Lecture 7:
Introduction to Applied ML

CMSC 25910

Spring 2022

The University of Chicago

THE UNIVERSITY OF CHICAGO

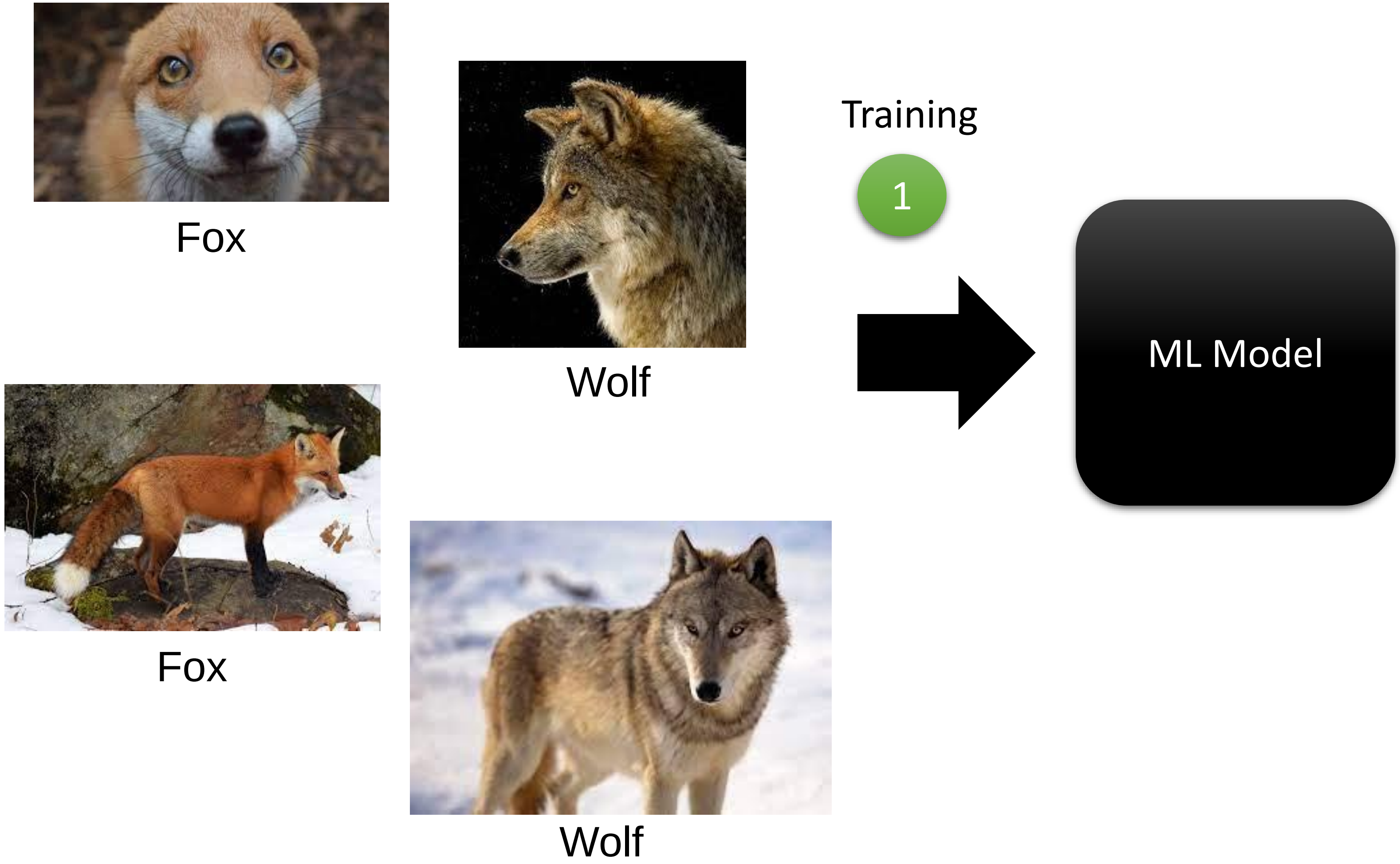
Goals and Intuition

Relationship Between Task & Methods

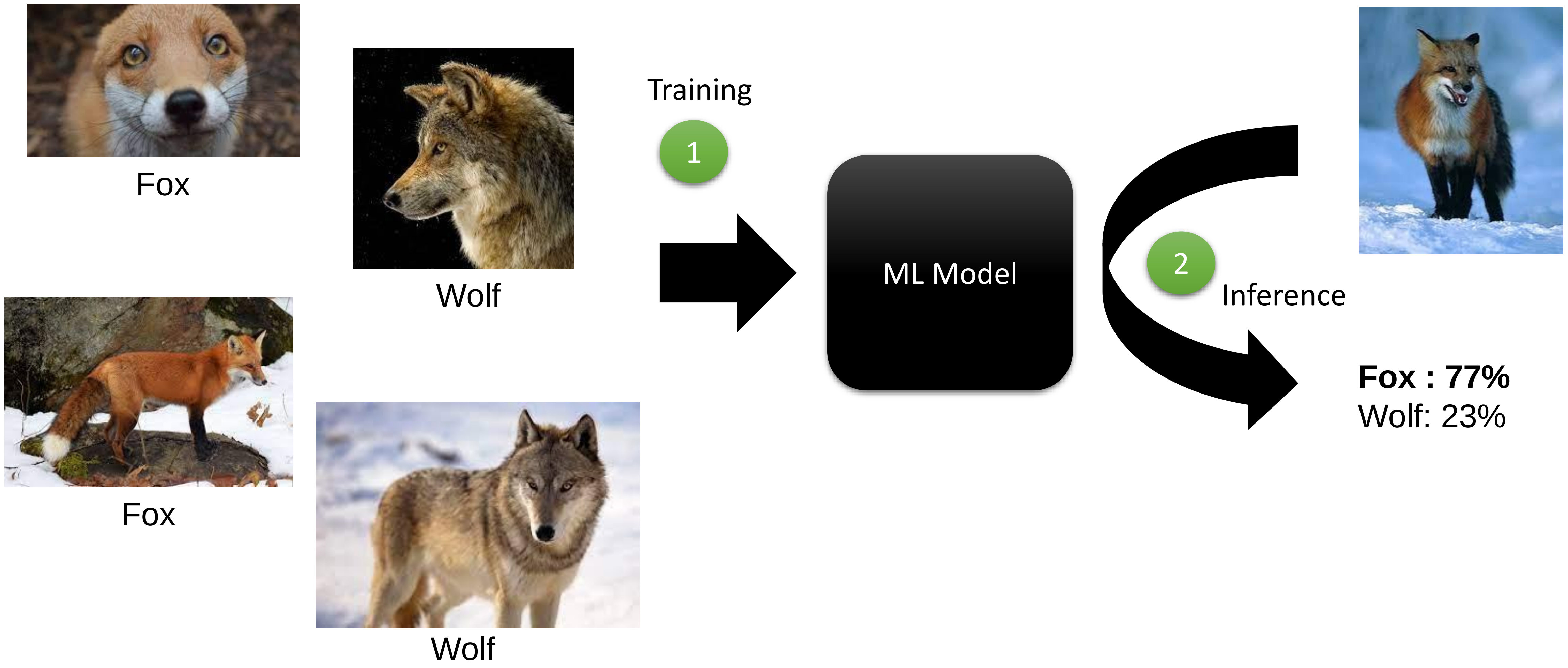
- Task: explain/describe data
  - Descriptive statistics (e.g., what percentage of people are late?)
- Task: use observed data to infer information about a population
  - Inferential statistics (e.g., what's the level of support for this candidate?)
- Task: draw a causal connection, explain
  - Experiments, quasi-experiments, human subjects, etc.
- Task: **predict** characteristics of **out-of-sample data**
  - Machine learning (prediction, forecasting, classification, etc.)

High-Level Intuition

High-Level Intuition



High-Level Intuition



Why we build models…

- To understand data
- To make predictions about *out-of-sample* data

Models

- “All models are wrong, but some are useful.” –George Box
- *“*Modelling in science remains, partly at least, an art. Some principles do exist, however, to guide the modeler. The first is that *all models are wrong*; some, though, are better than others and we can search for the better ones. At the same time we must recognize that eternal truth is not within our grasp.*”* -*McCullagh, P.; Nelder, J. A. (1983), Generalized Linear Models, Chapman & Hall, §1.1.4*.

