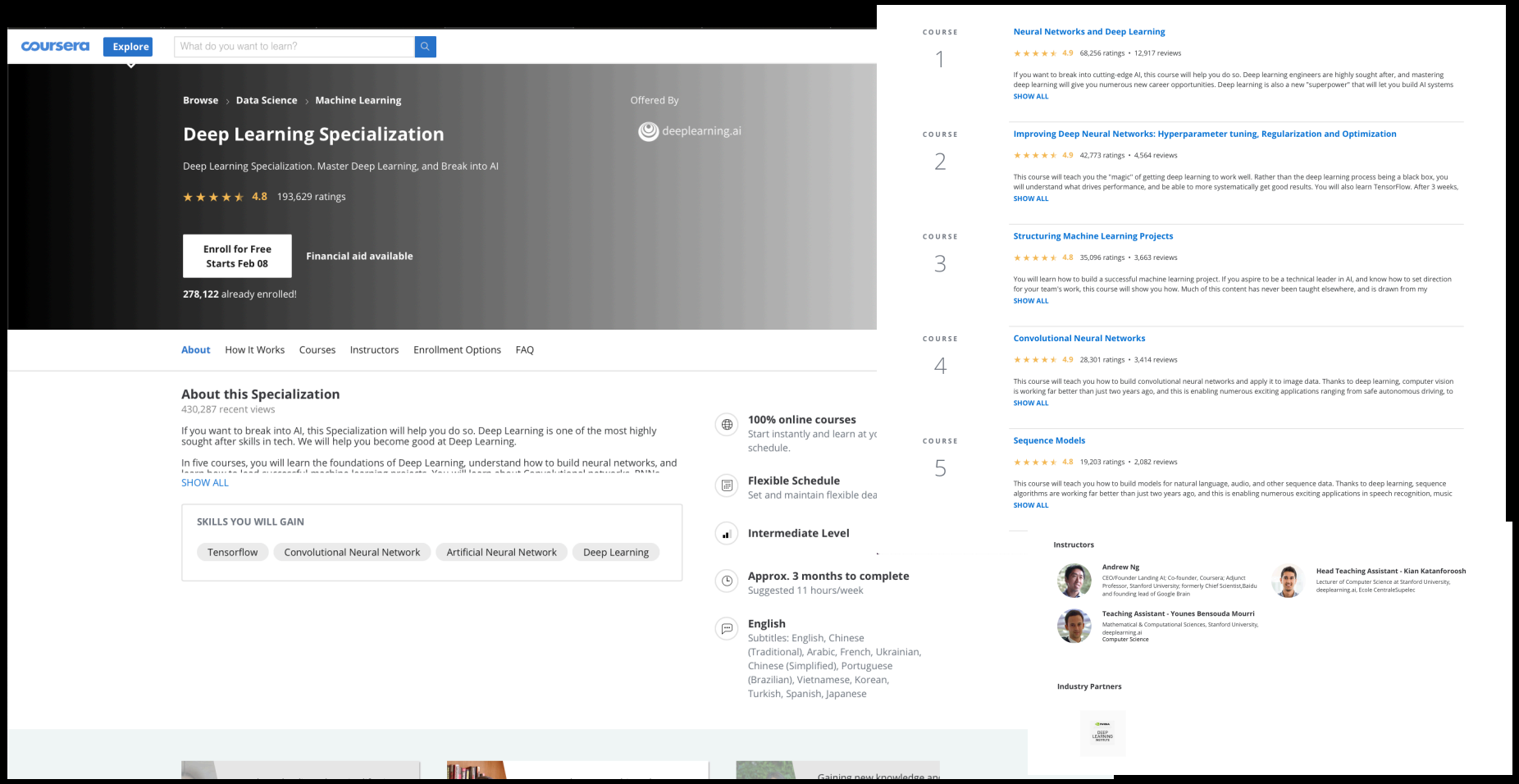
I have finished my first year studying machine learning at Georgia Tech. Last week, I studied wikipedia to implement Baum-Welch. The page lacked important details, and I thought, "Someone who knows machine learning should fix this."

And then, feeling 100% that impostor, I did.

8:53 AM · Dec 16, 2019 · Twitter Web App

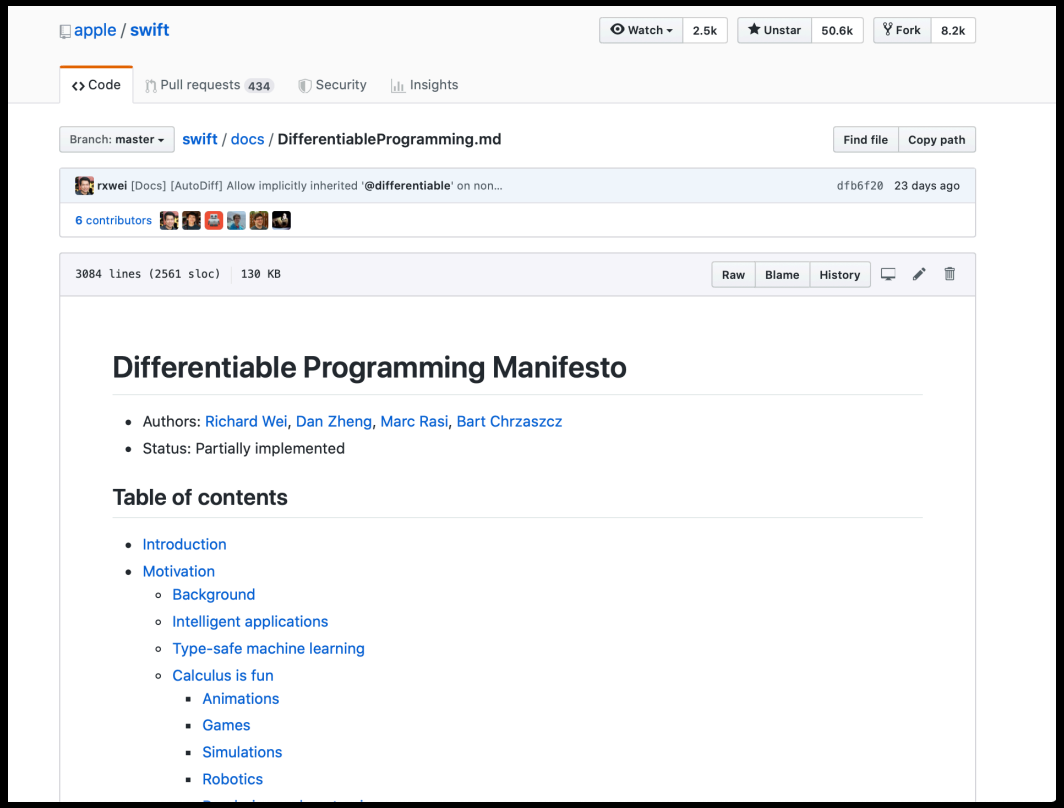


Aaron recommends



It’s for me!
and you…

Why should I care?



