

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



*be wary of statements like this

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.

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: Swift API Changes: None

Article

Getting a Core ML Model

Obtain a Core ML model to use in your app.

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

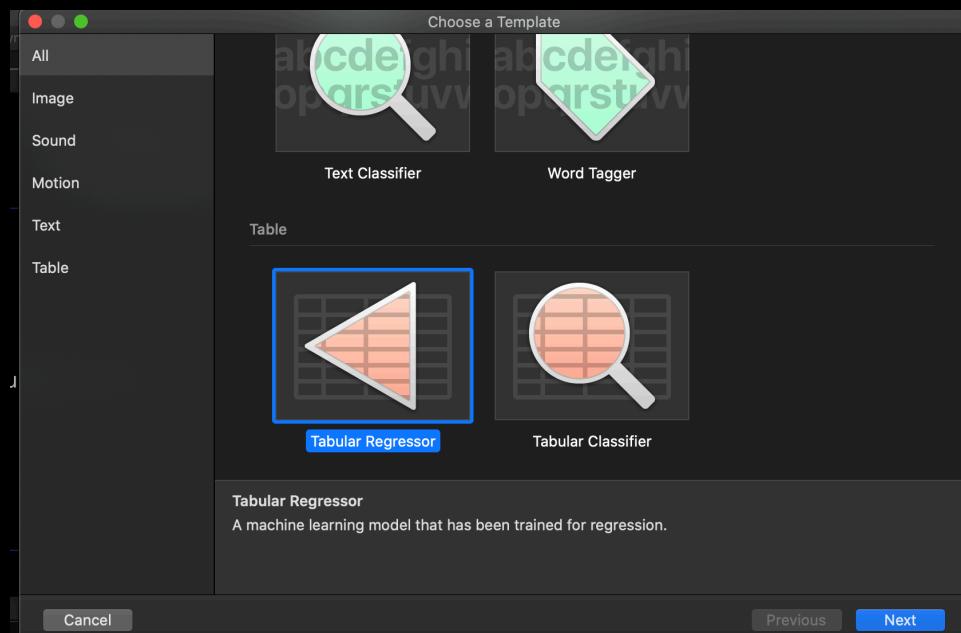
First Steps

- ⓘ [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?

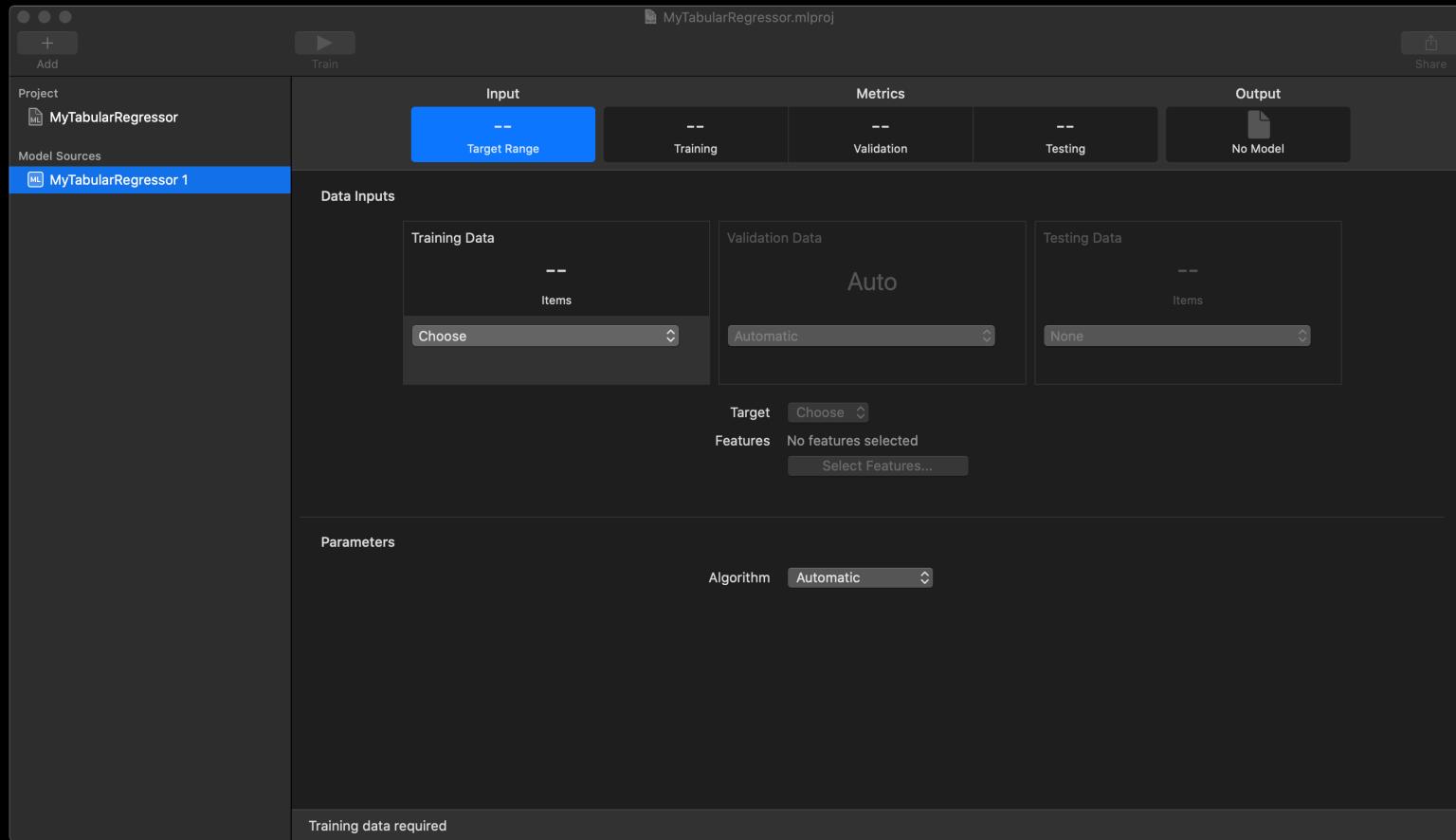
A screenshot of an Xcode playground window titled "MyPlayground". The code editor contains the following Swift code:

```
import CreateML
let builder = MLImageClassifierBuilder()
```

A warning message is visible in the status bar: "MLImageClassifierBuilder was deprecated in macOS 10.15; This feature is no longer supported." The status bar also shows "Ready to continue MyPlayground".

*RIP CreateMLUI

Not really sure what to do here



I guess it's not for me

2018

The screenshot shows the homepage of the Machine Learning Crash Course website. At the top, there is a navigation bar with links for "Courses", "Practica", "Guides", "Glossary", a search bar, a language selection dropdown, and a user profile icon. Below the navigation bar, there is a secondary navigation menu with links for "Crash Course", "Problem Framing", "Data Prep", "Clustering", "Recommendation", "Testing and Debugging", "GANs", and a "Send feedback" button. The main content area features a background image of a person writing on a whiteboard with mathematical notation like "Loss" and "model param θ ". Overlaid on this image is the title "Machine Learning Crash Course" and subtitle "with TensorFlow APIs". Below the title, a description reads "Google's fast-paced, practical introduction to machine learning". There are two buttons: a blue "Start Crash Course" button and a white "View prerequisites" button. At the bottom of the page, a white banner contains the text "A self-study guide for aspiring machine learning".

