INFO 2950: Intro to Data Science

Lecture 25
2023-11-27

Agenda

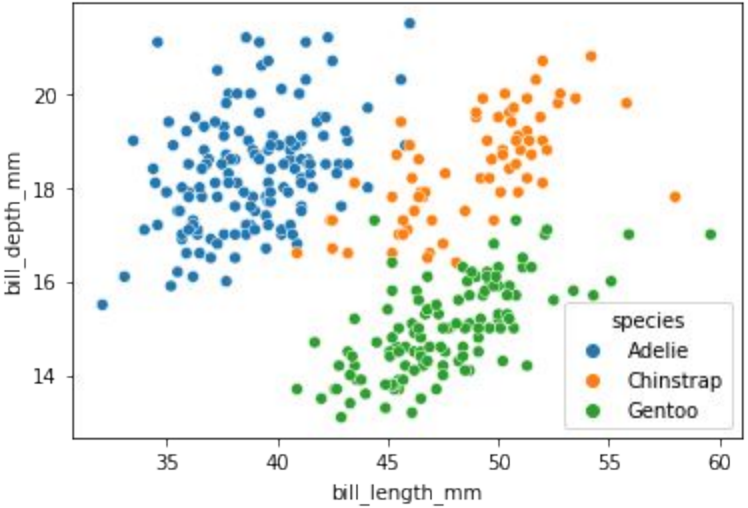
1. Classification
   a. Linear separability

2. Continuous output prediction
   a. Linear Regression
   b. Perceptrons
   c. Neural Networks

Classifying

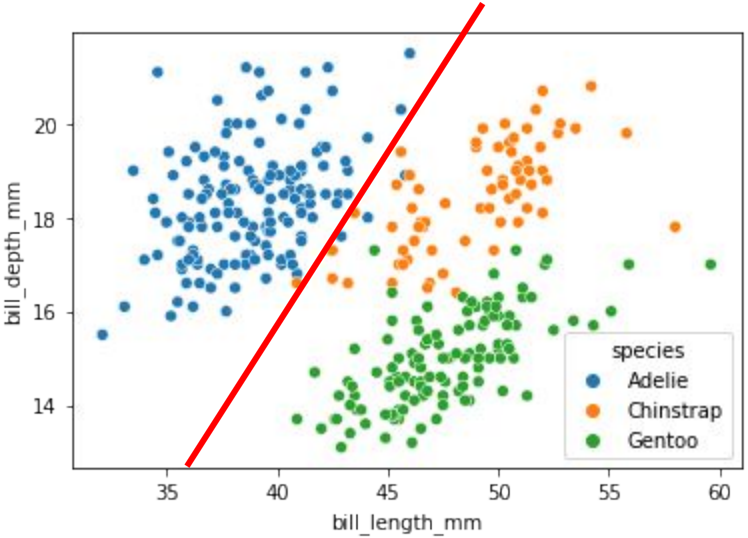
- To classify data (e.g., to predict whether binary output y is 0 or 1), we've learned how to use:
    - (Logistic) regression
    - Naive Bayes
    - K-means clustering

Linear separability



Given bill length and depth, can you draw a line that reliably separates Adelie penguins from Chinstrap and Gentoo penguins?

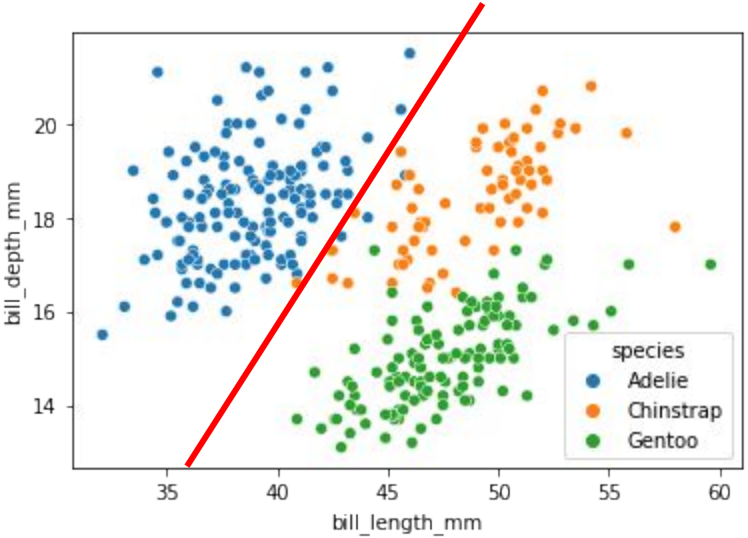
Linear separability



Given bill length and depth, can you draw a line that reliably separates Adelie penguins from Chinstrap and Gentoo penguins?

Yes! It's not perfect, but most of the blue dots (Adelie) are on one side, and most of the non-blue dots are on the other

Regression = Decision boundary



Math version:

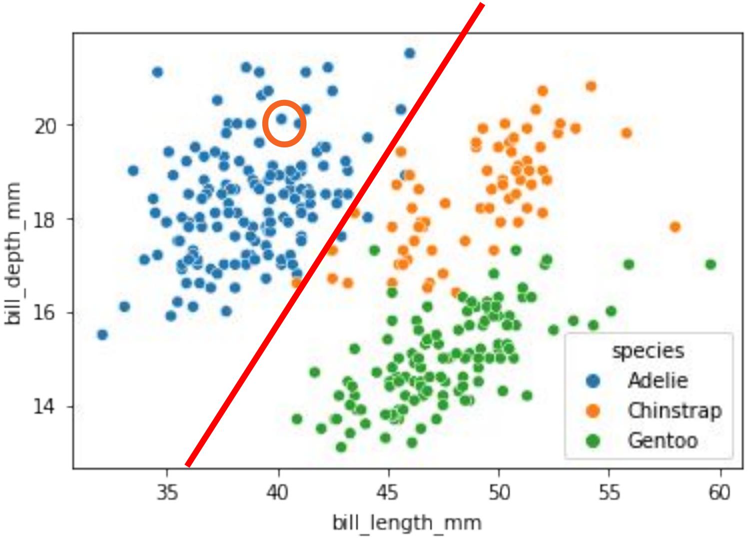
$\alpha = 25$

$\beta_{length} = -1.5$, $\beta_{depth} = 2.0$

Classification:

y = 1 if Adelie, 0 if Chinstrap, 0 if Gentoo

Calculate a linear output



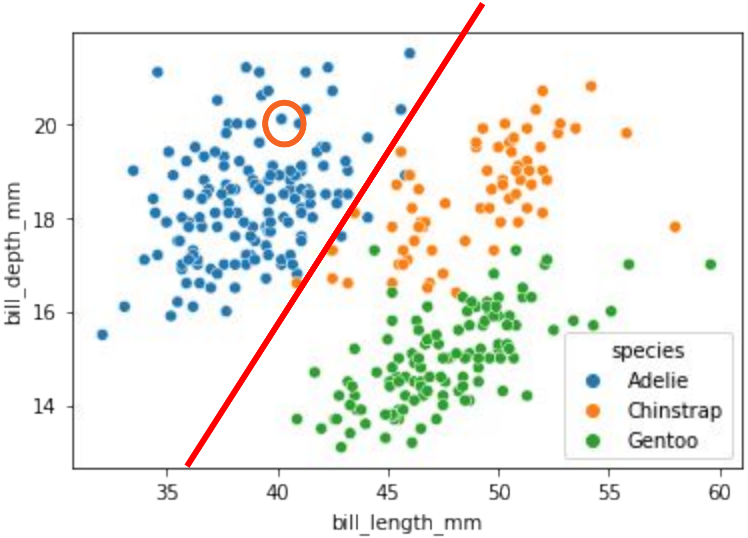
Math version:

$\alpha = 25$

$\beta_{length} = -1.5, \beta_{depth} = 2.0$

Is the value for length=40, depth=20 positive or negative?

Calculate a linear output



Math version:

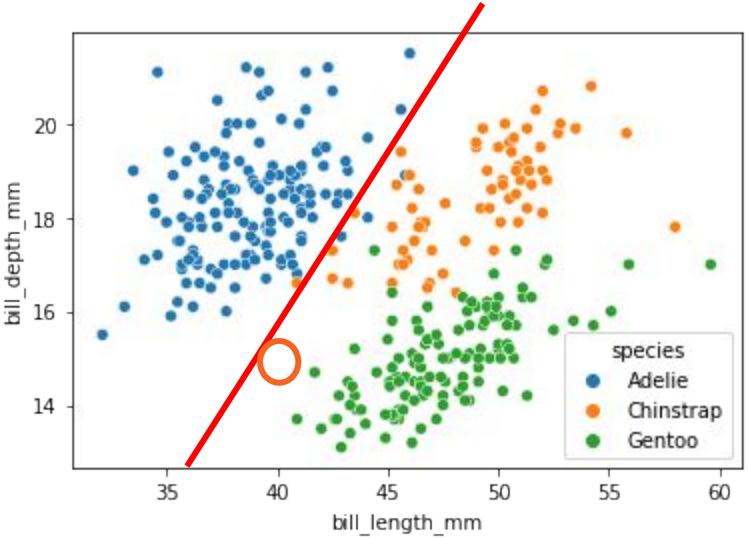
$\alpha = 25$

$\beta_{length} = -1.5$, $\beta_{depth} = 2.0$

Is the value for length=40, depth=20 positive or negative?

25 + (-60) + 40 = 5 *positive*

Calculate a linear output



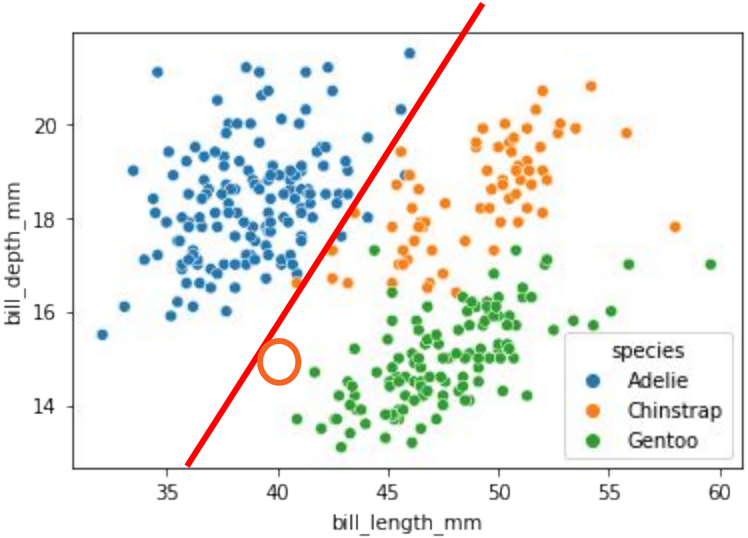
Math version:

$\alpha = 25$

$\beta_{length} = -1.5, \beta_{depth} = 2.0$

Is the value for length=40, depth=15 positive or negative?

Calculate a linear output



Math version:

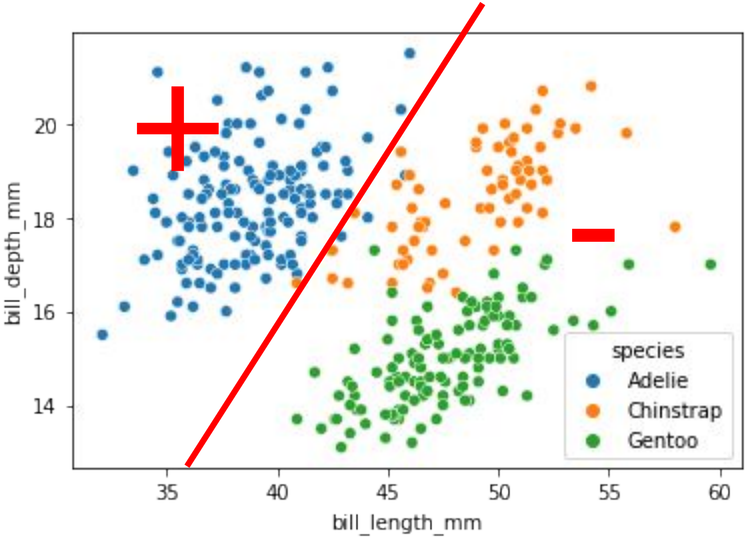
$\alpha = 25$

$\beta_{length} = -1.5$, $\beta_{depth} = 2.0$

Is the value for length=40, depth=15 positive or negative?

25 + (-60) + 30 = -5 ***negative***

Regions of positive / negative



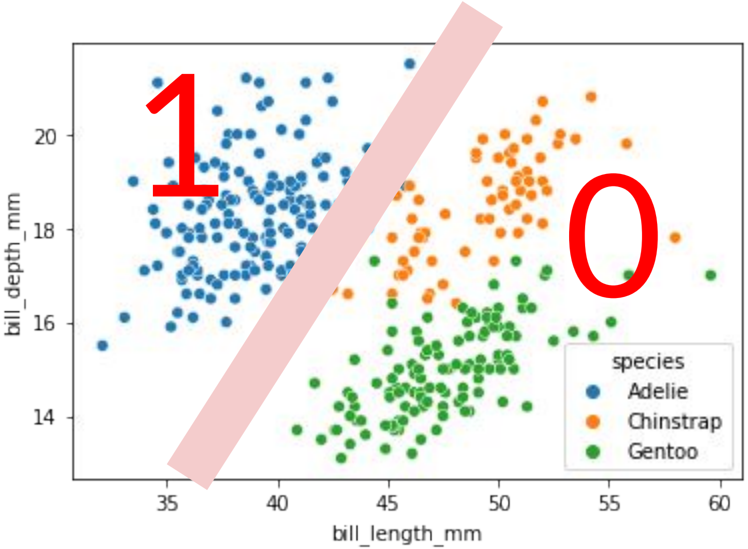
Math version:

$\alpha = 25$

$\beta_{length} = -1.5$, $\beta_{depth} = 2.0$

Applying these weights for any point left of the line gives us a positive value, and any point right of the line negative

logistic regression



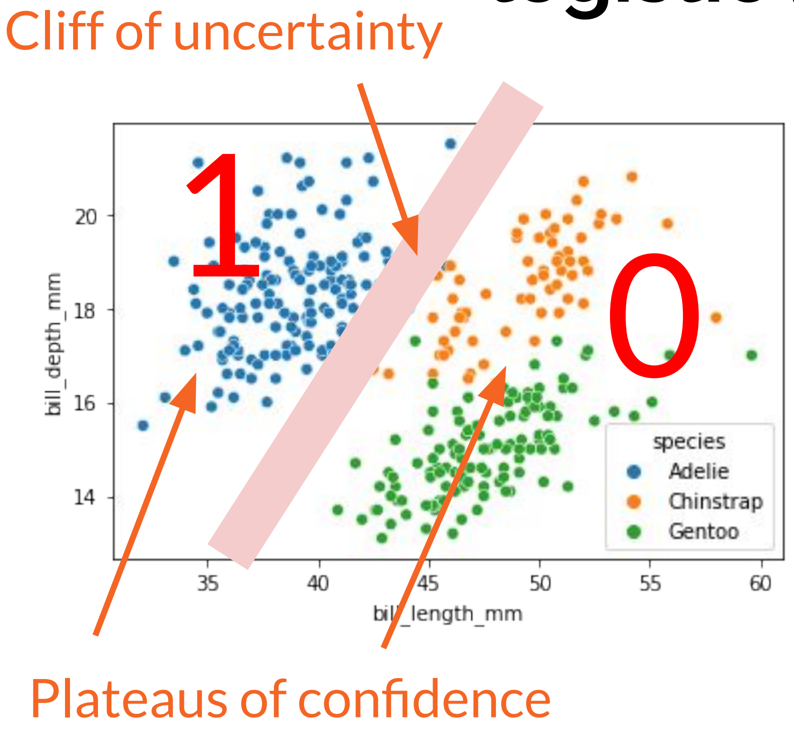
Math version:

$\alpha = 25$

$\beta_{length} = -1.5, \beta_{depth} = 2.0$

The logistic function squashes negative towards 0 and positive towards 1

logistic regression



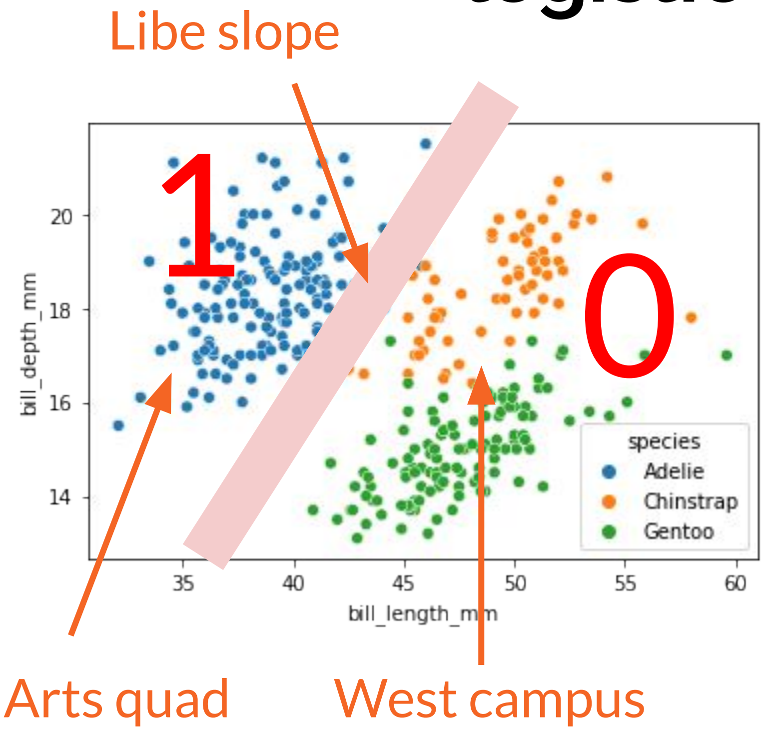
Math version:

$\alpha = 25$

$\beta_{length} = -1.5$, $\beta_{depth} = 2.0$

The logistic function squashes negative towards 0 and positive towards 1

logistic regression



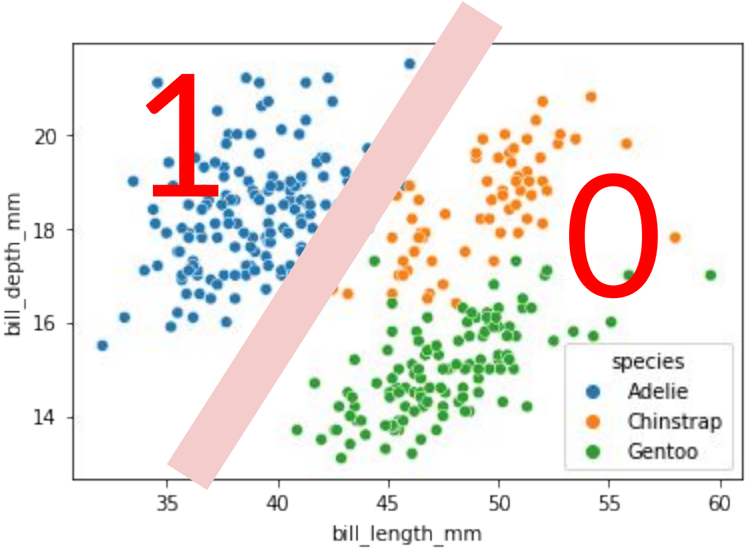
Math version:

$\alpha = 25$

$\beta_{length} = -1.5$, $\beta_{depth} = 2.0$

The logistic function squashes negative towards 0 and positive towards 1

Three views of Log. Reg.



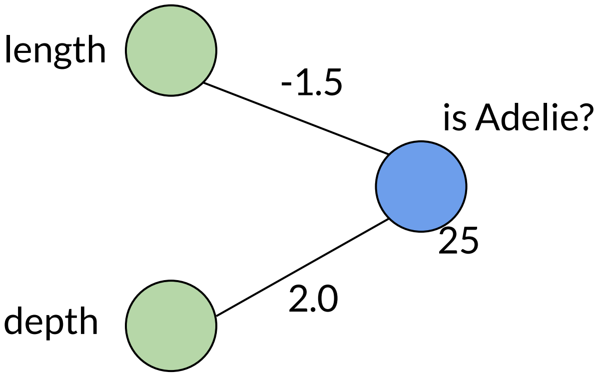
Mathematical expression

$$y = \sigma(\alpha + \beta_{length} X_{length} + \beta_{depth} X_{depth})$$

Python object

```
adelie_model.coef_, adelie_model.intercept_
(array([[-1.37,  1.98]]), array([24.29]))
```

A visual language for regression

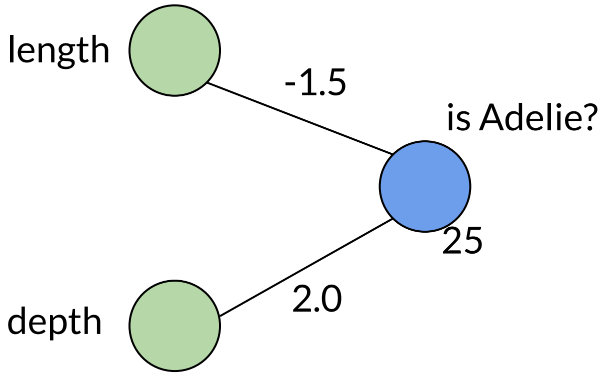


The *inputs* (length and depth) feed into the output (is the penguin Adelie?)

Each input has a *weight*

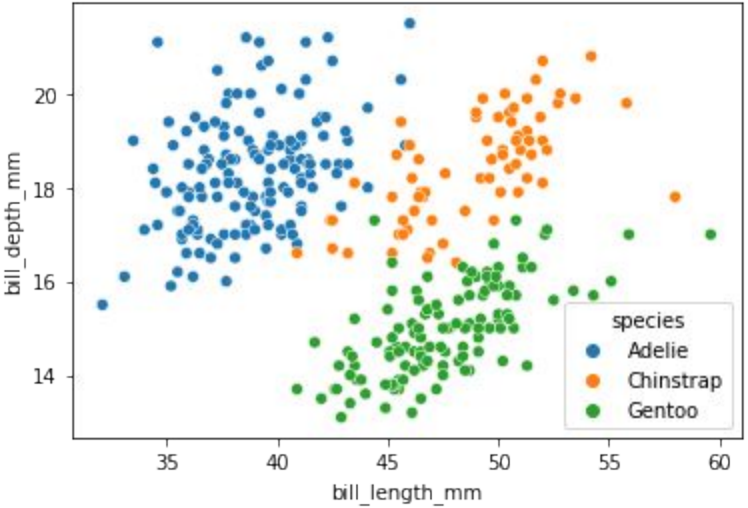
The output has a *bias* or *intercept*

A visual language for regression



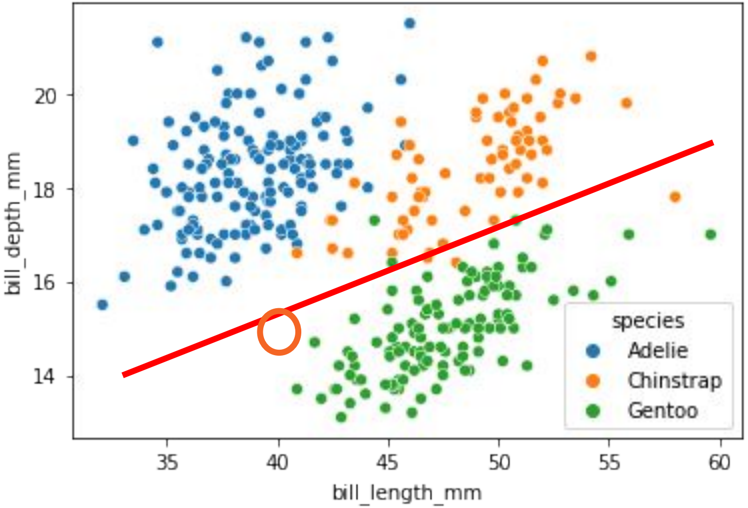
"To guess if a penguin is an Adelie, take -1.5 times the bill length plus 2.0 times the bill depth, and add 25. If the result is positive, guess yes (1), otherwise guess no (0)"

Linear separability



Given bill length and depth, can you draw a line that reliably separates Gentoo penguins from Chinstrap and Adelie penguins?

Linear separability



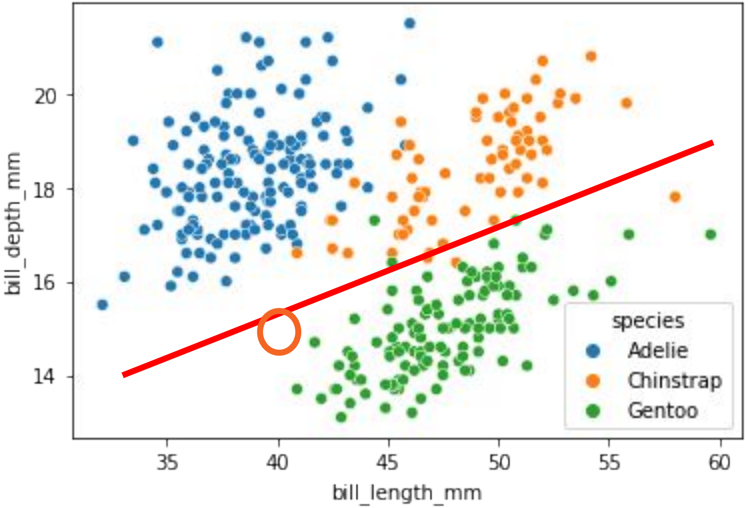
Math version:

$\alpha = 28$

$\beta_{length} = 0.5$, $\beta_{depth} = -3.0$

Is the value for length=40, depth=15 positive or negative?

Linear separability



Math version:

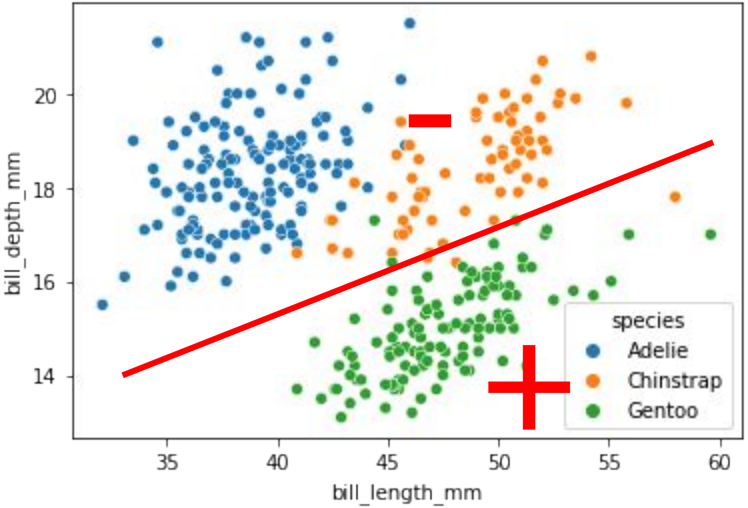
$\alpha = 28$

$\beta_{length} = 0.5$, $\beta_{depth} = -3.0$

Is the value for length=40, depth=15 positive or negative?

