



Neural Network Primer

Types of Artificial Intelligence

- Mathematical Models

- First known example: Linear Regression - invented by Gauss in 1800
- Classifier examples: logistic regression, support vector machine (SVM)
- Method: choose a model *i.e.* equation with parameters, choose a loss function, set the derivative to zero, solve for the roots, gives parameters that minimize the loss

- Expert Systems

- Knowledge Representation:
 - Create an ontology to define the terms
 - Create a triples database or knowledge graph to define the relationships
- Use First Order Logic and Deduction
 - Apply basic rules of logic to the knowledge and deduce specific facts

- Machine Learning

- Induction: Given lots and lots of examples, infer patterns, rules, or formulas

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



MACHINE LEARNING

Machine learning begins to flourish.



DEEP LEARNING

Deep learning breakthroughs drive AI boom.



1950's

1960's

1970's

1980's

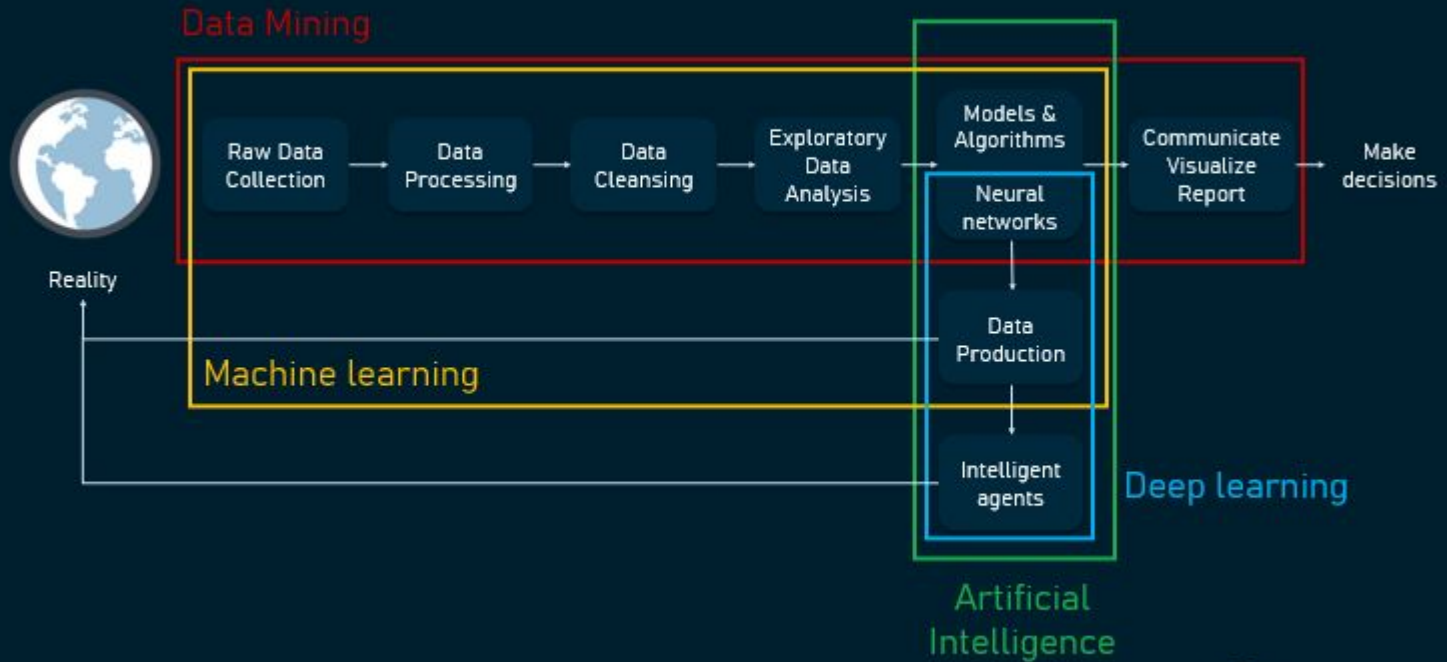
1990's

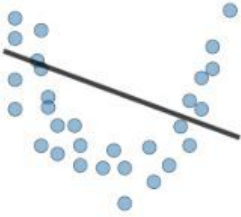


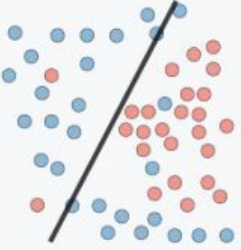
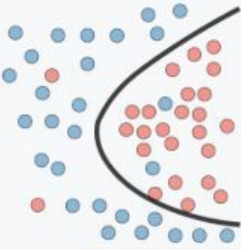
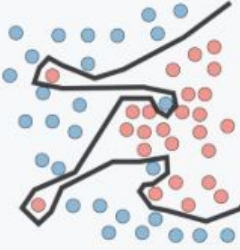

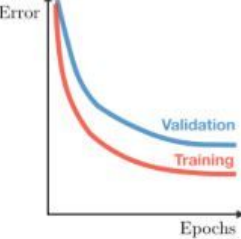
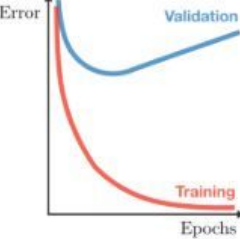
2000's

2010's

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

DATA SCIENCE



	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none"> • High training error • Training error close to test error • High bias 	<ul style="list-style-type: none"> • Training error slightly lower than test error 	<ul style="list-style-type: none"> • Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	<ul style="list-style-type: none"> • Complexify model • Add more features • Train longer 		<ul style="list-style-type: none"> • Perform regularization • Get more data

Classification vs Regression

- Classification task
 - Uses discrete mathematics
 - Predicts one class from a finite number of choices
 - Example: classify this image as cat or dog
 - Two types: binary classification, multiclass classification (the math is slightly different)
 - The softmax function is useful for multiclass
- Regression task
 - Uses continuous mathematics
 - Predicts a floating point number from an infinite range
 - Example: predict the time for this Olympic runner to complete this race
 - Warning: “logistic regression” is a classification algorithm with a really confusing name

Regression



What will be the temperature tomorrow?

84°



Fahrenheit

Classification



Will it be hot or cold tomorrow?

COLD

HOT



Fahrenheit

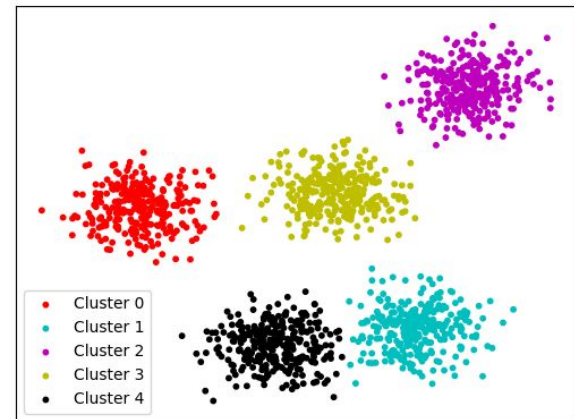
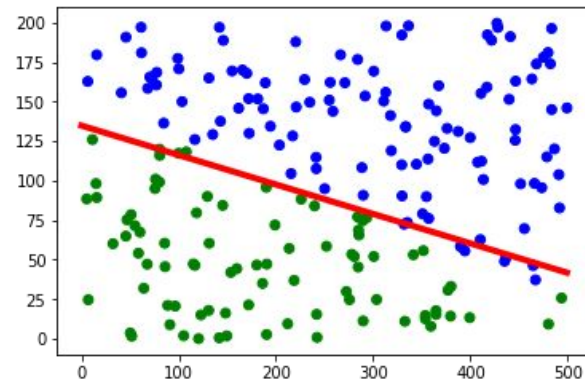
Supervised vs Unsupervised