Until

Machine Learning Crash Course

Courses Practice 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** (selected)
 - 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 starting point. At each step, the gradient of the loss is equal to the derivative (slope) of the curve, and tells you which direction is "colder." When there are multiple weights, the gradient is a vector of partial derivatives with respect to each weight.

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 gradient descent algorithm takes a step in the direction of the negative gradient in order to reduce loss as quickly as possible.

(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.

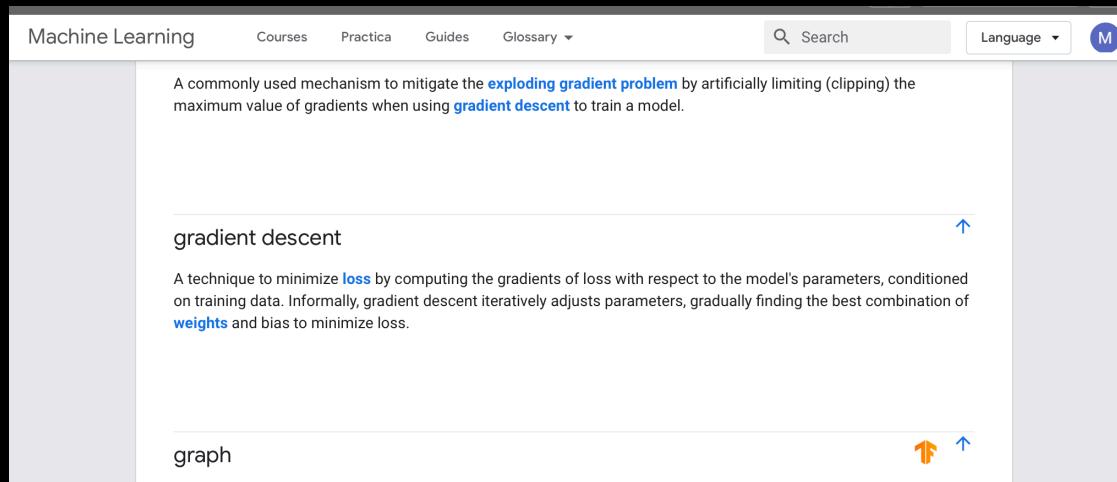
The number of dimensions in the vector is equal to the number of variables in the formula for f ; in other words, the vector falls within the domain space of the function. For instance, the graph of the following function $f(x, y)$:

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

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 Hillegass and Mikey Ward

OBJECTIVE-C PROGRAMMING
THE BIG NERD RANCH GUIDE
2ND EDITION



[Aaron Hillegass](#)
2,959 Tweets

 Big Nerd Ranch

Following

Aaron Hillegass
@AaronHillegass

Author of books on Cocoa, iOS, and Objective-C. Founder of Big Nerd Ranch, Inc. Currently studying machine learning at Georgia Tech.

Atlanta, GA ♂ [bignerdranch.com](#) Joined October 2009

66 Following 19.3K Followers

Aaron recommends

Coursera Explore What do you want to learn? 

Browse > Data Science > Machine Learning

Deep Learning Specialization

Deep Learning Specialization. Master Deep Learning, and Break into AI

★★★★★ 4.8 193,629 ratings

Enroll for Free Starts Feb 08 Financial aid available

278,122 already enrolled!

About How It Works Courses Instructors Enrollment Options FAQ

About this Specialization

430,287 recent views

If you want to break into AI, this Specialization will help you do so. Deep Learning is one of the most highly sought after skills in tech. We will help you become good at Deep Learning.

In five courses, you will learn the foundations of Deep Learning, understand how to build neural networks, and

SHOW ALL

SKILLS YOU WILL GAIN

Tensorflow Convolutional Neural Network Artificial Neural Network Deep Learning

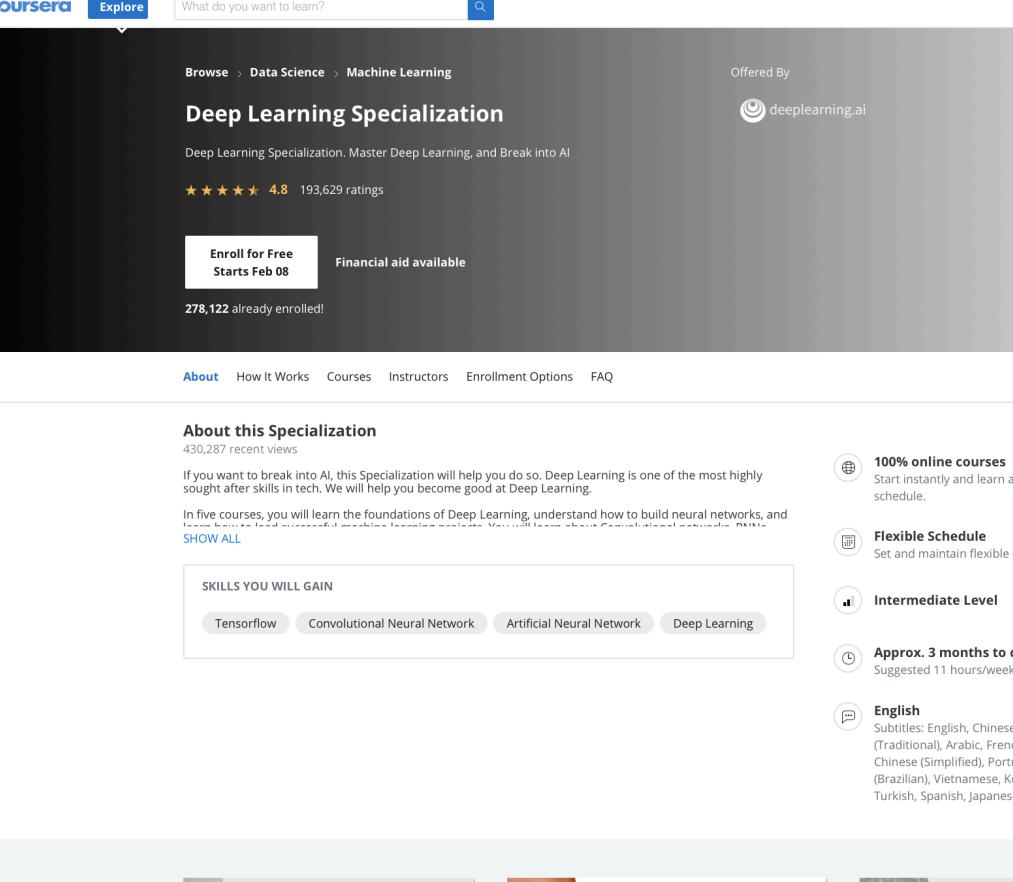
100% online courses
Start instantly and learn at your own pace.