28 + 20 - 45 = 3 ***positive***

Regions of positive and negative



Math version:

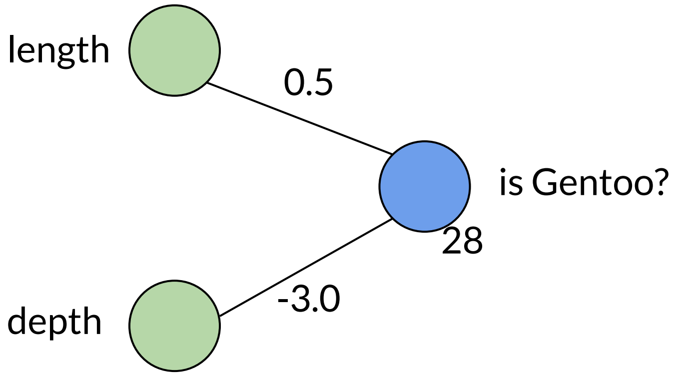
α = 28

β_{length} = 0.5, β_{depth} = -3.0

Is the value for length=40, depth=15 positive or negative?

28 + 20 - 45 = 3 *positive*

A visual language for regression

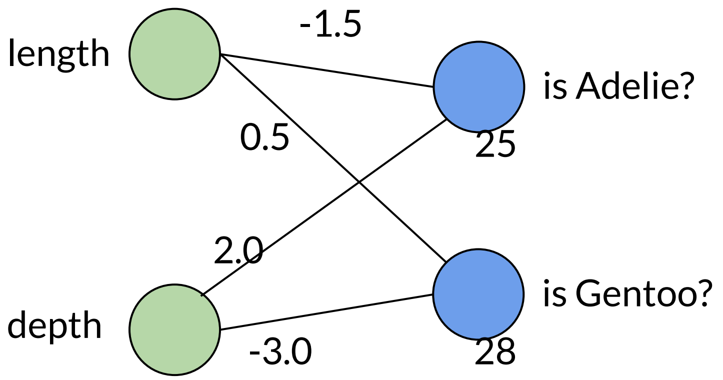


The *inputs* (length and depth) feed into the output (is the penguin Adelie?)

Each input has a *weight*

The output has a *bias* or *intercept*

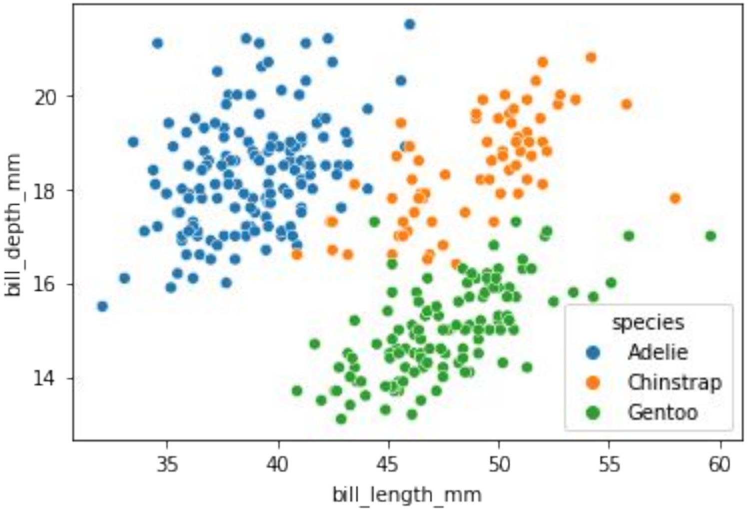
A visual language for regression



We can create more than one regression model from the same data!

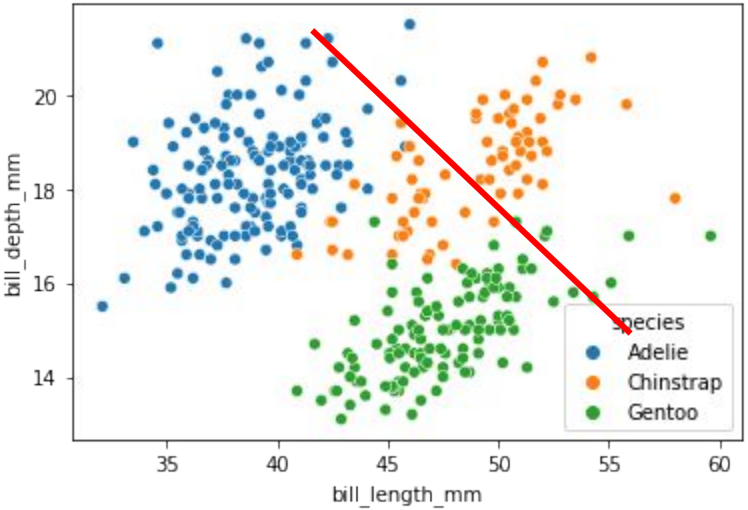
This is equivalent to drawing two decision boundaries

Linear separability



Given bill length and depth, can you draw a line that reliably separates Chinstrap penguins from Gentoo and Adelie penguins?

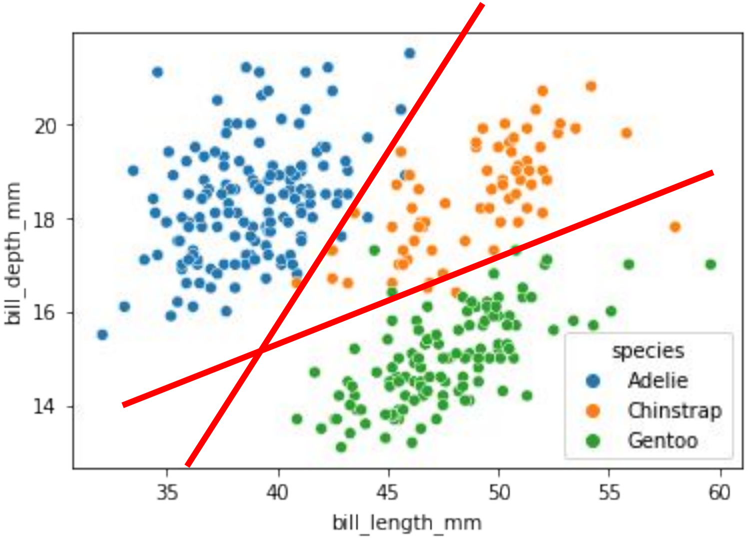
Linear separability



Given bill length and depth, can you draw a line that reliably separates Chinstrap penguins from Gentoo and Adelie penguins?

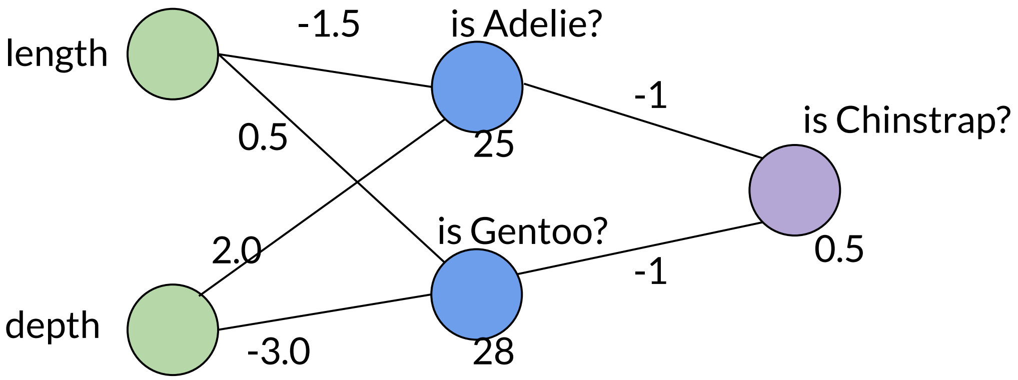
Not really, no

Outputs as inputs



If we combine the output of two linear classifiers, we can identify a non-linear region!

Multiple layers of regression



Outputs from layer 1 become inputs to layer 2

Classifying

- To classify data (e.g., to predict whether binary output y is 0 or 1), we've learned how to use:
    - (Logistic) regression
    - Naive Bayes
    - K-means clustering

Classifying

- To classify data (e.g., to predict whether binary output y is 0 or 1), we've learned how to use:
  - (Logistic) regression
  - Naive Bayes
  - K-means clustering
- **What if we have *continuous* output y?**

Classifying

- To classify data (e.g., to predict whether binary output y is 0 or 1), we've learned how to use:
  - (Logistic) regression
  - Naive Bayes
  - K-means clustering
- What if we have *continuous* output y? We can use **linear regression!**

Predicting continuous data

- We know we can do the following things with linear regressions:
    - **Predict** (continuous) outputs
    - **Summarize** relationships between input and output variables
    - Describe outliers/oddities

Predicting continuous data

- We know we can do the following things with linear regressions:
  - **Predict** (continuous) outputs
  - **Summarize** relationships between input and output variables
  - Describe outliers/oddities

What if we can get **more accurate predictions**

Predicting continuous data

- We know we can do the following things with linear regressions:
  - **Predict** (continuous) outputs
  - **Summarize** relationships between input and output variables
  - Describe outliers/oddities

What if we can get **more accurate predictions**

In exchange for **less interpretable results?**

Neural networks!

- Neural nets are **models**:
  - Still have multiple inputs x_i and make a single prediction $\hat{y}$
  - Still train on *train_set* and test on *test_set*

Neural networks!

- Neural nets are **models**:
  - Still have multiple inputs x_i and make a single $\boxed{\text{prediction } \hat{y}}$
  - Still train on *train_set* and test on *test_set*

Note: this prediction can be classification OR continuous

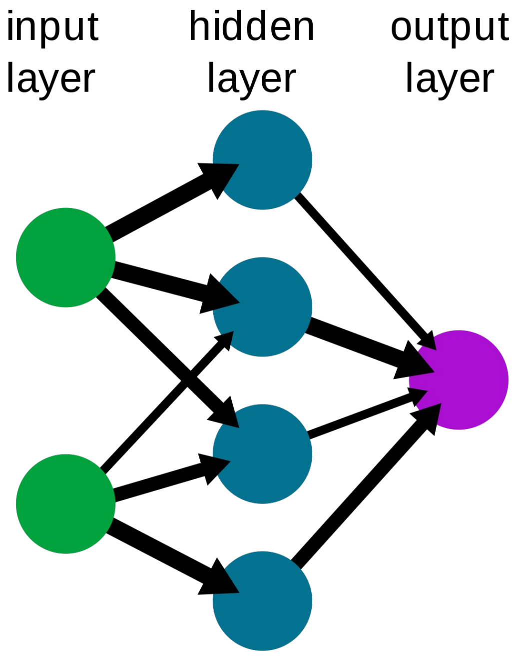
Neural networks!

- Neural nets are **models**:
  - Still have multiple inputs x_i and make a single prediction $\hat{y}$
  - Still train on *train_set* and test on *test_set*
- But, neural nets often outperform linear regressions on **big data** because they can account for **nonlinearities**

Neural nets work pretty well, but they aren't perfect!

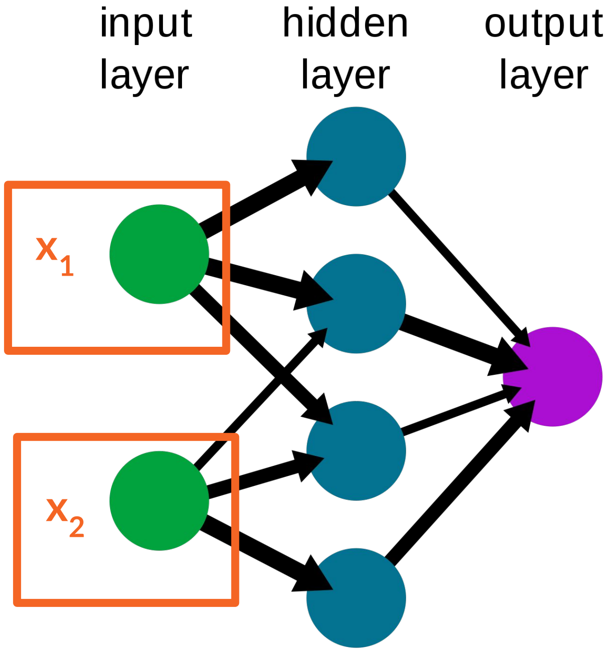
https://www.emergingtechbrew.com/stories/2022/02/16/the-creator-of-the-viral-neural-net-guesses-memes-twitter-account-explains-how-it-works

A simple neural net (NN)



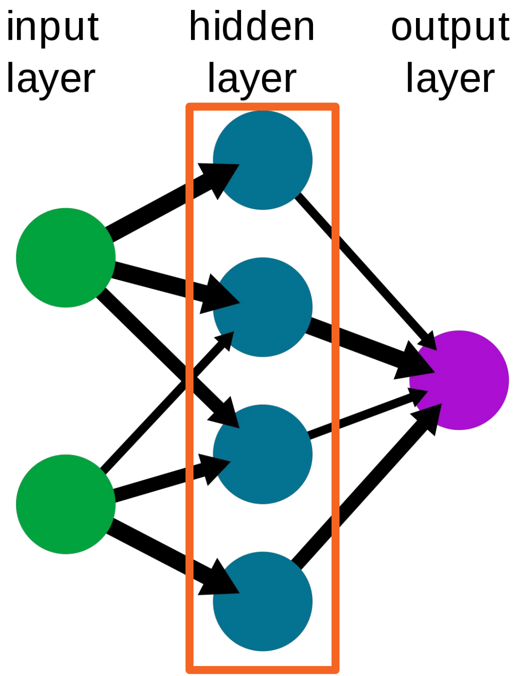
https://en.wikipedia.org/wiki/Neural_network

A simple neural net (NN)



https://en.wikipedia.org/wiki/Neural_network

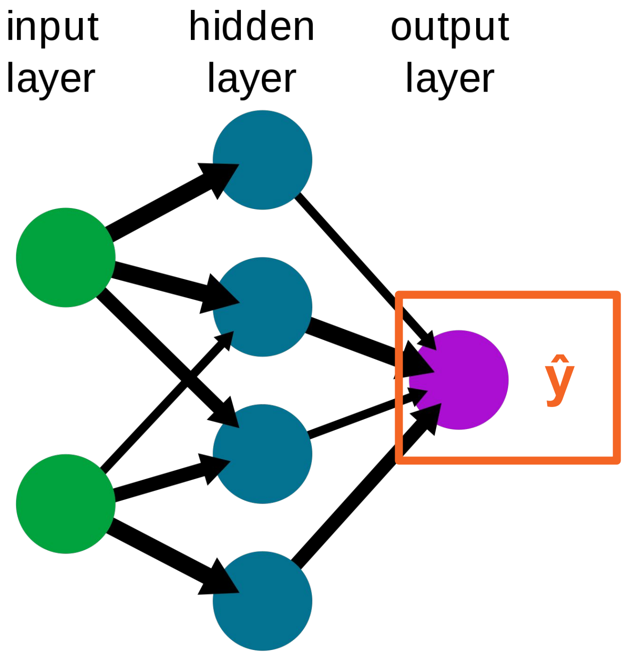
A simple neural net (NN)



Each circle is a "node" with an input and an output

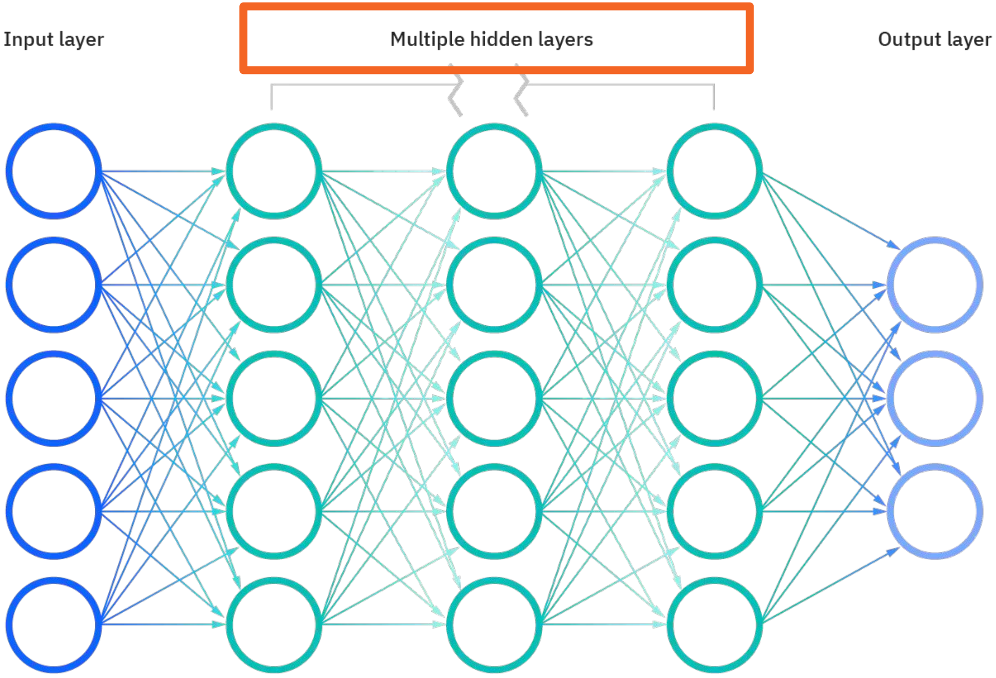
https://en.wikipedia.org/wiki/Neural_network

A simple neural net (NN)



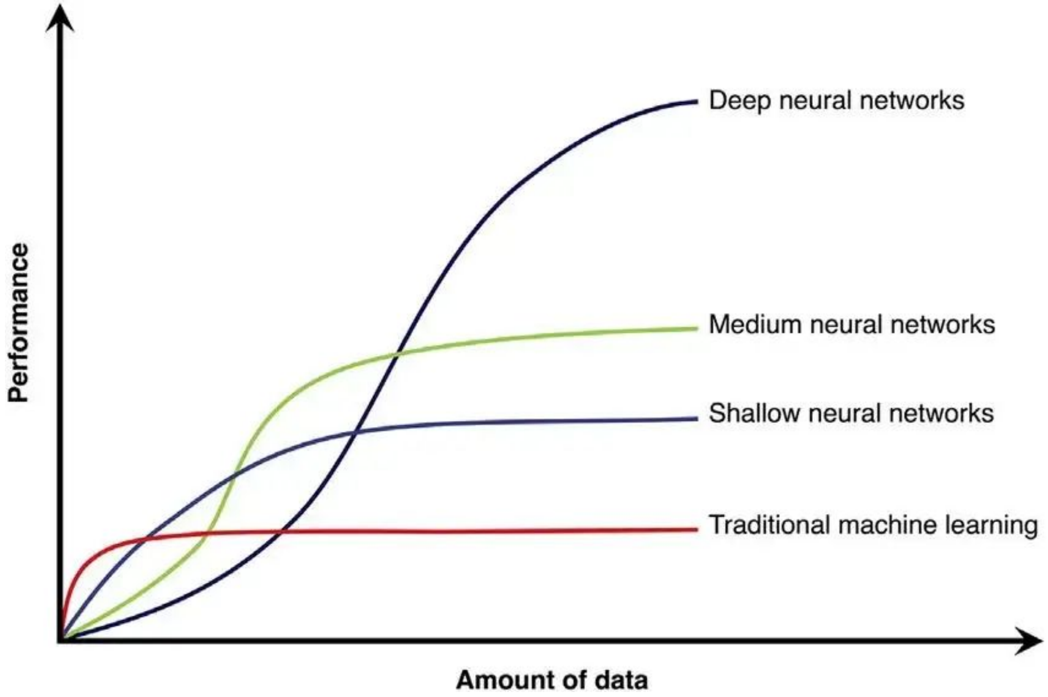
https://en.wikipedia.org/wiki/Neural_network

A deep neural net (NN)



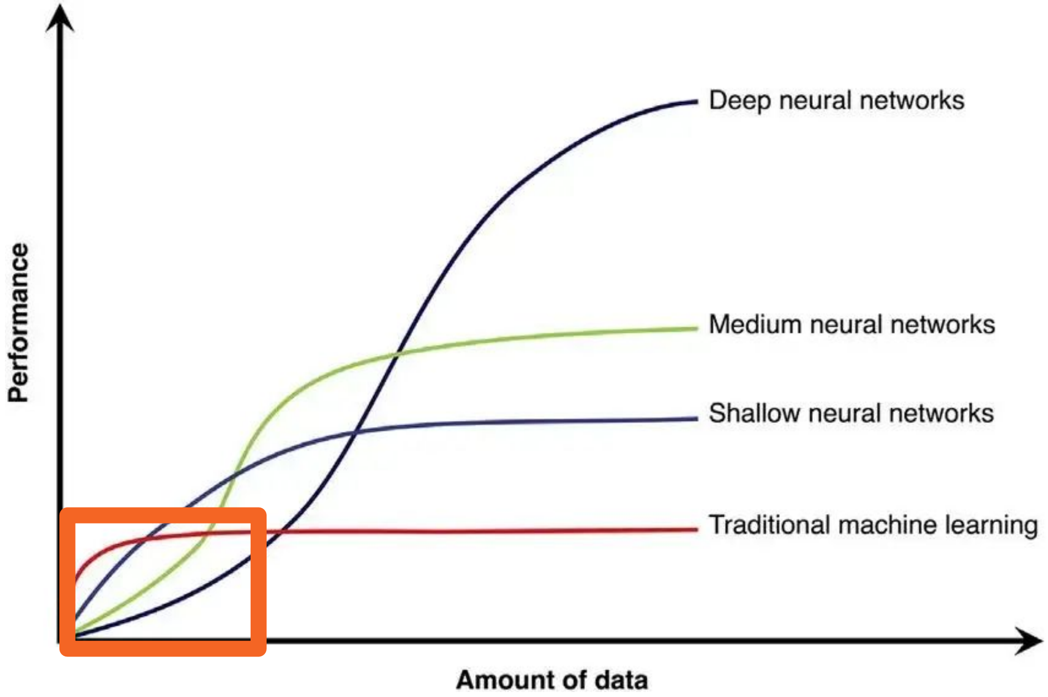
https://www.ibm.com/cloud/learn/neural-networks

Neural networks for big data



https://joonatan.medium.com/most-important-concepts-in-applied-machine-learning-11e1e5f33ff4

Neural networks for big data

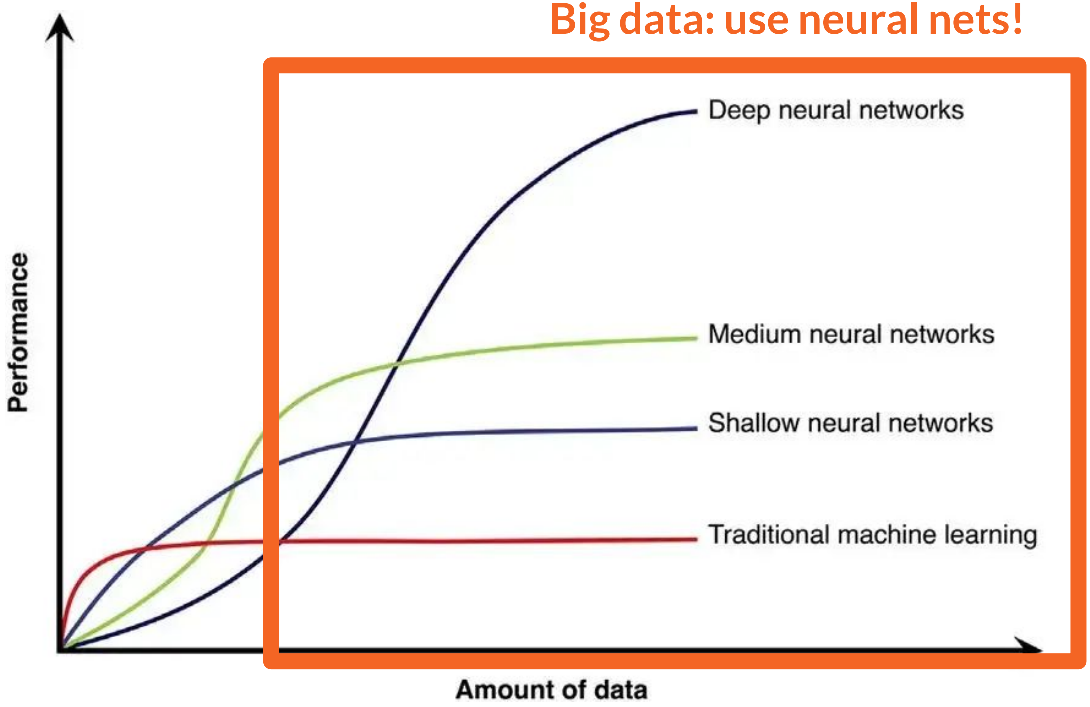


Small data: linear regression works great!

https://joonatan.medium.com/most-important-concepts-in-applied-machine-learning-11e1e5f33ff4

Neural networks for big data

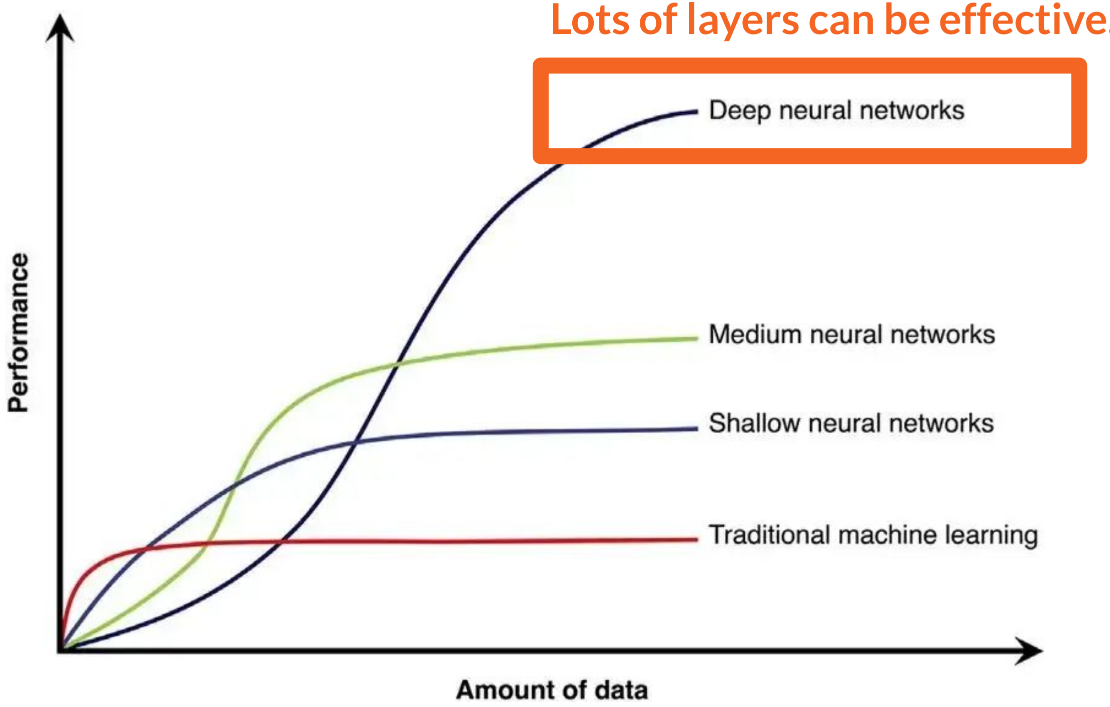
Big data: use neural nets!