- Supervised Learning

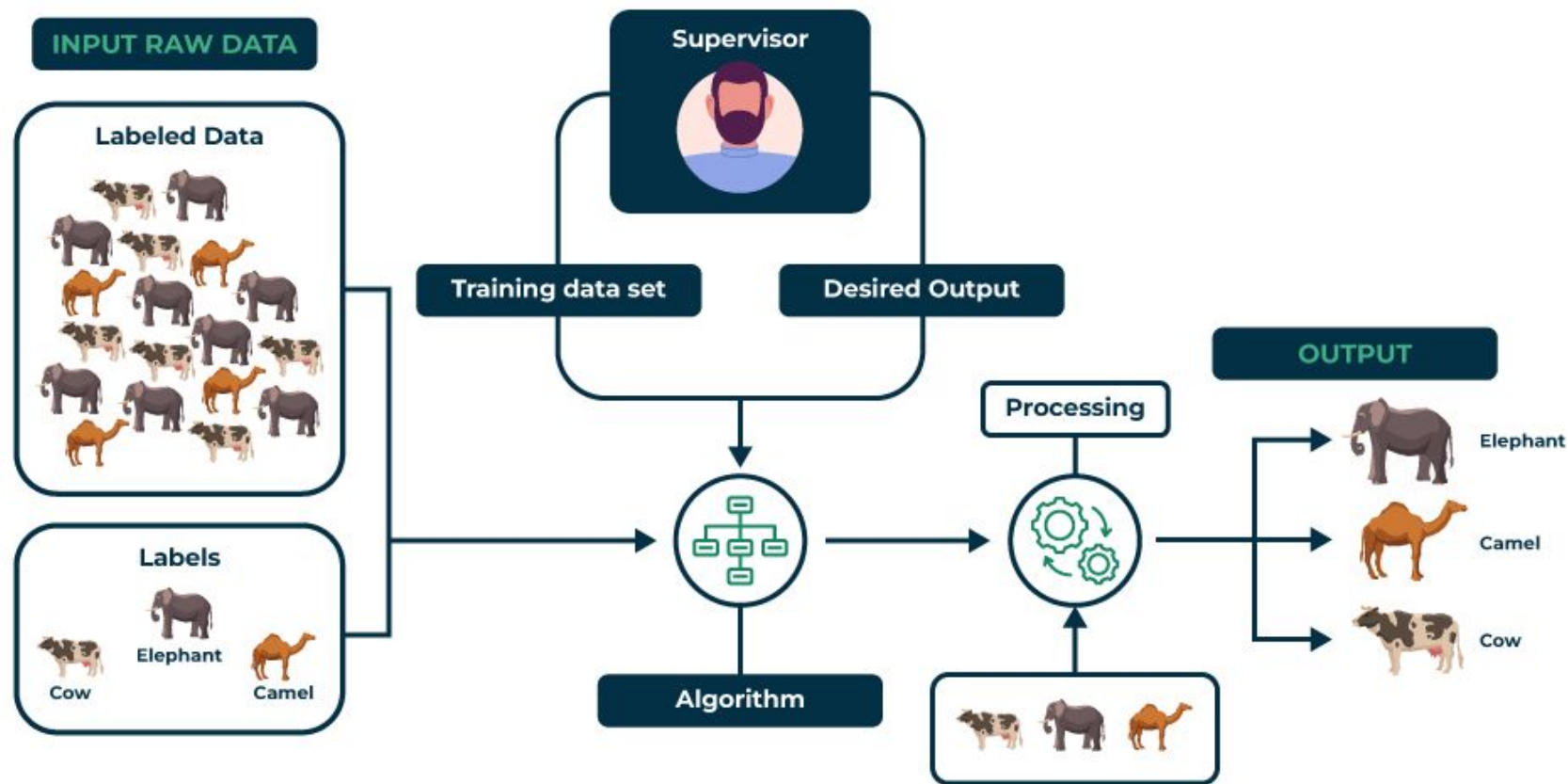
- The training data is labeled. We know the answers.
- Example: Hospital records for patients who were diagnosed with cancer or not.
- Goal: Train a **classifier**!

- Unsupervised Learning

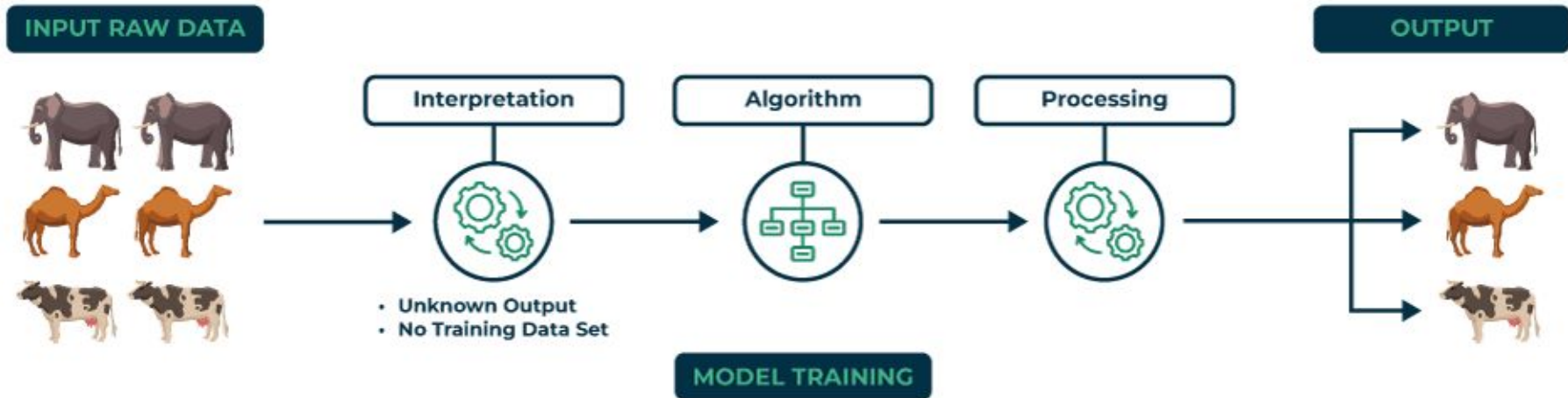
- The training data is not labeled. We don't know the answers.
- Example: Patient records from some hospital.
- Goal: Find patterns in the data.
- Example: The patient records seem to **cluster** into 5 groups. Members of each group are similar to each other and different from members of the other groups. Finding out what distinguishes each group could help us predict outcomes.