Regression Example

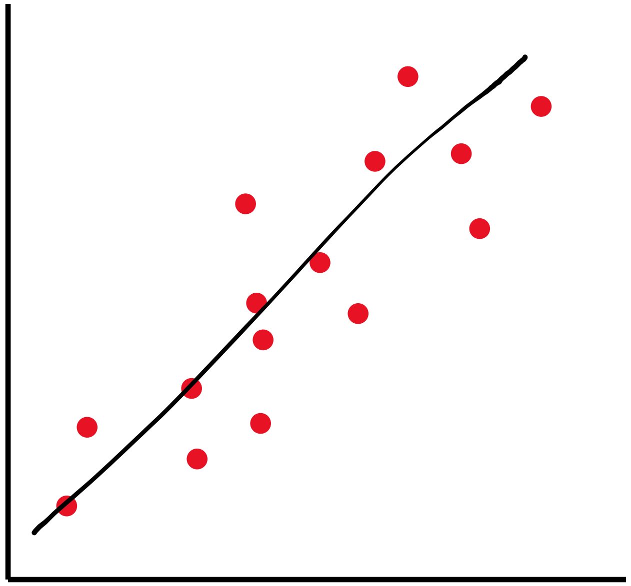
Let’s Build a Model To Understand Data

- Running example: a regression problem
- Example:

Name	Age	Department	Gender	Title	**Salary**
Jack	55	CS	M	Professor	??
Jane	27	Stats	F	Assistant Professor	??

Given these input vectors… …predict this input variable

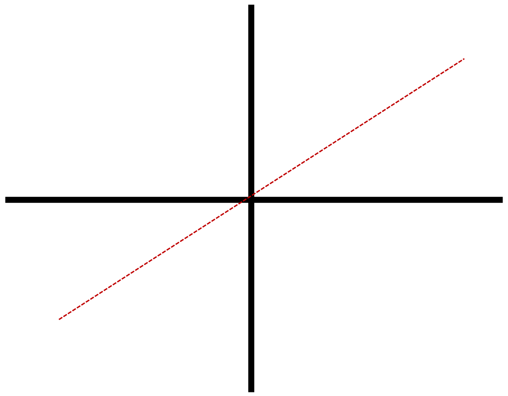
Building Intuition: Fitting a Line



Given Input Vector x, Predict y

- We need to choose a model to do that

$$\hat{y} = 0.3x$$



Parameters / weights

Output value / Explanatory

Input vector / predictor

$$\hat{y} = w^T x$$

$$x \in \mathbb{R}^n$$

$$y \in \mathbb{R}$$

$$w \in \mathbb{R}^n$$

Let’s Build a Model To Understand Data

- Running example: a regression problem
- Example:

Name	Age	Department	Gender	Title	**Salary**
Jack	55	CS	M	Professor	??
Jane	27	Stats	F	Assistant Professor	??

x_1, x_2, x_3, x_4, x_5 $\hat{y}$

Variables/Attributes/Columns become ‘features’ of the input vector

Linear Regression Model

- ‘Linear’ because of the relationship between x and y

$$\hat{y} = w^T x + b$$

Linear Regression Model

- 'Linear' because of the relationship between x and y

- A model is an assumption…
  - …of what function represents data *well*

$$\hat{y} = w^T x + b$$

- Once we've fixed a model…
  - …we <u>find</u> the parameters/weights *w* that make the model <u>perform well</u>

Linear Regression Model

- ‘Linear’ because of the relationship between x and y

- A model is an assumption…
  - …of what function represents data *well*

$$\hat{y} = w^T x + b$$

- Once we’ve fixed a model…
  - …we find the parameters/weights *w* that make the model perform well

We need a method to find those parameters

This suggests we need a performance metric

Our Data

- A dataset becomes a matrix
  - Each row is an input vector

Name	Age	Department	Gender	Title	Salary
Jack	55	CS	M	Professor	33000
Jill	23	Econ	F	Professor	32000
Josh	32	Bio	M	Staff	28000
Jenn	44	Bio	F	Associate Professor	24000
Jane	27	Stats	F	Assistant Professor	25000