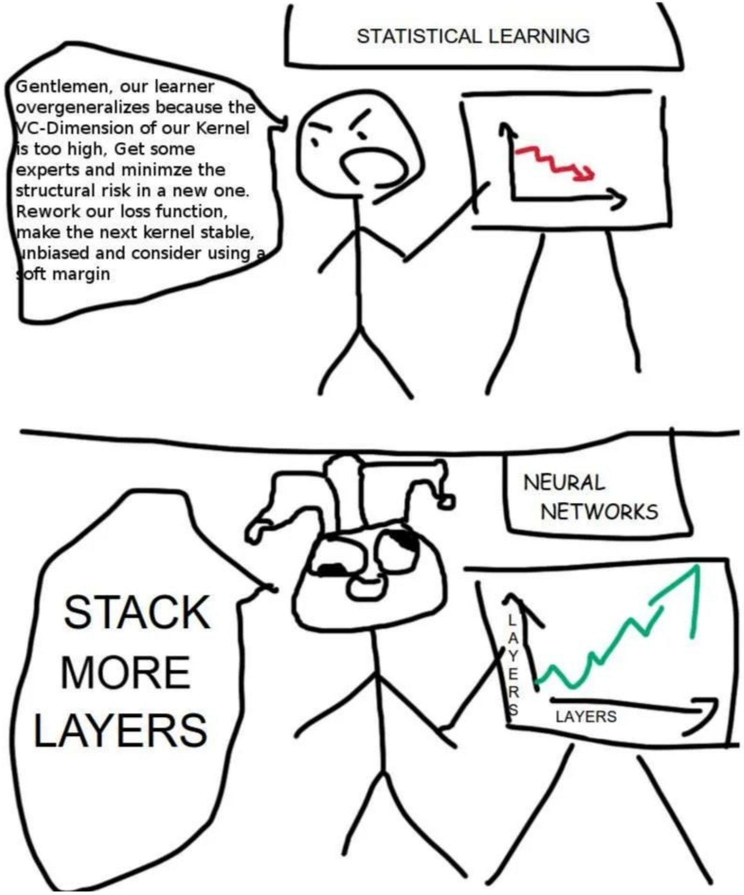
https://joonatan.medium.com/most-important-concepts-in-applied-machine-learning-11e1e5f33ff4

Neural networks for big data

Lots of layers can be effective...



https://joonatan.medium.com/most-important-concepts-in-applied-machine-learning-11e1e5f33ff4



https://www.reddit.com/r/ProgrammerHumor/comments/8c1i45/stack_more_layers/

Neural net drawbacks

- Neural nets are difficult for humans to interpret
- Neural nets are prone to overfitting on small data (needs big data to benefit from nonlinearities)
- Neural nets require a lot of compute
    - GPUs, CPUs, distributed computing (work in parallel)

Compute & sustainability

On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

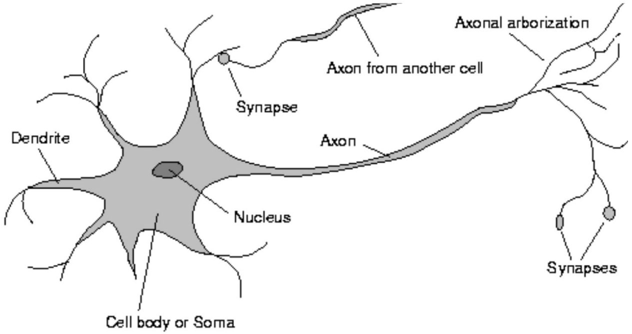
Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

- *"Training a single BERT base model (without hyperparameter tuning) on GPUs was estimated to require* ***as much energy as a trans-American flight.****"*

https://dl.acm.org/doi/pdf/10.1145/3442188.3445922

What is a neural net?

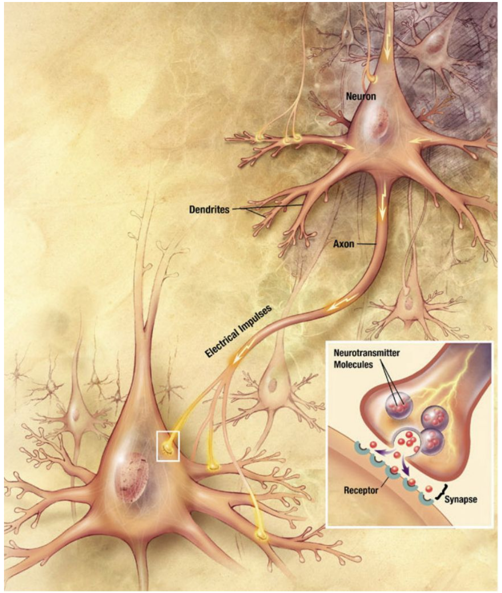
- Inspired by the human brain:
  - Different parts of the brain have different functions, which occur by firing neurons (10^{11})



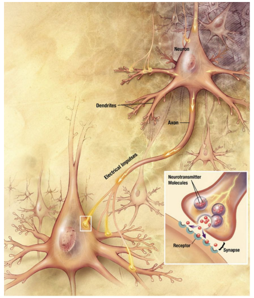
Based on slide by T. Finin, M. desJardins, L Getoor, R. Par
https://www.seas.upenn.edu/~cis5190/fall2017/lectures/11_NeuralNets.pdf

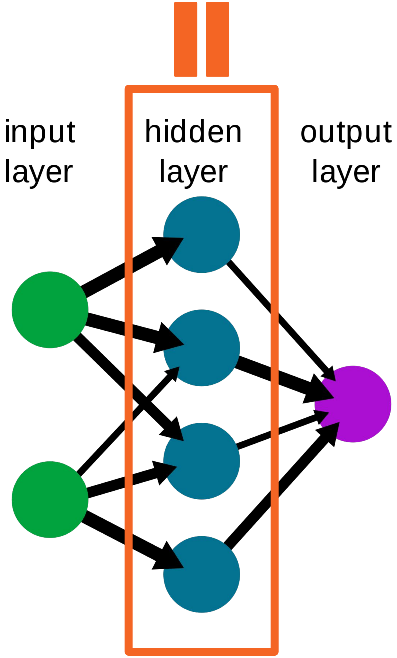
What is a neural net?

- Inspired by the human brain:
  - Different parts of the brain have different functions, which occur by firing neurons (10^{11})
  - We don’t exactly know how different brain functions are assigned (hard to interpret!), but we know neurons connect via synapses (10^{14})



Based on slide by T. Finin, M. desJardins, L Getoor, R. Par
https://www.seas.upenn.edu/~cis5190/fall2017/lectures/11_NeuralNets.pdf





What is a neural net?

- Inspired by the human brain:
  - Different parts of the brain have different functions, which occur by firing neurons (10^{11})
  - We don’t exactly know how different brain functions are assigned (hard to interpret!), but we know neurons connect via synapses (10^{14})

What is a neural net?

- Inspired by the human brain:
  - Different parts of the brain have different functions, which occur by firing neurons (10^{11})
  - We don't exactly know how different brain functions are assigned (hard to interpret!), but we know neurons connect via synapses (10^{14})
  - The output is "consciousness"

Based on slide by T. Finin, M. desJardins, L Getoor, R. Par

What is a neural net?

- Inspired by the human brain:
  - Different parts of the brain have different functions, which occur by firing neurons (10^{11})
  - We don't exactly know how different brain functions are assigned (hard to interpret!), but we know neurons connect via synapses (10^{14})
  - The output is "consciousness"

Computers: "only" 10^9 gates, 10^9 bits RAM. Brain wins!

Based on slide by T. Finin, M. desJardins, L Getoor, R. Par

Example: predicting house prices

- What are some input x’s that are useful to predict output y = house price?

Example: predicting house prices

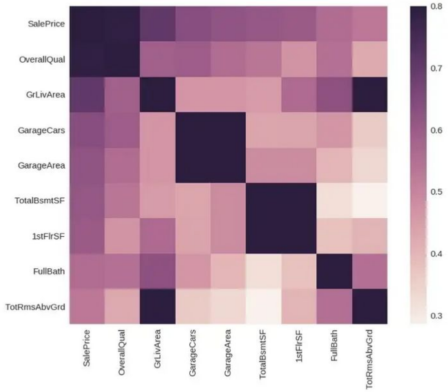
- What are some input x’s that are useful to predict output y = house price?
    - Square footage
    - # bed, # bath
    - Location
    - Neighborhood house $’s
    - …

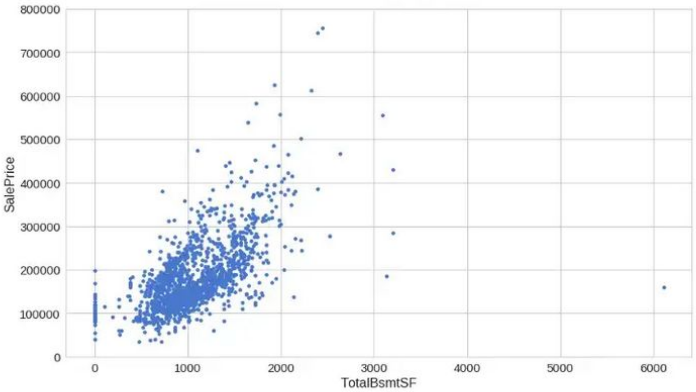
Refresher: linear regression

- Step 1: decide on your inputs and outputs
  - Plot/summarize…
    - Outliers
    - Collinearity
    - Missing data

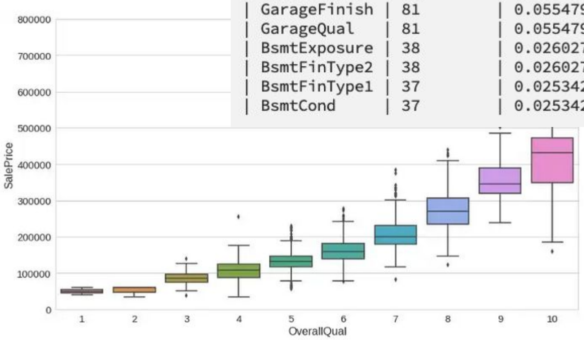
Refresher: linear regression

- Step 1: decide on your inputs and outputs
  - Plot/summarize...
    - Outliers
    - Collinearity
    - Missing data





	Row count	Percentage
PoolQC	1453	0.995205
MiscFeature	1406	0.963014
Alley	1369	0.937671
Fence	1179	0.807534
FireplaceQu	690	0.472603
LotFrontage	259	0.177397
GarageCond	81	0.055479
GarageType	81	0.055479
GarageYrBlt	81	0.055479
GarageFinish	81	0.055479
GarageQual	81	0.055479
BsmtExposure	38	0.026027
BsmtFinType2	38	0.026027
BsmtFinType1	37	0.025342
BsmtCond	37	0.025342



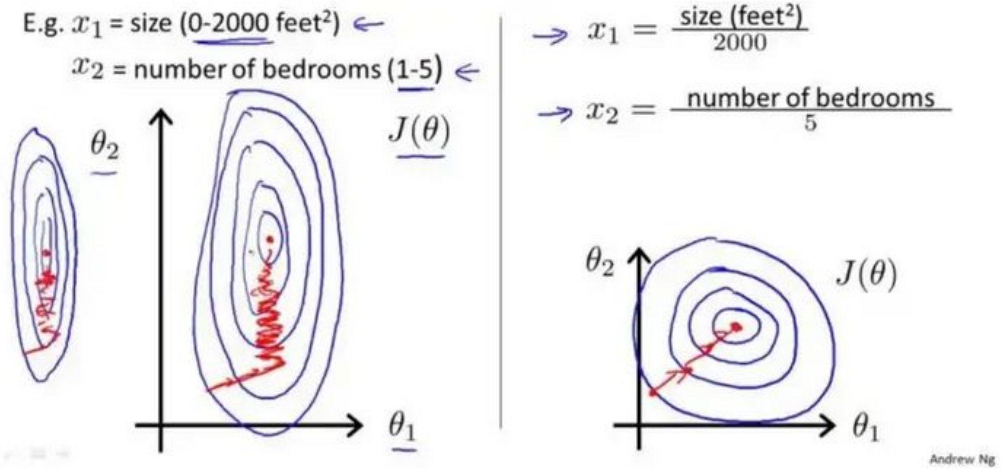
https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1

Refresher: linear regression

- Step 2: perform data preprocessing
    - Missing values (imputation)
    - Transformations, normalization

Refresher: linear regression

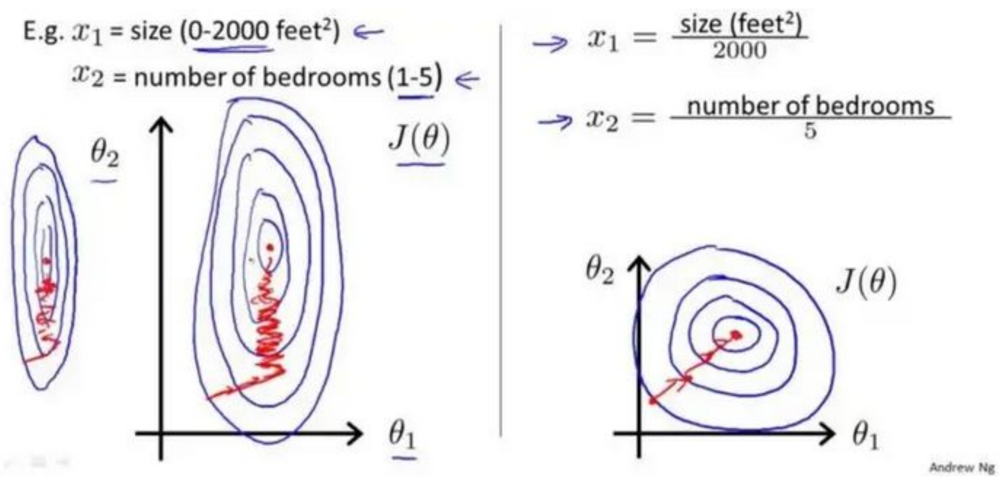
- Step 2: perform data preprocessing
  - Missing values (imputation)
  - Transformations, normalization



https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1

Refresher: linear regression

- Step 2: perform data preprocessing
    - Missing values (imputation)
    - Transformations, normalization



In addition to having more easily comparable coefficients, our algorithms (SGD) converge faster when data is scaled!

https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1

Refresher: linear regression

- Step 3: decide on an evaluation metric
    - Choose one based on data type of output y
        - For housing prices: ____

https://blog.jovian.ai/how-to-predict-housing-prices-with-linear-regression-b7c5a6fe1175

Refresher: linear regression

- Step 3: decide on an evaluation metric
  - Choose one based on data type of output y
    - For housing prices: **RMSE, MSE, MAE, MAPE, … any evaluation metric that is reasonable for *continuous output y***

https://blog.jovian.ai/how-to-predict-housing-prices-with-linear-regression-b7c5a6fe1175

Refresher: linear regression

- Step 3: decide on an evaluation metric
  - Choose one based on data type of output y
    - For housing prices, maybe RMSE
    - ```
      def rmse(predict,actual):
      return np.sqrt(np.mean(np.square(predict - actual)))
      ```

https://blog.jovian.ai/how-to-predict-housing-prices-with-linear-regression-b7c5a6fe1175
      ```

Refresher: linear regression

- Step 3: decide on an evaluation metric
  - Choose one based on data type of output y
    - For housing prices, maybe RMSE

- Step 4: split your data
  - Train / val / test sets
  - `X_train,X_test,y_train,y_test =train_test_split(X,y,test_size=0.4)`

https://blog.jovian.ai/how-to-predict-housing-prices-with-linear-regression-b7c5a6fe1175

Refresher: linear regression

- Step 5: run a regression
    - $y \sim x_1 + x_2 + x_3$

$$\boxed{Y(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0}$$

Different notation: instead of β's for coefficients, we now use *w* to represent "weights"

https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1

Refresher: linear regression

- Step 5: run a regression
  - $y \sim x_1 + x_2 + x_3 + \ldots$

```
model=LinearRegression().fit(X_train,y_train)
predictions_test=model.predict(X_test)
rmse(predictions_test,y_test)
```

https://blog.jovian.ai/how-to-predict-housing-prices-with-linear-regression-b7c5a6fe1175

Refresher: linear regression

- Step 5: run a regression
  - $y \sim x_1 + x_2 + x_3 + \ldots$

```
model=LinearRegression().fit(X_train,y_train)
predictions_test=model.predict(X_test)
rmse(predictions_test,y_test)
```

- Step 6: interpret results!
  - Predict, summarize, outliers/oddities

https://blog.jovian.ai/how-to-predict-housing-prices-with-linear-regression-b7c5a6fe1175

Model Recap

- Step 1: decide on inputs / outputs
- Step 2: data preprocessing
- Step 3: decide on an evaluation metric
- Step 4: split your data
- Step 5: run the model
- Step 6: interpret results

But what about neural nets?!

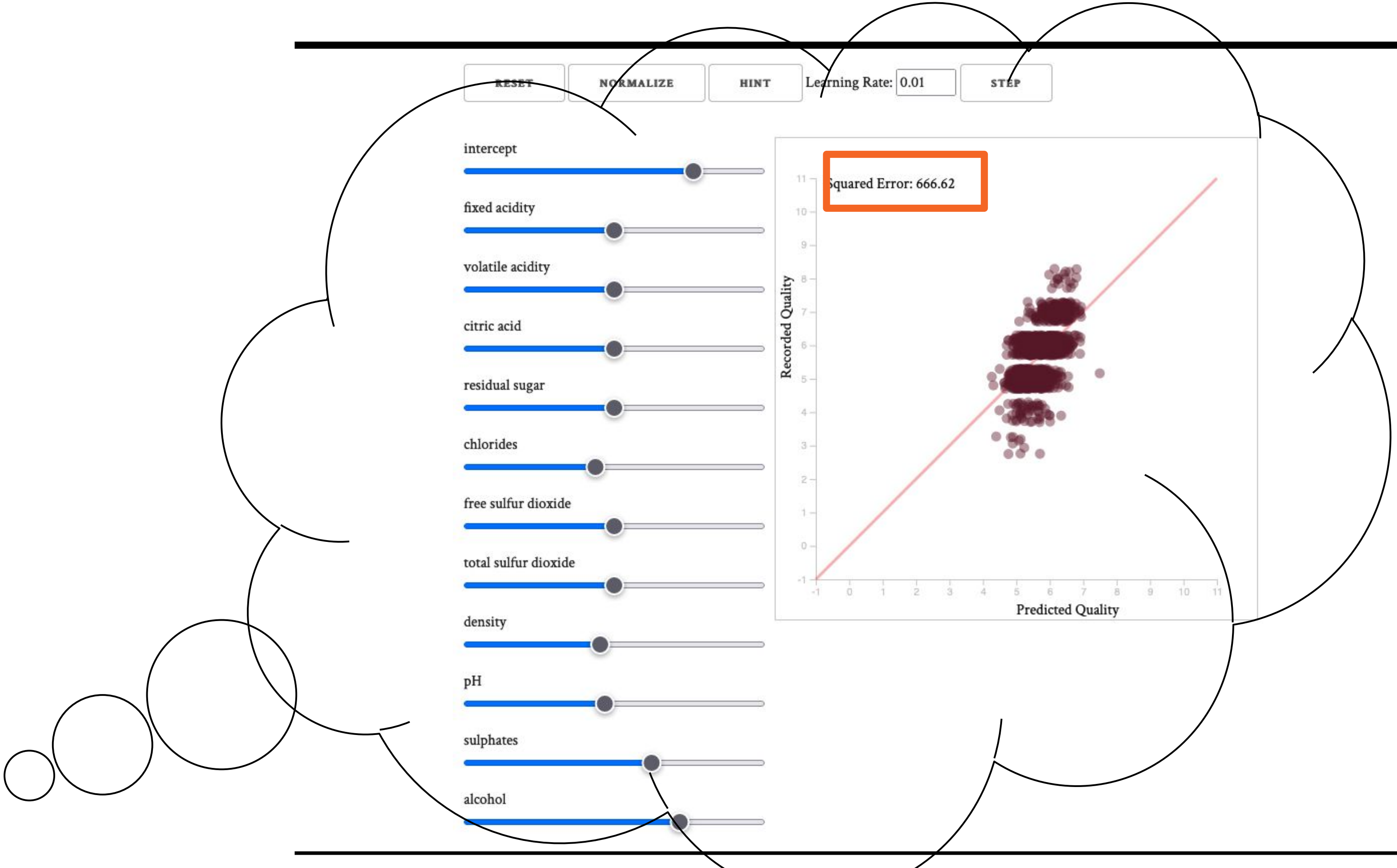
- A lot of these steps are the same!
  - Step 1: decide on inputs / outputs
  - Step 2: data preprocessing
  - Step 3: decide on an evaluation metric
  - Step 4: split your data
  - Step 5: run the model
  - Step 6: interpret results

We do this before even touching linear regression / neural nets!

But what about neural nets?!

- A lot of these steps are the same!
  - Step 1: decide on inputs / outputs
  - Step 2: data preprocessing
  - Step 3: decide on an evaluation metric
  - Step 4: split your data
  - Step 5: run the model
  - Step 6: interpret results

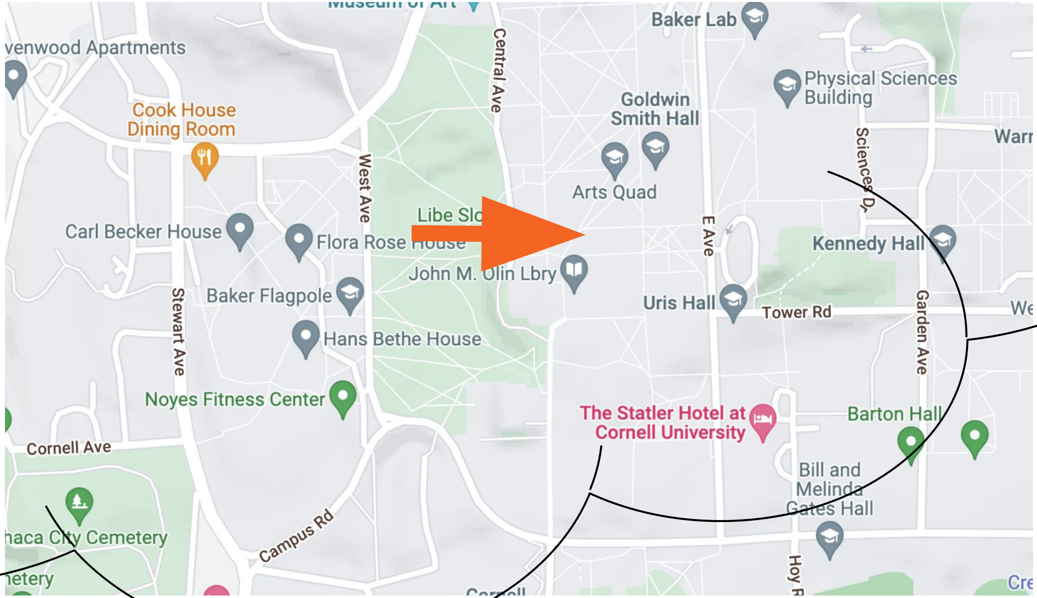
Caveat: more explanation needed here



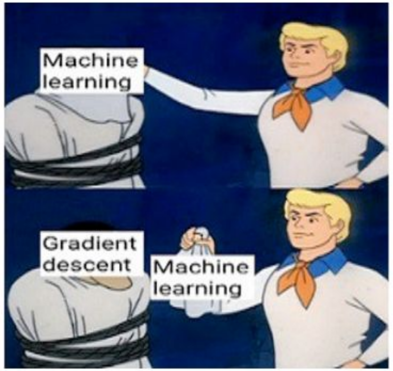
https://mimno.infosci.cornell.edu/info2950/interactive/wine.html

Gradient is a hint in multiple directions

The slope is steep in the East-West direction, but flat in the North-South direction