apple / swift

Watch 2.5k | Unstar 50.6k | Fork 8.2k

Code | Pull requests 434 | Security | Insights

Branch: master ▾ swift / docs / DifferentiableProgramming.md — Find file | Copy path

rxwei [Docs] [AutoDiff] Allow implicitly inherited '@differentiable' on non... dfb6f20 23 days ago

6 contributors

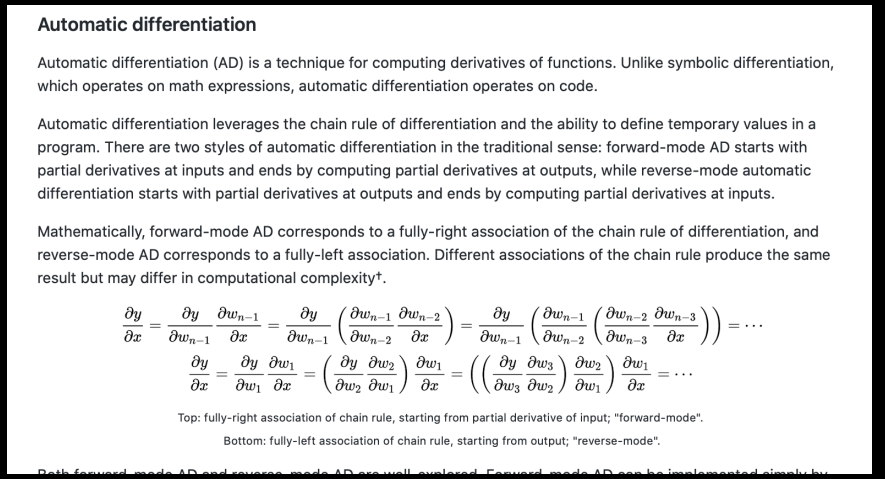
3084 lines (2561 sloc) 130 KB — Raw | Blame | History

Differentiable Programming Manifesto

- Authors: Richard Wei, Dan Zheng, Marc Rasi, Bart Chrzaszcz
- Status: Partially implemented

Table of contents

- Introduction
- Motivation
  - Background
  - Intelligent applications
  - Type-safe machine learning
  - Calculus is fun
    - Animations
    - Games
    - Simulations
    - Robotics



Automatic differentiation

Automatic differentiation (AD) is a technique for computing derivatives of functions. Unlike symbolic differentiation, which operates on math expressions, automatic differentiation operates on code.

Automatic differentiation leverages the chain rule of differentiation and the ability to define temporary values in a program. There are two styles of automatic differentiation in the traditional sense: forward-mode AD starts with partial derivatives at inputs and ends by computing partial derivatives at outputs, while reverse-mode automatic differentiation starts with partial derivatives at outputs and ends by computing partial derivatives at inputs.

Mathematically, forward-mode AD corresponds to a fully-right association of the chain rule of differentiation, and reverse-mode AD corresponds to a fully-left association. Different associations of the chain rule produce the same result but may differ in computational complexity†.

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_{n-1}}\frac{\partial w_{n-1}}{\partial x} = \frac{\partial y}{\partial w_{n-1}}\left(\frac{\partial w_{n-1}}{\partial w_{n-2}}\frac{\partial w_{n-2}}{\partial x}\right) = \frac{\partial y}{\partial w_{n-1}}\left(\frac{\partial w_{n-1}}{\partial w_{n-2}}\left(\frac{\partial w_{n-2}}{\partial w_{n-3}}\frac{\partial w_{n-3}}{\partial x}\right)\right) = \cdots$$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_1}\frac{\partial w_1}{\partial x} = \left(\frac{\partial y}{\partial w_2}\frac{\partial w_2}{\partial w_1}\right)\frac{\partial w_1}{\partial x} = \left(\left(\frac{\partial y}{\partial w_3}\frac{\partial w_3}{\partial w_2}\right)\frac{\partial w_2}{\partial w_1}\right)\frac{\partial w_1}{\partial x} = \cdots$$

Top: fully-right association of chain rule, starting from partial derivative of input; "forward-mode".

Bottom: fully-left association of chain rule, starting from output; "reverse-mode".

Both forward-mode AD and reverse-mode AD are well-explored. Forward-mode AD can be implemented simply by

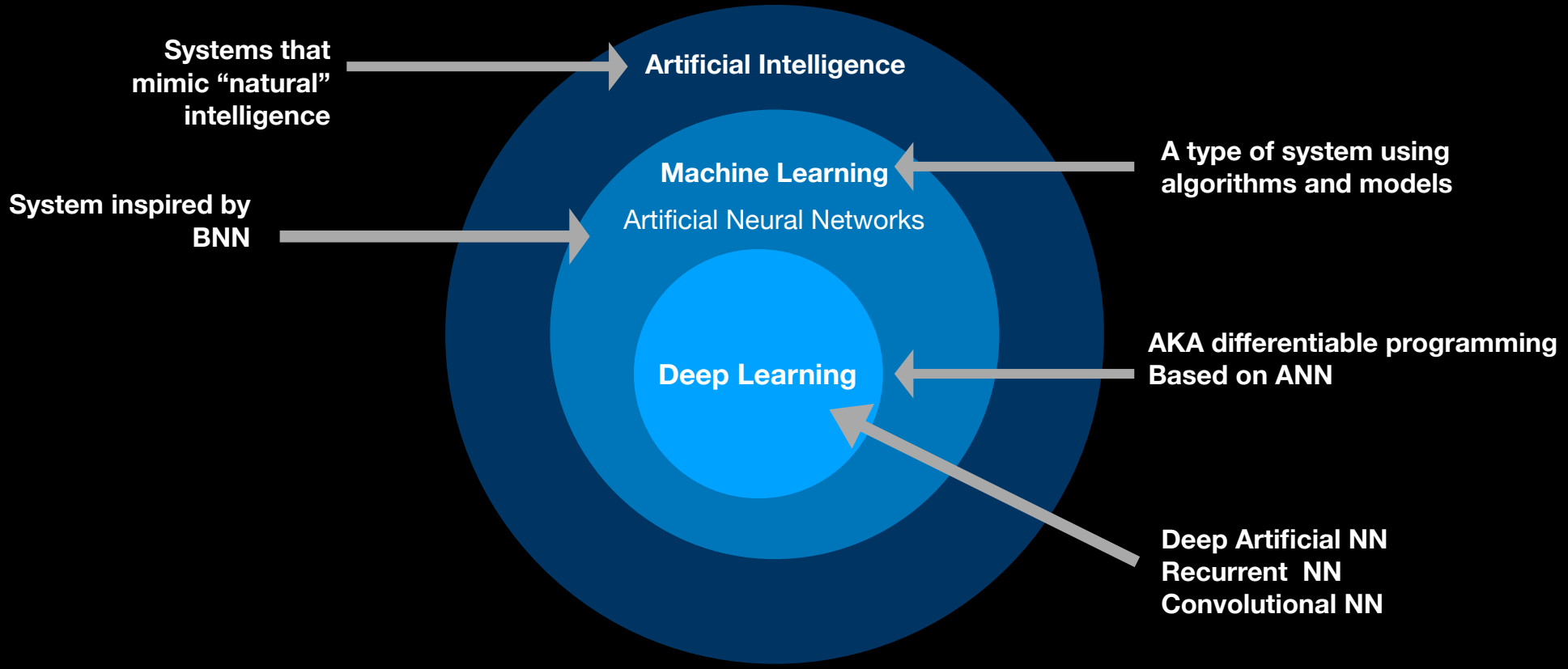
What did I learn?

