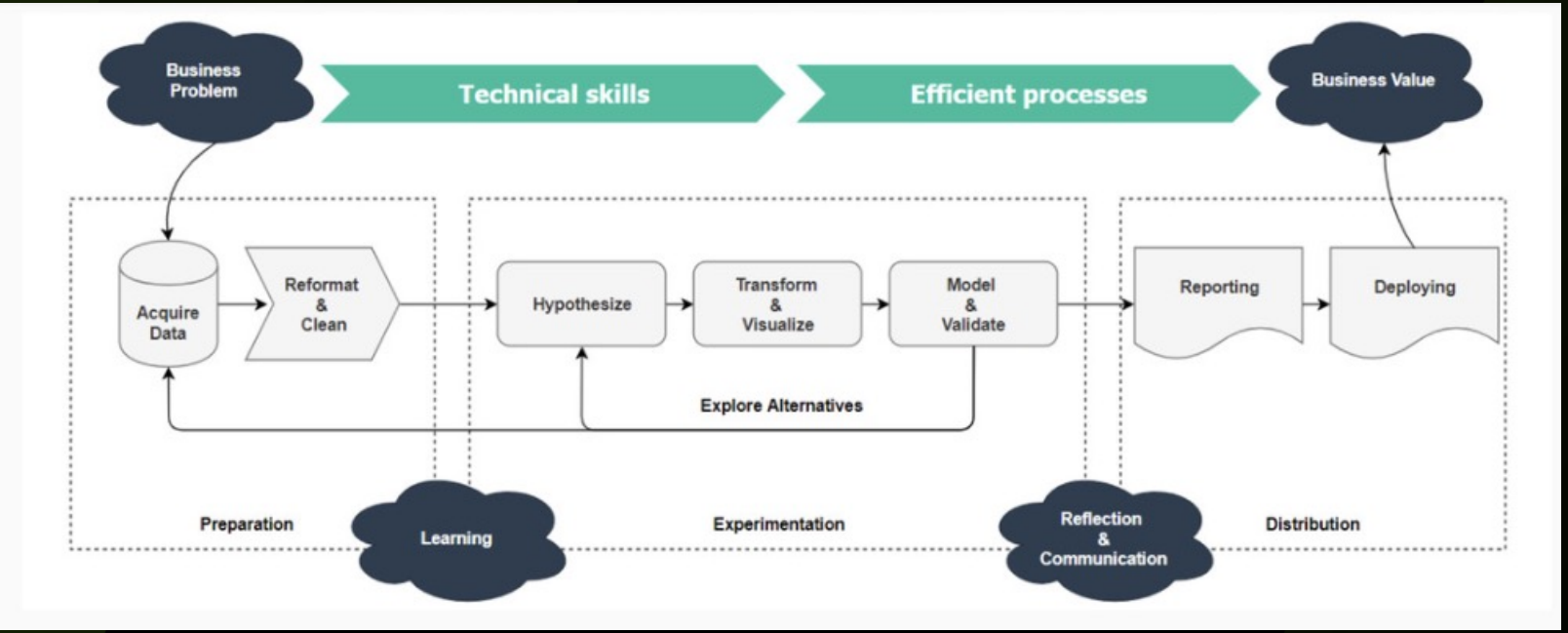
Machine Learning 101

A firehose introduction to all the Data Science things

The Data Science Process



- Understanding the business problem: 20%
- (Data) preparation: 60%
- Experimentation: 5%
- Distribution: 15%

”Data Scientist spend 60% of their time preparing data and the other 40% complaining about all the time they spend preparing data”

WARNING: This is not your mother’s (or father’s, for that matter) kind of programming!

A Few Words About Data

Your Dataset

Used as "test" data while experimenting with models

Used to ***independently*** test your selected and tuned model

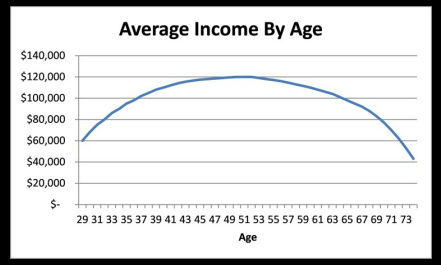
(IOW, use test data to determine you're really not done)

Learn from my suffering:

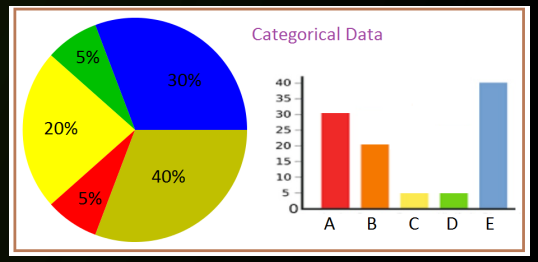
Have ***a lot more*** than "more than enough" data before attempting to use ML

…or…suffer (like me)!

A Little Metadata About Your Data: Numerical vs. Categorical



Can take a value within a range. Said range can be “discrete” (think ints) or “continuous” (think floats).



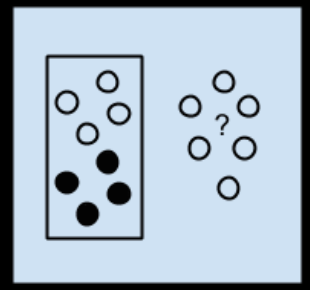
Can take a value within a set. Said set can be ”ordinal” (ordered e.g. t-shirt sizes) or “nominal” (not in any order e.g. t-shirt colors).

One person’s ordinal categorical variable is another person’s discrete numerical variable (figured out during data prep phase)

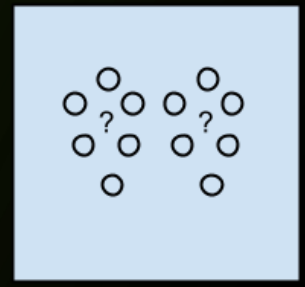
Text data (such as a person’s name or address) will often need to be transformed into one or more numerical and/or categorical values

Data with associated time stamps (“time series data”) can be “fun” and “interesting”

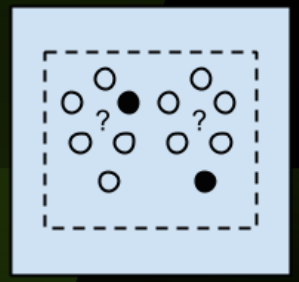
A Little More Metadata About Your Data: Supervised vs. Unsupervised



”Supervised” means your dataset comes with “labels” that split it into groups or mark data as “interesting” or not (labels are typically added by humans)



”Unsupervised” means your dataset does not come with any “metadata” (“clues” outside the data itself) that describe it