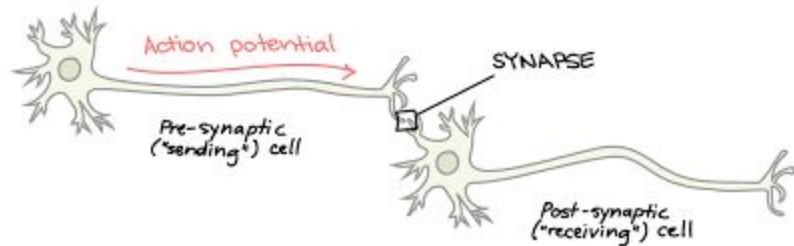
Supervised Learning



Unsupervised Learning

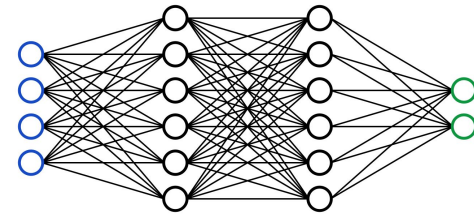


Neuron



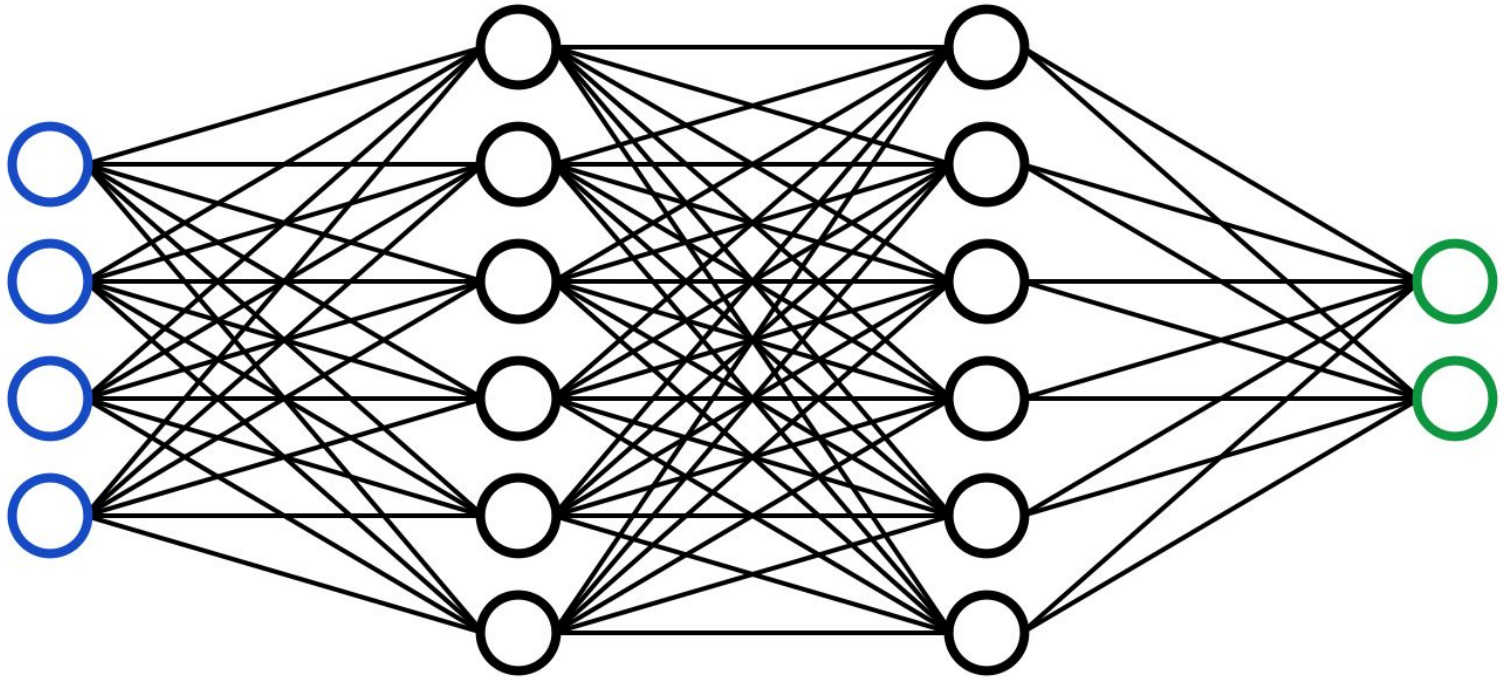
- Model of computation inspired by cells in brains of mammals
- Each neuron has multiple inputs
 - Neurons in brains connect at synapses
 - They communicate across a synapse by secreting a neurotransmitter such as dopamine.
 - Thus each neuron is receiving signals from several others
- Each neuron has a single output
 - Neurons in brains sense inputs from the previous layer of neurons
 - Each neuron decides whether to pass along a signal to the next layer of neurons
- Neurons must be trained
 - Neurons learn the rules during a child's development and education.
 - Example: a neuron might pass long intense signals but ignore small blips (noise filtering)

Artificial Neural Network (ANN)

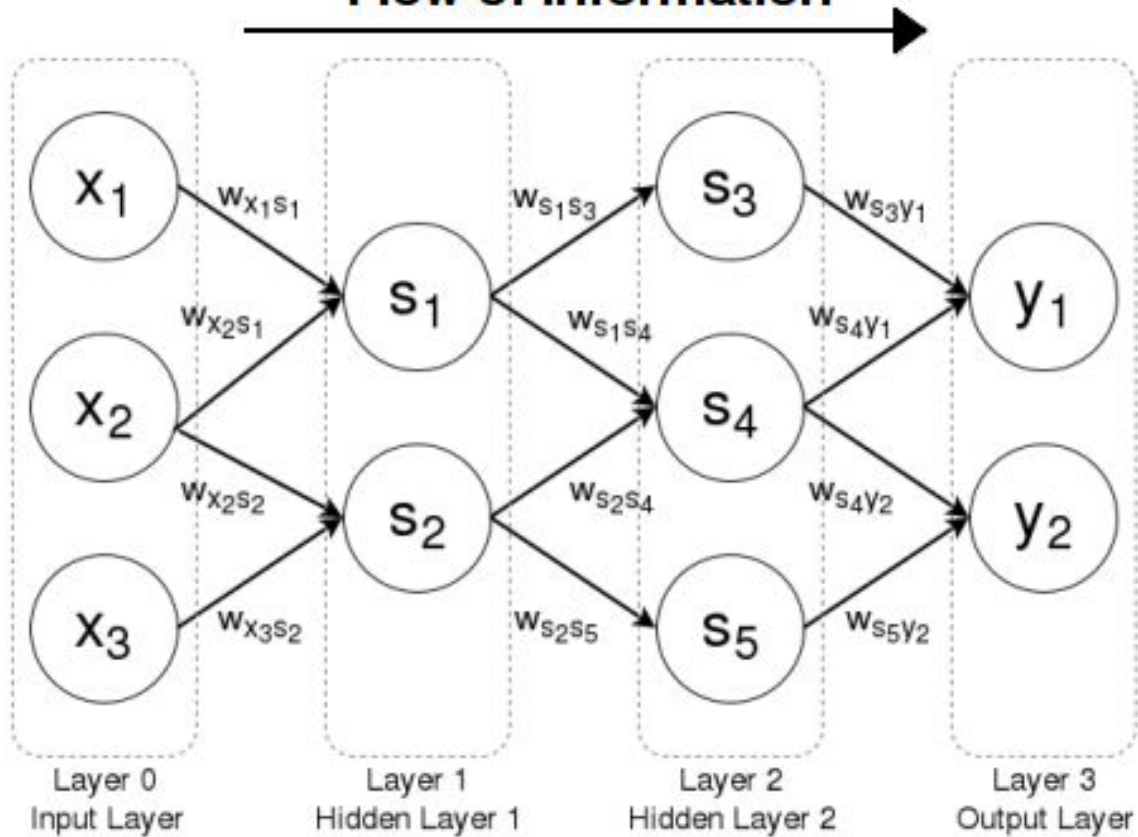


- ANN is a model of computation inspired by the human brain
 - The basic software module acts like a neuron
 - The neuron modules are stacked into layers
 - Each neuron takes input from the previous layer, sends output to the next layer
 - Each neuron is given an activation function but learns the parameters from the data
- The basic ANN goes by many names
 - Feed-forward neural network
 - Fully-connected neural network
 - Multi-Layer Perceptron (MLP)
 - [Later, we'll encounter more complex models: CNN, LSTM, Transformer, GAN]