What did I learn?

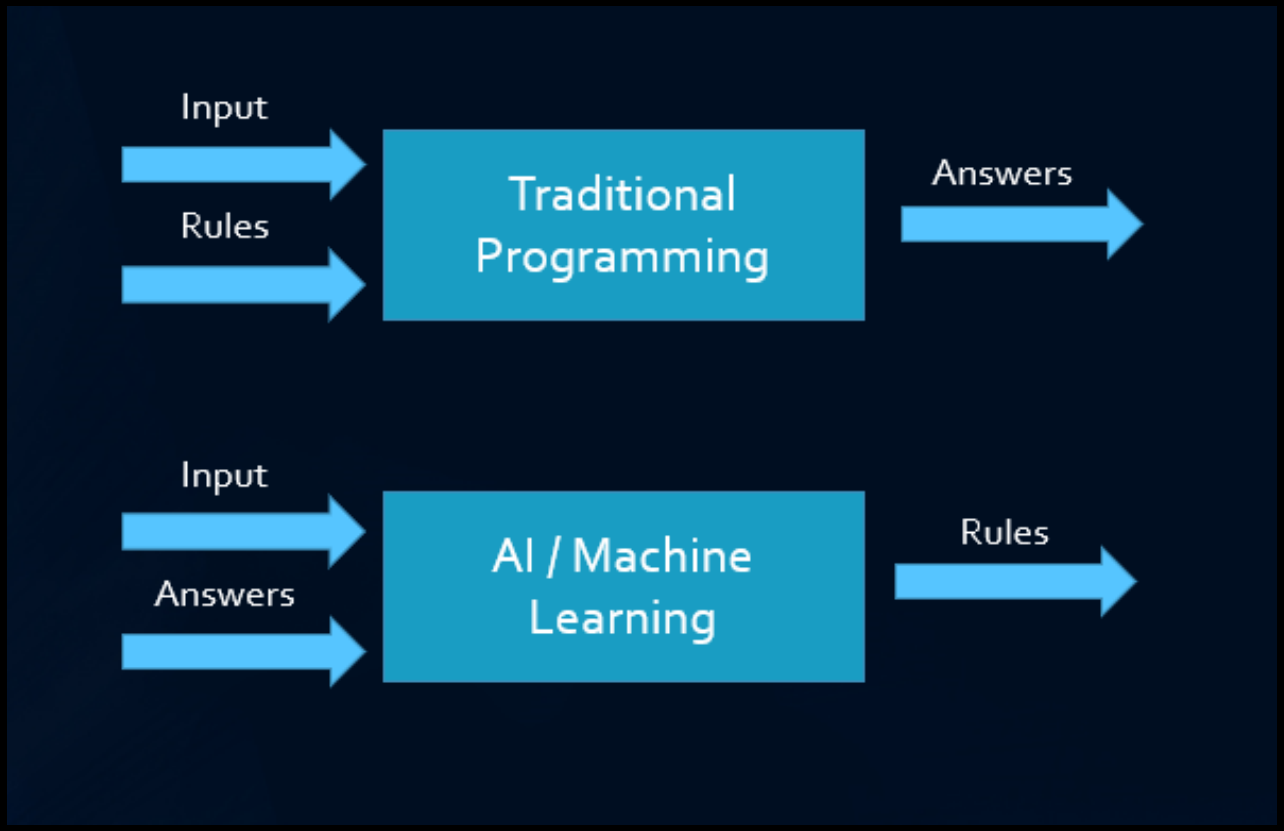
- **Coding** is the easy part, **theory** is the hard part
- Don’t be too eager to code, it’s less important
- It’s not as daunting as you might think
- Be patient, there is **a lot** to learn

Let’s move on - ML Building Blocks

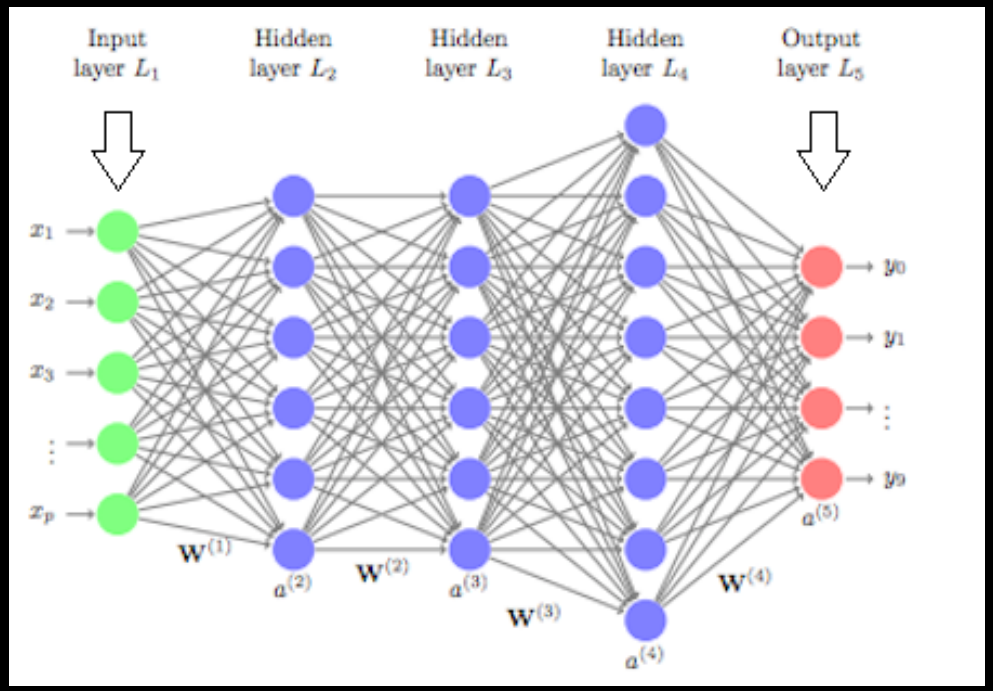
Terminology

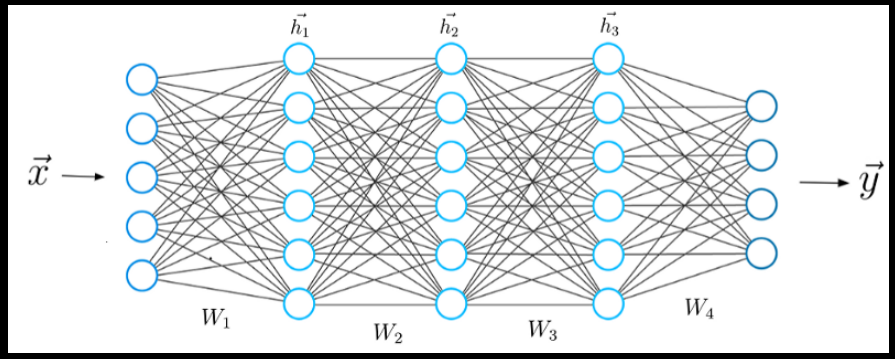


How does it compare to traditional programming?