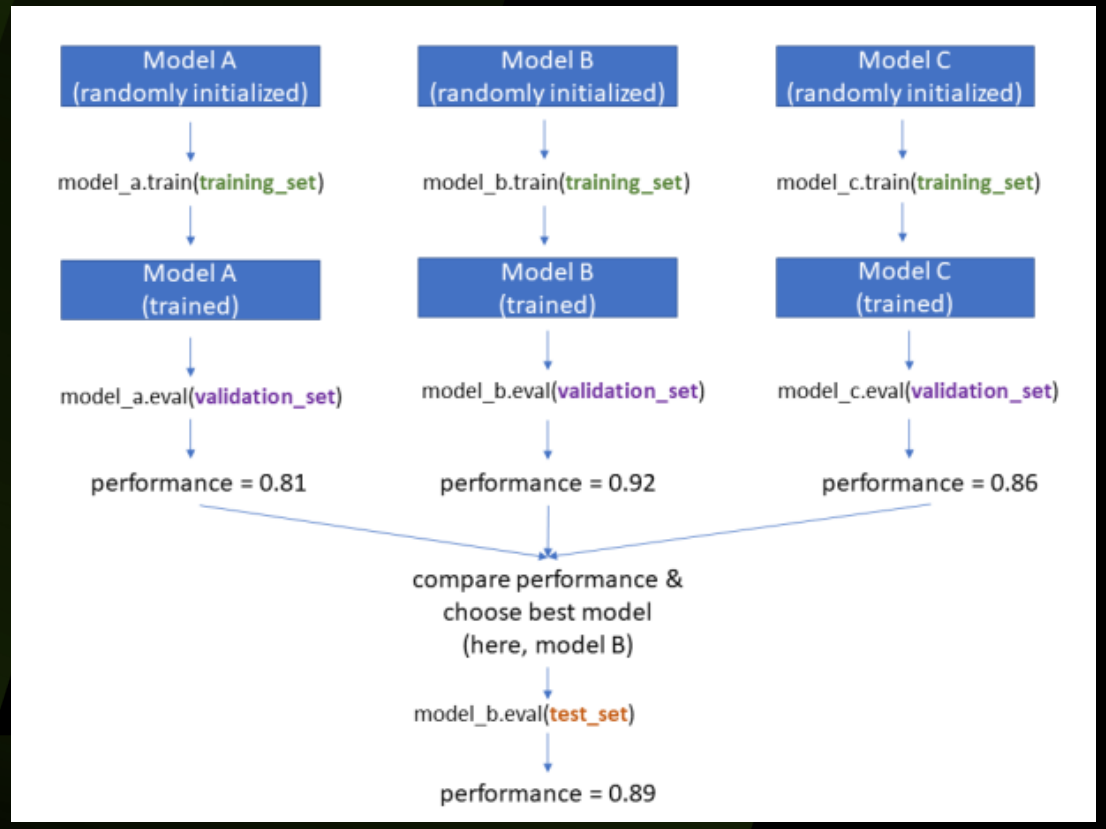


Being Data Science, nothing is simple, so of course there’s a hybrid kind of dataset called ”Semi-Supervised” where some of the data has labels

Pretty rare, I think, but check out: https://openai.com/blog/vpt/ (OpenAI using semi-supervised data to train a model to play Minecraft)

For completeness sake I’ll mention many consider “Reinforcement Learning” as separate from “Unsupervised”

Got data? Great. Now what?



Typically, lots of “feature” finding and model “hyperparameter” tweaking

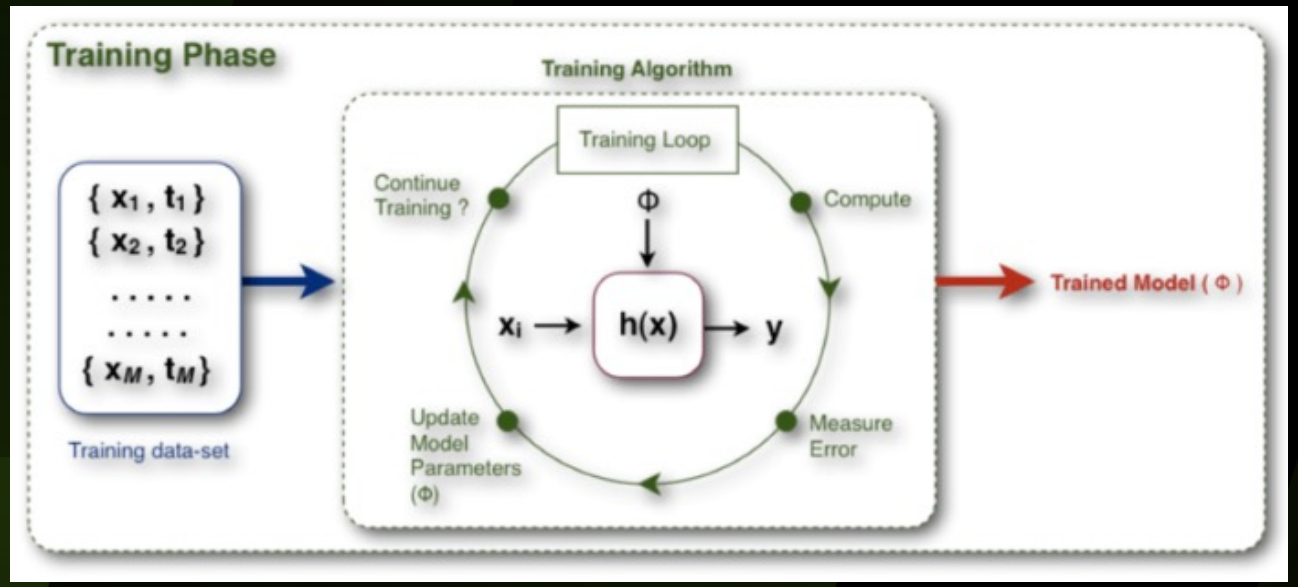
“Features” are the input values to a model. They can be straight from your “raw” data, but experience shows they’ll more likely be *derived* from your raw data.

Note: Data Scientists call the parameters that control a model’s behavior “hyperparameters” because someone needed a doctoral thesis

ML Truism:
“All models are wrong, but some are useful”

Lastly, do yourself a favor: always start with the simplest model possible

Basic (Model) Training



Initialized the model with random params (= random results)

Then for each training sample (x):

1. Compute output (y) using x: h(x)
2. Measure difference of t vs y ("Cost Function")
3. Exit loop if difference is "small enough"
4. Use the difference to update model params**

** Note: NOT hyperparams!

Test using test data (success=grab a beer)

(failure=grab two beers and try again)

Model Meta

Three basic types (and one ugly one)

* Curve Fitting (“Regression”) e.g. $f(x_1,x_2,\ldots)$ = continuous value(s)
* Grouping (“Classification”) e.g. $f(x_1,x_2,\ldots)$ = discreet type
* Analytics (“Description”) e.g. $f(x_1,x_2,\ldots)$ = meta data about $x_1,x_2,\ldots$

+ Helpers (enhancements to the above)

Pretty much any model of any type can be used for…

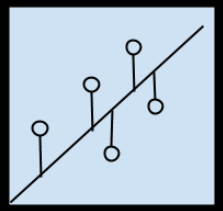
* Prediction
* Anomaly Detection
* Encoding
* Data Generation (text, images, sound, …)
* Data Mining (summation, correlation, …)
* Feature Selection

A Merciless Survey of ML Models

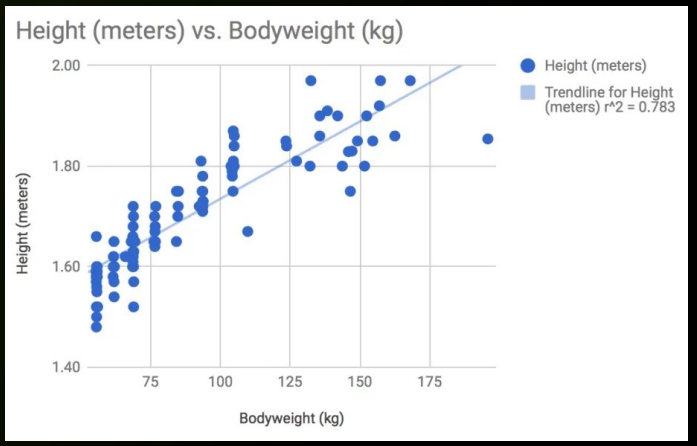
Buckle up and take a good hold of your laptop, we'll be pulling some serious mental G's

Mostly based on an unholy combination information from "A Tour of Machine Learning Algorithms" by Jason Brownlee and "A Tour of Machine Learning Algorithms" by Claire D. Costa

Regression



* Ordinary Least Squares Regression (OLSR)
* Linear Regression
* Logistic Regression
* Stepwise Regression
* Multivariate Adaptive Regression Splines (MARS)
* Locally Estimated Scatterplot Smoothing (LOESS)



Simple Linear Regression

Regularization

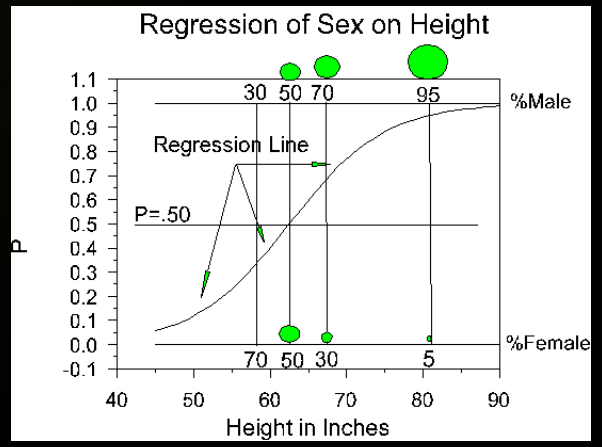
A popular *addition* to many algorithms (especially regression) that results in models with better generalization by biasing them against complexity during training