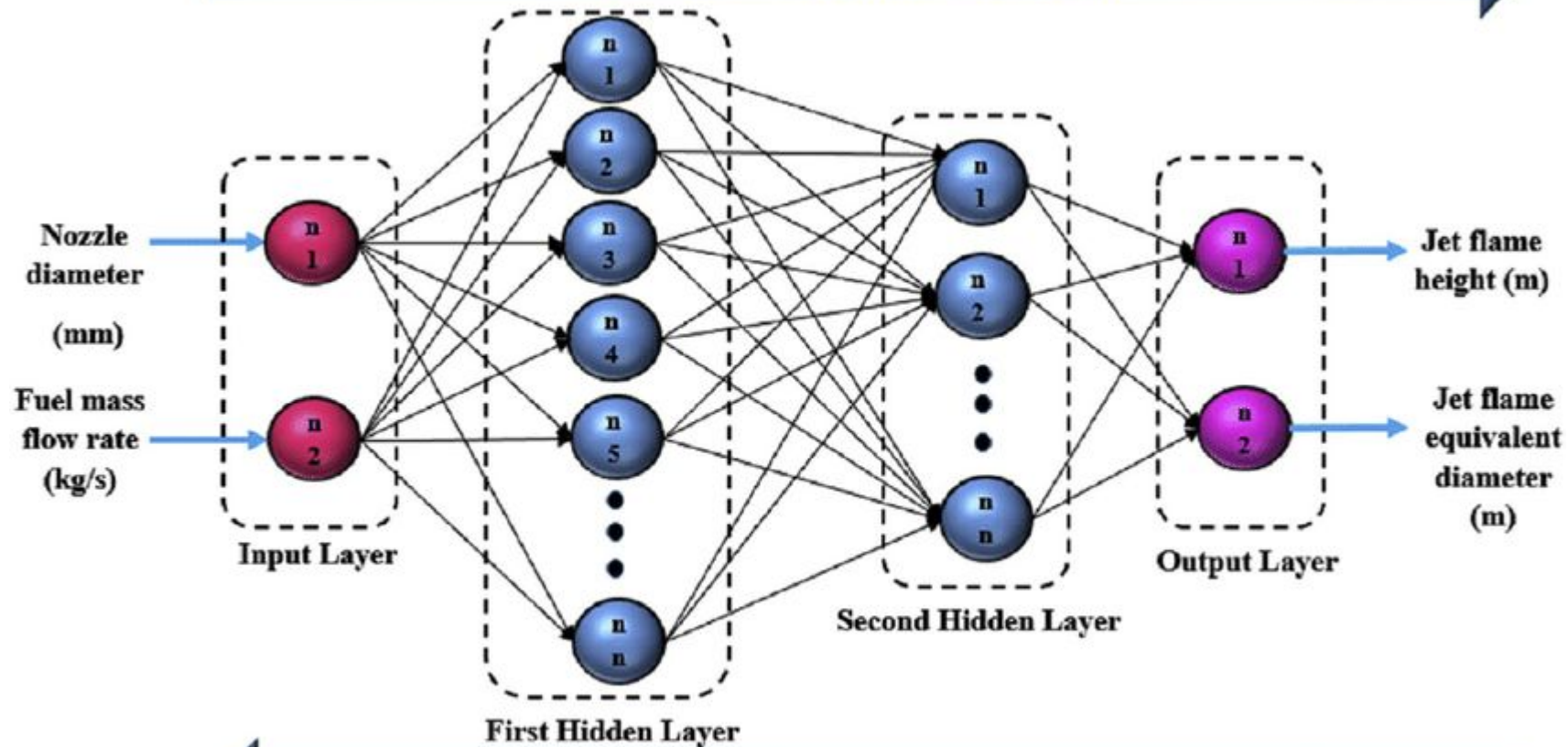
Feed Forward Neural Network



Flow of Information

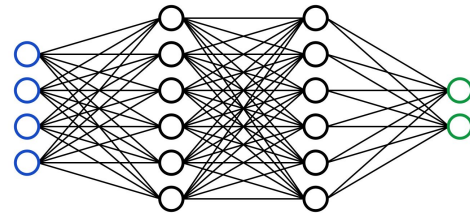


Feedforward of Information



Backpropagation of Errors

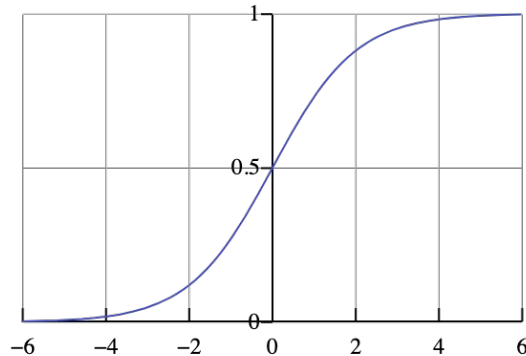
Deep Learning











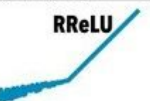

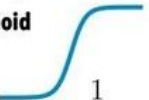
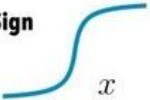

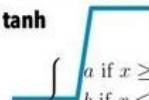
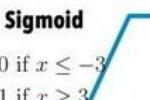
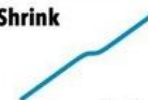

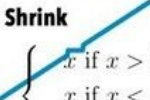
- Deep means 3 or more layers of neurons
 - Some famous models have dozens or hundreds of layers
 - Theorem: adding layers is more effective than adding neurons to existing layers
 - In practice: deeper models (more layers) outperform wider models (more neurons per layer)
 - But deeper models are harder to train. Exponential growth in #parameters to train.
- Retraining
 - Retraining is a technique to lower the cost of training
 - We “freeze” the early layers and only train the last layer(s)
 - ANN retains basic lessons *e.g.* word similarities or boundary recognition
 - This can reduce months of training at huge cost to days of training at low cost
 - Common use: retrain Google’s best image classifier on your company’s images
 - Another use: retrain OpenAI’s GPT on your company’s documents

Activation Functions

- We preload each neuron module with a rule
 - The rule says when to output a high value
 - Example: remember the neuron that acts like a noise filter
 - In training, neuron must learn the parameters for its rule
- We avoid linear activation functions
 - Linear rules like “output the sum of the inputs” have limited expressivity
 - Theorem: any network of linear functions can be reduced to a single linear function
- We choose non-linear activation functions
 - Common examples: sigmoid, tanh, relu
 - All these are non-linear and differentiable
 - Their output values don't grow exponentially
 - Which function is best? It depends, but the benefits of any one are usually small

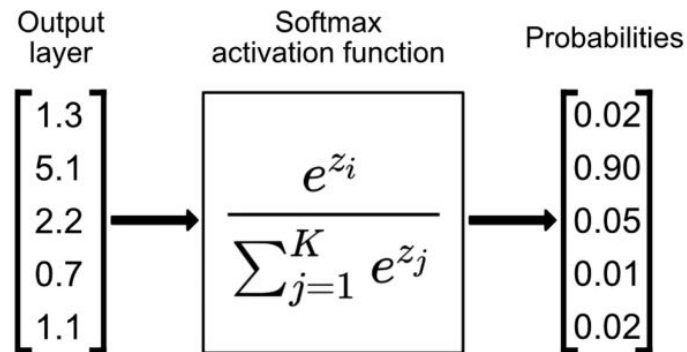


Neural Network Activation Functions: a small subset!