Flexible Schedule
Set and maintain flexible deadlines.

Intermediate Level

Approx. 3 months to complete
Suggested 11 hours/week

English
Subtitles: English, Chinese (Traditional), Arabic, French, Ukrainian, Chinese (Simplified), Portuguese (Brazilian), Vietnamese, Korean, Turkish, Spanish, Japanese

 Gaining new knowledge at your own pace.

COURSE 1 **Neural Networks and Deep Learning**
★★★★★ 4.9 68,256 ratings • 12,917 reviews
SHOW ALL

COURSE 2 **Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization**
★★★★★ 4.9 42,773 ratings • 4,564 reviews
SHOW ALL

COURSE 3 **Structuring Machine Learning Projects**
★★★★★ 4.8 35,096 ratings • 3,663 reviews
SHOW ALL

COURSE 4 **Convolutional Neural Networks**
★★★★★ 4.9 28,301 ratings • 3,414 reviews
SHOW ALL

COURSE 5 **Sequence Models**
★★★★★ 4.8 19,203 ratings • 2,082 reviews
SHOW ALL

Instructors

 **Andrew Ng**
CEO/Founder Landing AI; Co-founder, Coursera; Adjunct Professor, Stanford University; formerly Chief Scientist, Baidu and founding lead of Google Brain

 **Kian Katanforoosh**
Head Teaching Assistant - Kian Katanforoosh
Lecture of Computer Science at Stanford University, deeplearning.ai, Ecole CentraleSupélec

Teaching Assistant - Younes Bensouda Mouri
Mathematical & Computational Sciences, Stanford University, deeplearning.ai, Computer Science

Industry Partners



**It's for me!
and you...**

Why should I care?

The screenshot shows the GitHub repository page for `apple / swift`. The repository has 2.5k stars, 50.6k forks, and 8.2k issues. The `Code` tab is selected, showing a file named `DifferentiableProgramming.md`. The commit history shows a recent update by `rxwei` on `dfb6f20` 23 days ago. The file contains 3084 lines (2561 sloc) and is 130 KB. The content of the file is titled "Differentiable Programming Manifesto" and includes a table of contents and a list of authors.

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[†].

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_{n-1}} \frac{\partial w_{n-1}}{\partial x} = \frac{\partial y}{\partial w_{n-1}} \left(\frac{\partial w_{n-1}}{\partial w_{n-2}} \frac{\partial w_{n-2}}{\partial x} \right) = \frac{\partial y}{\partial w_{n-1}} \left(\frac{\partial w_{n-1}}{\partial w_{n-2}} \left(\frac{\partial w_{n-2}}{\partial w_{n-3}} \frac{\partial w_{n-3}}{\partial x} \right) \right) = \dots$$
$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_1} \frac{\partial w_1}{\partial x} = \left(\frac{\partial y}{\partial w_2} \frac{\partial w_2}{\partial w_1} \right) \frac{\partial w_1}{\partial x} = \left(\left(\frac{\partial y}{\partial w_3} \frac{\partial w_3}{\partial w_2} \right) \frac{\partial w_2}{\partial w_1} \right) \frac{\partial w_1}{\partial x} = \dots$$

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 easily by

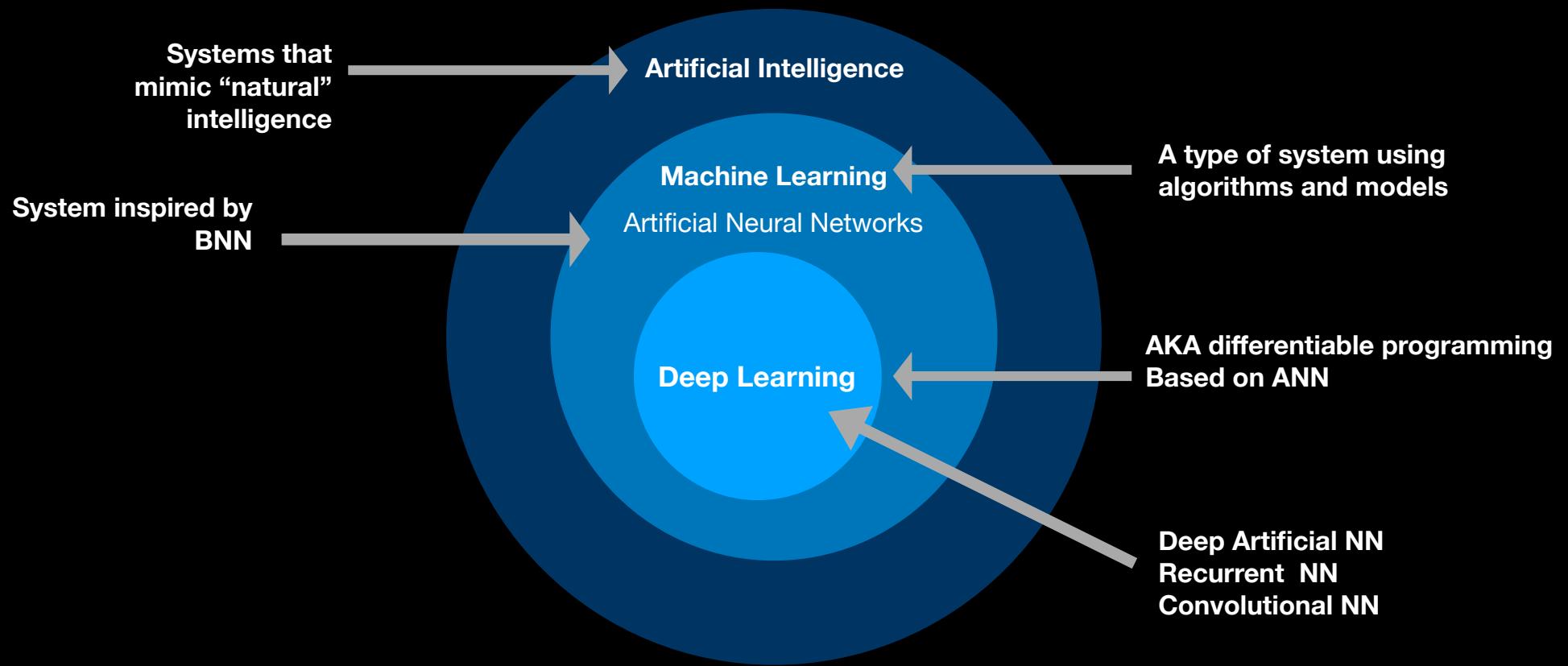
What did I learn?

What did I learn?