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

L1 or “Lasso” regression

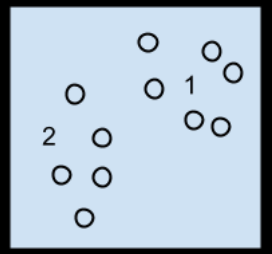
$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

L2 or “Ridge” regression (note the **2)



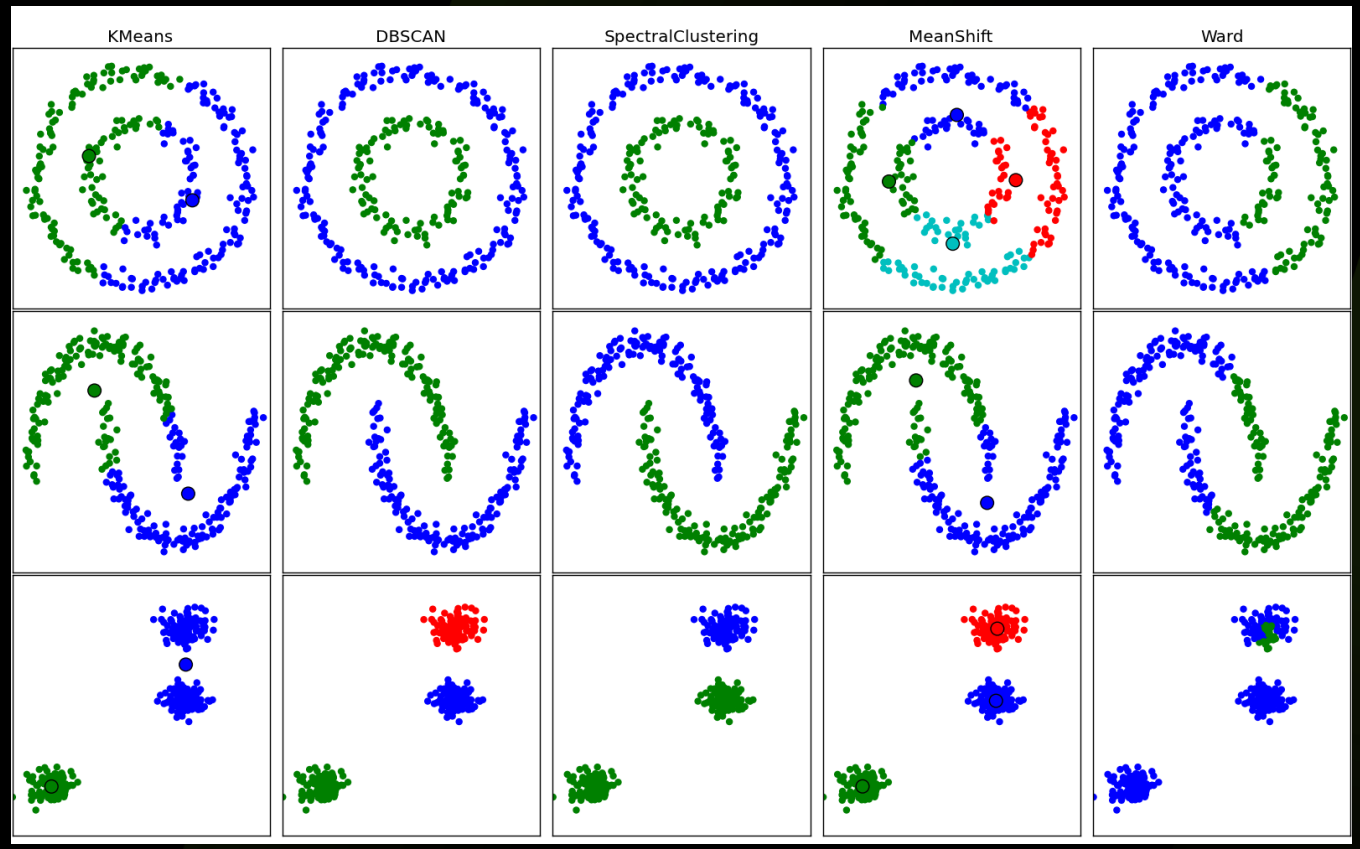
Simple Logistic Regression

Clustering

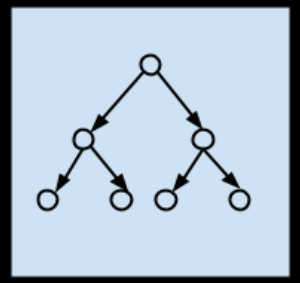


* KMeans
* DBScan
* SpectralClustering
* MeanShift
* Ward
* KMedians
* Expectation Maximization
* Hierarchical Clustering

‘nuff Said?

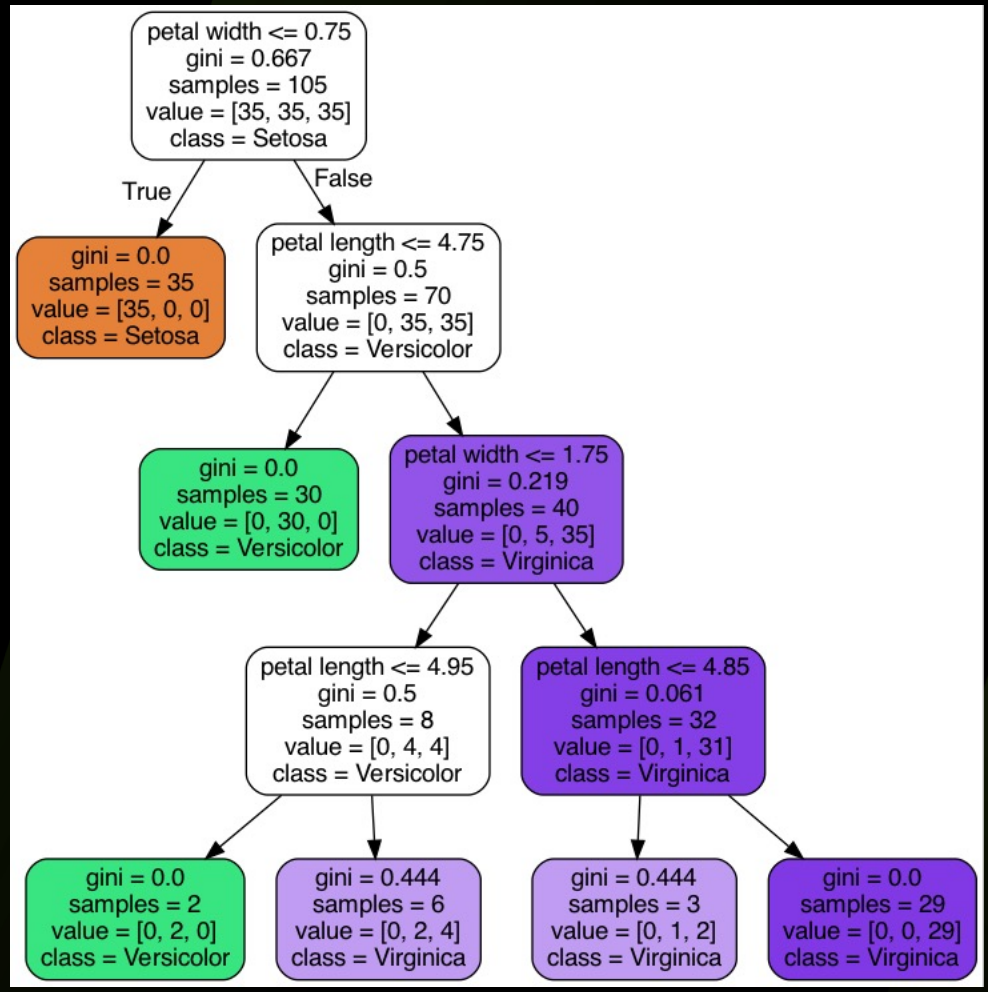


Decision Trees



* Chi-squared Automatic Interaction Detection (CHAID)
* Classification and Regression Tree (CART)
* Iterative Dichotomiser 3 (ID3)
* C4.5 and C5.0
* M5
* Conditional Decision Trees
* Random Forest (tree ensemble)
* Gradient Boost (another ensemble w/o the math pun)