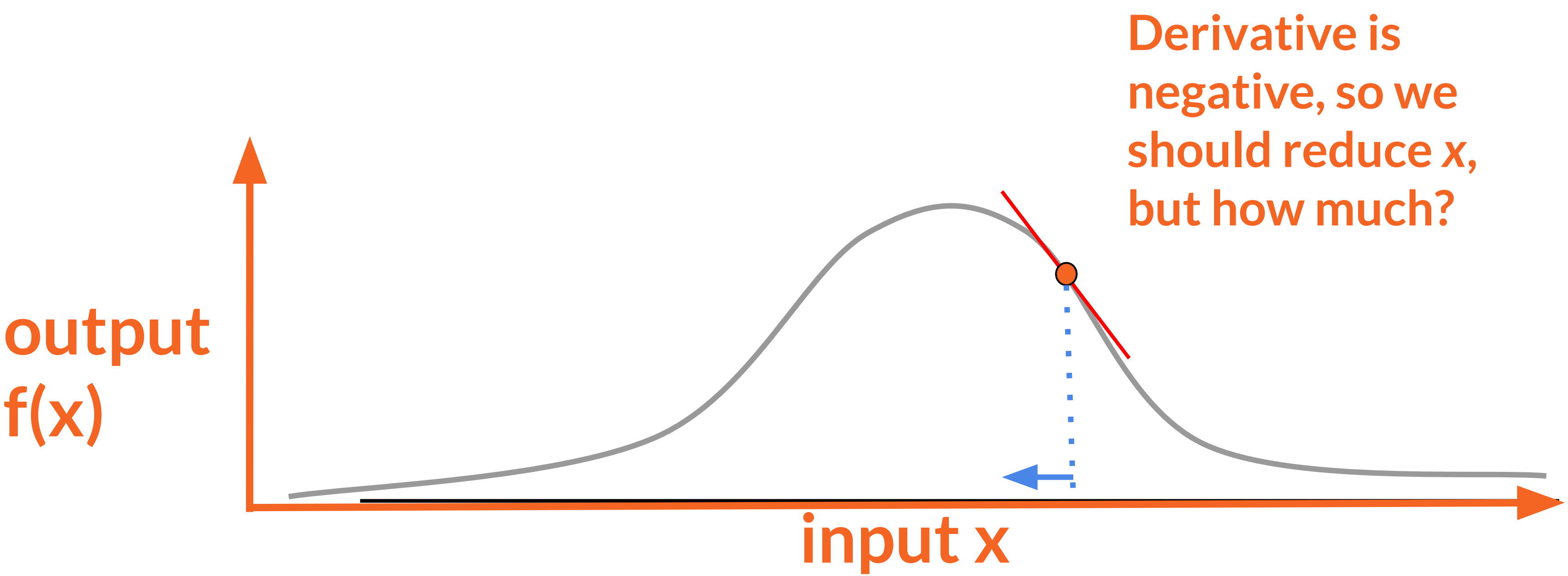
Takeaways on gradients



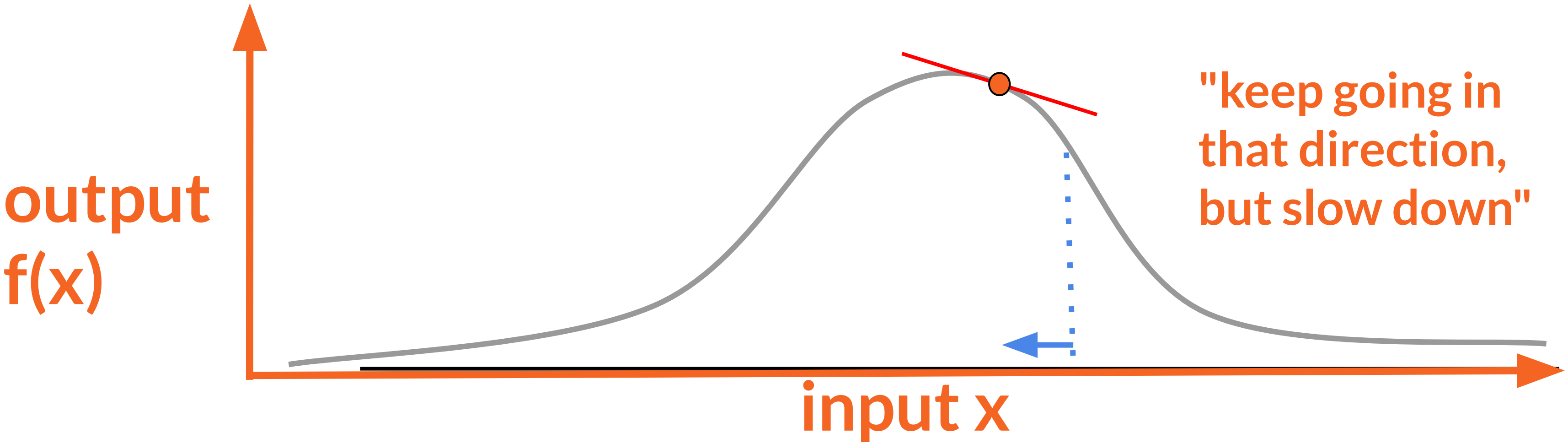
- "Stochastic gradient descent" is used to find minima / maxima for complicated models (e.g. multivariable regression)
  - choose a **learning** rate to do this efficiently
  - this is the core of modern machine **learning**!

https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a

Stochastic Gradient Descent (SGD)



SGD: move β in a direction specified by the gradient



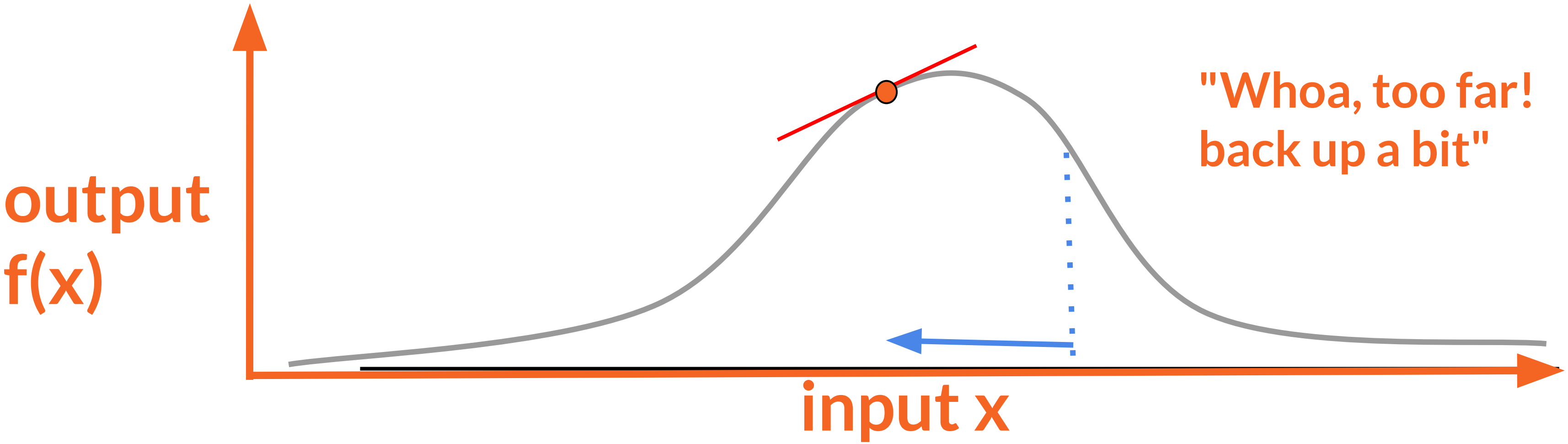
Slope still negative, but less so:

"keep going in that direction, but slow down"

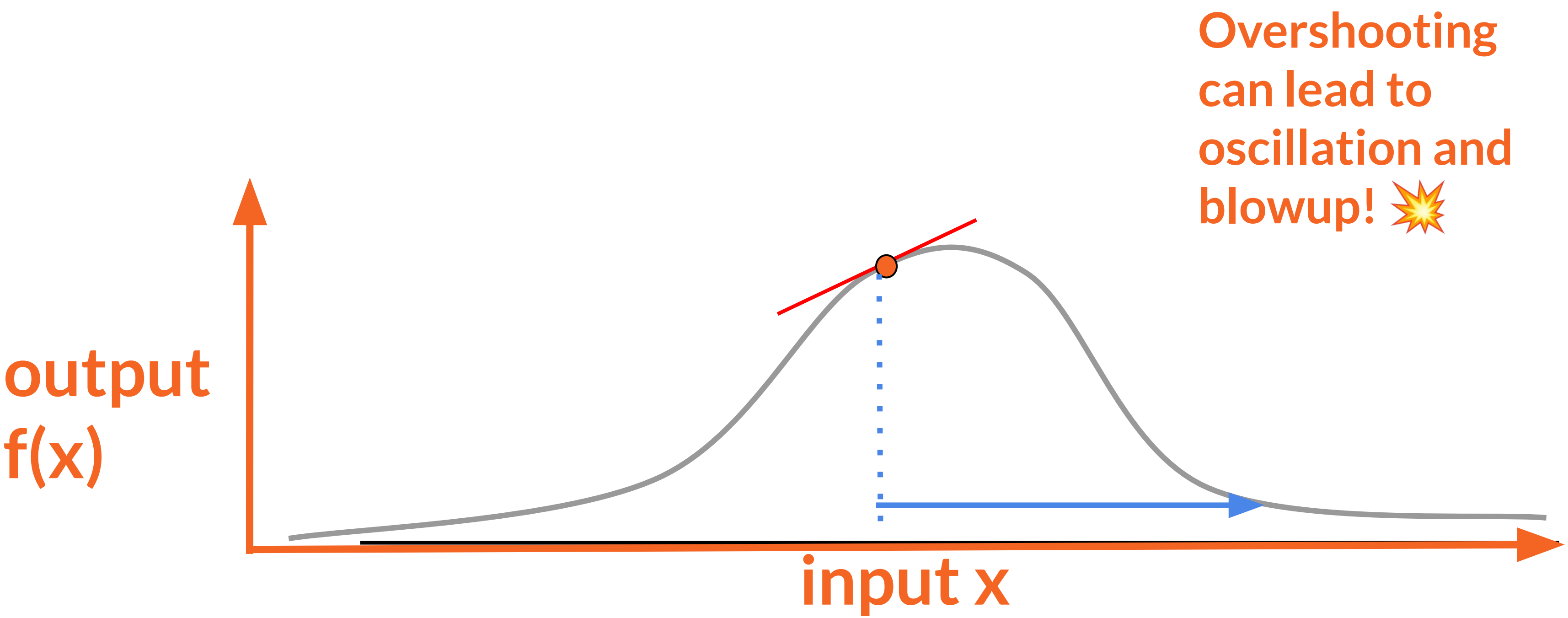
SGD: "step size" or "learning rate" is important

If you take a big step you may overshoot:

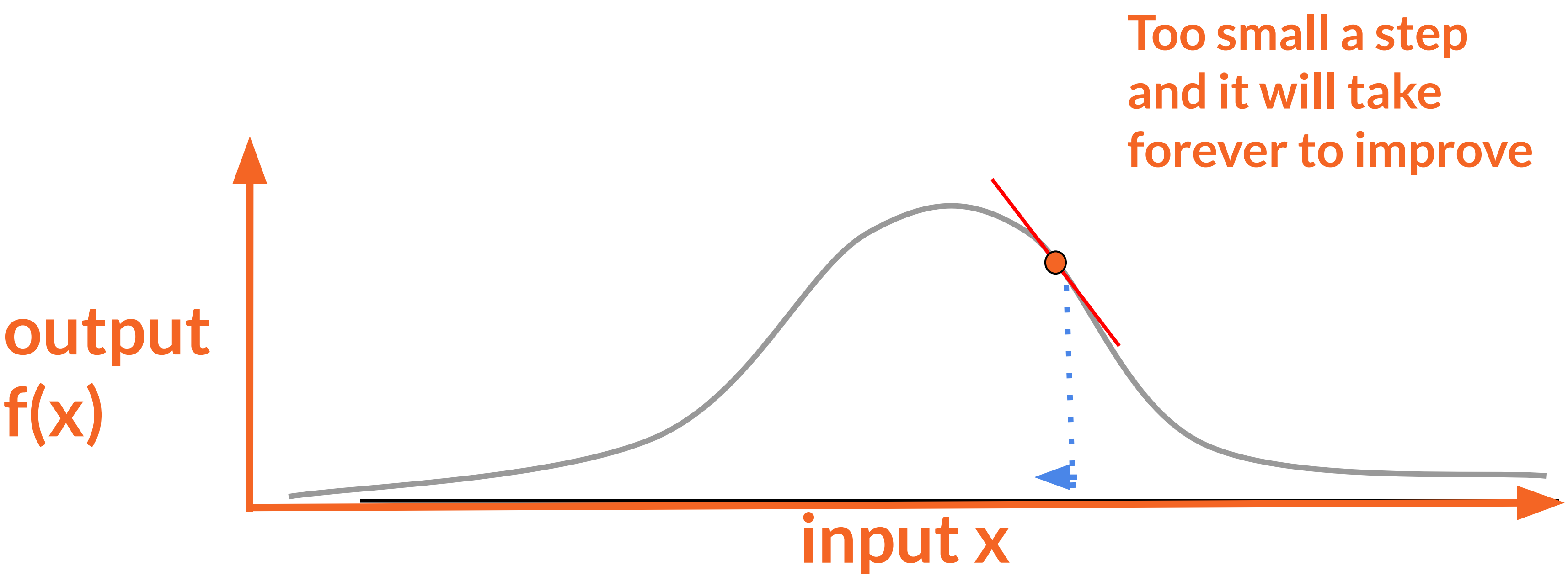
"Whoa, too far! back up a bit"



SGD: "step size" or "learning rate" is important



SGD: "step size" or "learning rate" is important

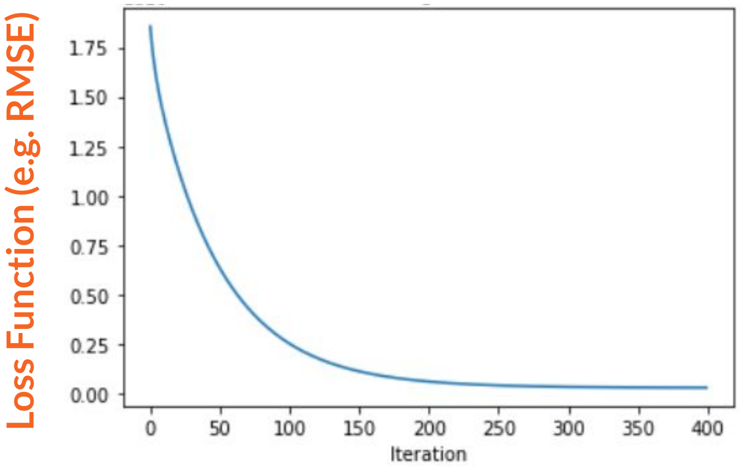


Step 3: decide on an evaluation metric

- In machine learning, we use an algorithm (e.g. SGD) that steps towards convergence and “learns” when we’ve reached a local minimum
  - “Loss function” = how you evaluate when to stop
  - RMSE is one choice of loss function

Step 3: decide on an evaluation metric

- In machine learning, we use an algorithm (e.g. SGD) that steps towards convergence and "learns" when we've reached a local minimum



https://medium.com/@ben_lau93/house-prices-prediction-using-andrew-ngs-machine-learning-algorithm-31b29a81acb8

Step 3: decide on an evaluation metric

- In Step 5, we've used sklearn to run:

```
model=LinearRegression().fit(X_train,y_train)
```

- But, we can re-define "fit" to include things like…

```
def fit(self, X, y, n_iter=100000, lr=0.01):
    # [gradient descent code that runs for n_iter iterations and
    # uses lr learning rate step sizes]
    return self
```

https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1

Step 3: decide on an evaluation metric

- In Step 5, we've used sklearn to run:

```
model=LinearRegression().fit(X_train,y_train)
```

- But, we can re-define "fit" to include things like...

```
def fit(self, X, y, n_iter=100000, lr=0.01):
    # [gradient descent code that runs for n_iter iterations and
    # uses lr learning rate step sizes]
    return self
```

```
model=LinearRegression().fit(X_train,y_train, 2000, 0.01)
```

https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1

Step 3: decide on an evaluation metric

- Your loss should be smoothly converging to a local min
- Debug by checking for NaNs or repeated input data, changing learning rate, etc. if you see plots like:

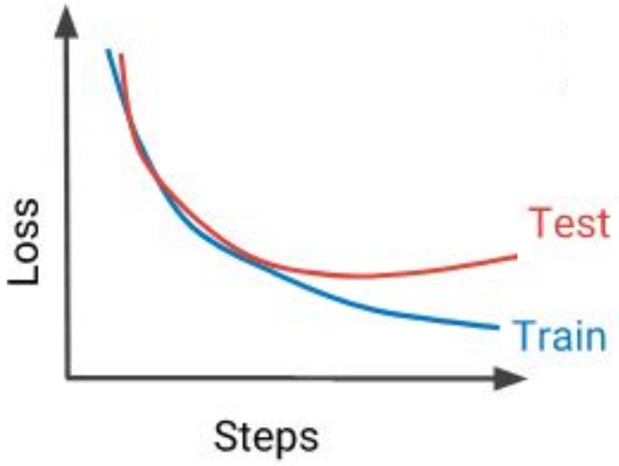
https://developers.google.com/machine-learning/testing-debugging/metrics/interpretic

Step 3: decide on an evaluation metric

- It's important to check the loss for each of your train / val / test sets separately, to ensure no overfitting is happening!

Step 3: decide on an evaluation metric

- It's important to check the loss for each of your train / val / test sets separately, to ensure no overfitting is happening!

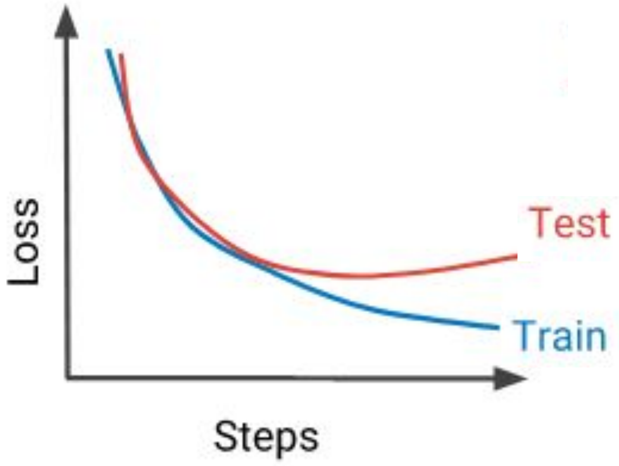


Is this indicative of overfitting?

https://developers.google.com/machine-learning/testing-debugging/metrics/interpretic

Step 3: decide on an evaluation metric

- It's important to check the loss for each of your train / val / test sets separately, to ensure no overfitting is happening!



This is overfit! We're performing way better on train than test! (We want to *minimize* loss)

https://developers.google.com/machine-learning/testing-debugging/metrics/interpretic

But what about neural nets?!

- A lot of these steps are the same!
  - Step 1: decide on inputs / outputs
  - Step 2: data preprocessing
  - Step 3: decide on an evaluation metric
  - Step 4: split your data
  - Step 5: run the model
  - Step 6: interpret results

Specifically, make sure you have code to plot loss function

But what about neural nets?!

- A lot of these steps are the same!
  - Step 1: decide on inputs / outputs
  - Step 2: data preprocessing
  - Step 3: decide on an evaluation metric
  - Step 4: split your data
  - Step 5: run the model
  - Step 6: interpret results

How do we do this with neural nets?

1 min break & attendance!

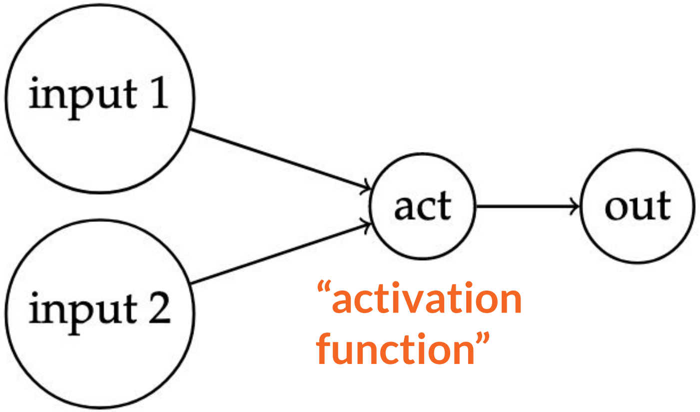
tinyurl.com/7haktr95

But what about neural nets?!

- A lot of these steps are the same!
  - Step 1: decide on inputs / outputs
  - Step 2: data preprocessing
  - Step 3: decide on an evaluation metric
  - Step 4: split your data
  - Step 5: run the model
  - Step 6: interpret results

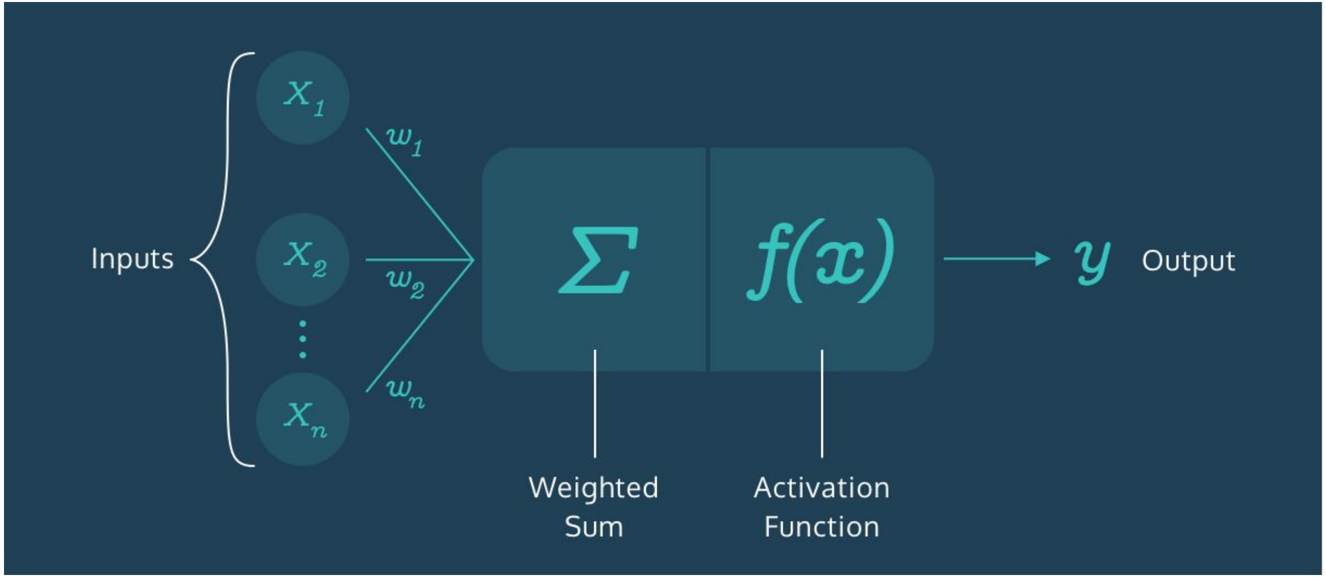
How do we do this with neural nets?

Perceptron



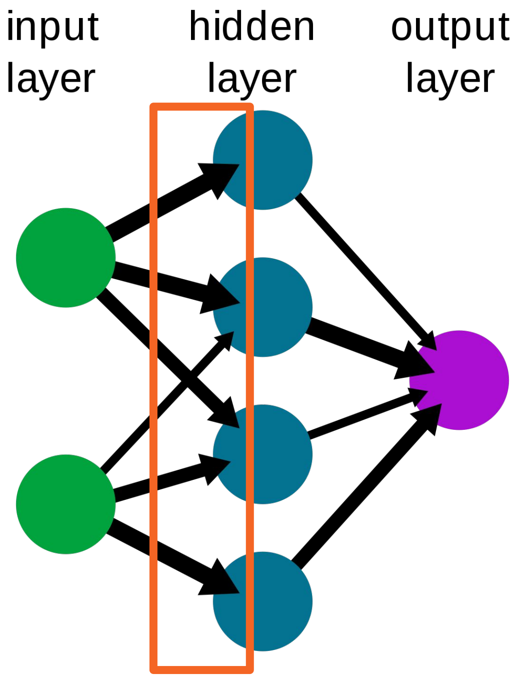
https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Perceptron (Minsky-Papert, 1969)



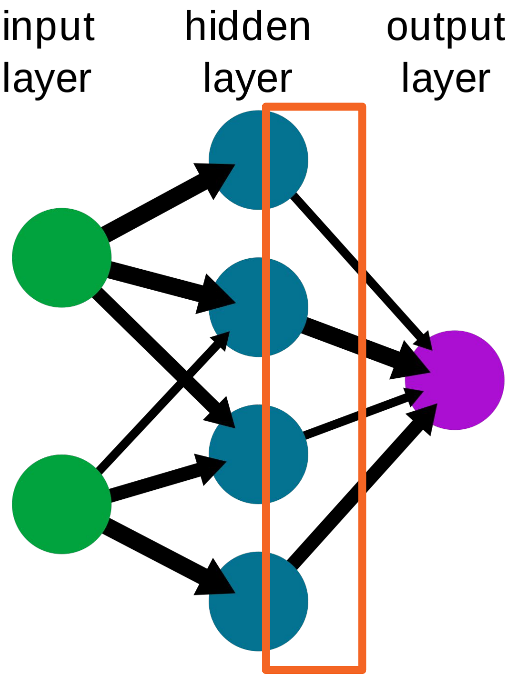
https://www.codecademy.com/learn/perceptrons-and-neural-nets-skill-path/modules/perceptrons-skill-path/cheatsheet