ReLU  $\max(0, x)$	GELU  $\frac{x}{2} \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} (x + ax^3) \right) \right)$	PReLU  $\max(0, x)$
ELU  $\begin{cases} x & \text{if } x > 0 \\ \alpha(x \exp x - 1) & \text{if } x < 0 \end{cases}$	Swish  $\frac{x}{1 + \exp -x}$	SELU  $\alpha(\max(0, x) + \min(0, \beta(\exp x - 1)))$
SoftPlus  $\frac{1}{\beta} \log(1 + \exp(\beta x))$	Mish  $x \tanh \left(\frac{1}{\beta} \log(1 + \exp(\beta x)) \right)$	RReLU  $\begin{cases} x & \text{if } x \geq 0 \\ ax & \text{if } x < 0 \text{ with } a \sim \mathcal{R}(l, u) \end{cases}$
HardSwish  $\begin{cases} 0 & \text{if } x \leq -3 \\ x & \text{if } x \geq 3 \\ x(x+3)/6 & \text{otherwise} \end{cases}$	Sigmoid  $\frac{1}{1 + \exp(-x)}$	SoftSign  $\frac{x}{1 + x }$
Tanh  $\tanh(x)$	Hard tanh  $\begin{cases} a & \text{if } x \geq a \\ b & \text{if } x \leq b \\ x & \text{otherwise} \end{cases}$	Hard Sigmoid  $\begin{cases} 0 & \text{if } x \leq -3 \\ 1 & \text{if } x \geq 3 \\ x/6 + 1/2 & \text{otherwise} \end{cases}$
Tanh Shrink  $x - \tanh(x)$	Soft Shrink  $\begin{cases} x - \lambda & \text{if } x > \lambda \\ x + \lambda & \text{if } x < -\lambda \\ 0 & \text{otherwise} \end{cases}$	Hard Shrink  $\begin{cases} x & \text{if } x > \lambda \\ x & \text{if } x < -\lambda \\ 0 & \text{otherwise} \end{cases}$

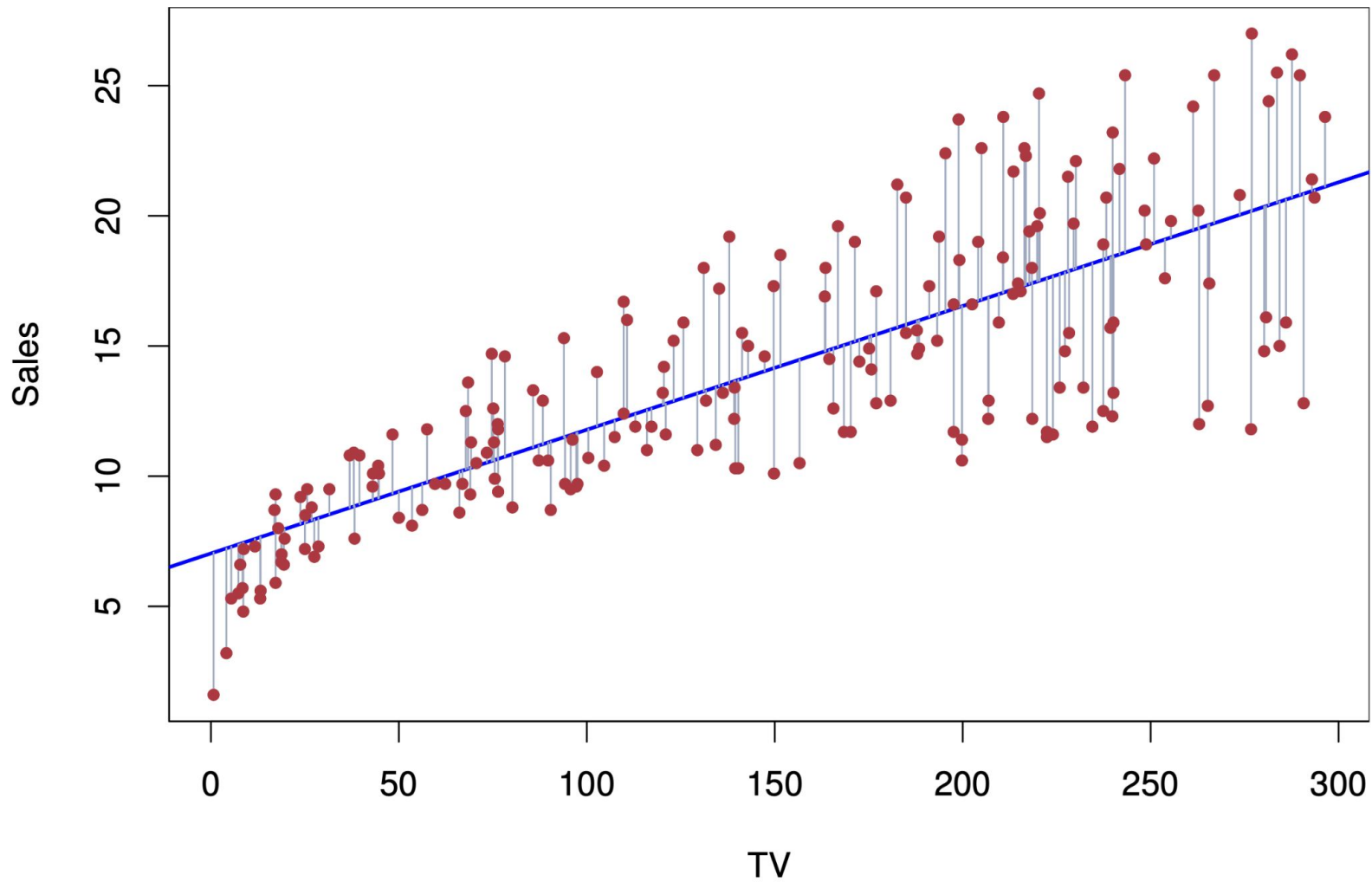
Softmax Function

- The softmax function is useful for multiclass classification
 - Note: “softmax” has slightly different meanings in mathematicians vs machine learning
- Goal:
 - Decide the most likely class given scores for each class
- Method:
 - For each image, the model generates a score relative to each class.
 - Mathematically convert the scores to probabilities that sum to 1.
 - Return the index of the class with the highest probability.
- Example:
 - Say the classes are: 1=car, 2=truck, 3=motorcycle.
 - Say the model probabilities for some image come to: car=20%, truck=50%, motorcycle=30%.
 - Then softmax returns 2.