FYI: Gini (impurity) is the probability of incorrectly classifying a randomly chosen element in the dataset and giving it a random class



DT trained to predict Iris species from petal width and length

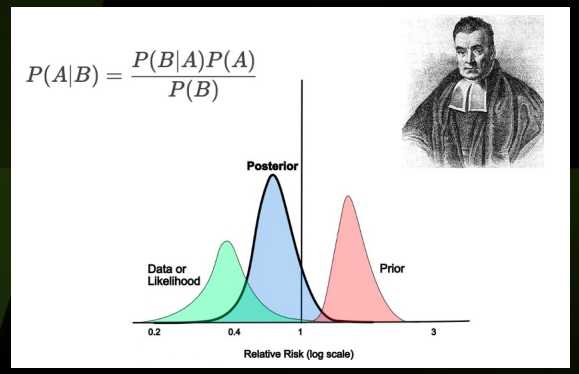
Three More

that I haven't used much but want you to be aware of so I'm glossing over them to get to the "good stuff"

Bayesian

* Naive Bayes
* Gaussian Naive Bayes
* Multinomial Naive Bayes
* Averaged One-Dependence Estimators
* Bayesian Belief Network (BBN)
* Bayesian Network (BN)

Classic statistics-based algos

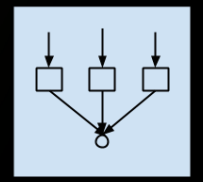


Instance-Based

* k-Nearest Neighbor (kNN)
* Learning Vector Quantization (LVQ)
* Self-Organizing Map (SOM)
* Locally Weighted Learning (LWL)

Extracts info based on "neighborhoods"

Ensembles

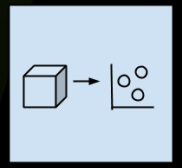


* Boosting
* Bootstrapped Aggregation (Bagging)
* AdaBoost
* Weighted Average (Blending)
* Stacked Generalization (Stacking)
* Gradient Boosting Machines (GBM)

Combining independently trained models to achieve synergy

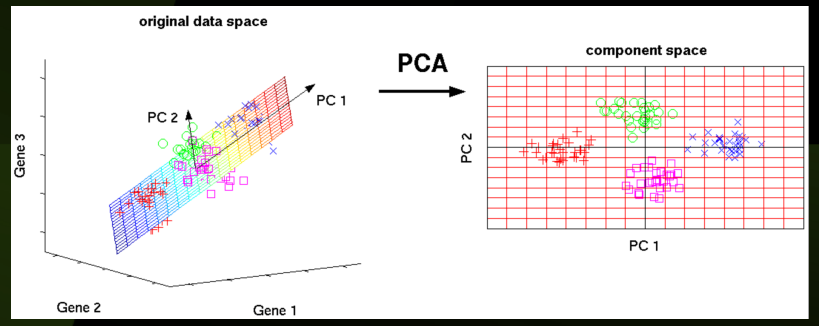
And Another Three Used Mostly For Data Analysis

Dimension Reduction

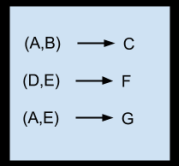


* Principal Component Analysis (PCA)
* Principal Component Regression (PCR)
* Partial Least Squares Regression (PLSR)
* Sammon Mapping
* Multidimensional Scaling (MDS)
* Projection Pursuit
* Linear Discriminant Analysis (LDA)
  - (Linear, Mixture, Quadratic, Flexible)

Good for simplifying and visualizing multidimensional data



Association Rule Learning



* Apriori
* Eclat

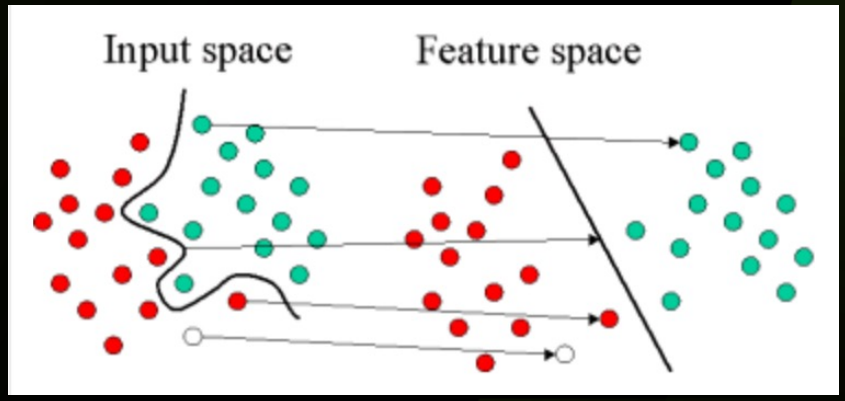
"Often used to discover or mine for interesting patterns and relationships"

- Not Me

Support Vector Machines

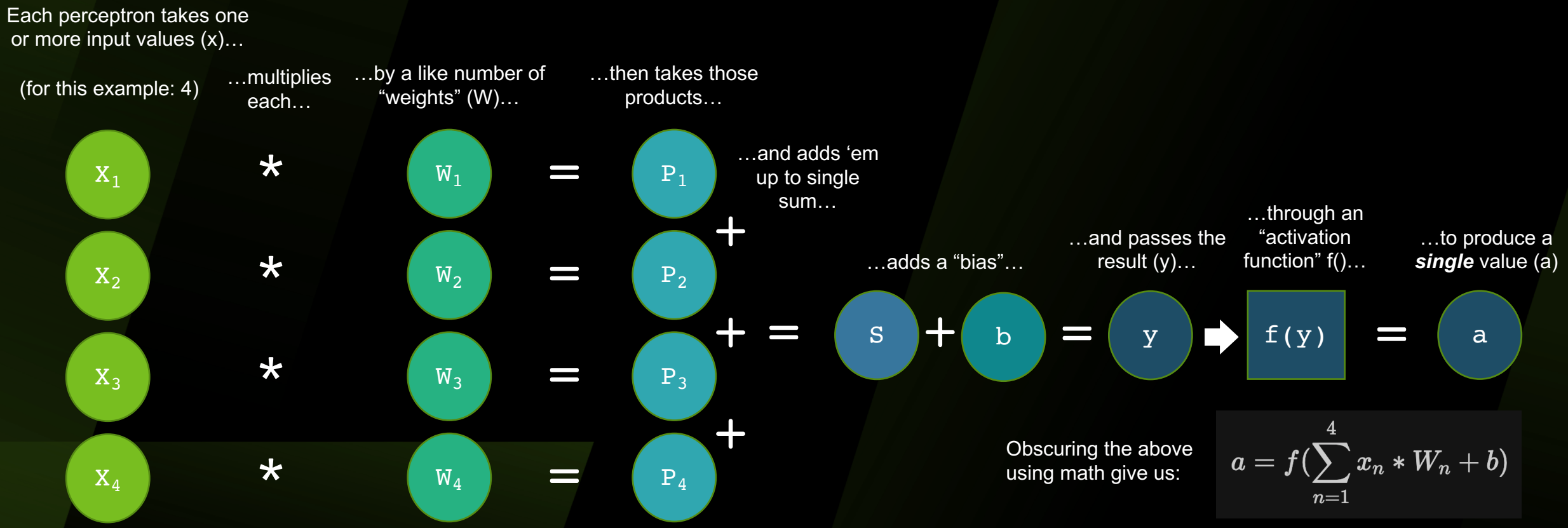
* Linear
* Non-Linear

Leverages a kernel function to transform your data into a higher dimensional space so it can be separated using hyperplanes – which you've got to admit is pretty cool!



Artificial Neural Networks (FINALLY!)