Train-Test Split

Dataset contains the target variable / label

- A dataset becomes a matrix
  - Each row is an input vector

	Name	Age	Department	Gender	Title	Salary
Training dataset	Jack	55	CS	M	Professor	33000
	Jill	23	Econ	F	Professor	32000
	Josh	32	Bio	M	Staff	28000
Test dataset	Jenn	44	Bio	F	Associate Professor	24000
	Jane	27	Stats	F	Assistant Professor	25000

Performance Metric

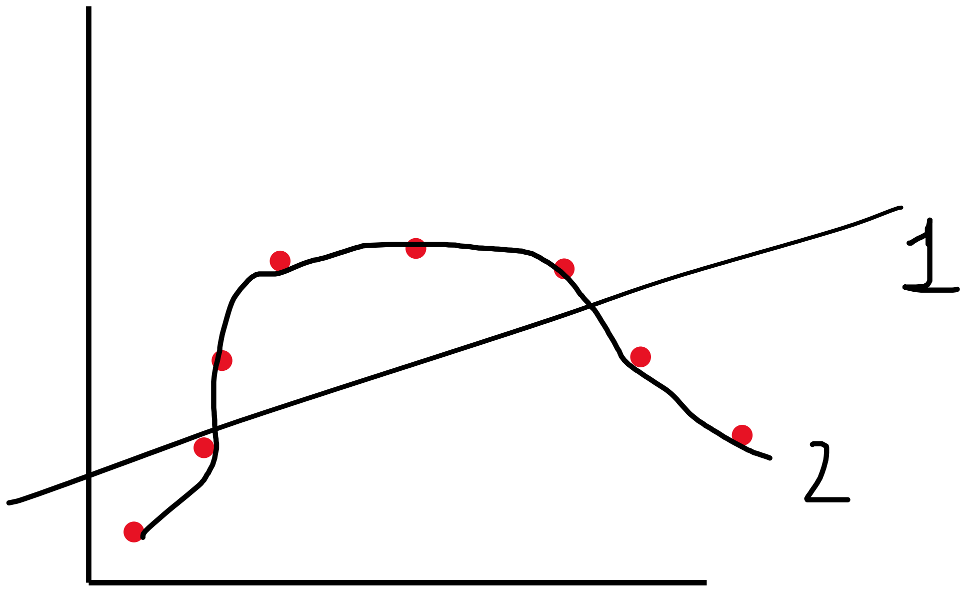
- Mean Squared Error (MSE)
  - Error decreases to 0 when *predicted y = ground-truth y*

$$\mathrm{MSE}_{\mathrm{test}} = \frac{1}{m} \sum_i (\hat{\boldsymbol{y}}^{(\mathrm{test})} - \boldsymbol{y}^{(\mathrm{test})})_i^2.$$

m test examples

- Goal: We want the model to perform well on the test data, which has “never been seen before” (out-of-sample data)

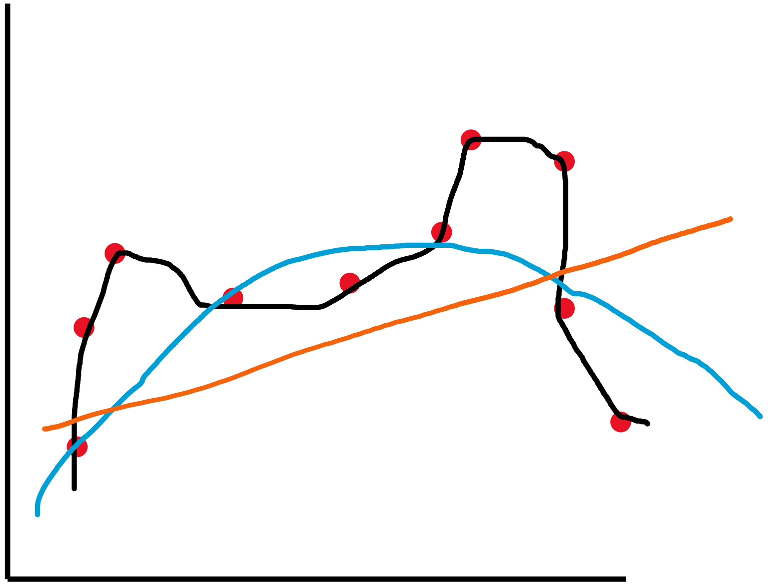
Building Intuition…



- “*With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.*”
  - John von Neumann (born Neumann János Lajos)