Deep Artificial Neural Network - DNN



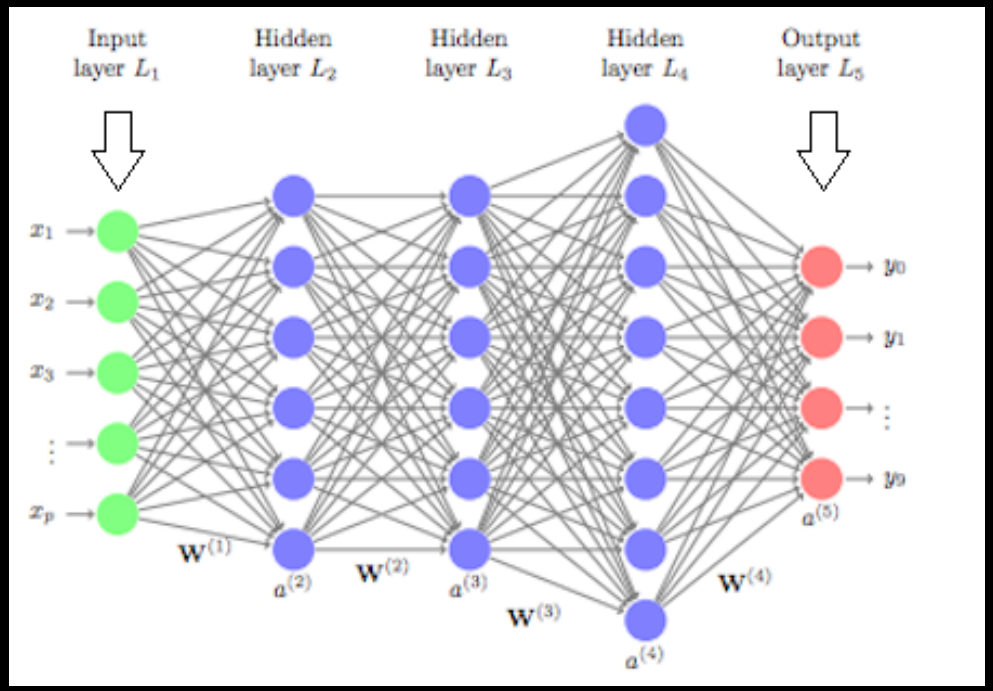


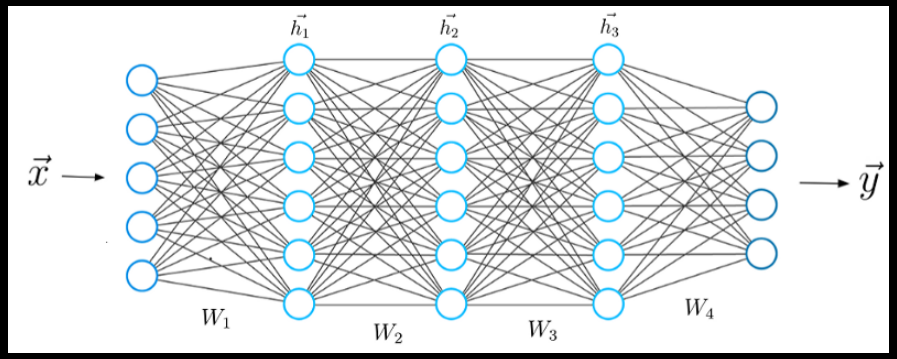
A ML model is just a function

Make a deep neural network

- Prepare the data
- Loop for a number of iterations
  - Forward step to make a prediction
  - Compute the loss
  - Backward step to update parameters
- Use the model to make predictions