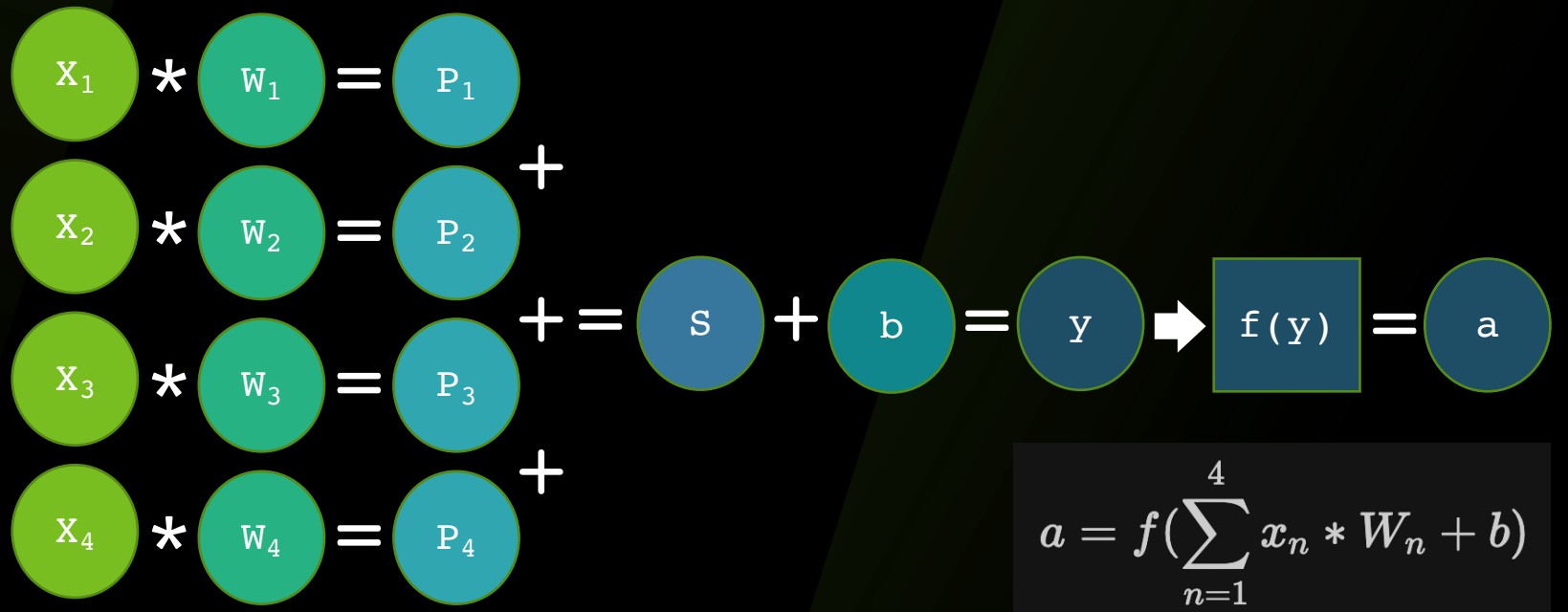
All ANN's are made up of "perceptrons" ← analogous to a neuron in your brain



Obscuring the above using math give us:

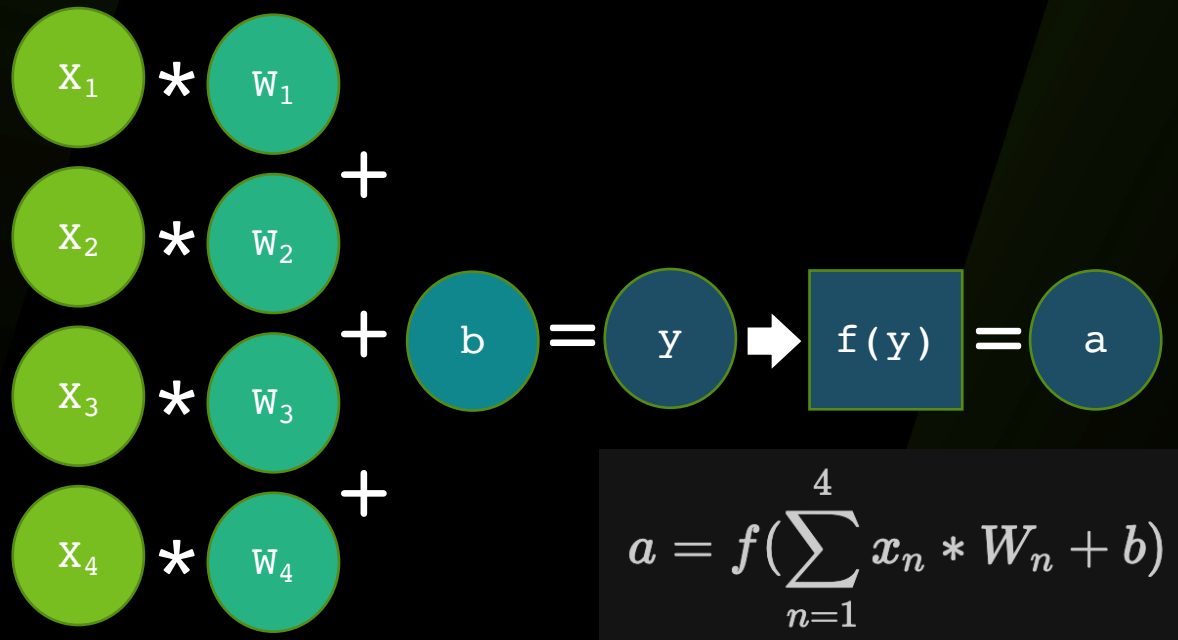
$$a = f(\sum_{n=1}^{4} x_n * W_n + b)$$

Take the Perceptron Diagram…



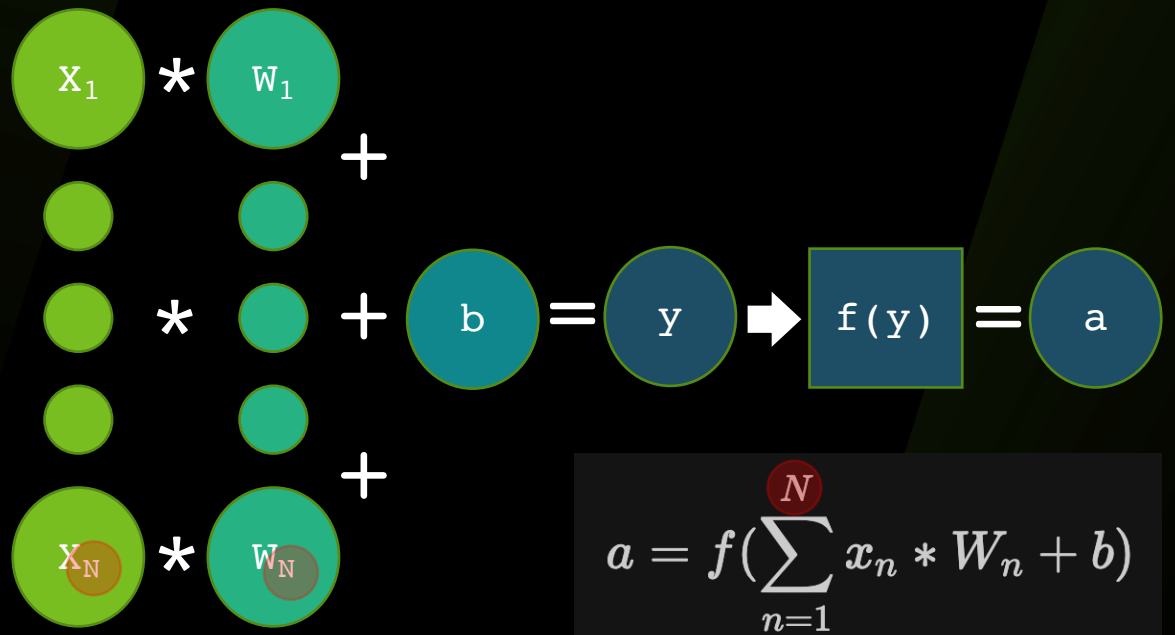
$$a = f(\sum_{n=1}^{4} x_n * W_n + b)$$

...and Simplify It



$$a = f(\sum_{n=1}^{4} x_n * W_n + b)$$

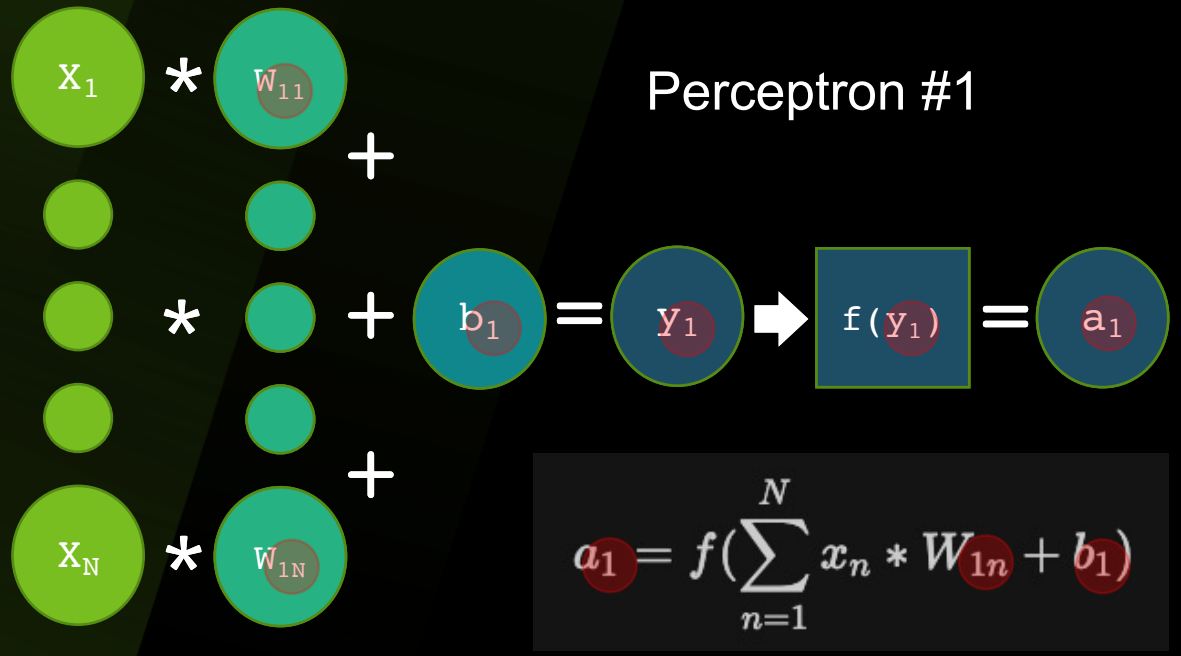
Expand it to Handle “N” Inputs



$$a = f(\sum_{n=1}^{N} x_n * W_n + b)$$

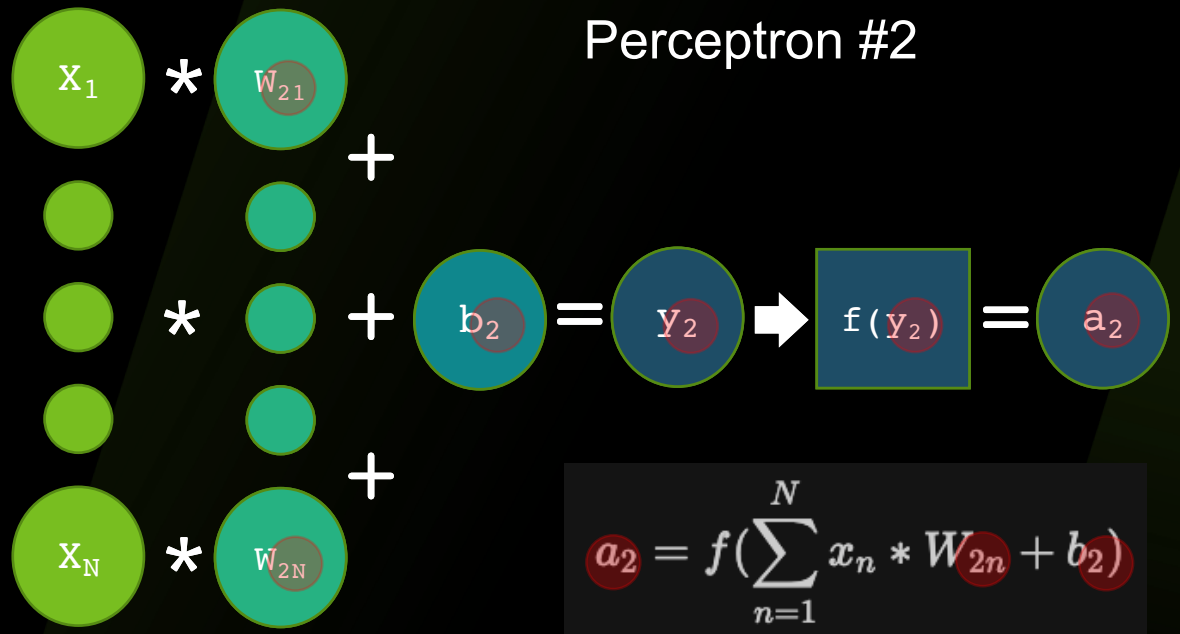
But note! We still only have one output. Sad!

Want two outputs? Use two perceptrons!



Perceptron #1

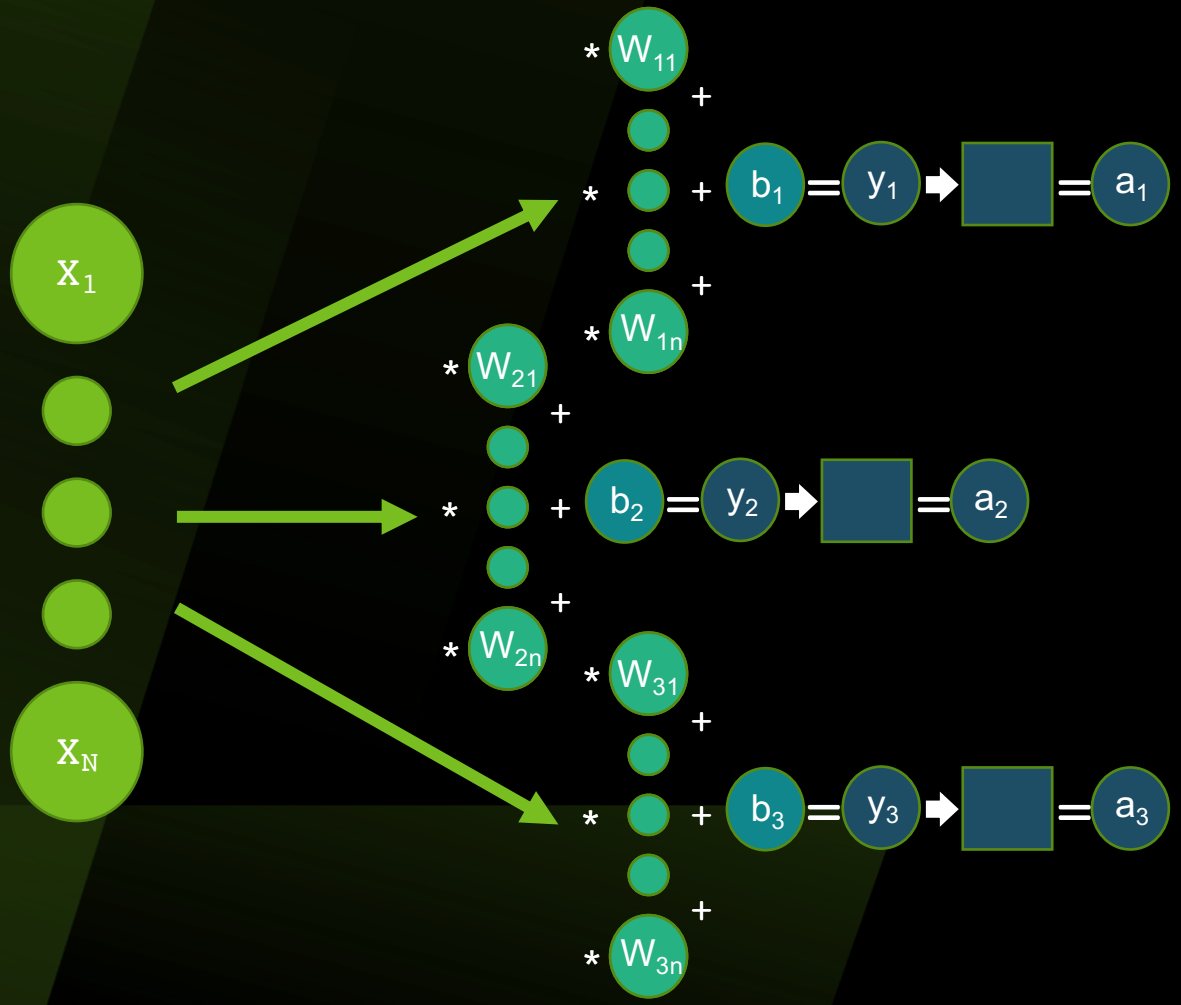
$$a_1 = f(\sum_{n=1}^{N} x_n * W_{1n} + b_1)$$



Perceptron #2

$$a_2 = f(\sum_{n=1}^{N} x_n * W_{2n} + b_2)$$

Want three outputs? You guessed it!