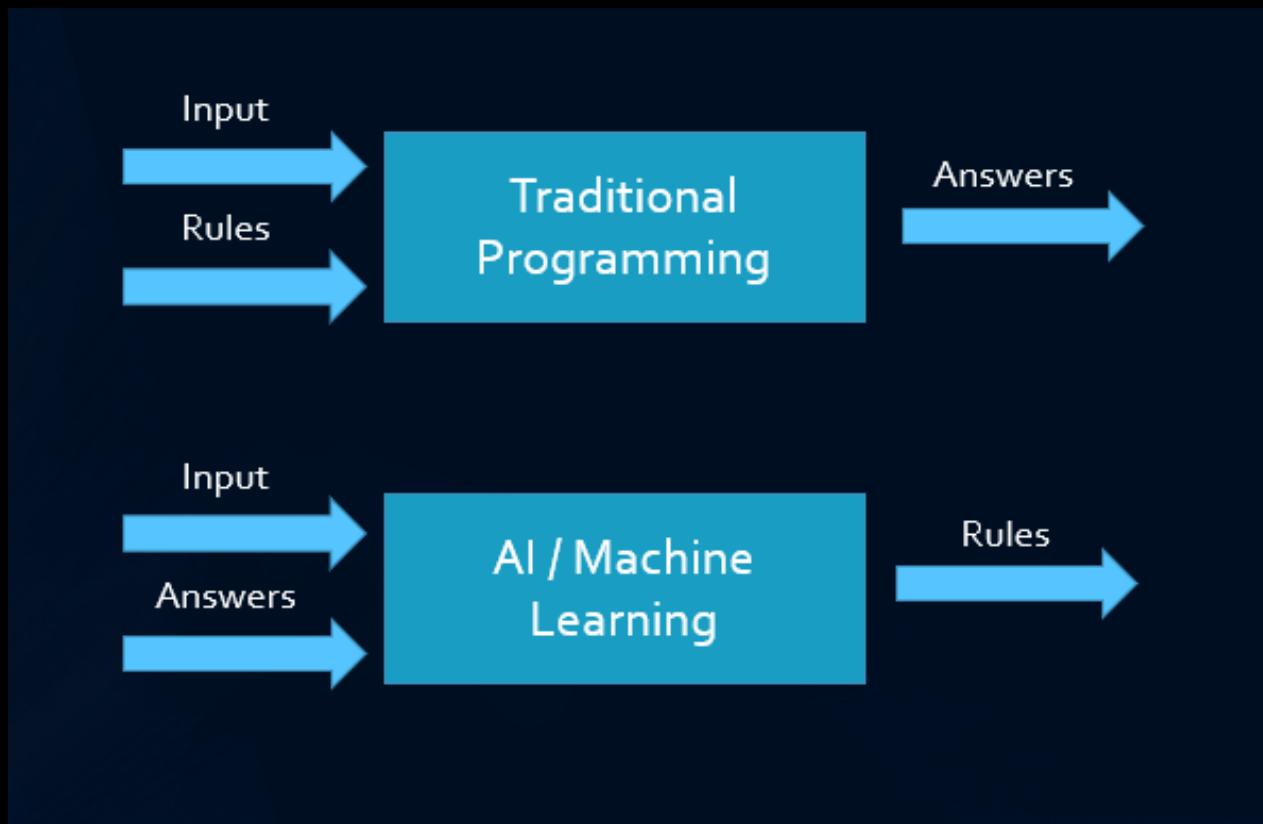
- **Coding** is the easy part, **theory** is the hard part
- Don't be too eager to code, it's less important
- It's not as daunting as you might think
- Be patient, there is **a lot** to learn

Let's move on - ML Building Blocks

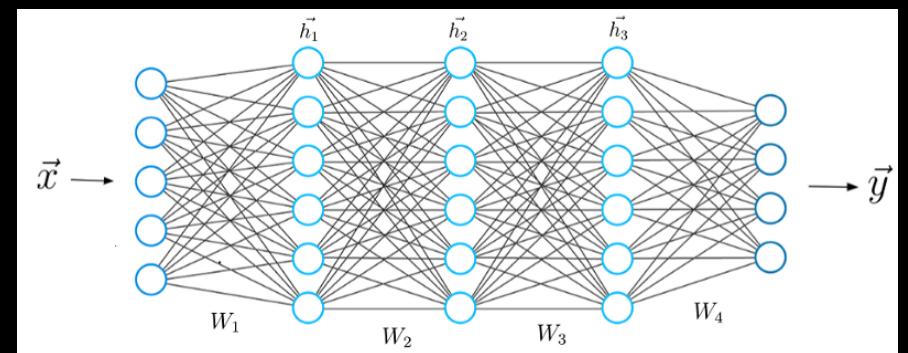
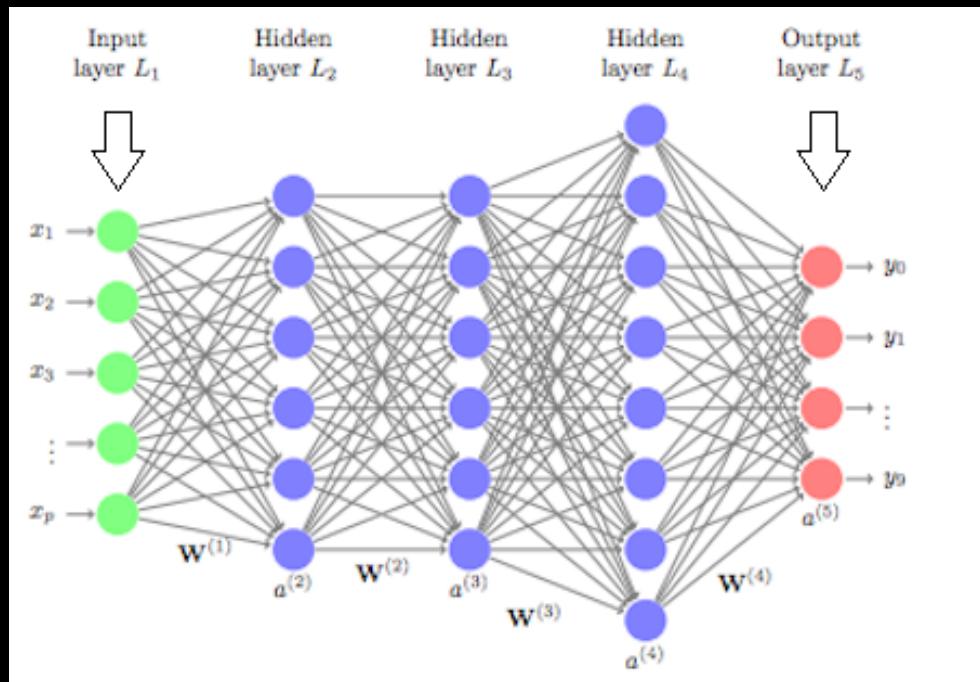
Terminology



How does it compare to traditional programming?