Single-layer perceptron



Each circle is a "node" with an input

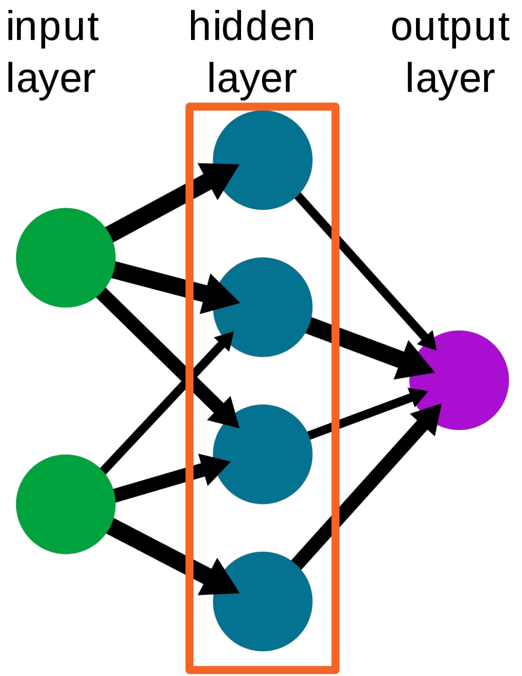
Single-layer perceptron



Each circle is a "node" with an input and an output

https://en.wikipedia.org/wiki/Neural_network

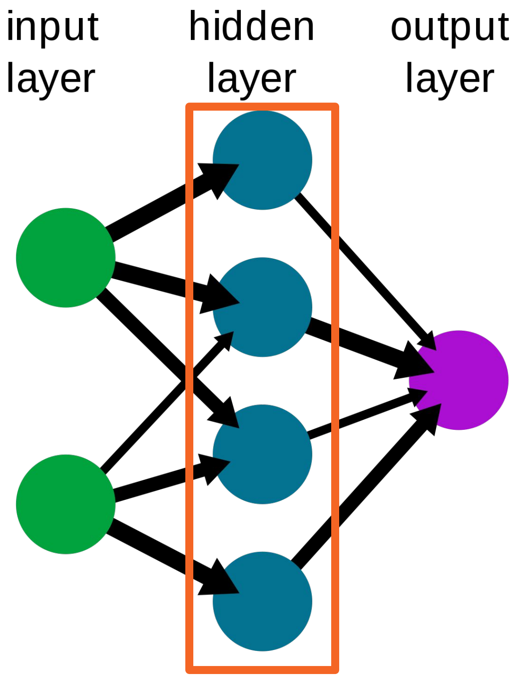
Single-layer perceptron



Each node also has a corresponding *weight* and *bias*

https://en.wikipedia.org/wiki/Neural_network

Single-layer perceptron

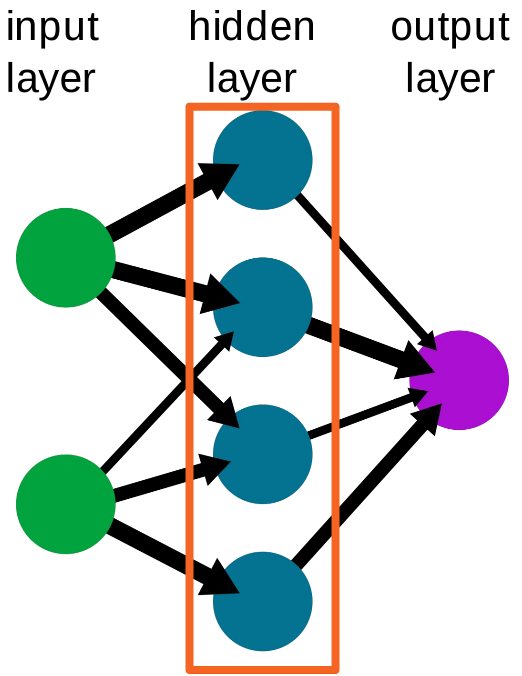


Each node also has a corresponding *weight* and *bias*

coefficient **intercept**

https://en.wikipedia.org/wiki/Neural_network

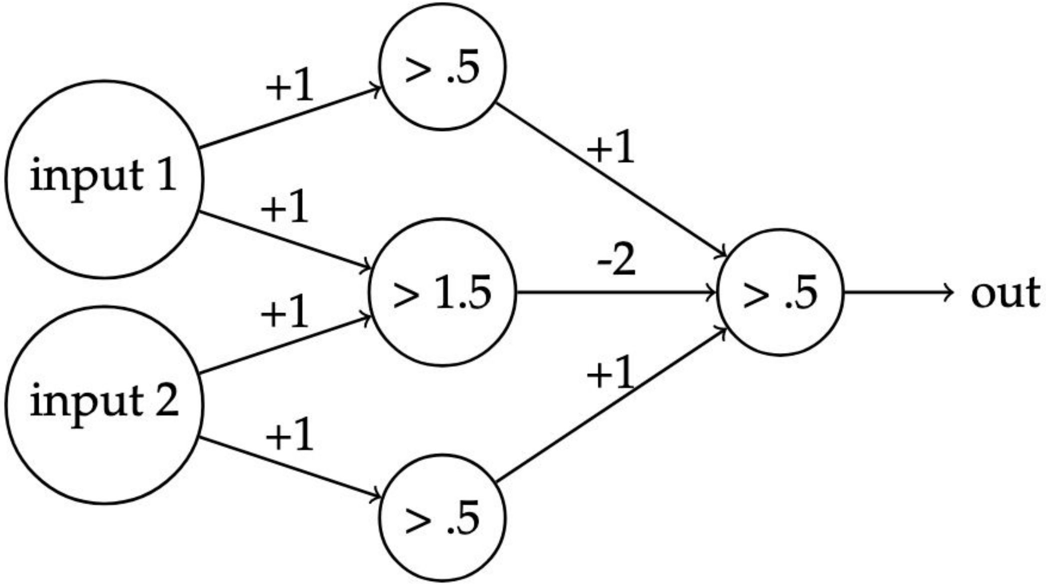
Single-layer perceptron



Think of each node as its own linear regression

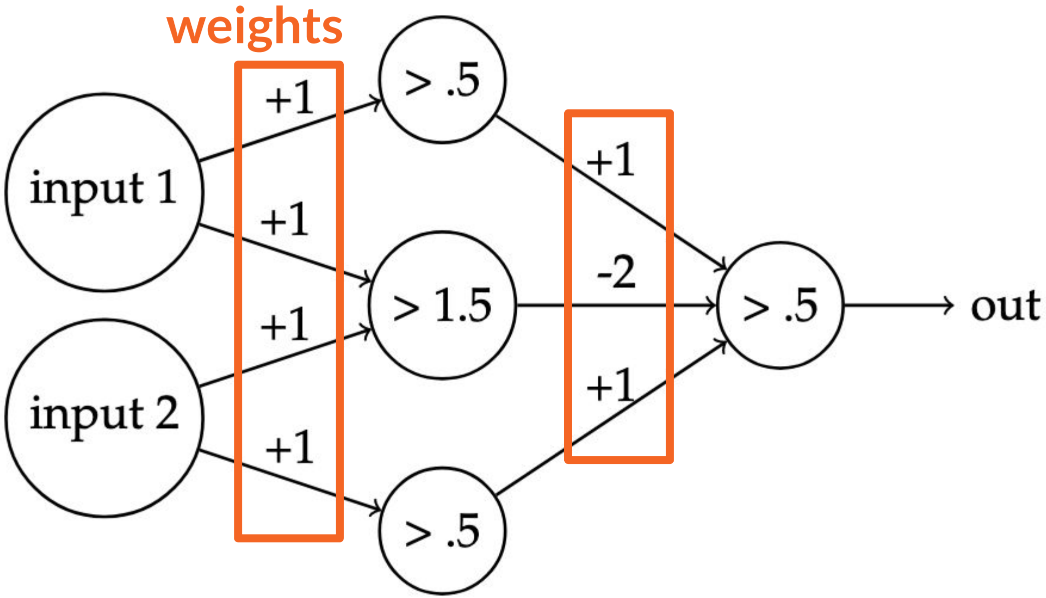
https://en.wikipedia.org/wiki/Neural_network

Multi-layer perceptron: XOR, 1985



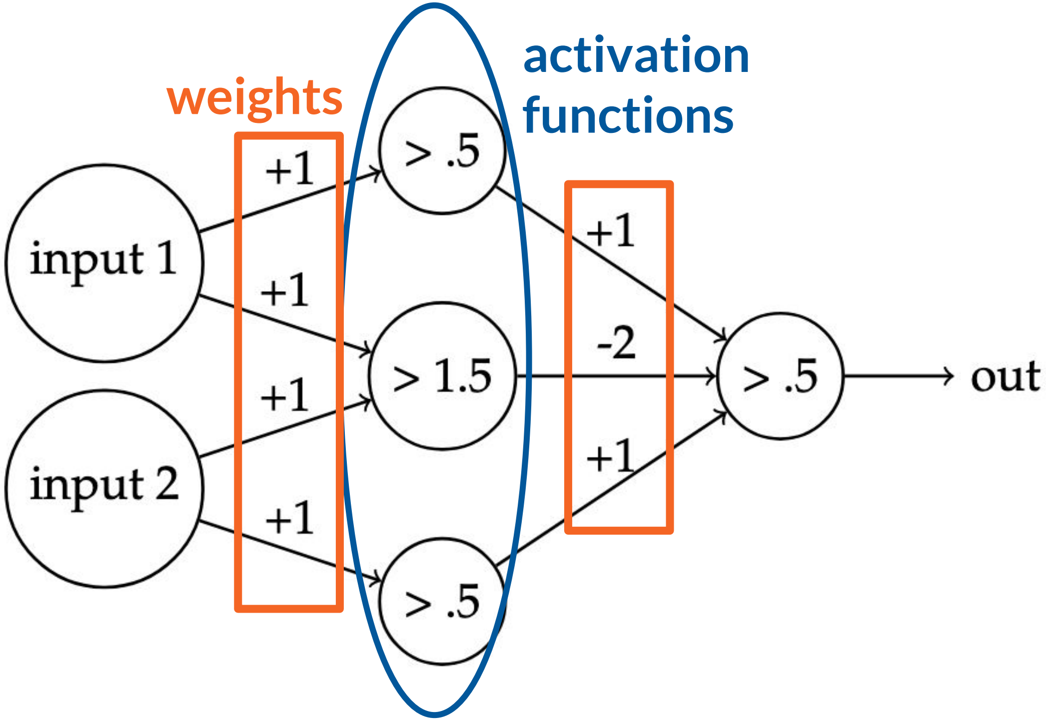
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985



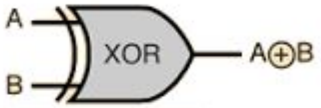
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985

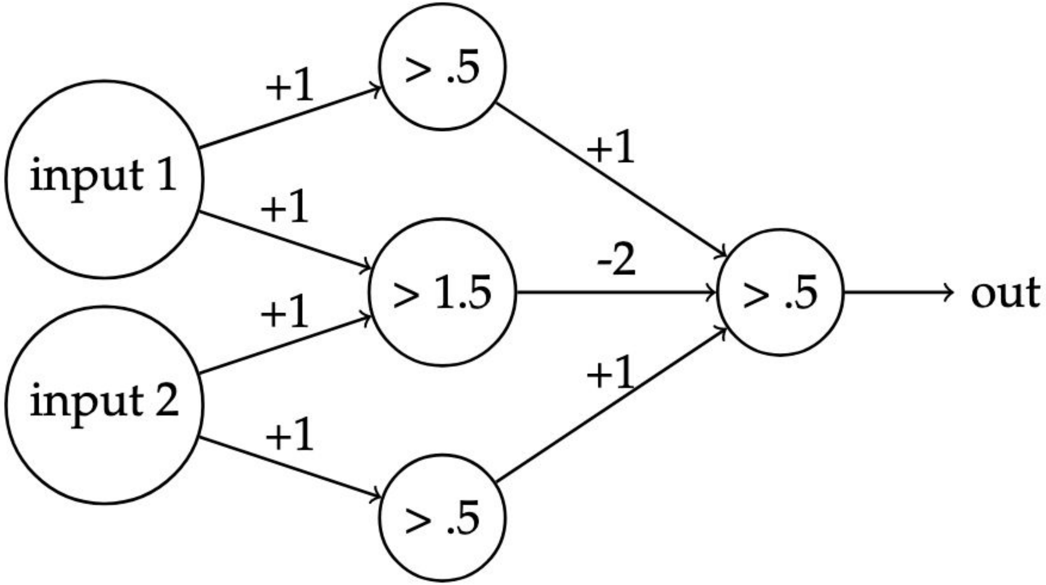


Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985

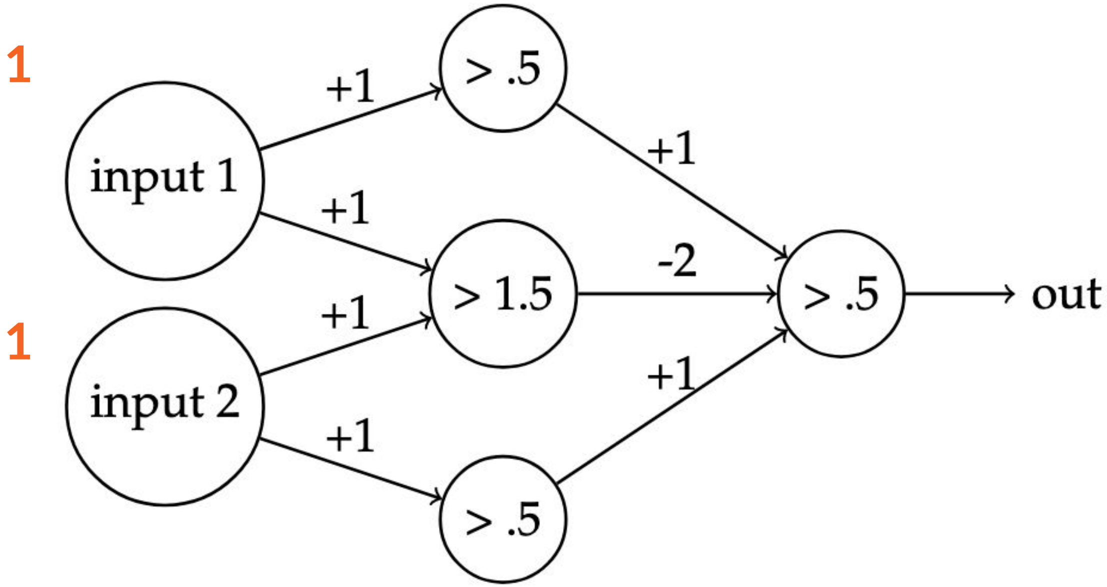


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

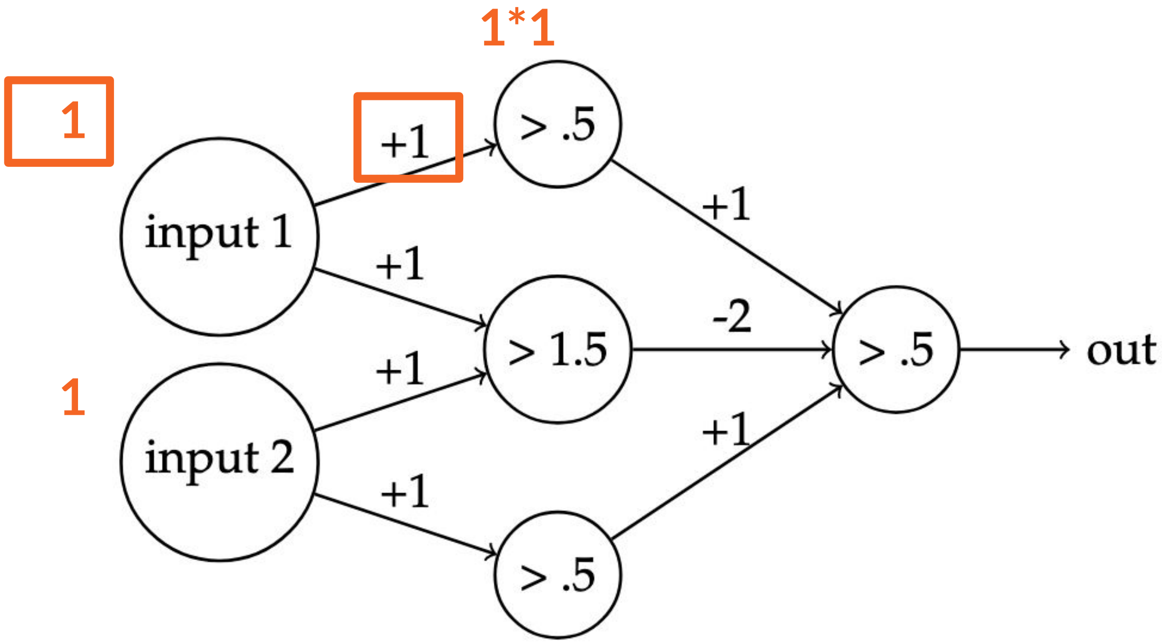


http://hyperphysics.phy-astr.gsu.edu/hbase/Electronic/xor.html Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985

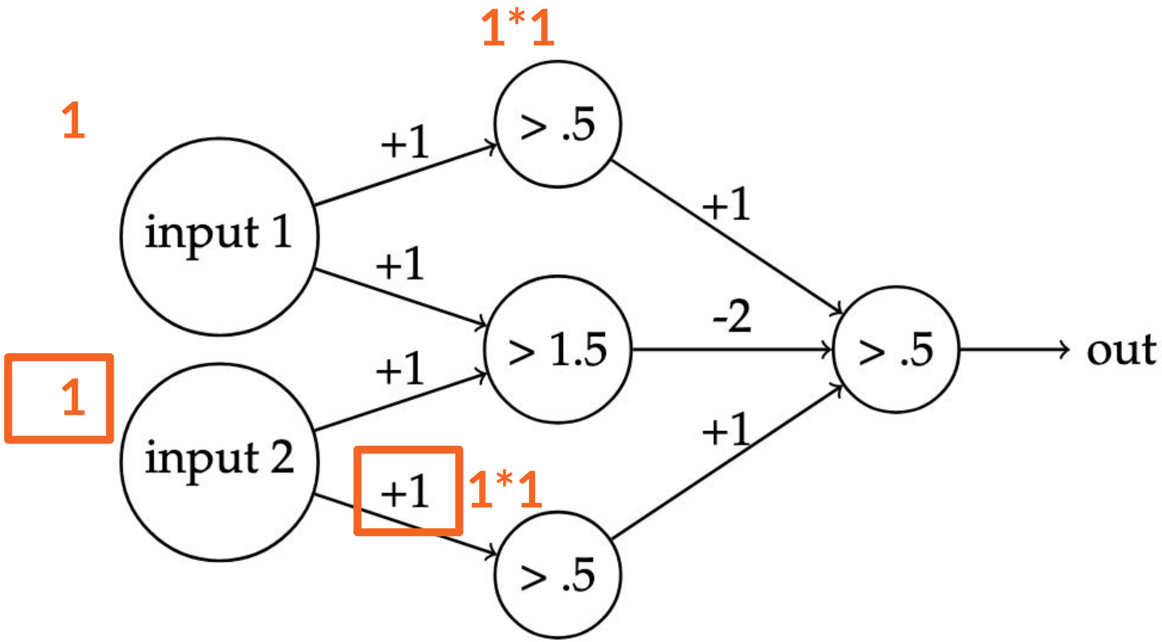


Multi-layer perceptron: XOR, 1985



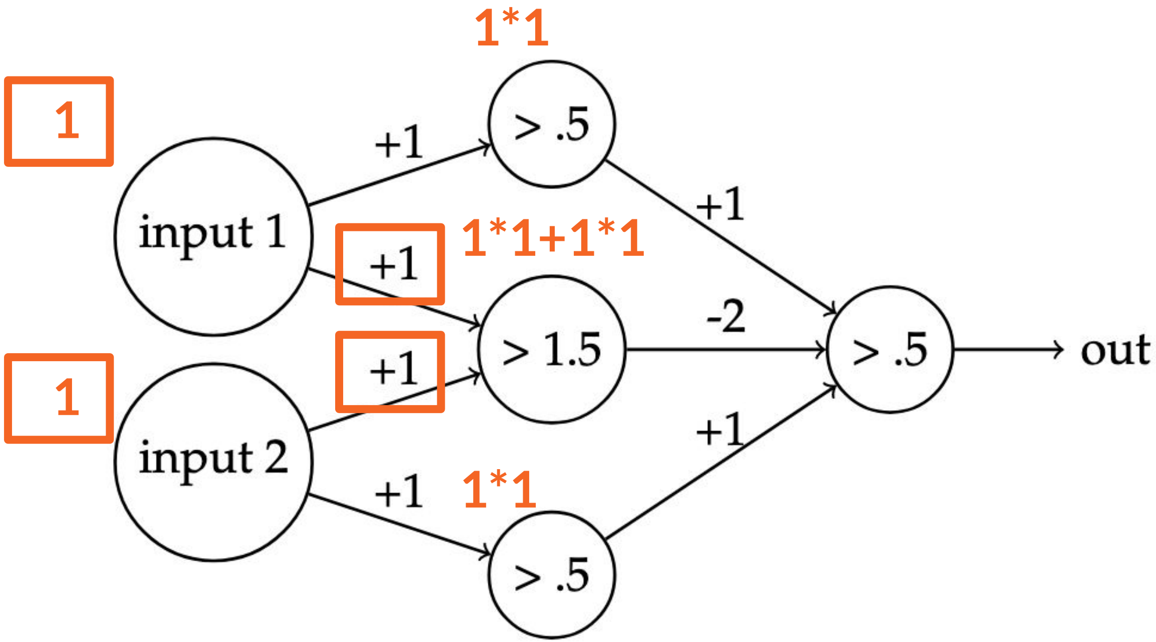
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985



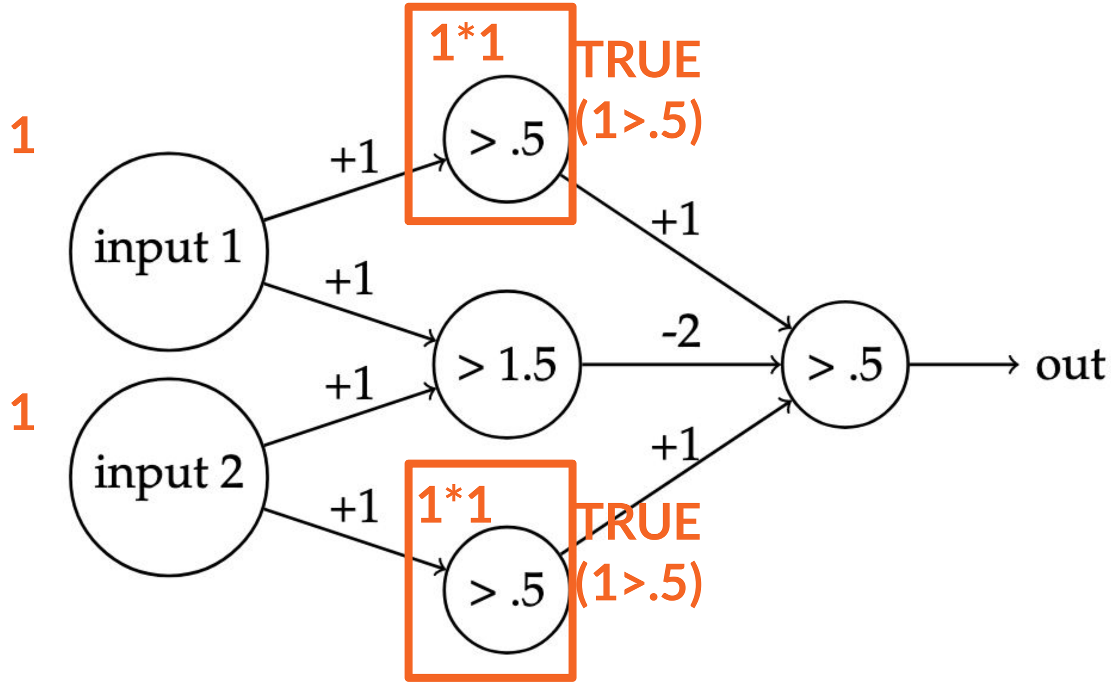
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985



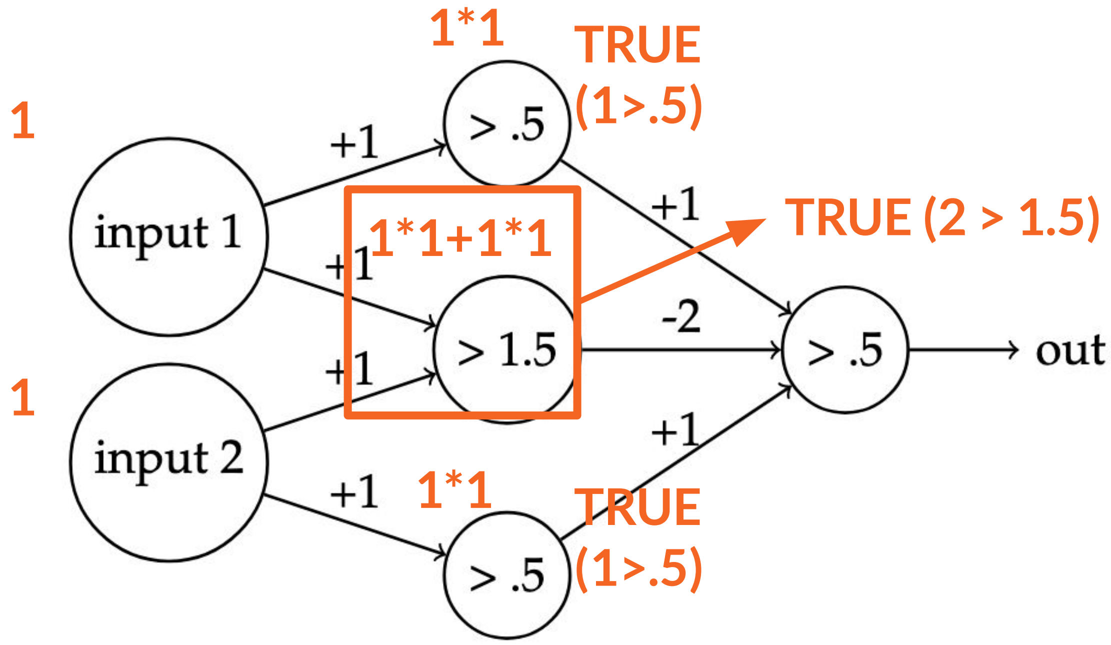
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985



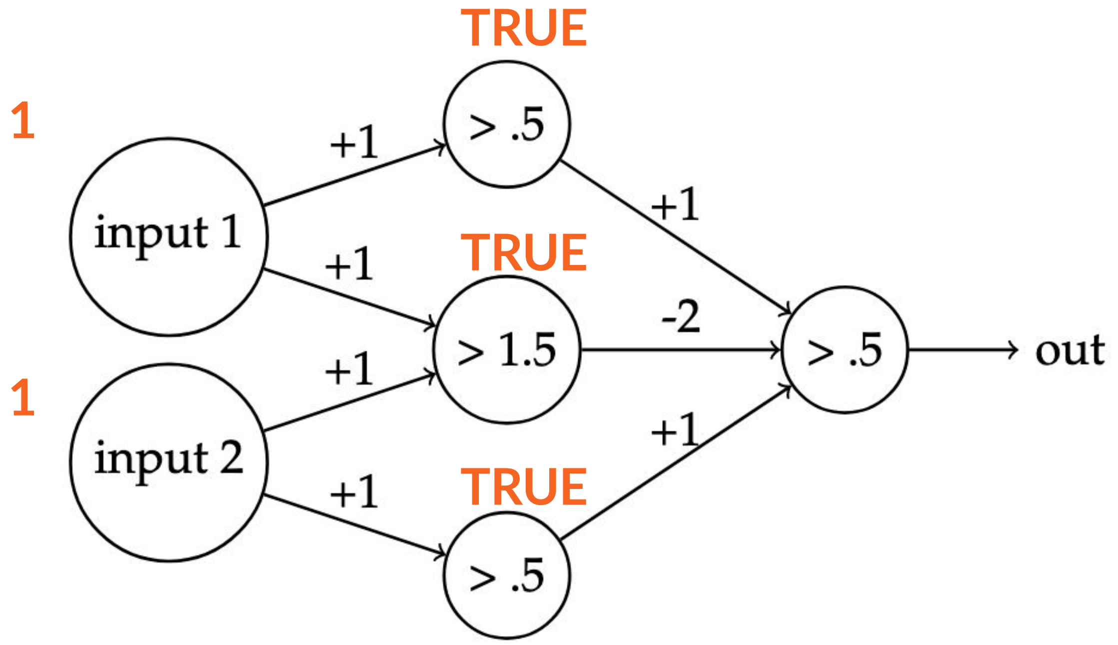
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985