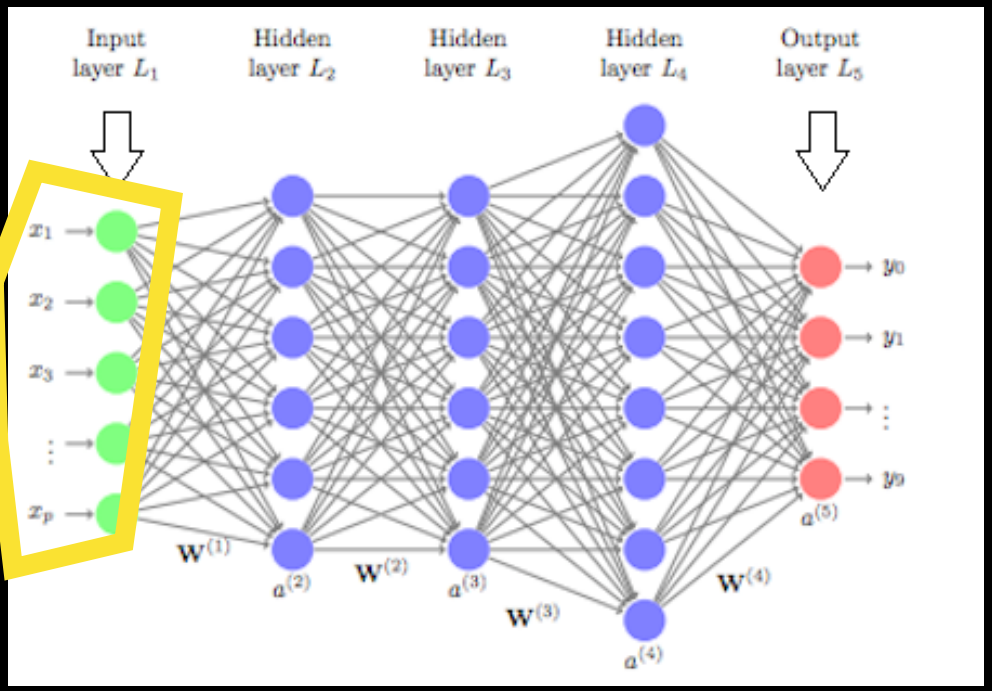
Forward step: Forward Propagation





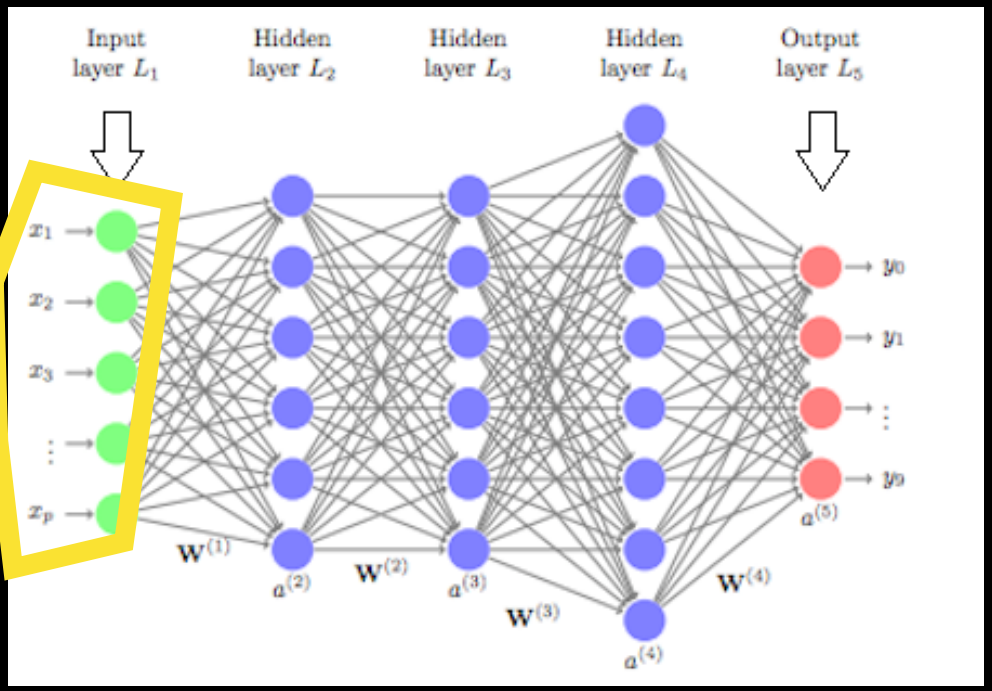
What is X?

- X is our input feature “vector” or array
- Recall that a 2D array is a matrix
- Input array has a “shape” or dimension
- Number of X features by number of examples
- We write this as (nx, m)



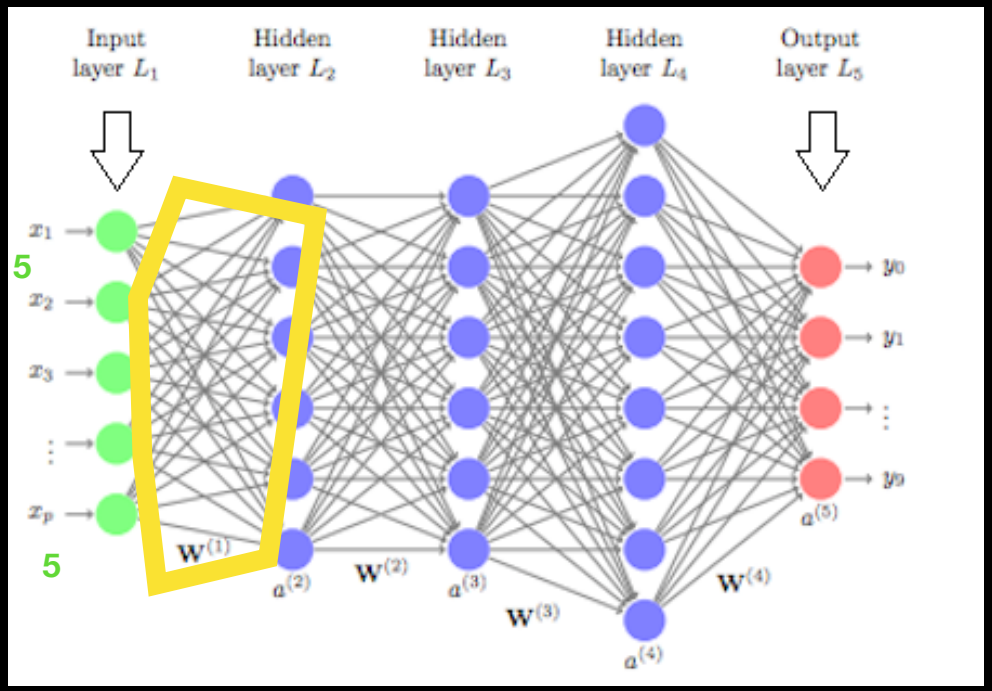
Make it real

- We write this as (nx, m)
- One feature - one example
- [[1]]
- 2 features, 2 examples
- [[1,1], [2,2]]



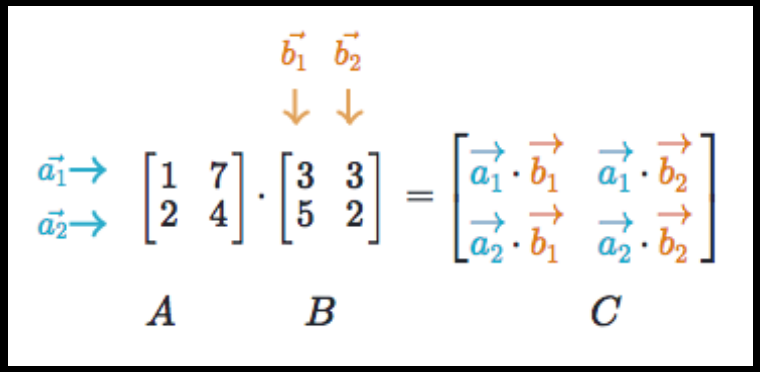
Next step - weighted sum

- We have another matrix of weights, called W
- The weight matrix has dimensions of number of hidden units by number of units in previous layer
- e.g. W1 has a shape of (7,5)
- We compute the matrix "dot product" (**dot product** is the sum of the **products** of the corresponding entries)
- = **X · W1**