Deep Artificial Neural Network - DNN

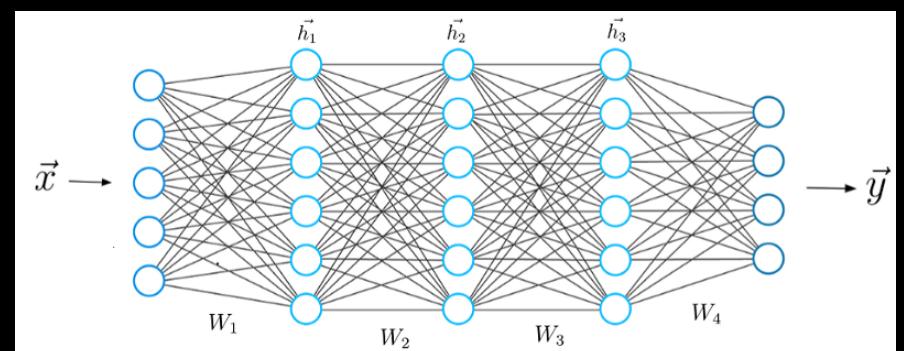
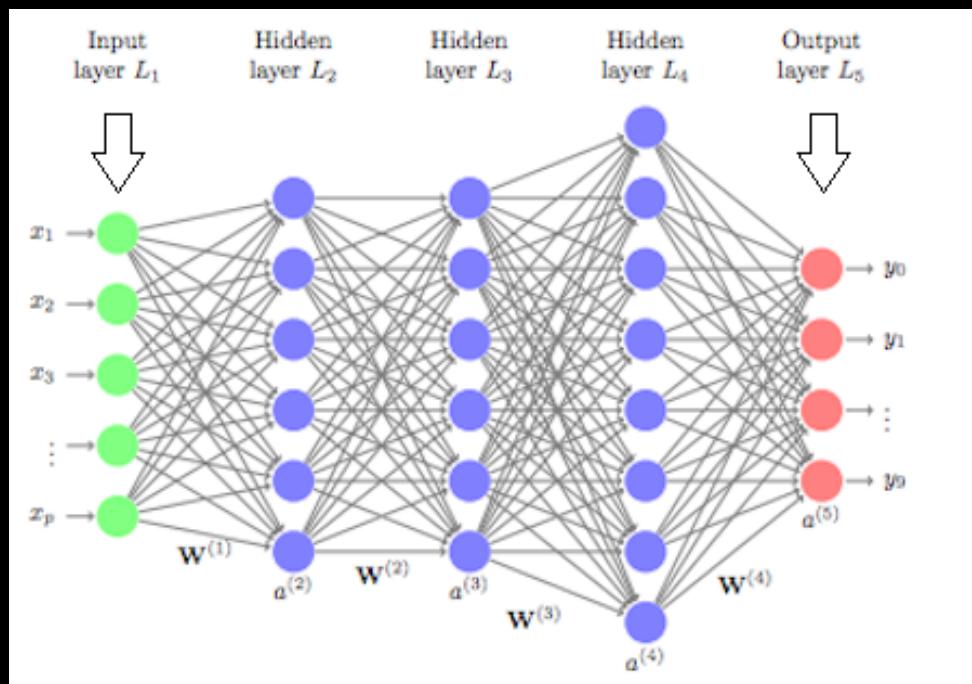


A ML model is just a function

Make a deep neural network

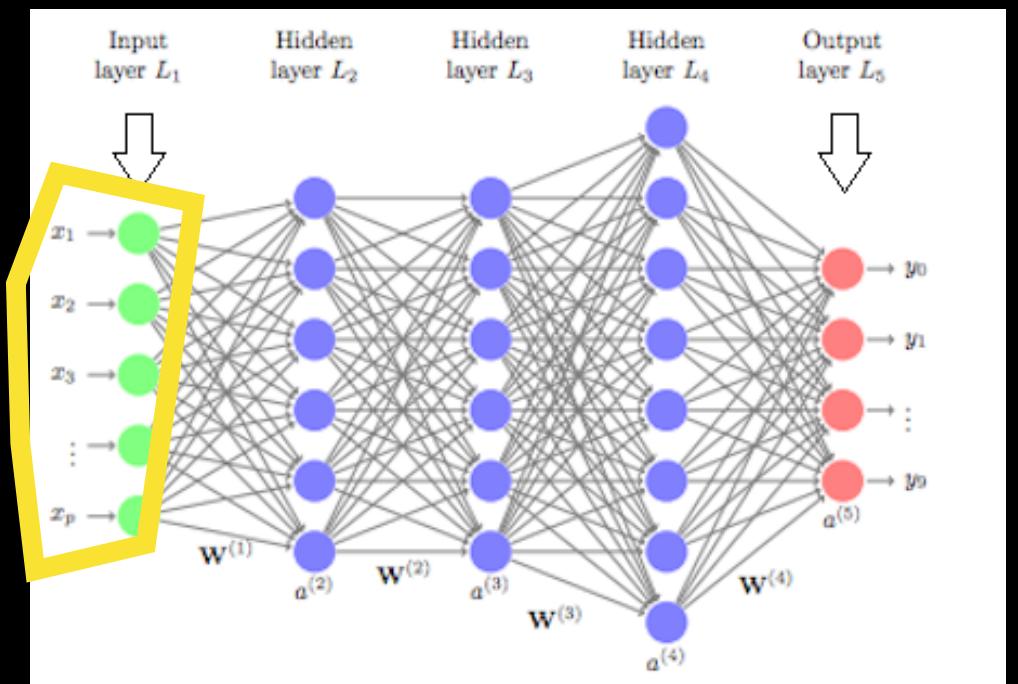
- Prepare the data
- Loop for a number of iterations
 - Forward step to make a prediction
 - Compute the loss
 - Backward step to update parameters
- Use the model to make predictions

Forward step: Forward Propagation



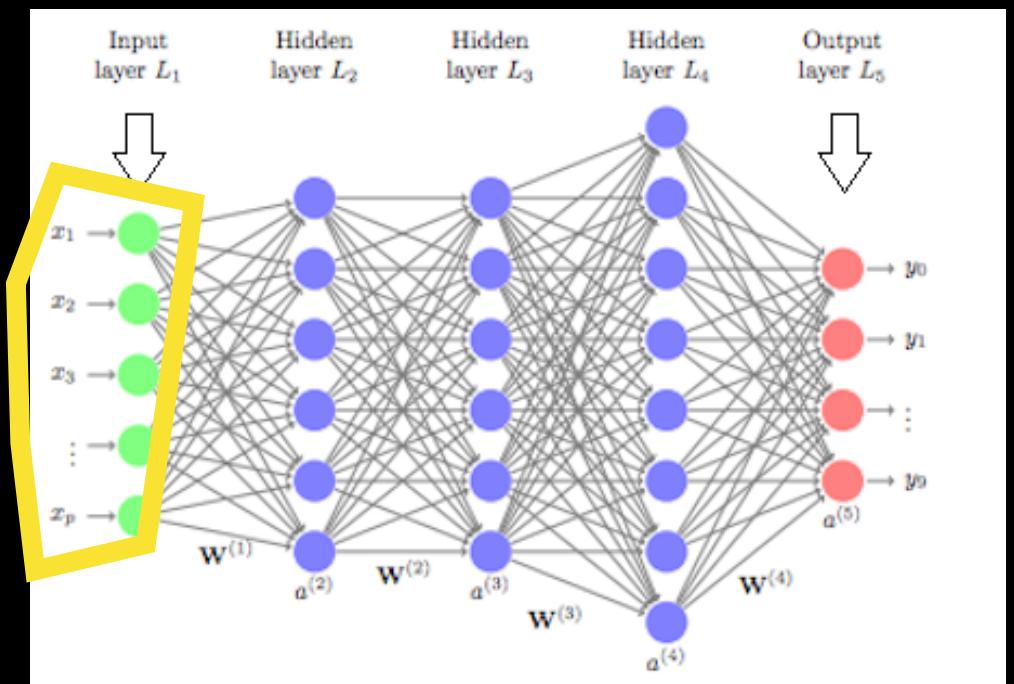
What is X?