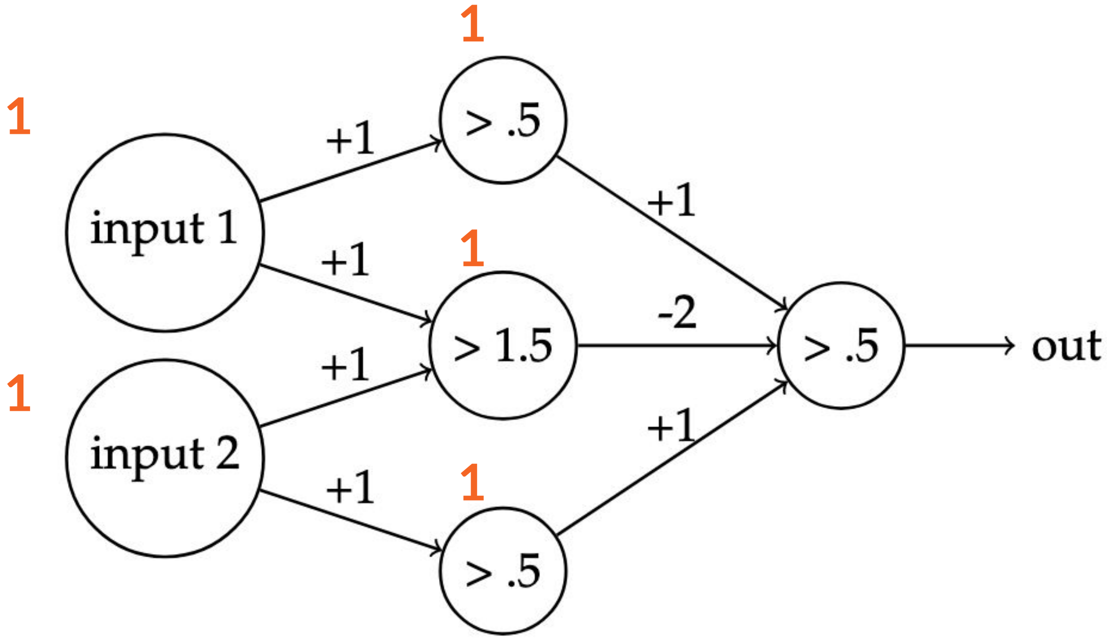
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985



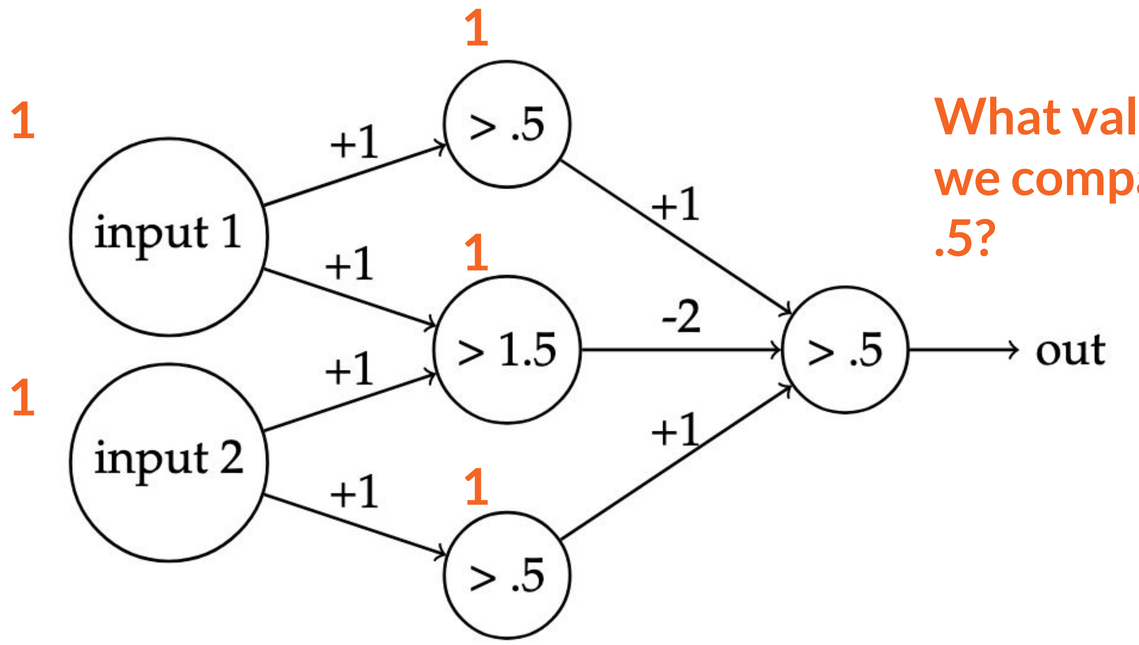
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985



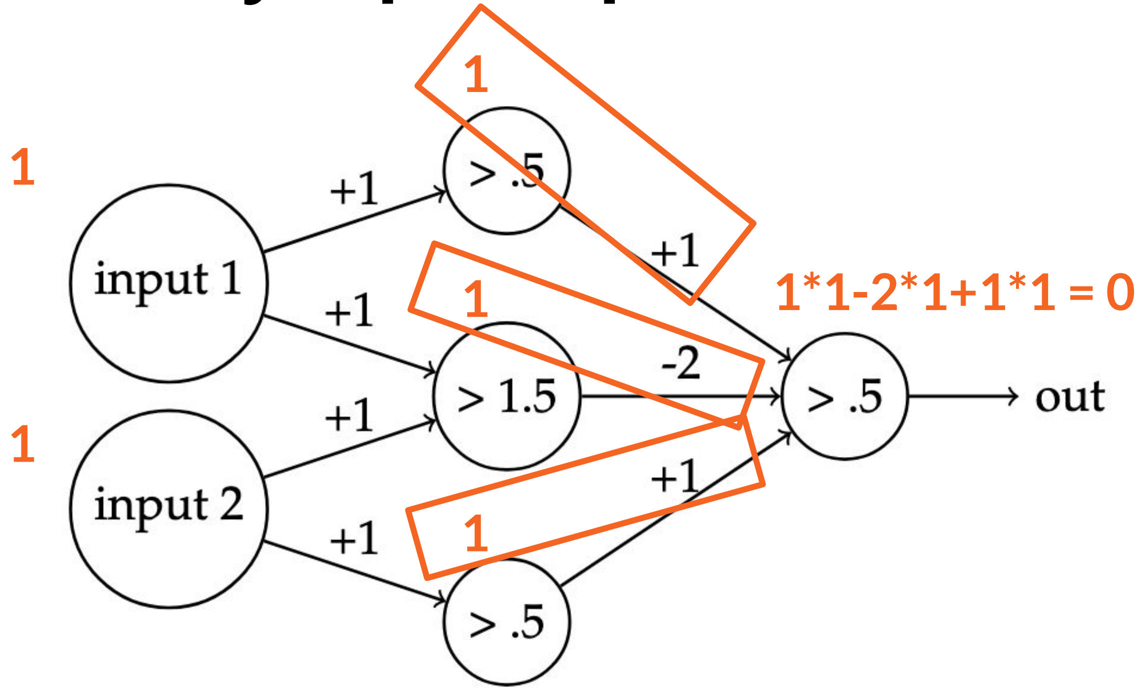
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985

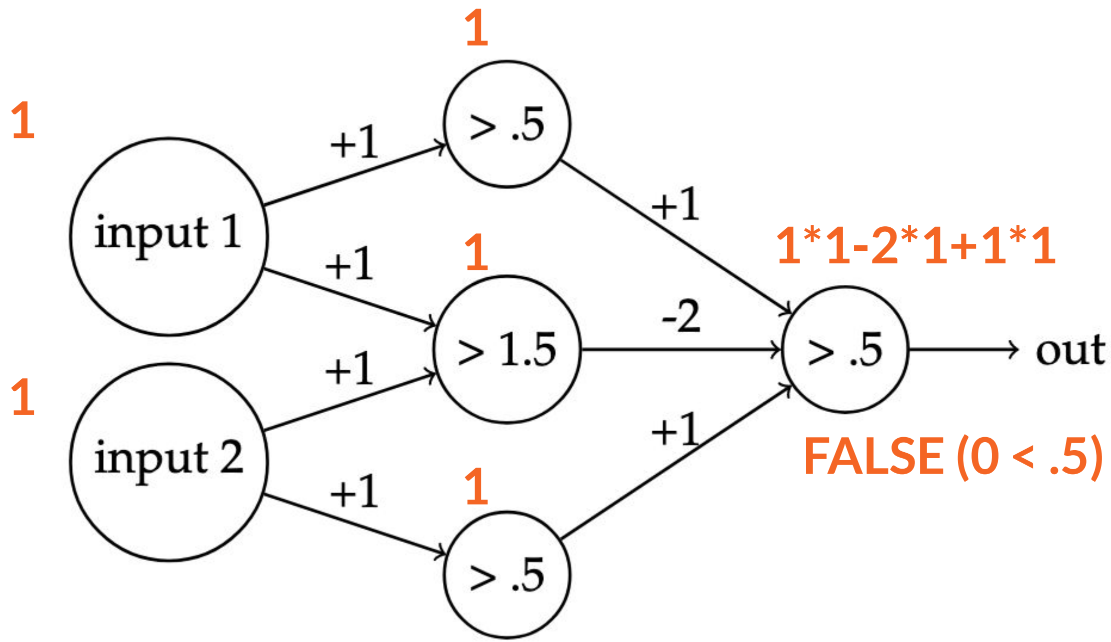


What value do we compare to .5?

Multi-layer perceptron: XOR, 1985



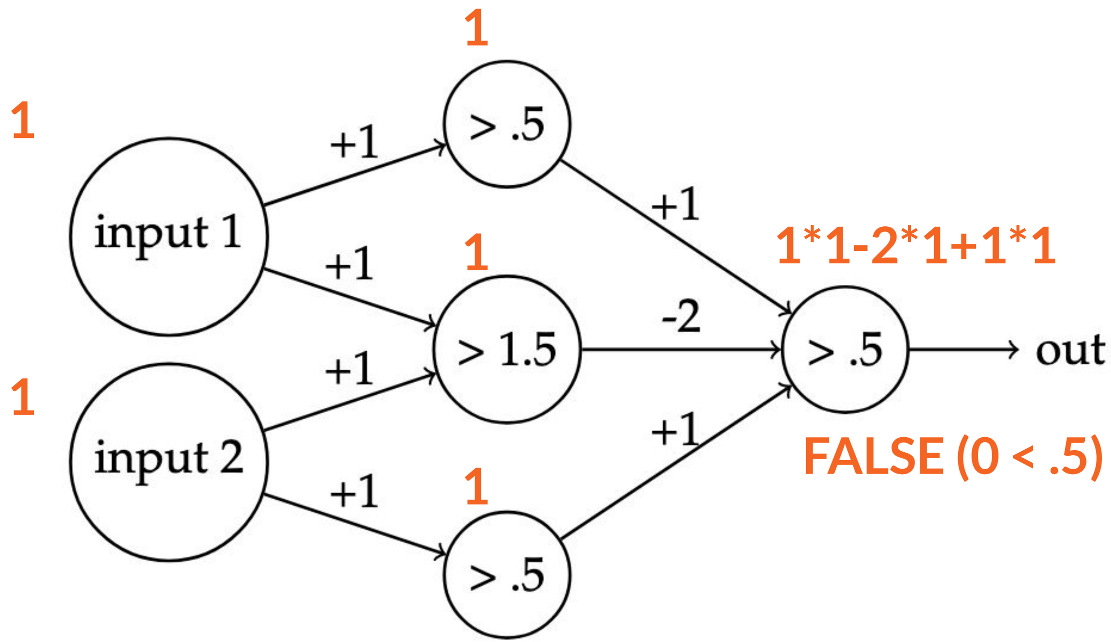
Multi-layer perceptron: XOR, 1985



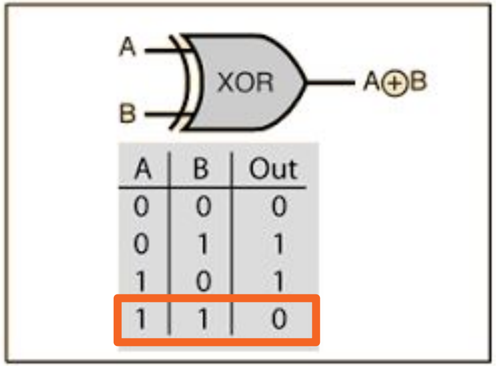
Rumelhart, Hinton & Williams. https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf

Multi-layer perceptron: XOR, 1985

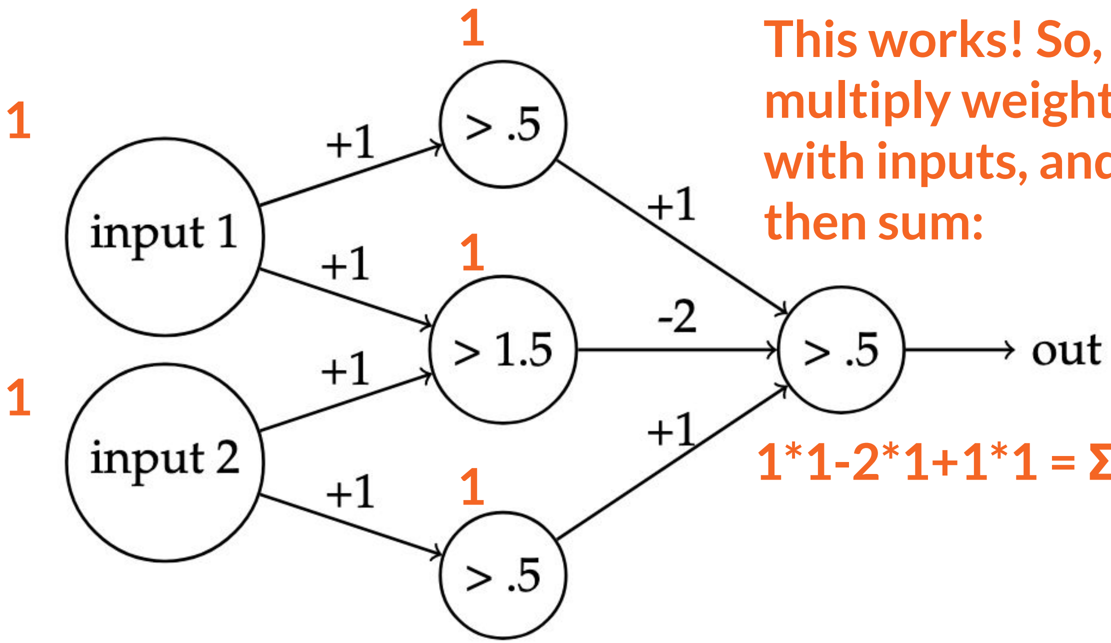
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0



Multi-layer perceptron: XOR, 1985



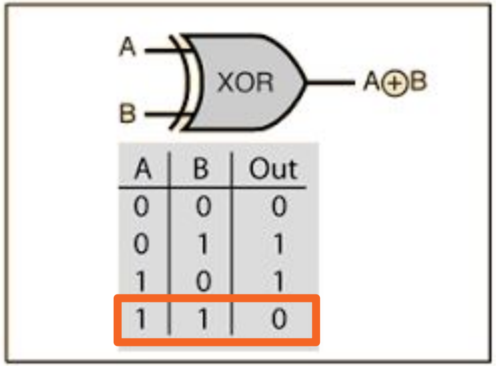
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

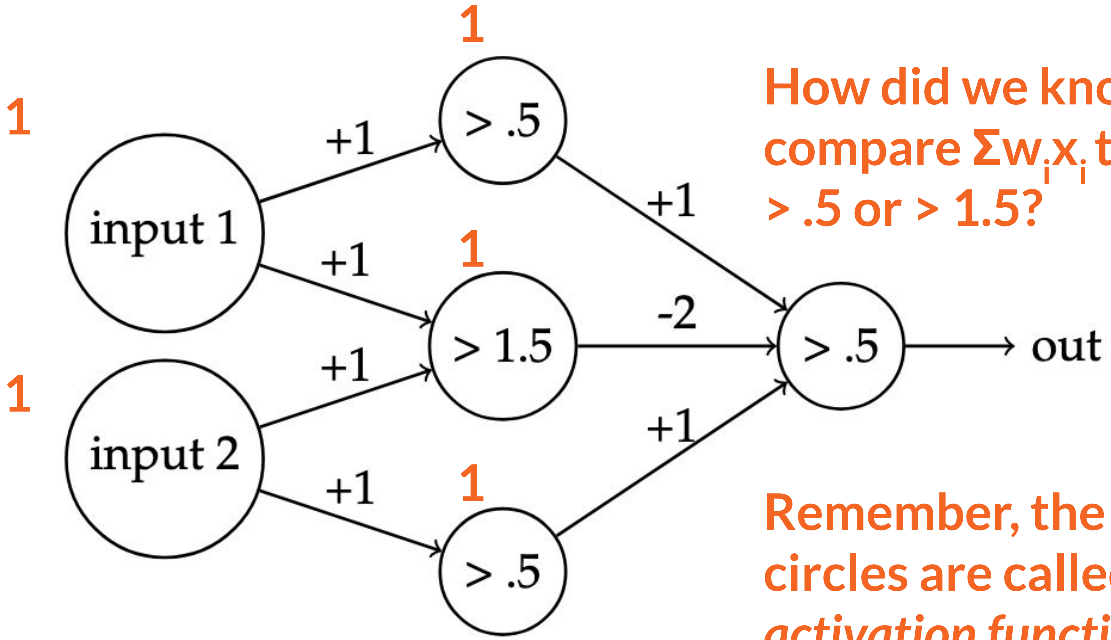


This works! So, we multiply weights with inputs, and then sum:

$1*1-2*1+1*1 = \Sigma w_i x_i$

Multi-layer perceptron: XOR, 1985

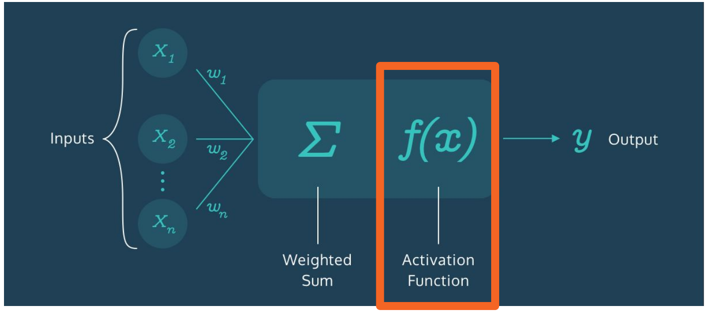




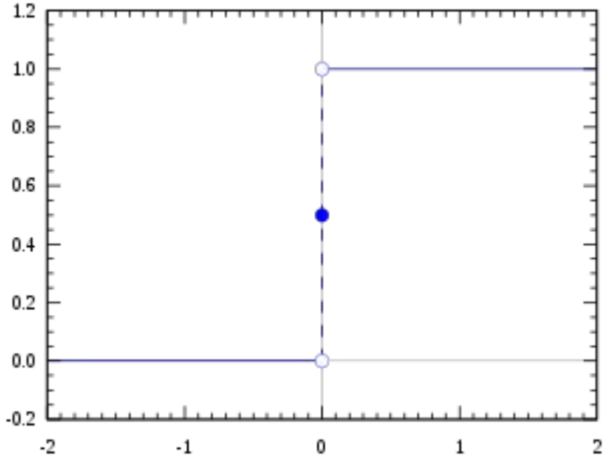
How did we know to compare $\Sigma w_i x_i$ to > .5 or > 1.5?

Remember, the circles are called *activation functions*

Common activation functions

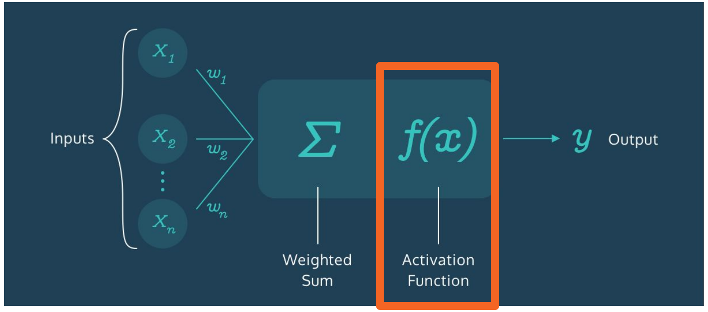


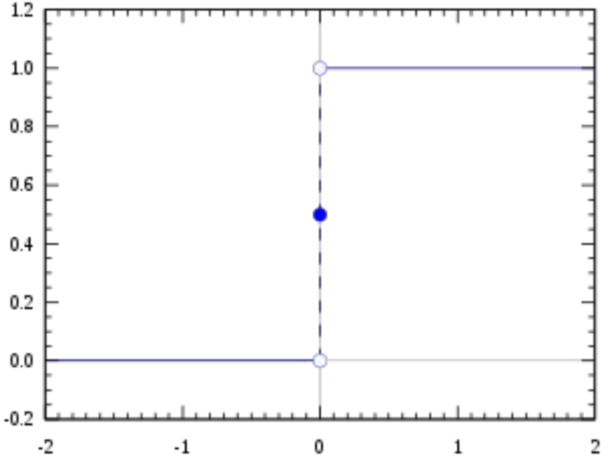
Step function: f(x) = $\begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$



https://www.codecademy.com/learn/perceptrons-and-neural-nets-skill-path/modules/perceptrons-skill-path/cheatsheet
https://en.wikipedia.org/wiki/Heaviside_step_function

Common activation functions



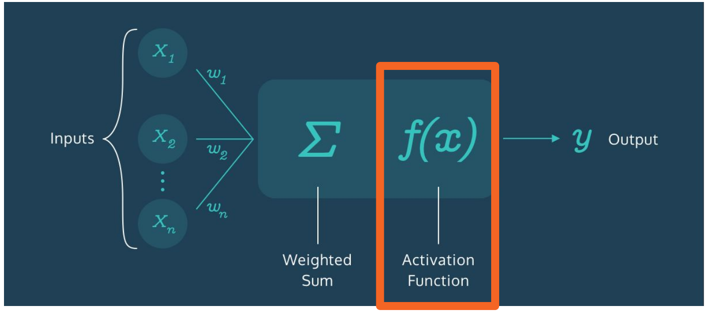


Step function: f(x) = $\begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$

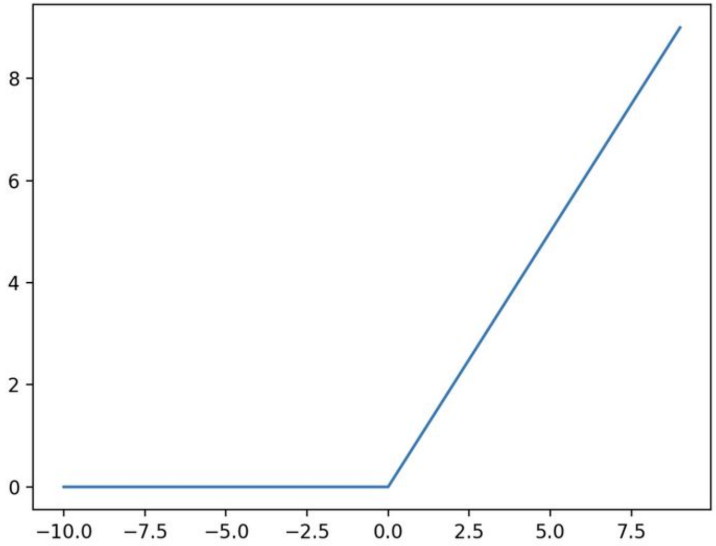
(We compared $\Sigma w_i x_i$ to being over/under .5 or 1.5 instead of over/under 0, which is just shifting a step function!) $= \begin{cases} 1 \text{ if } \Sigma w_1 x_1 \geq 0 \\ 0 \text{ if } \Sigma w_1 x_1 < 0 \end{cases}$

https://www.codecademy.com/learn/perceptrons-and-neural-nets-skill-path/modules/perceptrons-skill-path/cheatsheet
https://en.wikipedia.org/wiki/Heaviside_step_function

Common activation functions

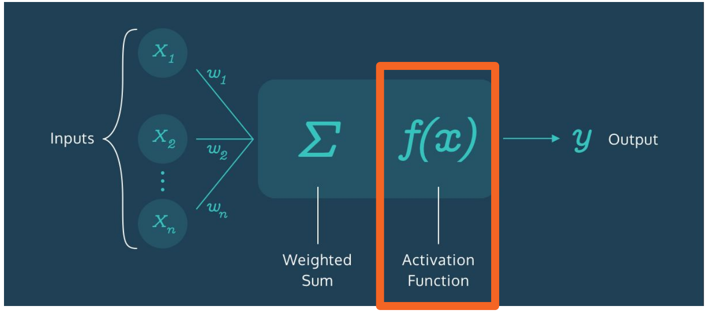


Rectified Linear Units (ReLUs): $f(x) = \max(0, x)$

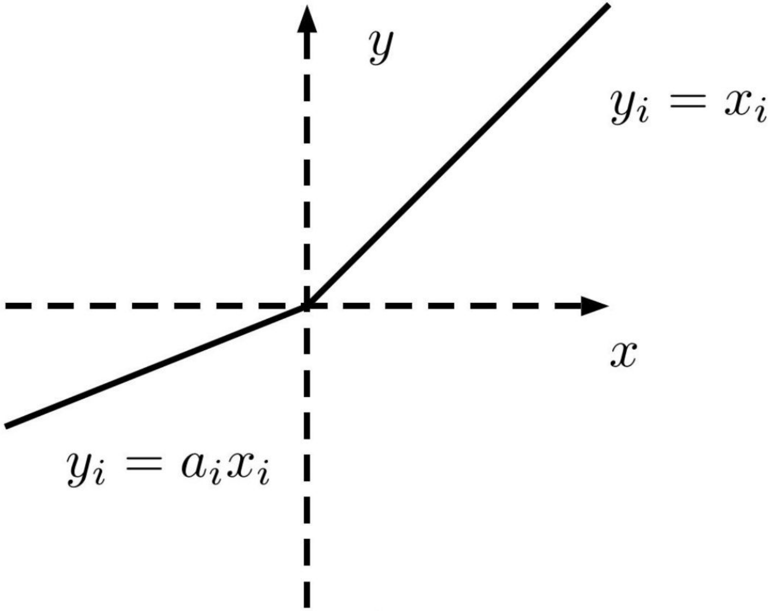


https://www.codecademy.com/learn/perceptrons-and-neural-nets-skill-path/modules/perceptrons-skill-path/cheatsheet
https://paperswithcode.com/method/relu