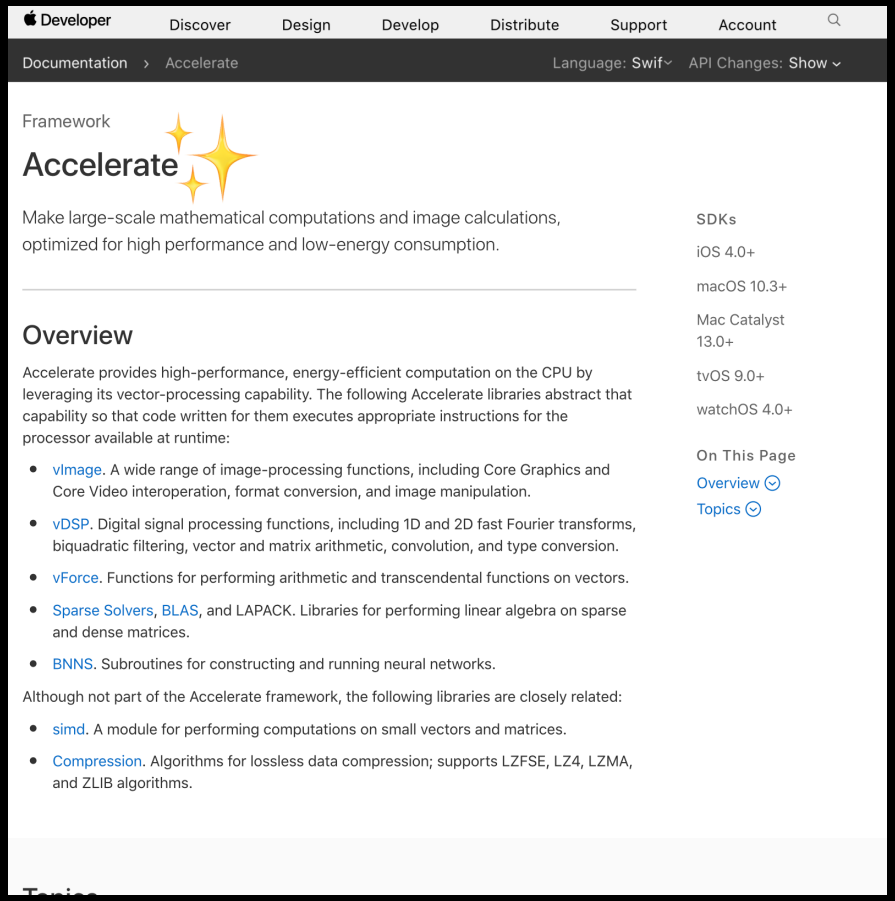
Single instruction, multiple data (SIMD)

```swift
import simd

let X = simd_float2x2(rows: [
    SIMD2<Float>(x: 1, y: 7),
    SIMD2<Float>(x: 2, y: 4),
])

let Y = simd_float2x2(rows: [
    SIMD2<Float>(x: 3, y: 3),
    SIMD2<Float>(x: 5, y: 2),
])

let dotProduct = matrix_multiply(X, Y)

print(dotProduct) // simd_float2x2([[38.0, 26.0], [17.0, 14.0]])
```



Why do we use vectors and not just array?

- Performance!
- Vectorized implementations are much faster
- Parallel computation

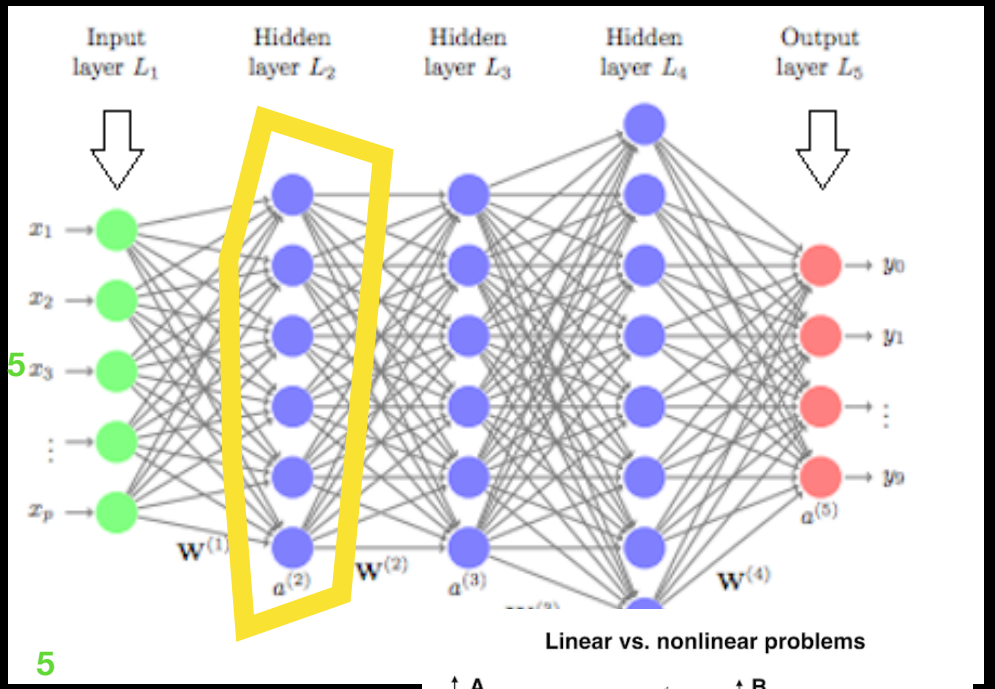


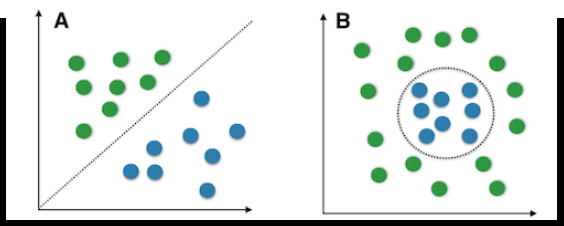
Why do we use vectors and not just array?

```
var x = [Float]()
for _ in 0...10_000 {
    x.append(Float(drand48() * 10_000))
}

var time = Date()
let y = x.map {
    return sqrt($0)
}
print("non vectorized: \(time.timeIntervalSinceNow)") // non vectorized:
0.16907799243927002

import Accelerate.vecLib

time = Date()
var n2 = Int32(x.count)
var y2 = [Float](repeating: 0, count: x.count)

vvsqrtf(&y2, x, &n2)
print("vectorized: \(time.timeIntervalSinceNow)") // "vectorized: 0.007734894752502441\n"
```

Next step - activation

- Why do we need an activation?
- Many linear functions are still just a giant linear function
- Non linear functions allow model to learn non linear relationships

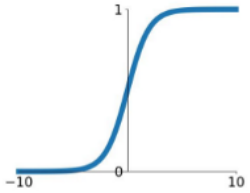




Common Activations

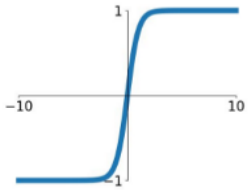
Sigmoid

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



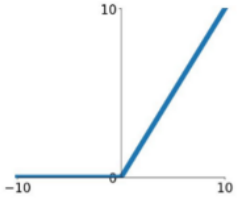
tanh

$$\tanh(x)$$



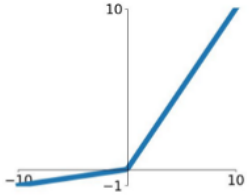
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