- 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 (n_x, m)



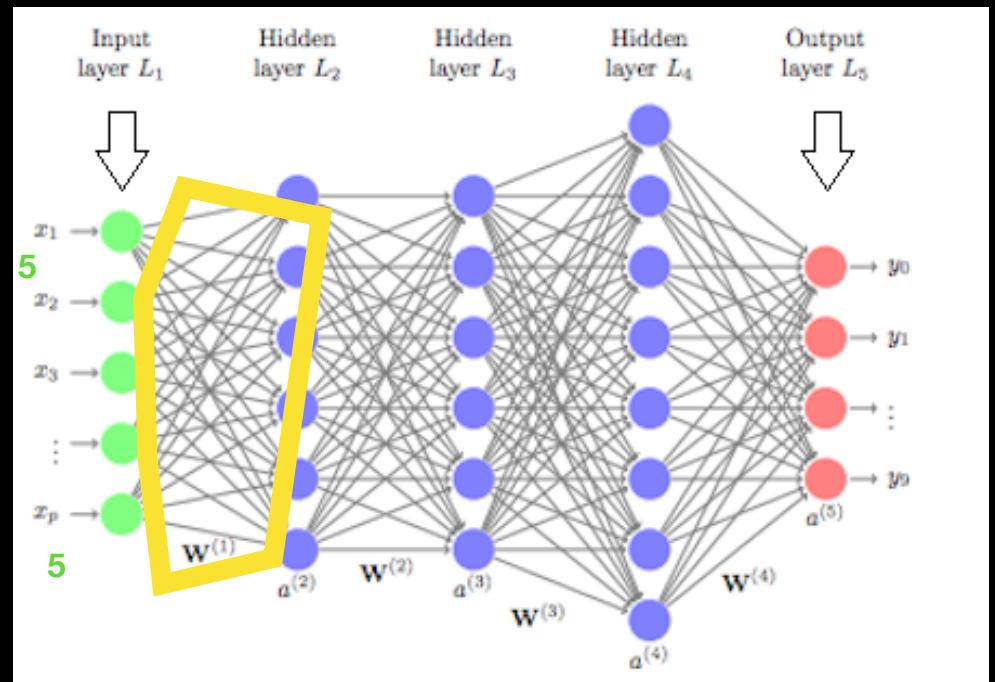
Make it real

- We write this as (n_x, 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. W_1 has a shape of $(7, 5)$
- We compute the matrix "dot product" (**dot product** is the sum of the **products** of the corresponding entries)
- $= X \cdot W_1$



Single instruction, multiple data (SIMD)

```
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]])
```

$$\begin{array}{c} \vec{b}_1 \quad \vec{b}_2 \\ \downarrow \quad \downarrow \\ \vec{a}_1 \rightarrow \quad \begin{bmatrix} 1 & 7 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 3 & 3 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} \vec{a}_1 \cdot \vec{b}_1 & \vec{a}_1 \cdot \vec{b}_2 \\ \vec{a}_2 \cdot \vec{b}_1 & \vec{a}_2 \cdot \vec{b}_2 \end{bmatrix} \\ \vec{a}_2 \rightarrow \\ A \qquad \qquad \qquad B \qquad \qquad \qquad C \end{array}$$

Why do we use vectors and not just array?

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

The screenshot shows the Apple Developer website with the URL [https://developer.apple.com/documentation/accelerate](#). The page title is "Accelerate". The main content area describes the Accelerate framework as a "Framework" for performing large-scale mathematical computations and image calculations, optimized for high performance and low-energy consumption. It features a logo with three yellow stars. Below the description, there's an "Overview" section detailing the framework's purpose and the libraries it contains: vImage, vDSP, vForce, Sparse Solvers, BLAS, LAPACK, and BNNS. The "Related Libraries" section lists simd and Compression. On the right side, there are sections for "SDKs" (iOS 4.0+, macOS 10.3+, Mac Catalyst 13.0+, tvOS 9.0+, watchOS 4.0+), and links for "On This Page", "Overview", and "Topics".