Common activation functions

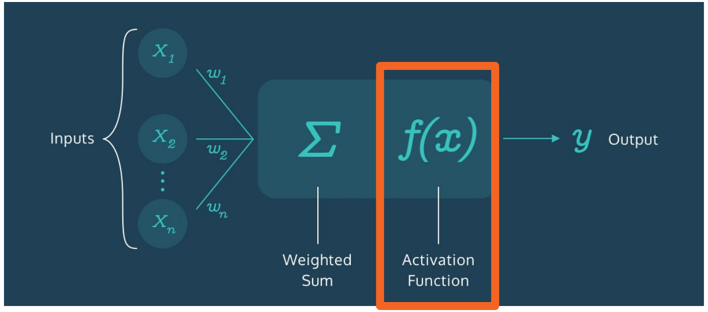


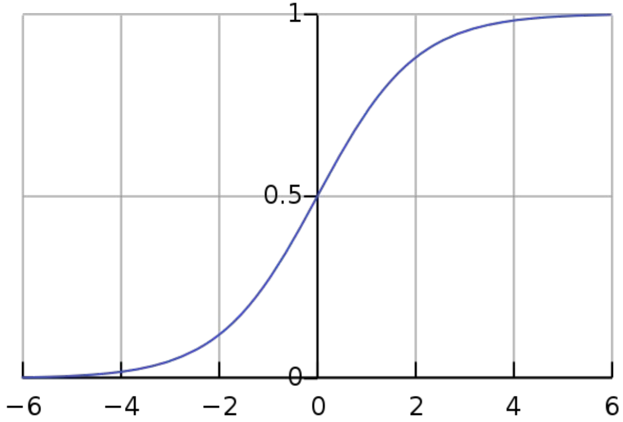
Leaky ReLU:



https://www.codecademy.com/learn/perceptrons-and-neural-nets-skill-path/modules/perceptrons-skill-path/cheatsheet
https://production-media.paperswithcode.com/methods/new_act.jpg

Common activation functions

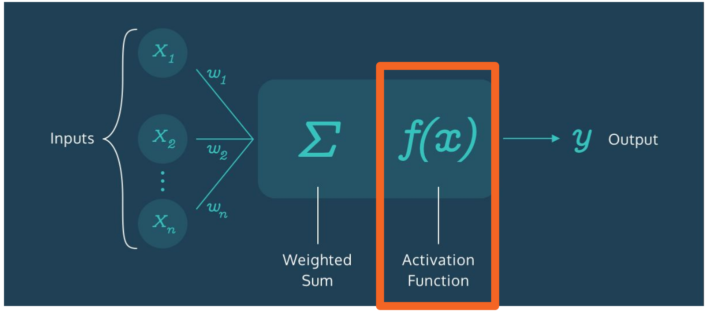


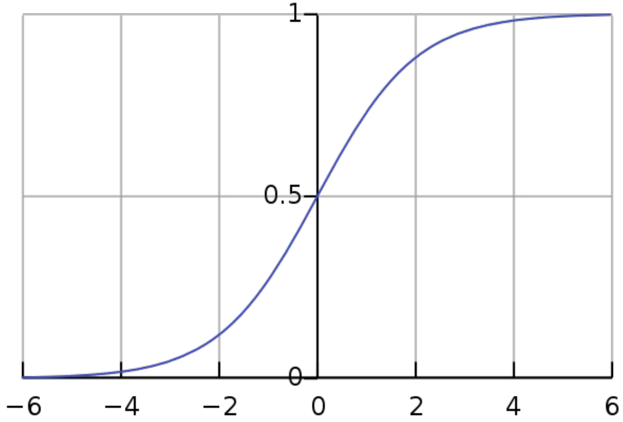


What's this function called?

https://www.codecademy.com/learn/perceptrons-and-neural-nets-skill-path/modules/perceptrons-skill-path/cheatsheet

Common activation functions



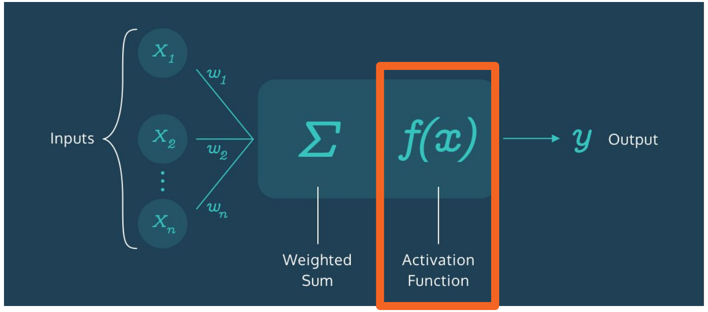


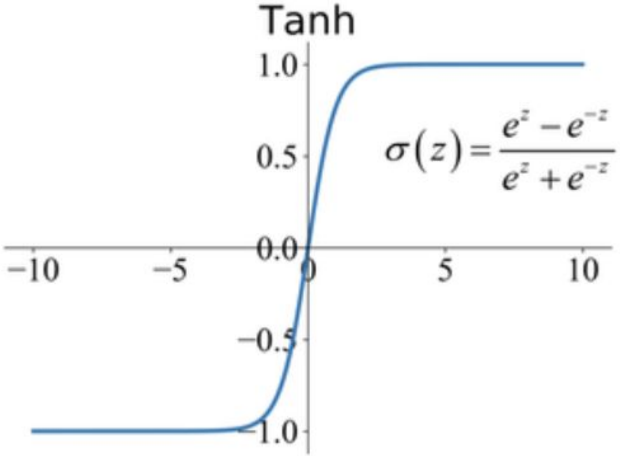
Sigmoid (logistic) activation function: $g(z) = \dfrac{1}{1 + e^{-z}}$

https://www.codecademy.com/learn/perceptrons-and-neural-nets-skill-path/modules/perceptrons-skill-path/cheatsheet

Andrew Ng, https://slideplayer.com/slide/12620833/76/

Common activation functions





Hyperbolic tangent (Tanh) looks like sigmoid, but goes from -1 to 1 instead of 0 to 1.

Neural nets

- NNs are just multi-layer perceptrons!

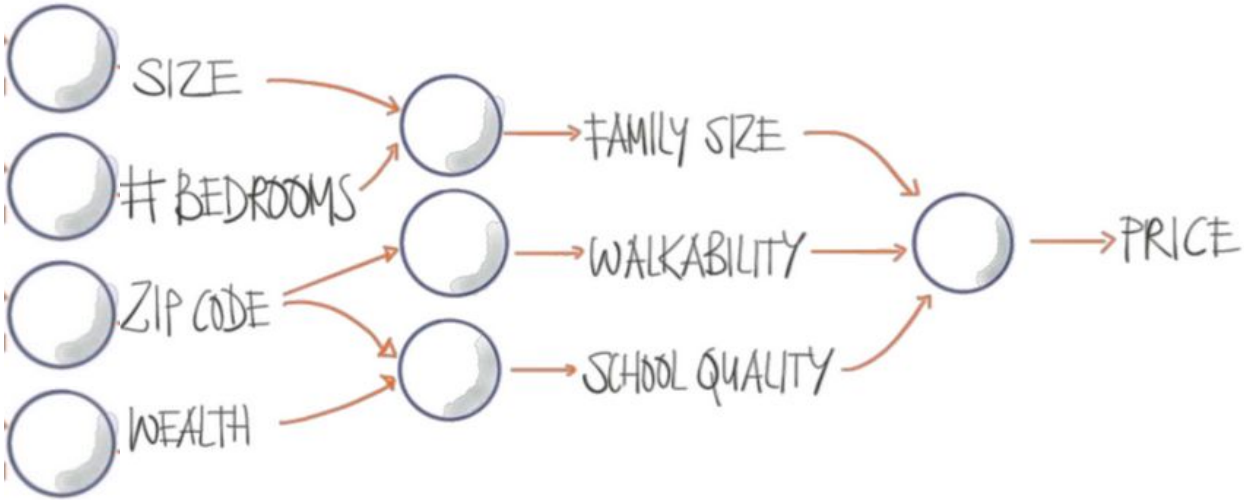
Neural nets

- NNs are just multi-layer perceptrons!
- How do you get weight and bias values?
    - Similar to regressions: “training the model” results in a weight (coefficient) and bias (intercept) for each node

Neural nets

- NNs are just multi-layer perceptrons!
- How do you get weight and bias values?
    - Similar to regressions: “training the model” results in a weight (coefficient) and bias (intercept) for each node
    - Randomly initialize your weights. Iterate:
        - **Gradient descent**: finds where to minimize loss function
        - **Backpropagation**: updates weights based on gradient of loss

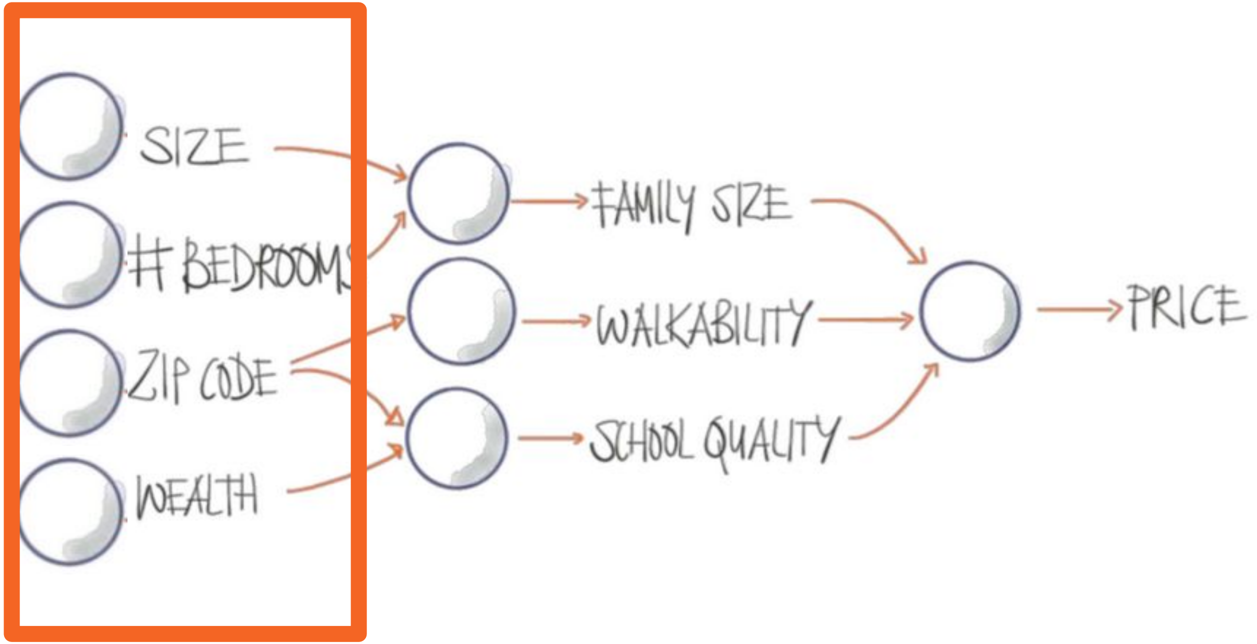
Interpreting neural nets



https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/discussion/111538

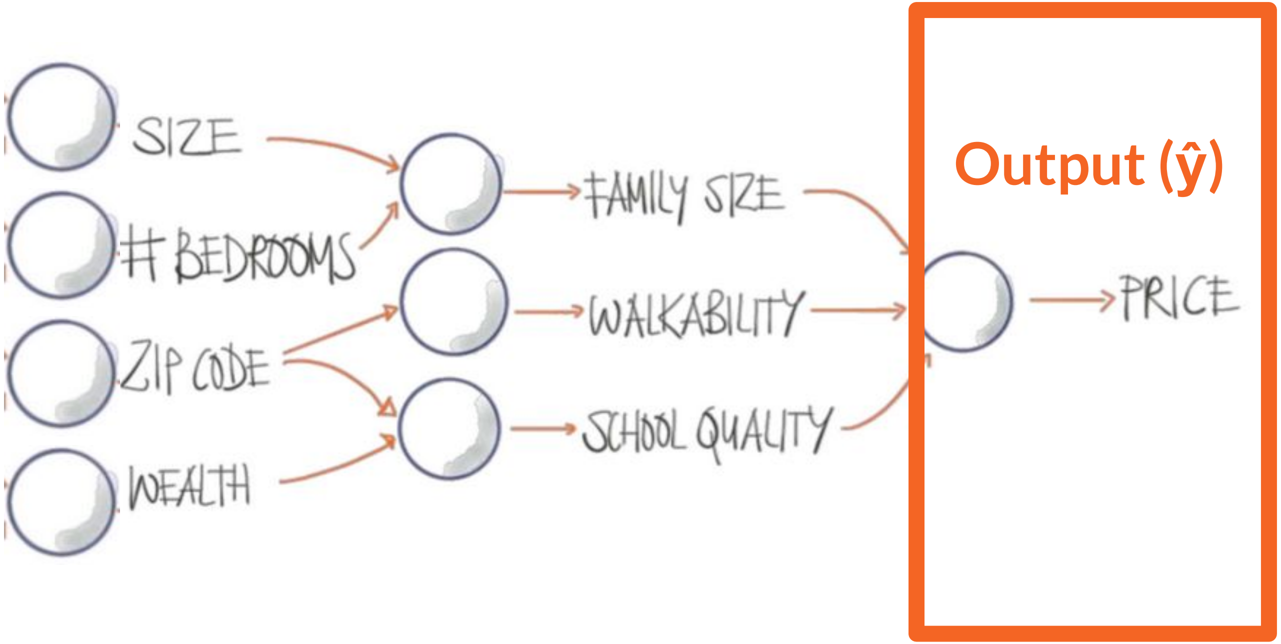
Interpreting neural nets

Input (x's):
known



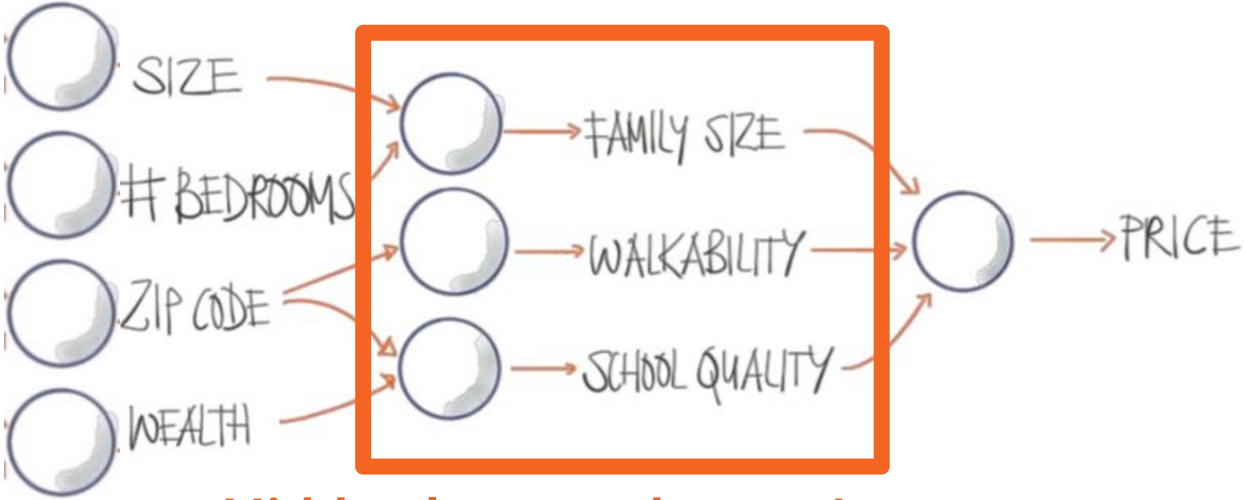
https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/discussion/111538

Interpreting neural nets



https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/discussion/111538

Interpreting neural nets



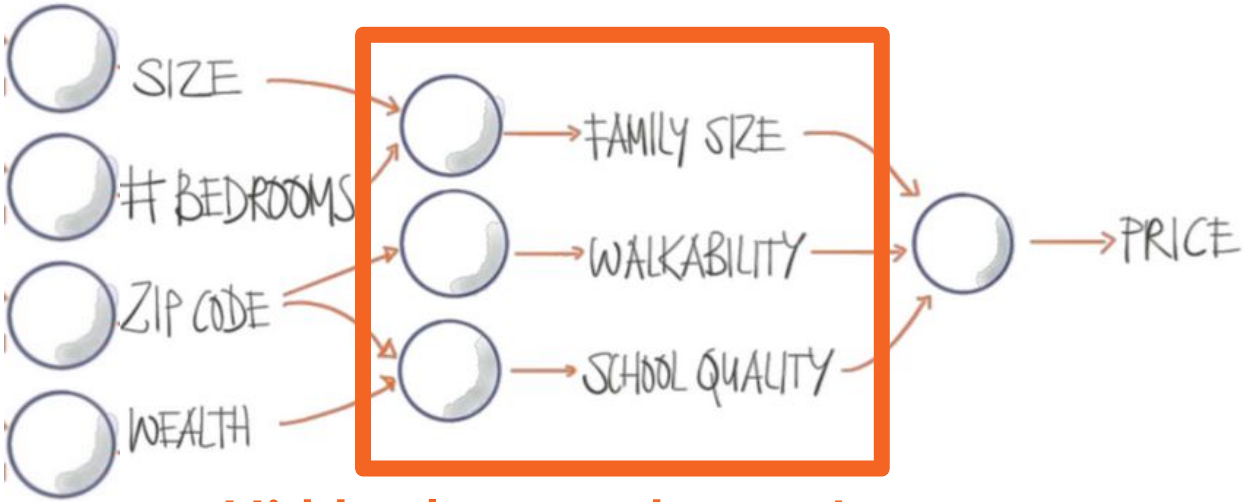
Hidden layer: unknown!

We're just guessing the meanings of each node

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/discussion/111538

Neural net layers

*Similar to how we had to guess concepts for SVD decompositions; in practice this is much harder for NNs

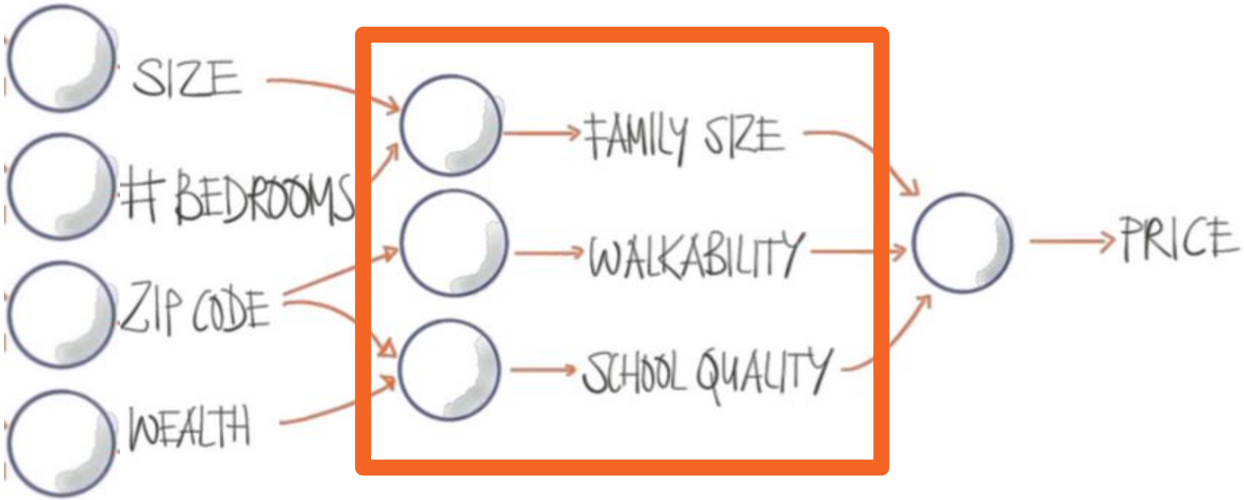


Hidden layer: unknown!

We're just guessing the meanings of each node

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/discussion/111538

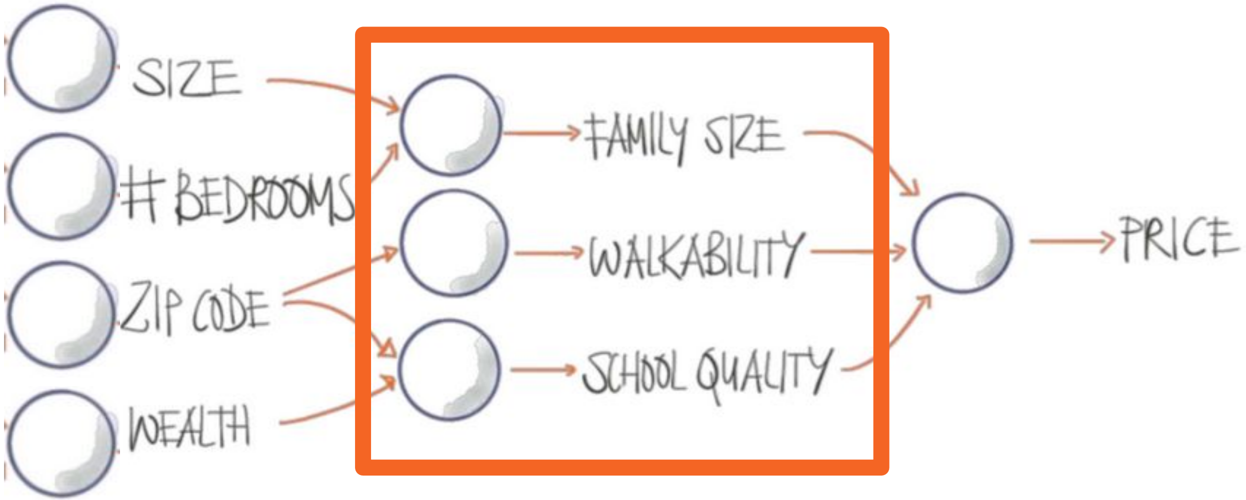
Neural net hidden layer



What we have control over: # hidden layers, # neurons in each layer, and what “activation” each layer uses

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/discussion/111538

Neural net hidden layer

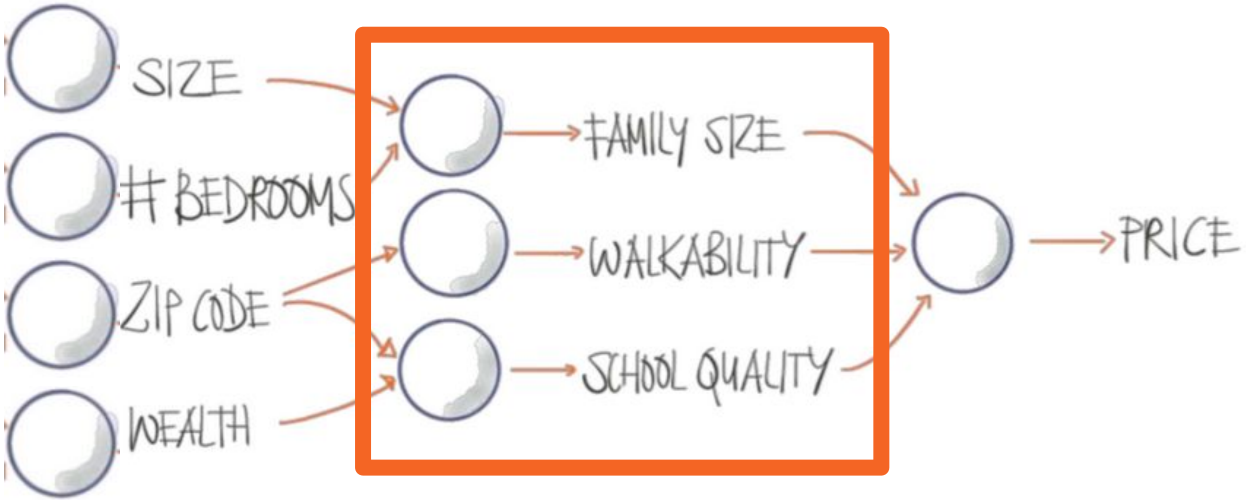


"Neural net architecture"

What we have control over: # hidden layers, # neurons in each layer, and what "activation" each layer uses

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/discussion/111538

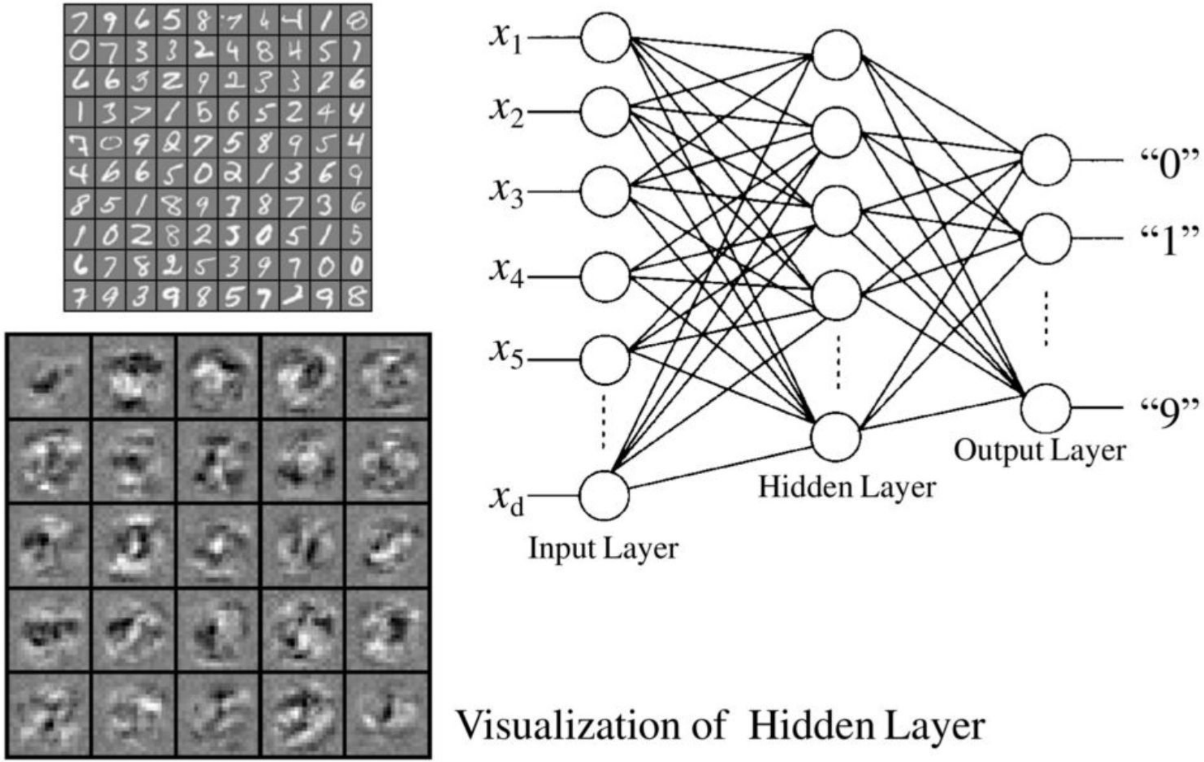
Neural net hidden layer



What we have control over: # hidden layers, # neurons in each layer, and what "activation" each layer uses

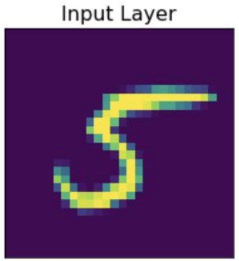
Should always be reported in your research for reproducibility!