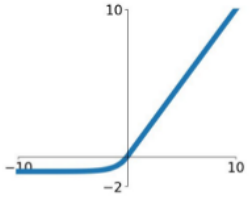


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

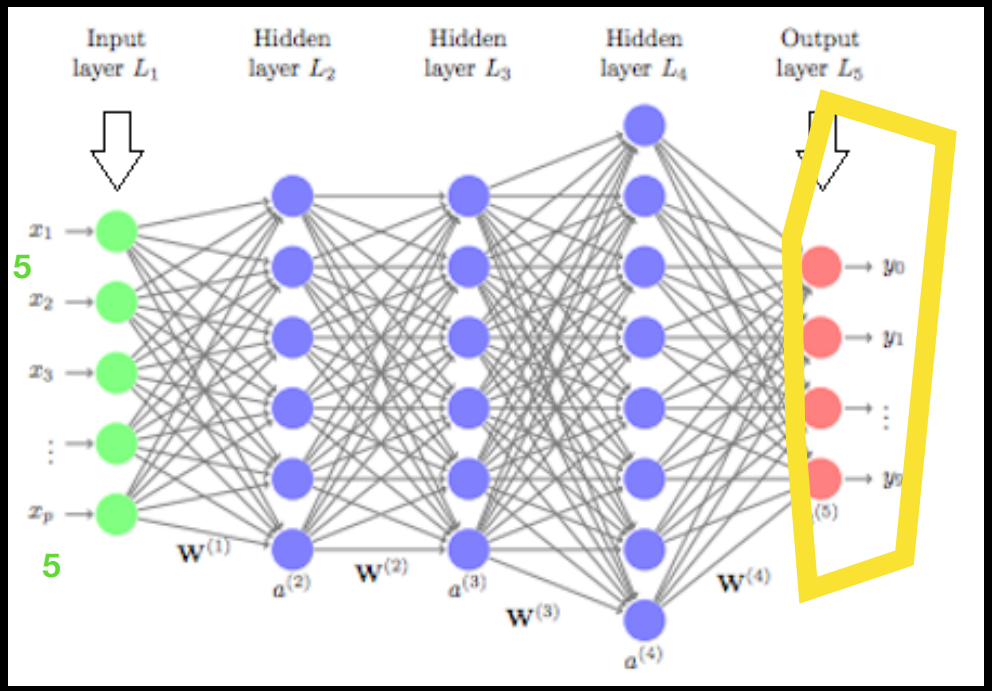
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



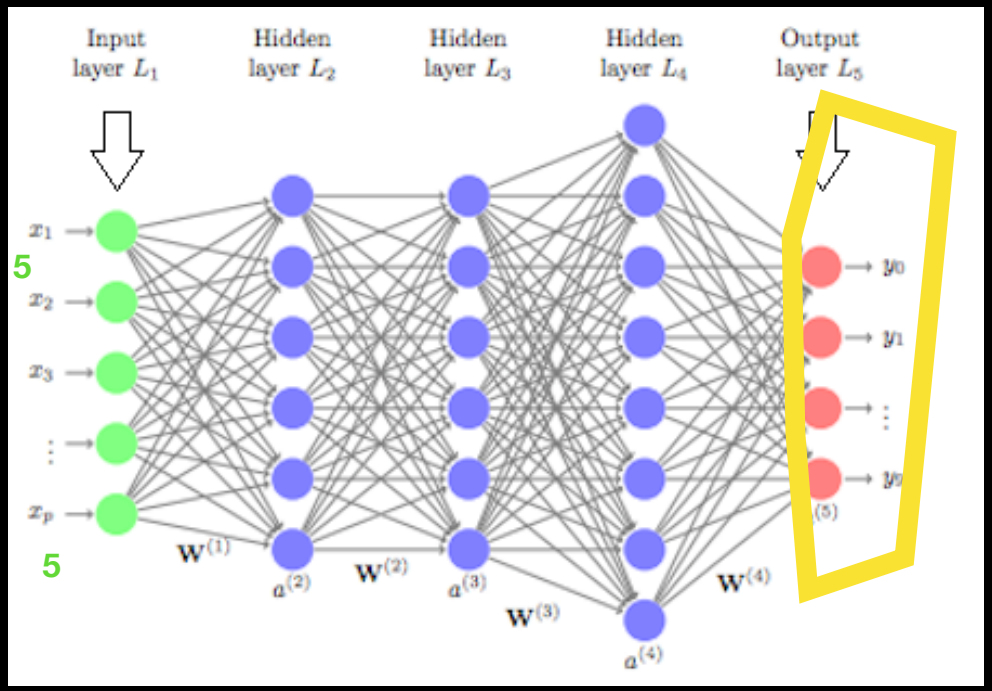
Next step - prediction and loss

- Prediction denoted ŷ (y hat)
- $\hat{y} = \sigma(WX + b)$
- Linear regression output is a value (continuous)
- Logistic regression outputs a probability (between 0 and 1)



Next step - prediction and loss

- Loss is difference between prediction and “true” label
- Linear regression uses mean squared error, logistic regression uses cross entropy
- This is called a cost function
- Low loss means performing well on data set



Introducing Cost Function

Given the current weights (parameters), how good are we doing at predicting?

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

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

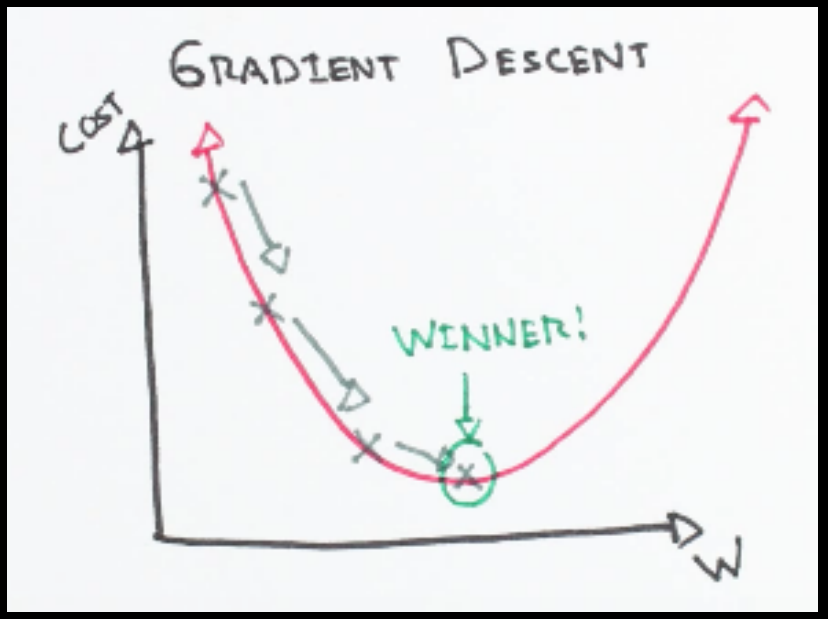
Now we got backwards

Backpropagation

But my friends call me backprop

Backprop

- Compute gradients with respect to weights
- For gradient descent, we update the parameters with the gradients
- Backprop and gradient descent are the “automatic” part of neural networks, it is how our model learns to better fit the data



Gradient Descent

It’s objective is to find the parameters that minimize the cost function

Answers “What is the weight that will minimize the loss?”