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

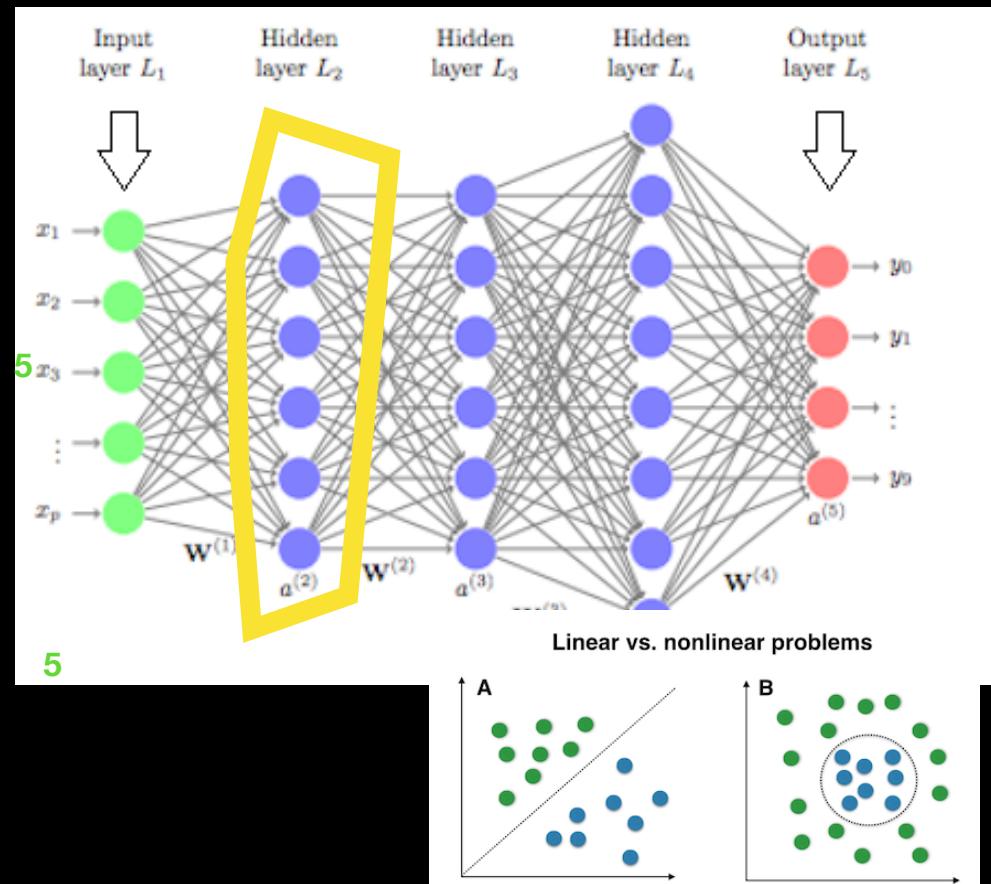
import Accelerate.vecLib

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

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

Next step - activation

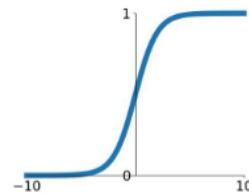
- Why do we need an activation?
- Many linear functions are still just a giant linear function
- Non linear functions allow model to learn non linear relationships



Common Activations

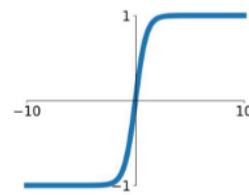
Sigmoid

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



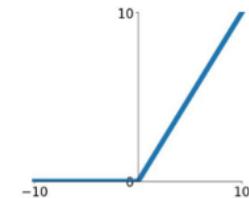
tanh

$$\tanh(x)$$



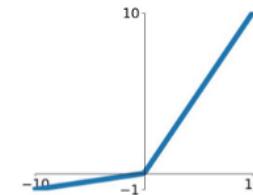
ReLU

$$\max(0, x)$$



Leaky ReLU

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

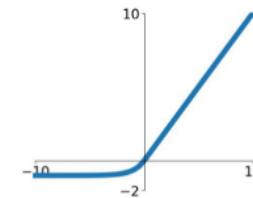


Maxout

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

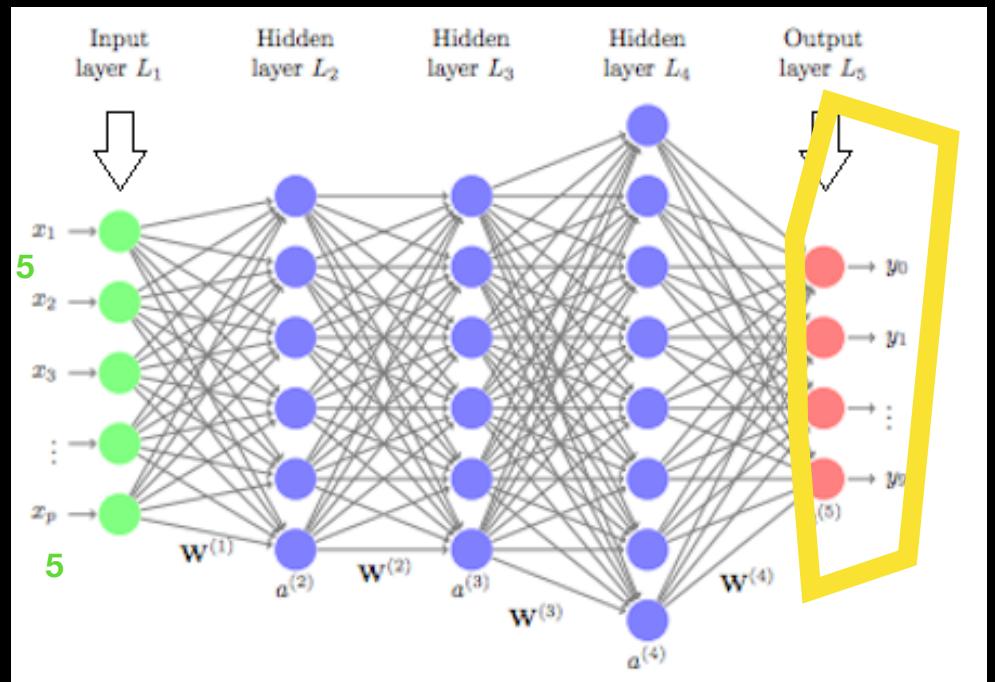
ELU

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



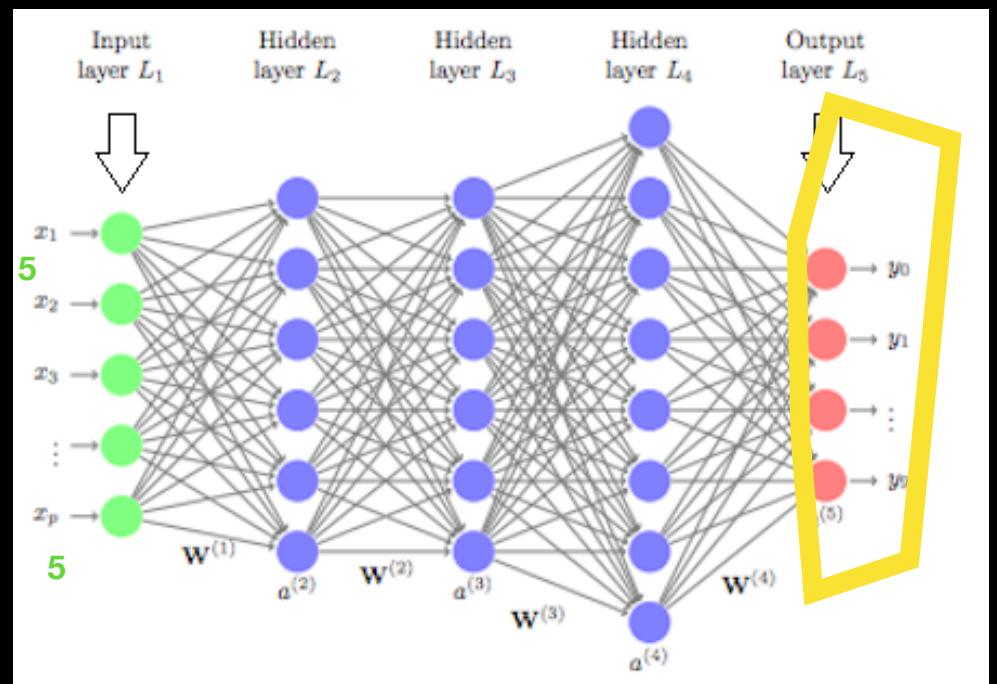
Next step - prediction and loss

- Prediction denoted \hat{y} (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?

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

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” ($\theta_0, \theta_1, \theta_2$)

$h_\theta(x^{(i)})$ The algorithm’s prediction for the i^{th} training example using the parameters θ .