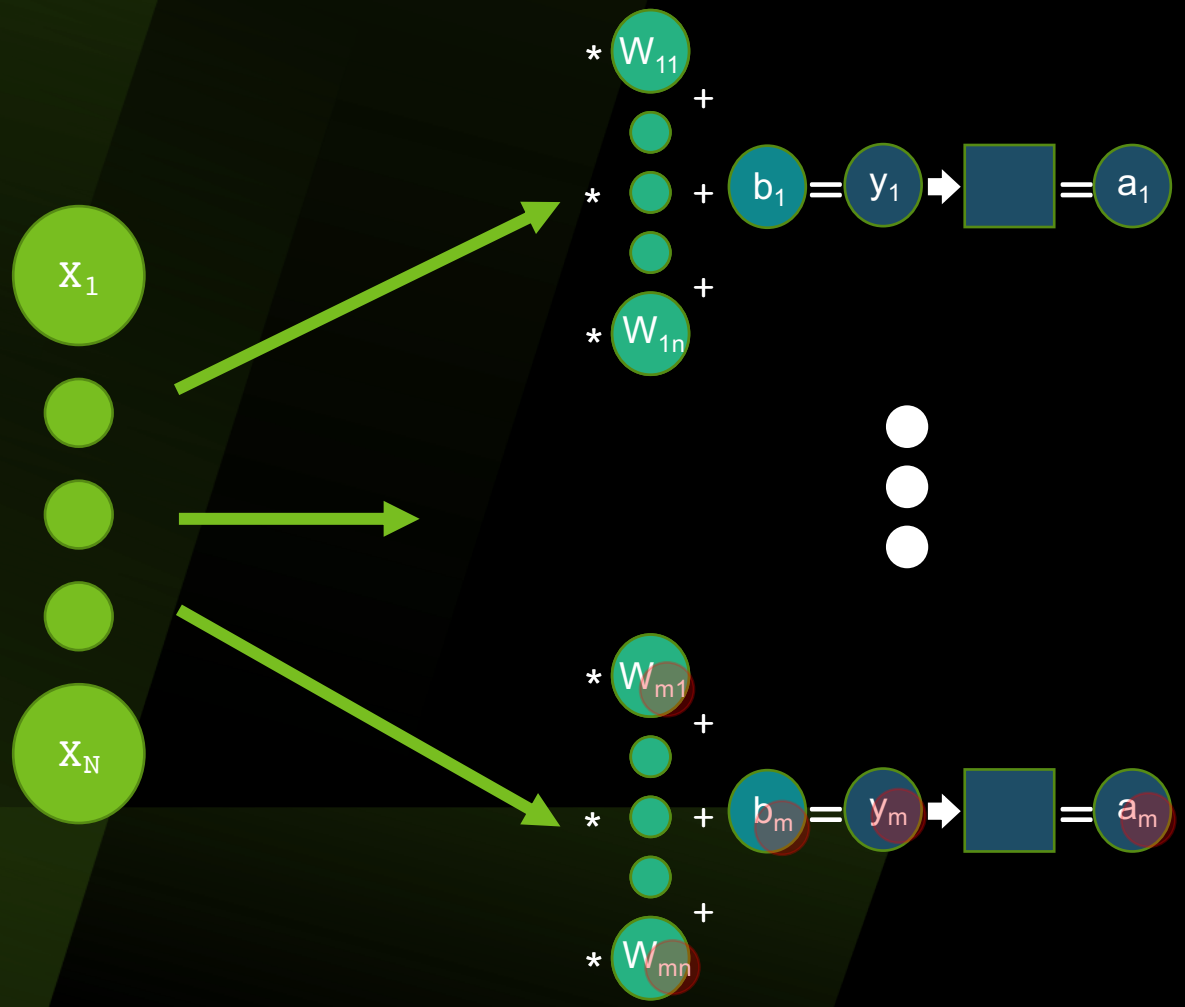


$$a_1 = f(\sum_{n=1}^{N} x_n * W_{1n} + b_1)$$

$$a_2 = f(\sum_{n=1}^{N} x_n * W_{2n} + b_2)$$

$$a_3 = f(\sum_{n=1}^{N} x_n * W_{3n} + b_3)$$

Want “M” outputs? Voila!

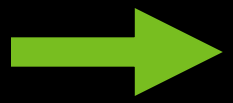


$$a_1 = f(\sum_{n=1}^{N} x_n * W_{1n} + b_1)$$

$$a_m = f(\sum_{n=1}^{N} x_n * W_{mn} + b_m)$$

We all know loops suck. Answer: matrix math!

$$a_1 = f(\sum_{n=1}^{N} x_n * W_{1n} + b_1)$$



$$\begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} = f\left(\begin{bmatrix} W_{1,1} & W_{1,2} & \cdots & W_{1,n} \\ W_{2,1} & W_{2,2} & \cdots & W_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ W_{m,1} & W_{m,2} & \cdots & W_{m,n} \end{bmatrix} \times \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \right)$$

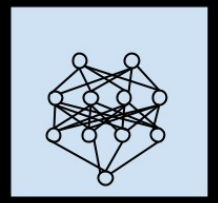
$$a_m = f(\sum_{n=1}^{N} x_n * W_{mn} + b_m)$$

This a set of perceptrons that takes N inputs and produces M outputs.

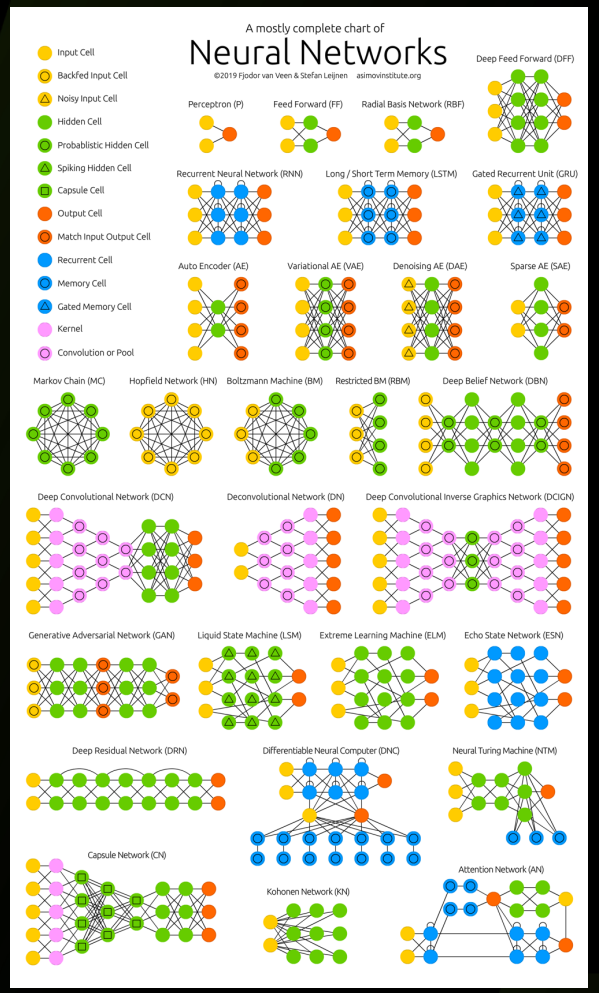
Such an arrangement of perceptrons is called a “layer”

Wanna go “Deep”?