The GD update

perform many iterations of update rule

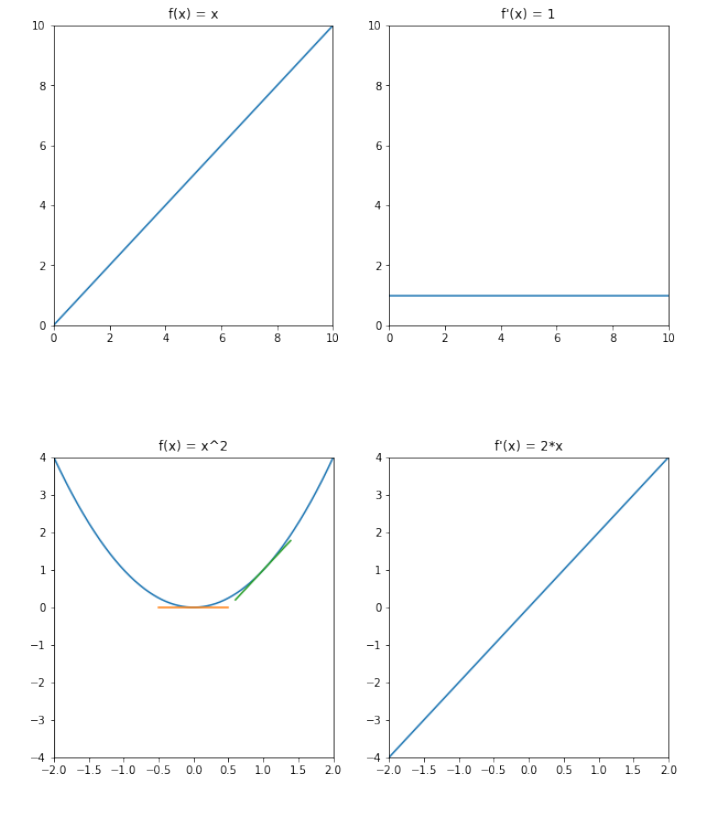
$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$$

:= update current value in place

α How big of a step to take each iteration

d/d**θ** * J**(θ) derivative of J with respect to theta**

What is a derivative?



Derivative of f at x can be summarized as a single real number f'(x) such that $f(x + \varepsilon) \approx f(x) + f'(x) * \varepsilon$

“how quickly function output changes when you make small changes to the function’s input”

What is a derivative?

Measures how sensitive a function output is to changes in input.

ex. $y = f(x)$

derivative of f w.r.t x measures how much y changes with respect to a change in x

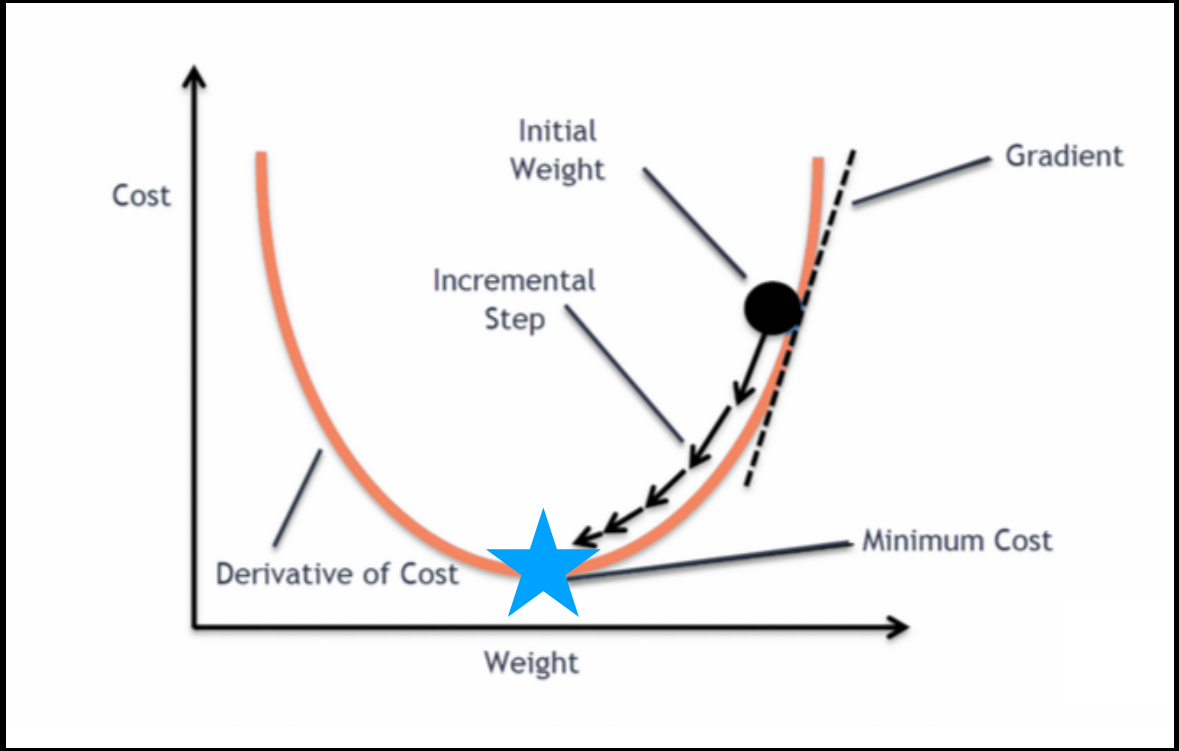
Can be written as df/dx or dx or d/dx f(x).

When there are multiple parameters, we call it a partial derivative and use ∂

Rules	Function	Derivative
Multiplication by constant	cf	cf'
Power Rule	x^n	nx^{n-1}
Sum Rule	$f + g$	$f' + g'$
Difference Rule	$f - g$	$f' - g'$
Product Rule	fg	$f\,g' + f'\,g$
Quotient Rule	f/g	$(f'\,g - g'\,f)/g^2$
Reciprocal Rule	$1/f$	$-f'/f^2$

https://www.mathsisfun.com/calculus/derivatives-rules.html

Why is that useful?



Convergence

- We repeat this iterative update until our model is changing the weights very little on each iteration, it is at this point that we might say our model has converged

So what next?

- We evaluate the model on unseen data (validation set, test set)
- We make adjustments, repeat
- Use the model to make predictions
- There is a lot of neat stuff that we didn't talk about, regularization, optimization techniques, different model architectures, ..

Demo with Tensorflow Visualizer

Conclusion / Review

Take aways for you

- Be a beginner again every once in a while
- Embrace your confusion
- Learn the theories behind ML and deep learning to get more out of it
- If I've confused you, it's because I'm a poor teacher and not because you can't understand it

Citations and Sources

- Google Machine Learning Glossary - developers.google.com/machine-learning/glossary
- Coursera Deep Learning Specialization - https://www.coursera.org/specializations/deep-learning
- https://www.mathsisfun.com/calculus/derivatives-rules.html
- https://github.com/apple/swift/blob/master/docs/DifferentiableProgramming.md

Questions