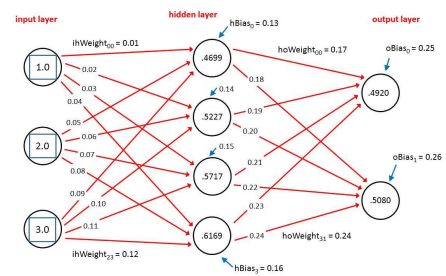
Model, Hyperparameters, Parameters

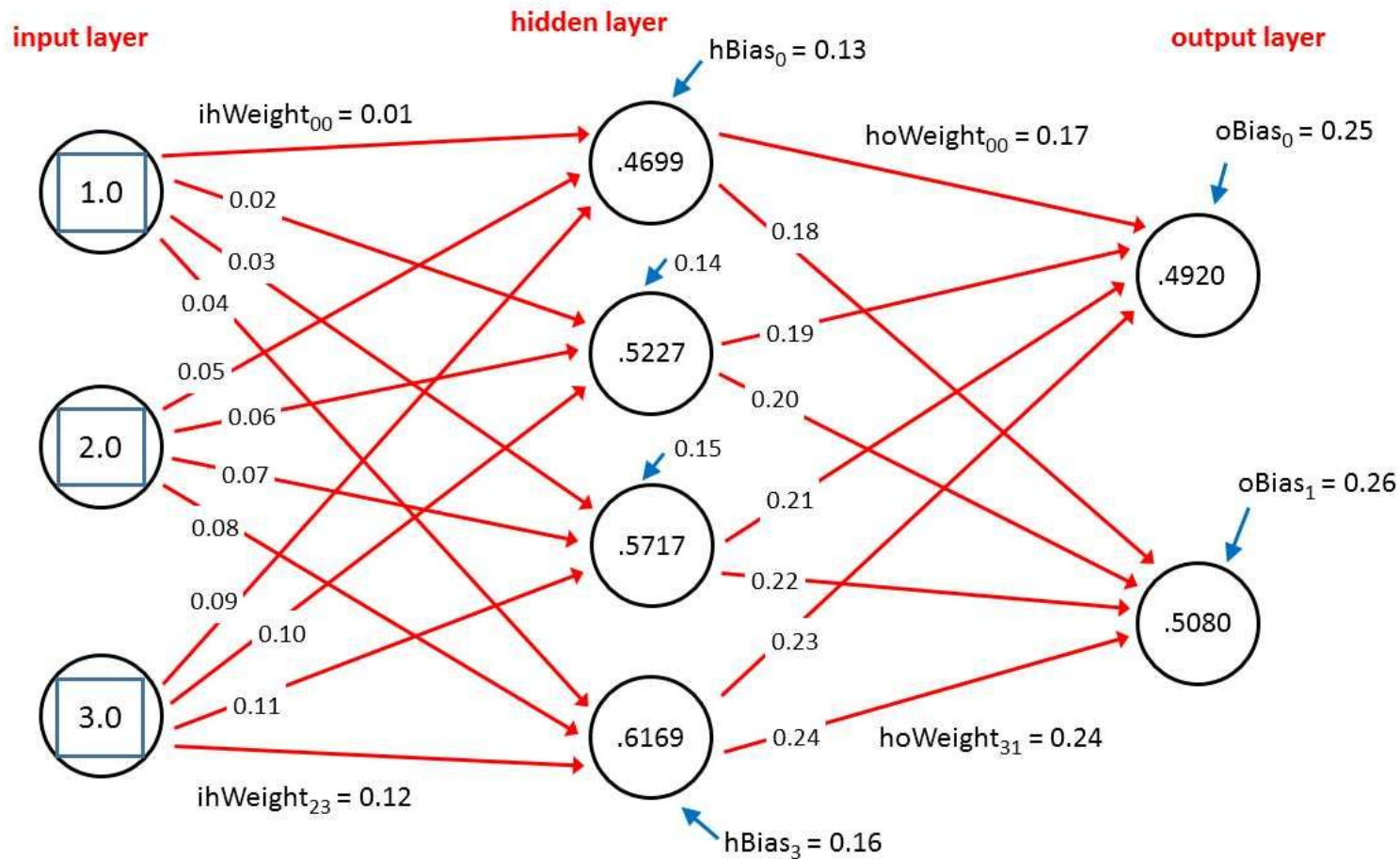
- A model is a mathematical function
 - Example: $y = \mathbf{m}x + \mathbf{b}$ is a linear model
 - The model embodies assumptions e.g. assume a linear relationship
 - Neural networks are just (very complicated) mathematical functions
- Models hyperparameters are chosen
 - These are the design decisions made by the experimenter
 - Examples: # layers, # neurons per layer, which activation functions
- Model parameters are learned
 - We learn the ideal parameter values from the data
 - Example, for model $y = \mathbf{m}x + \mathbf{b}$, choose best values for \mathbf{m} and \mathbf{b} by linear regression on data
 - Today's generative AI models have billions of trained parameters



The forward computation in ANNs

- A neural network is a computation graph
 - Each neuron computes and outputs a number based on its inputs
 - Each neuron receives inputs from neurons in the previous layer
 - Each neuron sends outputs to neurons in the next layer
 - The last layer makes the final output
- Prediction: the trained network computes an output given an input
 - Example: input the pixel values of an image, output a yes or no
 - Forward computations are usually fast (much faster than training)
 - The entire network could be reduced to a (very complicated) mathematical function
- Loss function
 - Choose a formula for quantifying the performance
 - Examples: number wrong, percentage wrong, root mean square error (RMSE)