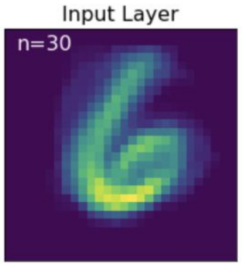
https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/discussion/111538



Slide from Andrew Ng https://slideplayer.com/slide/12620833/76/

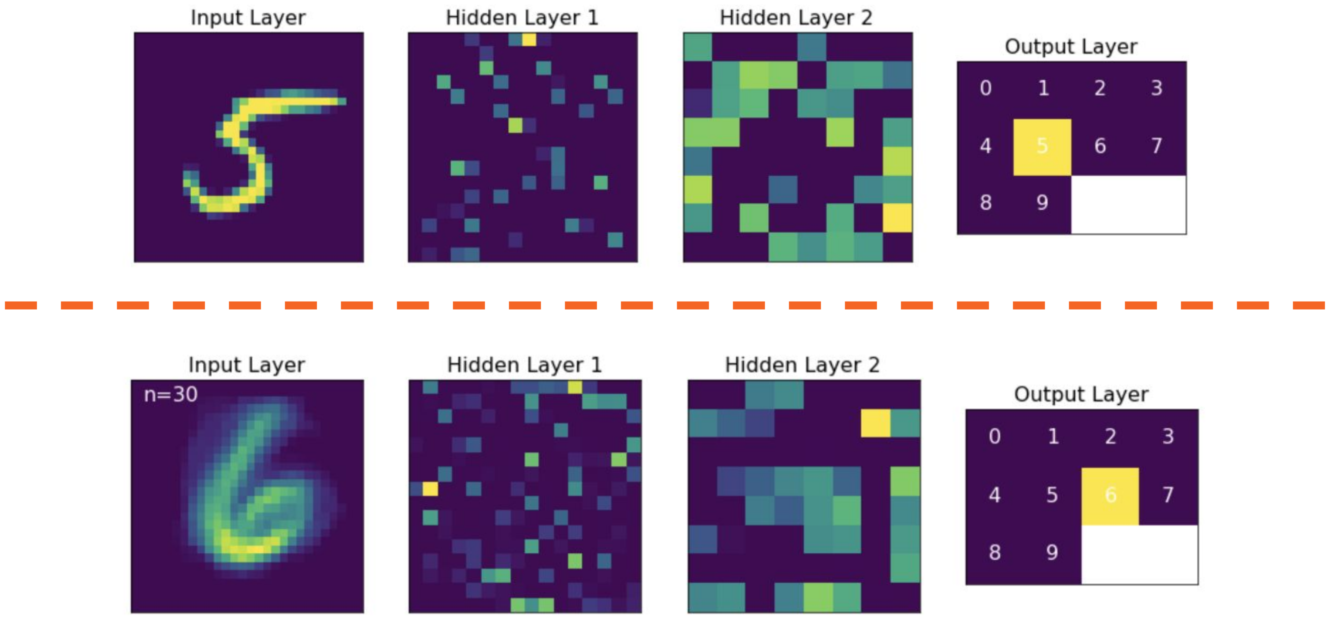
MNIST data





https://www.kaggle.com/code/mommermi/mnist-neural-network-visualization

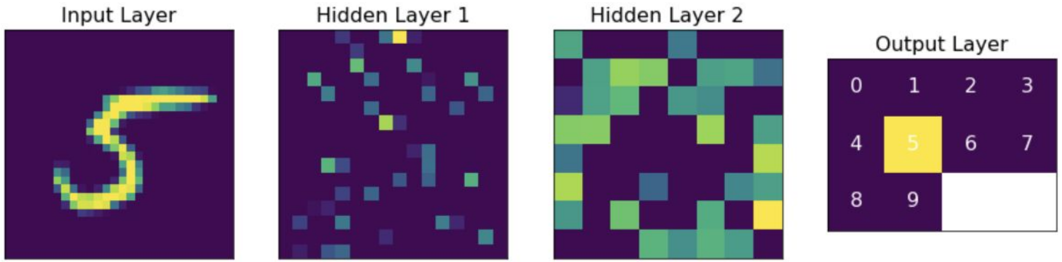
Which neurons are triggered?

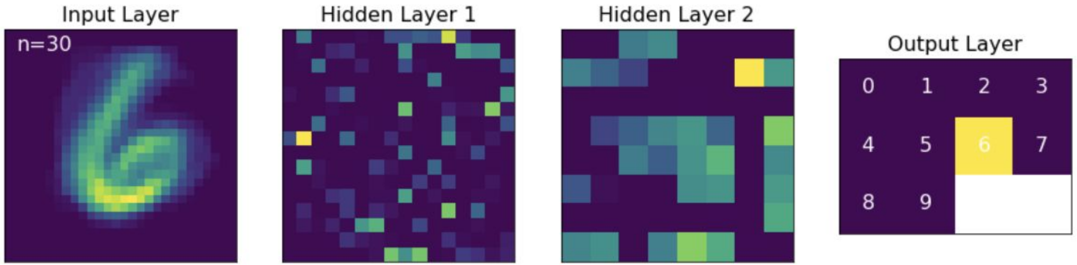


https://www.kaggle.com/code/mommermi/mnist-neural-network-visualization

Which neurons are triggered?

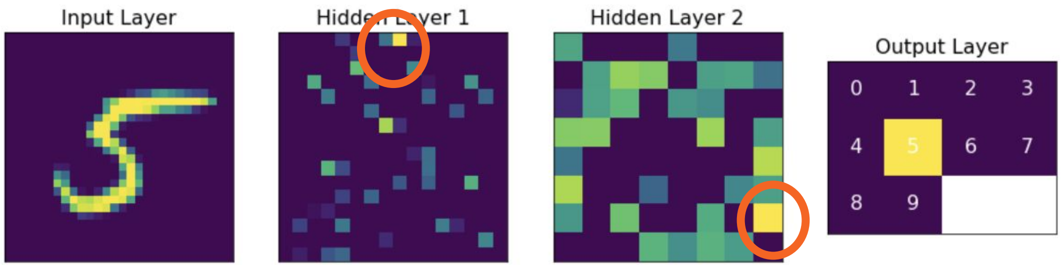
Yellow = more important
Blue = less important



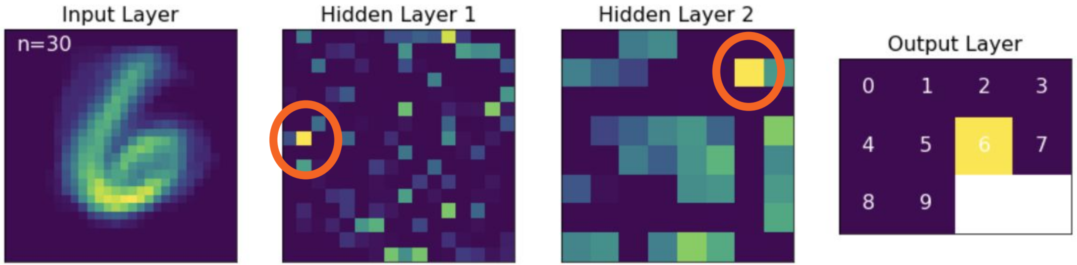


https://www.kaggle.com/code/mommermi/mnist-neural-network-visualization

Which neurons are triggered?

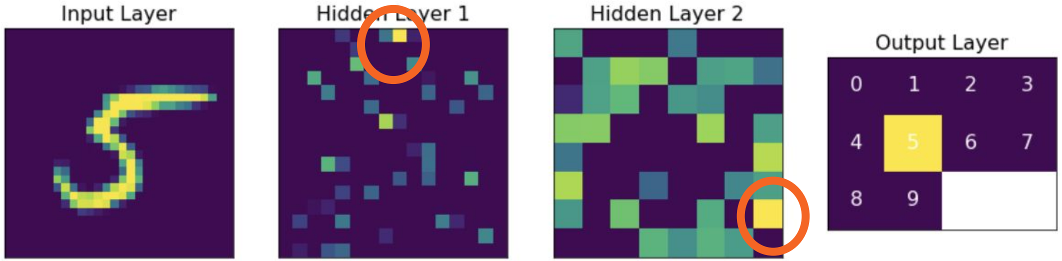


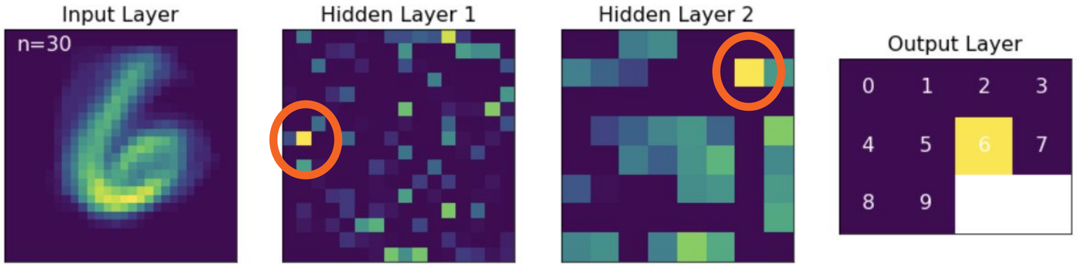
The "important" neurons are different for different inputs!



https://www.kaggle.com/code/mommermi/mnist-neural-network-visualization

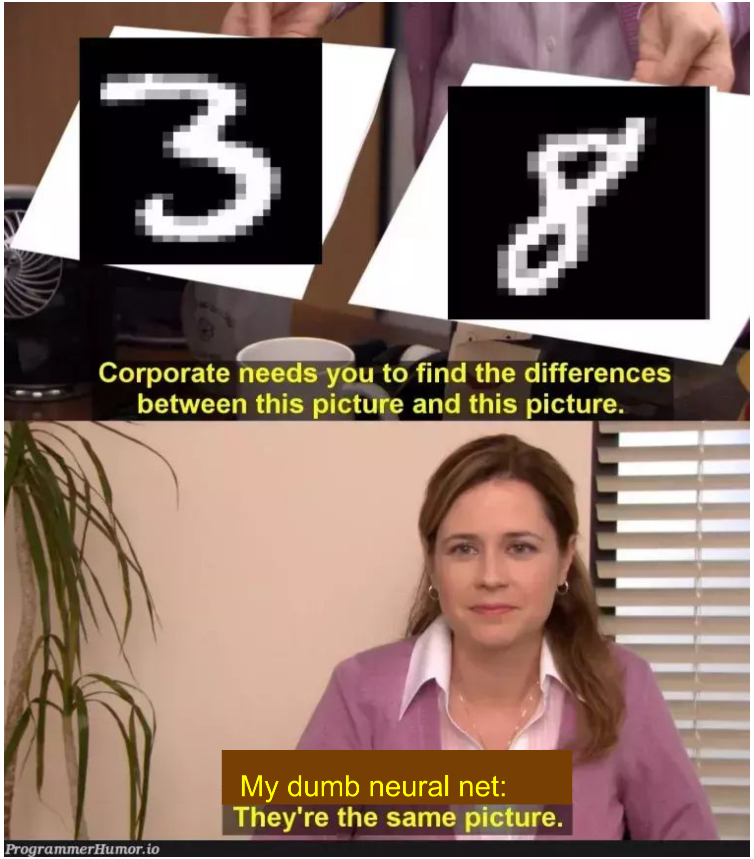
Which neurons are triggered?





We can't really tell what each hidden layer's "concept" is, but we tell the NN is using them to correctly noticed that 5!=6

https://www.kaggle.com/code/mommermi/mnist-neural-network-visualization



https://programmerhumor.io/programming-memes/neural-networks-are-difficult/

How do we do this in Python?

- *More libraries!*
    - Tensorflow (v1 and v2)
    - Keras
    - Jax
    - PyTorch
    - Theano

Running neural nets

- A lot of these steps are the same!
  - Step 1: decide on inputs / outputs
  - Step 2: data preprocessing
  - Step 3: decide on an evaluation metric
  - Step 4: split your data
  - Step 5: run the model
  - Step 6: interpret results

numpy, scipy, scikit-learn, etc.

Can use, e.g., Keras

E.g., in Keras:

Import relevant packages

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
```

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

E.g., in Keras:

Import relevant packages

Deal with your NN architecture

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
```

Deals with gradient descent

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

E.g., in Keras:

Define NN

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(1))model.compile(optimizer='Adam',loss='mes')
```

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

E.g., in Keras:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(1))model.compile(optimizer='Adam',loss='mes')
```

Define NN

Add 4 hidden layers, each with 19 neurons and ReLU activation function

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

E.g., in Keras:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(1))model.compile(optimizer='Adam',loss='mes')
```

Define NN

Add output layer (final single neuron)

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

E.g., in Keras:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(1))model.compile(optimizer='Adam',loss='mse')
```

Define NN

Add output layer (final single neuron)

Define loss function

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

E.g., in Keras:

Fit the model by taking "batches" of 128 data rows at a time, sweeping over the full data set 400 times ("epochs")

```
model.fit(x=X_train,y=y_train,
          validation_data=(X_test,y_test),
          batch_size=128,epochs=400)
model.summary()
```

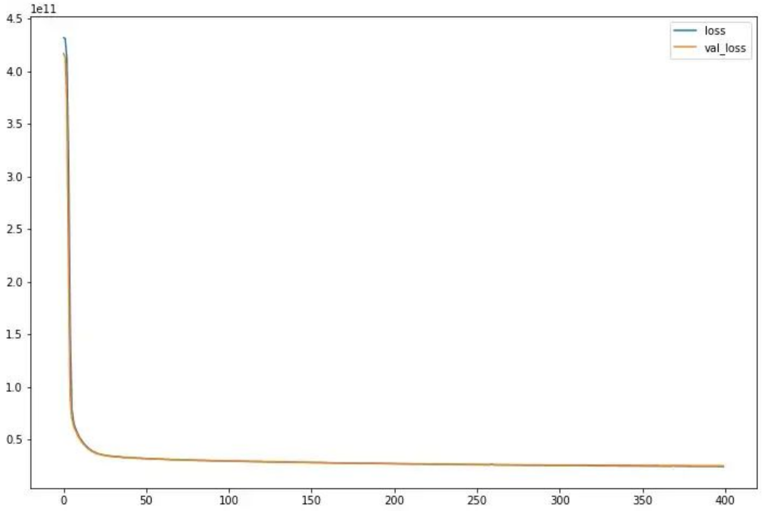
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                multiple                  418
_________________________________________________________________
dense_1 (Dense)              multiple                  380
_________________________________________________________________
dense_2 (Dense)              multiple                  380
_________________________________________________________________
dense_3 (Dense)              multiple                  380
_________________________________________________________________
dense_4 (Dense)              multiple                  20
=================================================================
Total params: 1,578
Trainable params: 1,578
Non-trainable params: 0
```

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

E.g., in Keras:

```
loss_df = pd.DataFrame(model.history.history)
loss_df.plot(figsize=(12,8))
```

Plot loss over epochs



https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

E.g., in Keras:

```
y_pred = model.predict(X_test)
```

Predict y-hats

```
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print('VarScore:',metrics.explained_variance_score(y_test,y_pred))
```

Use sklearn to check evaluation metrics

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

Interpret: Regression vs. NN

Which is better?

Note: "Variance score" refers to "explained variance score" (similar to R^2)

Model: Keras Neural Net

```
Mean Absolute Error(MAE): 96667.89
Mean Squared Error(MSE): 24912134897.75
Root Mean Squared Error(RMSE): 157835.78
Variance score: 80.84
```

```
*********************************
```

Model: Sklearn Multiple Linear Regression

```
Mean Absolute Error(MAE): 124516.17
Mean Squared Error(MSE):39763621927.16
Root Mean Squared Error(RMSE):199408.18
Variance score: 69.42
```

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

Interpret: Regression vs. NN

Lower error and higher explainability is better!

Model: Keras Neural Net

```
Mean Absolute Error(MAE): 96667.89
Mean Squared Error(MSE): 24912134897.75
Root Mean Squared Error(RMSE): 157835.78
Variance score: 80.84

*********************************
```

Model: Sklearn Multiple Linear Regression

```
Mean Absolute Error(MAE): 124516.17
Mean Squared Error(MSE):39763621927.16
Root Mean Squared Error(RMSE):199408.18
Variance score: 69.42
```

https://towardsdatascience.com/house-prices-prediction-using-deep-learning-dea265cc3154

Make sure to think about the broader implications of a poorly-performing model!

People with no idea about AI, telling me my AI will destroy the world

Me wondering why my neural network is classifying a cat as a dog…

https://journeys.dartmouth.edu/folklorearchive/2020/06/03/title-ai-cant-classify-a-cat/

“Parameters” in machine learning

- Model **parameters** are the α/βs/weights that are assigned to each variable
  - These are internally set through the model’s learning process

“Parameters” in machine learning

- Model **parameters** are the α/βs/weights that are assigned to each variable
    - These are internally set through the model’s learning process
- **Hyperparameters** are specific model settings
    - These are established prior to model learning/training
    - Hyperparameters do not change during model training; they guide training

Hyperparameters

- Hyperparameters are the ***arguments*** you can pass into a model when instantiating it
- There can be **many** hyperparameters for a single model

Hyperparameters

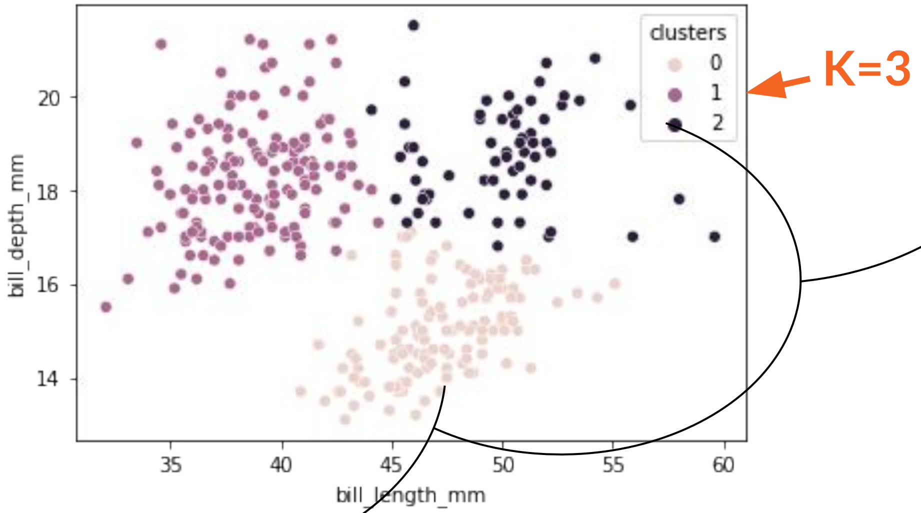
- Hyperparameters are the ***arguments*** you can pass into a model when instantiating it
- There can be **many** hyperparameters for a single model
- Choosing the best hyperparameters is an important part of the machine learning process
- Different hyperparameters can affect each other
    - Sort of like interaction effects, but with model performance!

Hyperparameters

Can you think of any hyperparameters we've already encountered?

- Hyperparameters are the ***arguments*** you can pass into a model when instantiating it
- There can be **many** hyperparameters for a single model
- Choosing the best hyperparameters is an important part of the machine learning process
- Different hyperparameters can affect each other
    - Sort of like interaction effects, but with model performance!

Output: k-means clustering



Examples of hyperparameters

- K-Means Clustering:
  - Number of clusters
  - `clustering = KMeans(n_clusters=5)`
- TF-IDF
  - Max/min number of documents a word can appear in
  - `tfidf_vectorizer = TfidfVectorizer(min_df=5)`
- SGD
  - Learning rate
- Neural networks
  - Model architecture (hidden layers, nodes, etc.)

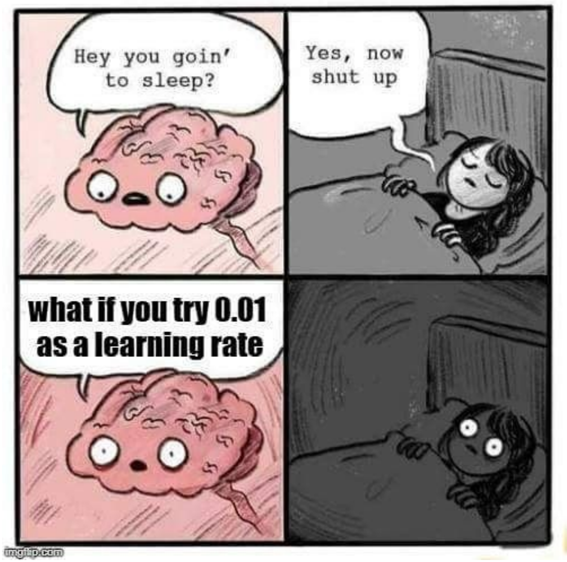
Examples of hyperparameters

How do we choose these numbers??

The best hyperparameter values vary based on model, dataset, and evaluation metric

- K-Means Clustering:
  - Number of clusters
  - `clustering = KMeans(n_clusters=5)`
- TF-IDF
  - Max/min number of documents a word can appear in
  - `tfidf_vectorizer = TfidfVectorizer(min_df=5)`
- SGD
  - Learning rate
- Neural networks
  - Model architecture (hidden layers, nodes, etc.)

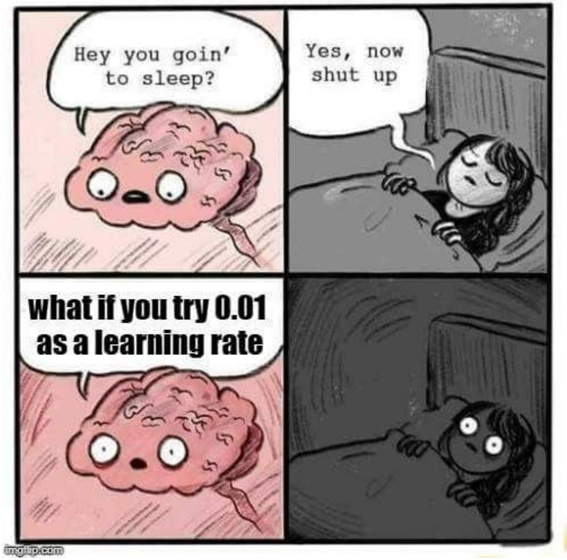
Hyperparameter tuning



- We always want to have the best-performing model
- How do we choose the hyperparameters that will "**optimize**" performance?
  - One option: manually choose a combination of a few hyperparameters and then see which model performs best

https://twitter.com/mlinterview/status/993257627244204032

Hyperparameter tuning



- We always want to have the best-performing model
- How do we choose the hyperparameters that will "**optimize**" performance?
  - One option: manually choose a combination of a few hyperparameters and then see which model performs best
  - Better option: algorithmically "search" for best combinations of hyperparameters

https://twitter.com/mlinterview/status/993257627244204032

Hyperparameter tuning

- **Better option:** algorithmically "search" for best combinations of hyperparameters
  - This is known as **hyperparameter optimization** or **hyperparameter tuning**

- Hyperparameter tuning tries various combinations of hyperparameters and how they affect model performance

Main steps of hyperparameter tuning

1. Choose a model and a set of relevant hyperparameters

Main steps of hyperparameter tuning

1. Choose a model and a set of relevant hyperparameters
2. Define an evaluation function (like F1 or MSE)

Main steps of hyperparameter tuning

1. Choose a model and a set of relevant hyperparameters
2. Define an evaluation function (like F1 or MSE)
3. Iterate through hyperparameters

Main steps of hyperparameter tuning

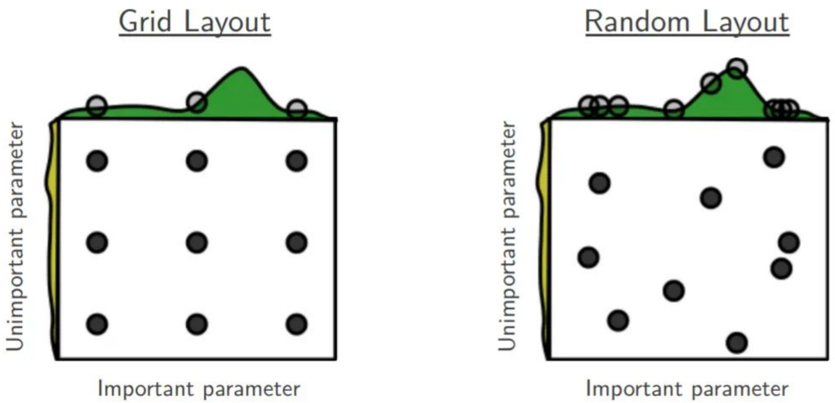
1. Choose a model and a set of relevant hyperparameters
2. Define an evaluation function (like F1 or MSE)
3. Iterate through hyperparameters
4. For each combination of hyperparameter values, evaluate model performance (using step 2!)

Main steps of hyperparameter tuning

1. Choose a model and a set of relevant hyperparameters
2. Define an evaluation function (like F1 or MSE)
3. Iterate through hyperparameters
4. For each combination of hyperparameter values, evaluate model performance (using step 2!)
5. Select combination of hyperparameters that resulted in best performance

Main steps of hyperparameter tuning

1. Choose a model and a set of relevant hyperparameters
2. Define an evaluation function (like F1 or MSE)
3. **Iterate through hyperparameters**
4. For each combination of hyperparameter values, evaluate model performance (using step 2!)
5. Select combination of hyperparameters that resulted in best performance



Most common hyperparameter tuning procedures:

- **Grid search:** evaluate every combination of hyperparameter values
- **Random search:** evaluate hyperparameter values drawn from a distribution

https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318

Demo: Neural Network Playground

https://playground.tensorflow.org/