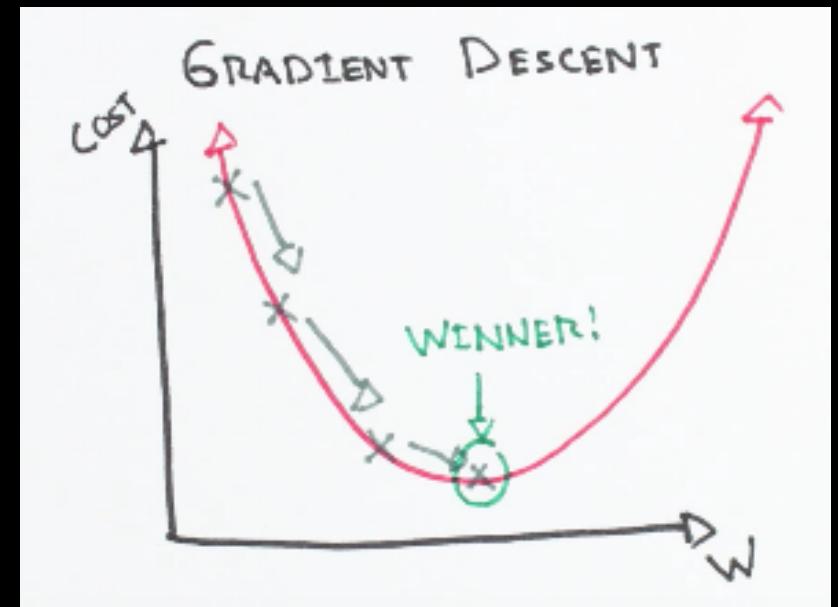
Now we got backwards

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

Its 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

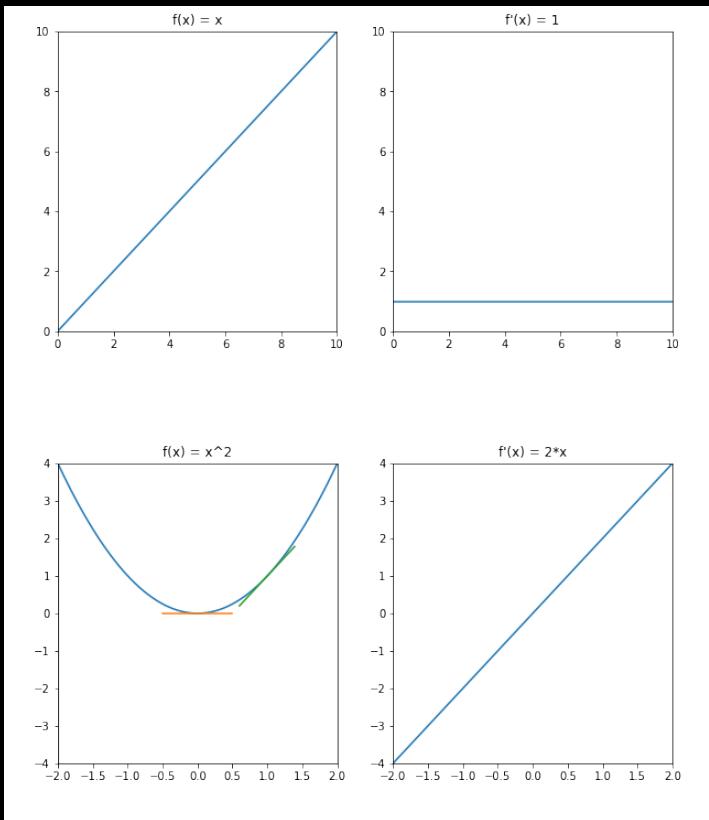
$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$$

\coloneqq update current value in place

a How big of a step to take each iteration

$d/d\theta * J(\theta)$ 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 $f(x + \varepsilon) \sim= f(x) + f'(x) * \varepsilon$

“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

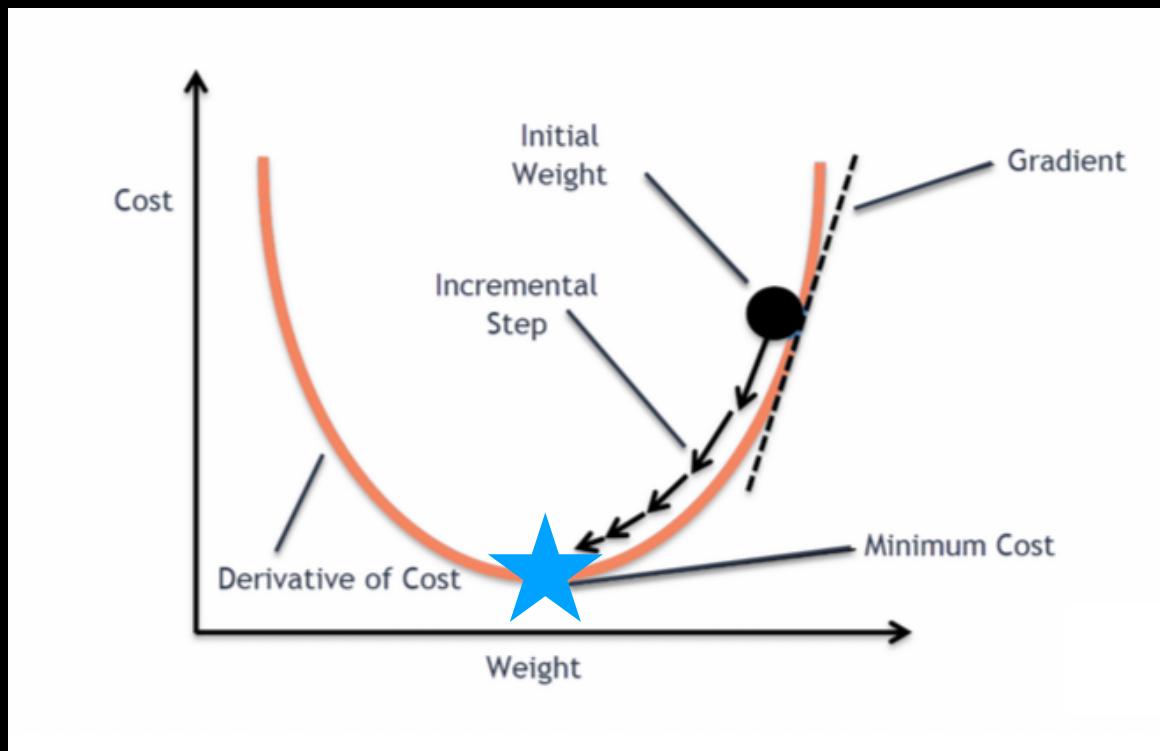
Can be written as df/dx or dx or $d/dx f(x)$.

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

Rules	Function	Derivative
Multiplication by constant	cf	cf'
<u>Power Rule</u>	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