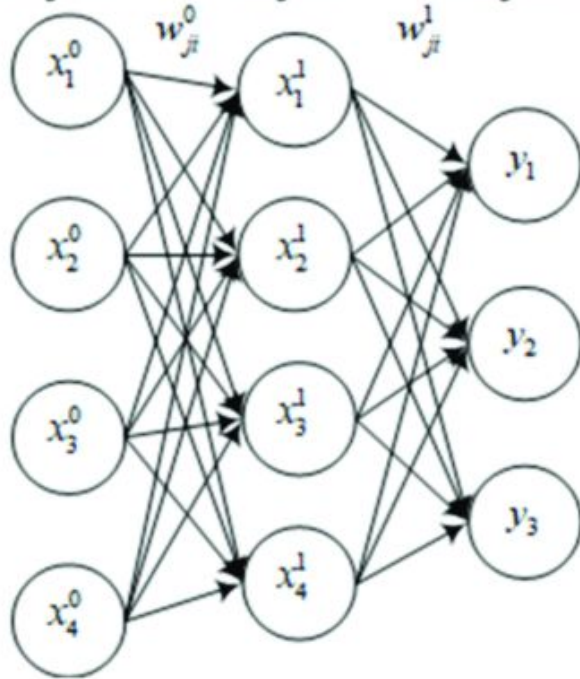




The backward computation in ANNs

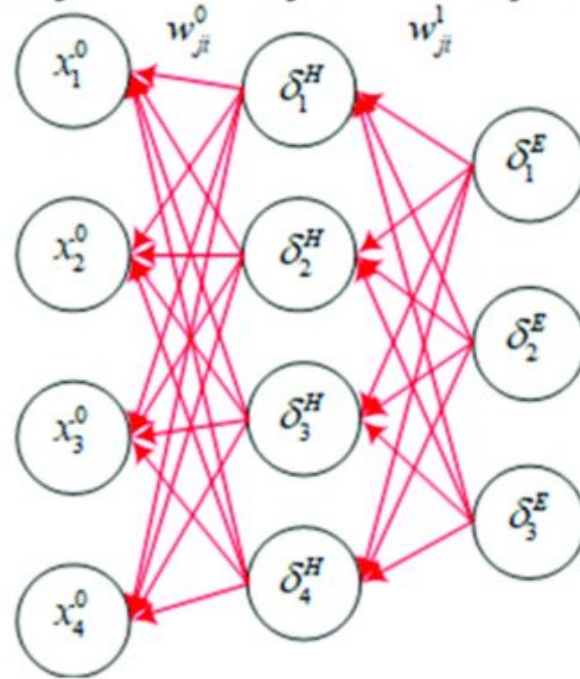
- Problem: we need parameter values for our model
 - Example 1: our model is $y=mx+b$, but what are **m** and **b**?
 - Example 2: our neural network (2 layers, 64 neurons/layer) contains random values
 - For some mathematical models, there is a closed form solution (e.g. roots of derivative)
 - For neural networks, there is no easy solution
 - Exhaustive search is intractable because the search space is huge
- Solution: stochastic gradient descent (SGD)
 - A: Start with random values.
 - B: Randomly select samples. Run the model. Measure the loss.
 - C: If the loss is small, stop, else adjust the parameters slightly to fix the loss. Go back to B.
 - Gotcha: SGD finds local minima. To find others, repeat the whole process starting at A.
- But how exactly to adjust parameters?
 - Backpropagation. Published by Hinton in 1986. Requirement: the loss function is differentiable.
 - Math: Work backward, starting at last layer. Compute the partial derivative (**pd**) of the loss relative to each neuron **N**. Adjust each **N**'s parameters, informed by the sign and magnitude of its **pd**.
 - Put simply: If the model's predicted value was too overly positive, and the **pd** relative to **N** was overly positive, then adjust **N**'s parameters in the negative direction.

Input layer Hidden layer Output layer



Forward = prediction

Input layer Hidden layer Output layer



Backward = error propagation

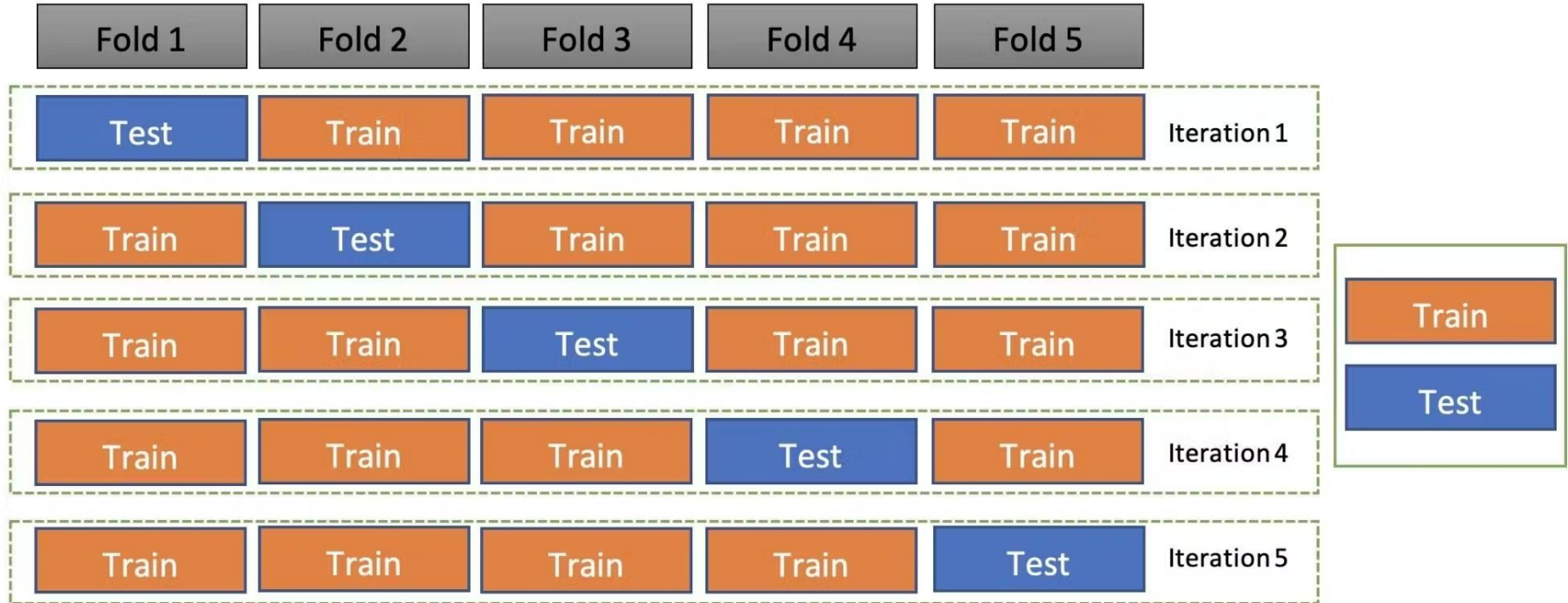
The train:test data partition

- Problem: modelers overfit the training data
 - Scientist iteratively improves the model until it performs really well on the training data
 - But how will the final model perform on future, unseen data?
- Solution: a separate test subset
 - Before starting, randomly partition the data into train:test subsets. Typically 80%:20%
 - Don't even look at the test data! Use it only once as the “final exam.”
- But what if I fail the final exam?
 - No publication. Months of time and grant money wasted!
 - Of course, people are tempted to start over and try again, but that is dishonest.
 - You need ANOTHER test set that is truly unseen. This may be infeasible.
 - To avoid this, try validation (next slide)

The (train:valid):test data partition

- Problem: you can only use the testing subset once
 - You don't want to fail the “final exam”
- Solution: simulate the final exam with train:valid partitions
 - Start with the training subset (typically 80% after 20% is removed for testing)
 - Then, pretend the training subset is your entire dataset, and partition it!
 - Randomly partition the training subset into train:valid subsets. Typically 80%:20%
 - Train the model on the reduced training subset. Test the model on the validation subset
 - Ok to train, validate, change model, & repeat as needed
- 5-Fold Cross Validation (5FCV)
 - Commonly used technique to prepare for the “final exam” on the test set
 - Make 5 random partitions of the training subset
 - For each partition: train on the other 4 partitions, validate on this partition
 - Measure performance using mean & standard deviation over 5 folds
 - Finally, retrain the model on the full training subset, and test it on the testing subset.