Deep Learning



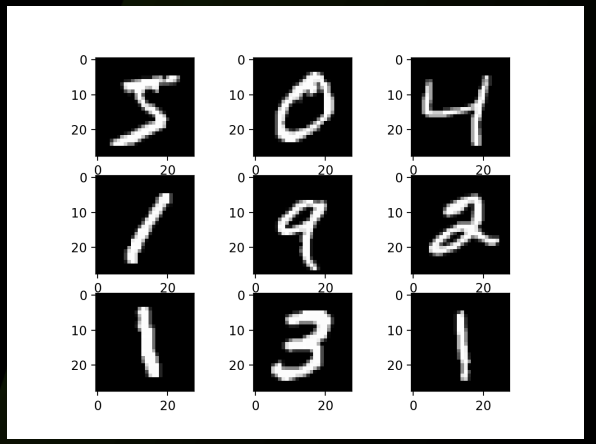
* Convolutional Neural Network (CNN)
* Recurrent Neural Networks (RNNs)
* Long Short-Term Memory Networks (LSTMs)
* Stacked Auto-Encoders
* Deep Boltzmann Machine (DBM)
* Deep Belief Networks (DBN)



Let's be shallow…

…and build a ANN to recognize these →

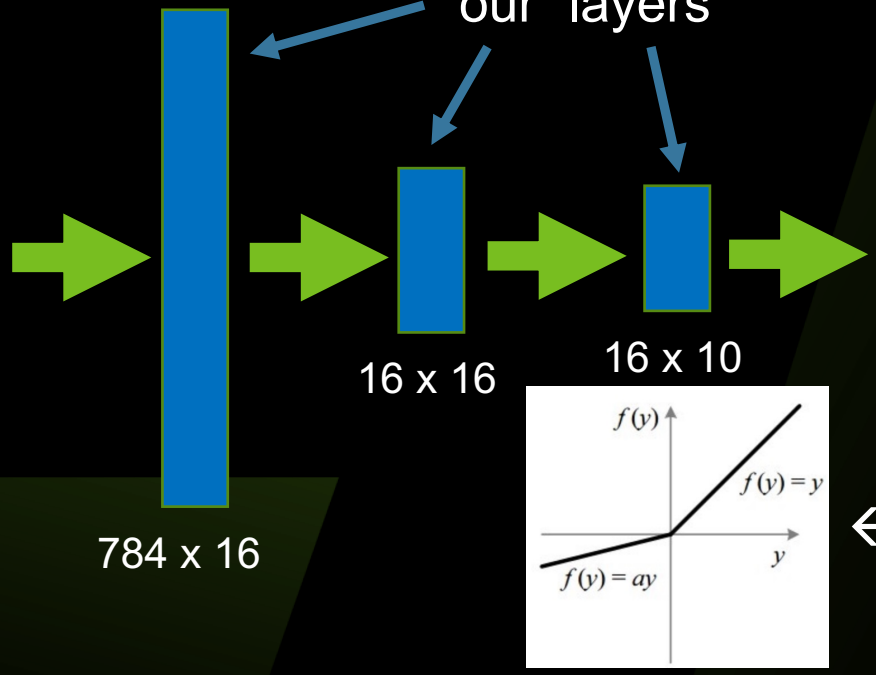
Our data…



70,000 labeled monochrome 28x28 images of handwritten digits [0..9]

Split: 60k for training, 10k for testing

28x28 → 784x1



10 outputs

Select the largest as the answer (e.g. if the largest value is at position six then the ANN is "recognizing" the input as a "6")

← Our "activation function" (LeakyReLU)

For a given training sample…

How our ANN will learn

Use mathemagic to ”spread” the error across the weights and biases in this layer and update them so it’ll do better next time

error = mean((output – expected)^2)

And different mathemagic to figure out the error from the previous layer

28x28 → 784x1

784 x 16

16 x 16

16 x 10

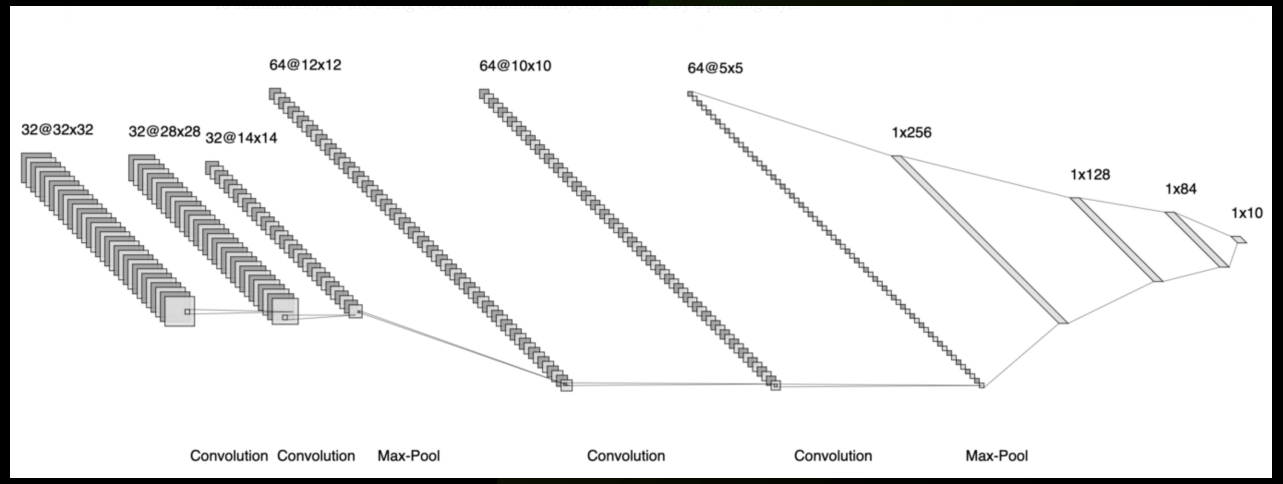
repeat

output		expected	
0.4		0.0	
0.2		0.0	
0.3		0.0	
0.3		0.0	
0.6		0.0	
0.1		1.0	“It’s a 5!”
0.8	“It’s a 6!”	0.0	
0.2		0.0	
0.7		0.0	
0.1		0.0	

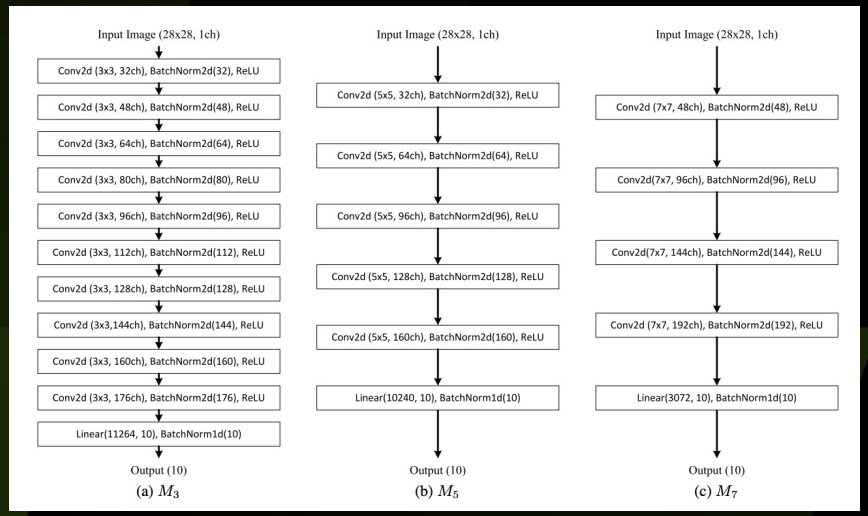
This is called “backpropagation”

Code!

Most comprehensible: 99.41%
(https://towardsdatascience.com/going-beyond-99-mnist-handwritten-digits-recognition-cfff96337392)



Best accuracy: 99.91%
(https://paperswithcode.com/sota/image-classification-on-mnist)



References

https://github.com/heck/ml101

https://www.linkedin.com/pulse/data-science-workflow-matt-dancho/

https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

https://www.asimovinstitute.org/neural-network-zoo/

https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/

https://openai.com/blog/vpt/

https://bdtechtalks.com/2022/04/11/openai-dall-e-2/

https://towardsdatascience.com/a-tour-of-machine-learning-algorithms-466b8bf75c0a

https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/