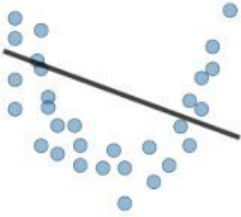


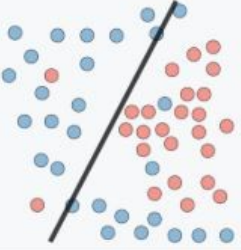
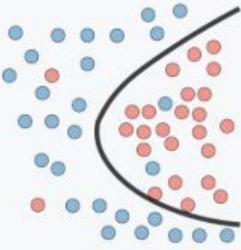
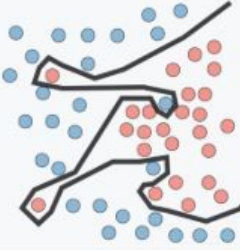

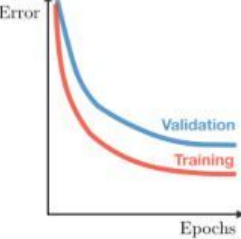
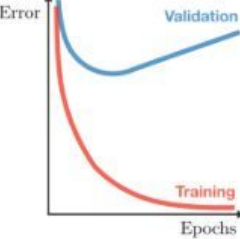
Five Fold Cross Validation



Usually, there is an additional test set that remains unseen during training.

Overfitting vs Generalization

- Problem: Overfitting
 - Our models get too smart! After extensive training, $\text{loss} == 0$
 - These “overfit” models may behave stupidly on unseen data
 - Analogy: student who memorizes all the review problems, then fails the final exam
- Goal: our model should generalize and work on unseen data
- Mathematical solutions
 - “Regularization” describes multiple techniques for generalizing mathematical models
 - Example: penalize models for their complexity ($\# \text{parameters}$ or $\text{abs}(\text{parameter})$)
- Computational solutions (two of many)
 - Method 1, early stopping. Stop the training before the loss hits zero. (But how early?)
 - Method 2, dropout. Published by Hinton in 2012. Disable different random portions of the neurons while you train.

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none"> • High training error • Training error close to test error • High bias 	<ul style="list-style-type: none"> • Training error slightly lower than test error 	<ul style="list-style-type: none"> • Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	<ul style="list-style-type: none"> • Complexify model • Add more features • Train longer 		<ul style="list-style-type: none"> • Perform regularization • Get more data

Python and Jupyter Notebooks

- Why Python?
 - Python 3 is fully object-oriented and very popular
 - Google and Facebook programmers used Python for internal machine learning research
 - Their libraries are now public and open source.
 - These libraries are object-oriented and well documented. Great resources!
 - For speed, the linear algebra is coded in C++ with Python bindings
- Why Notebooks?
 - These are web pages implemented with JavaScript
 - They rely on a web server that runs a Python interpreter for each page
 - Notebooks show the code, actual results, and documentation, all in one page!
 - The documentation is written in markdown language, then rendered in HTML

Google CoLab

- Service offered by Google
 - First-come, first-served. You may not get a CPU, even if you pay.
 - Basic usage is free (requires a gmail account)
 - Paid usage *e.g.* \$10/month - more likely to get resources
- Runs on the Google Cloud
 - Includes various CPU types *e.g.* standard, high RAM
 - Includes various GPU types *e.g.* T4, V100, A100
 - Can access data on your Google Drive (requires a drive account)
 - Can interact with some non-Google resources *e.g.* GitHub
- Pre-installed: Python 3, Jupyter Notebooks, Google libraries
 - And you can install other libraries on your virtual machine

scikit-learn, Tensorflow, PyTorch, Keras

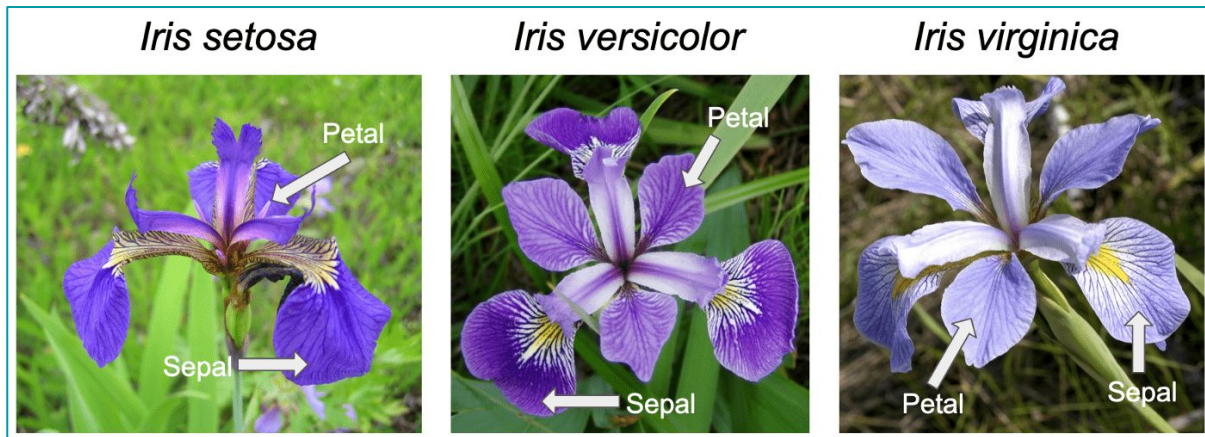
- **scikit-learn**
 - Open source. Implements many mathematical machine learning models.
- **Tensorflow**
 - Was internal to Google, now open source. Implements many neural network models.
- **PyTorch**
 - Was internal to Facebook, now open source. Implements many neural network models.
- **Keras**
 - Was designed by a Google scientist.
 - Originally an abstraction layer, supporting either Tensorflow or PyTorch as its backend.
 - Now open source, has been folded into Tensorflow. Can be hard to tell which is which.

Another common dataset: Iris

One genus, three species, 50 flowers per species

Four features per flower: width & length of petal & sepal

Challenge: train a model to classify an iris given these measurements



This 1936 dataset was collected by botanist E. Anderson and used by statistician **R.A. Fisher** to demonstrate linear discriminant analysis, which he invented.

This dataset is used in many demos and is available on many machine learning platforms.

Preprocessing

		Features (4 total)				Labels (species converted to int)
		sepal length	sepal width	petal length	petal width	species
Samples (first 5 are shown)	0	5.1	3.5	1.4	0.2	0
	1	4.9	3.0	1.4	0.2	0
	2	4.7	3.2	1.3	0.2	0
	3	4.6	3.1	1.5	0.2	0
	4	5.0	3.6	1.4	0.2	0

Train

Train

Test

Train

Train

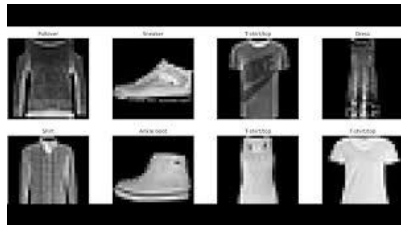
Random partition:
80% Train, 20% Test.

Supervised Learning:

Using labeled samples to train a model to predict those labels.

Commonly used datasets: MNIST

- NIST = National Institute of Standards and Technology
 - US federal agency
- MNIST Digits
 - Images of the 10 single digits (0,1,2,3...,9)
 - Taken from envelopes handled by the US Postal Service.
 - Very clean: black & white, uniform size, perfectly cropped and centered, no overlap.
 - Partitioned into training and testing data. Both are labeled.
- Fashion MNIST
 - Images of 10 types of clothing: shirt, shoe, etc.
 - Very clean: black & white, uniform size, perfectly cropped, centered, oriented.
 - Partitioned into training and testing data. Both are labeled.



Example

The GitHub repo for this book:

Has one notebook for all of chapter 1:

https://github.com/ageron/handson-ml3/blob/main/01_the_machine_learning_landscape.ipynb