Higher Capacity Models

- We can increase the capacity of the model by adding more parameters; this will help with obtaining a ‘better’ fit



$$\hat{y} = w^T x$$

$$x \in \mathbb{R}^n$$

$$y \in \mathbb{R}$$

$$w \in \mathbb{R}^n$$

Goal

- We want to find parameters w using the training dataset

We want to achieve
a low training error

Optimization

- We want to find parameters *w* using the training dataset

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}_{\mathrm{train}} = 0$$

- This is an optimization problem that we know how to solve well; we can find the minimum MSE
- Consider that we run this optimization with the training data. What will happen when we run it on the test data?

Challenges

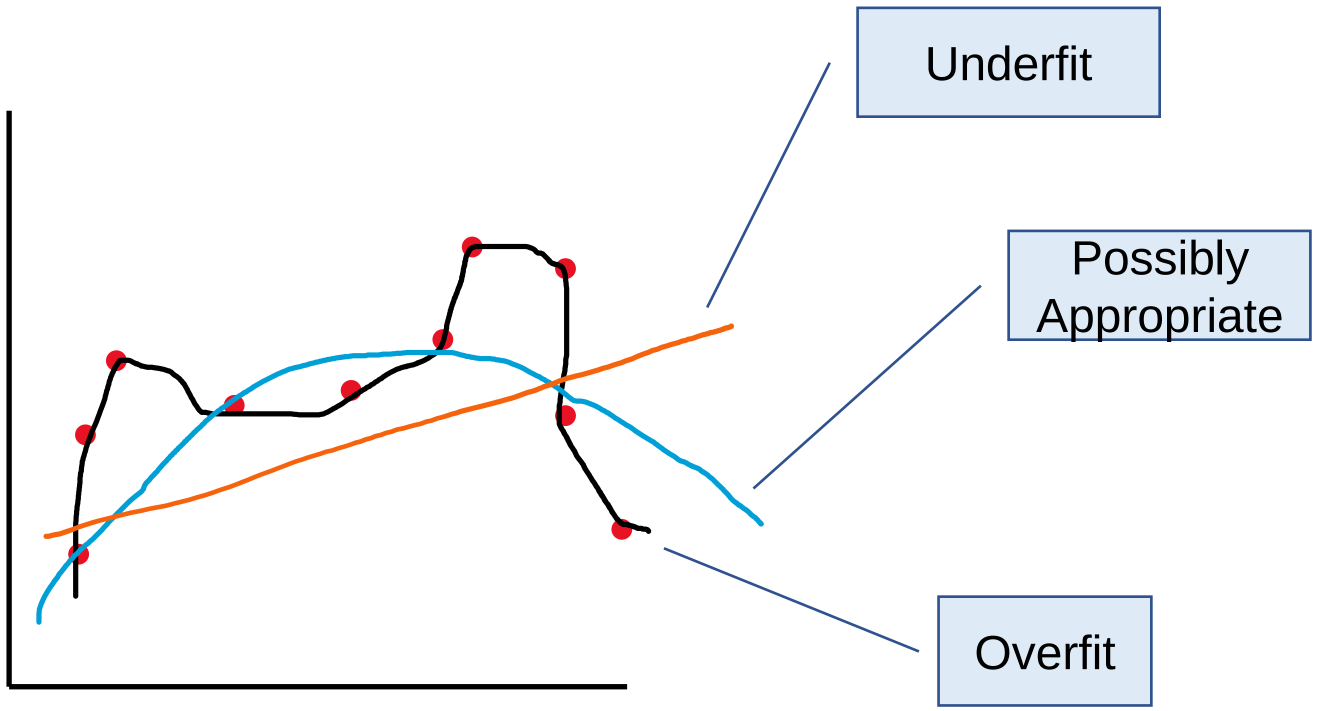
Challenges For Machine Learning

- Learn parameters so the model performs well on unseen data
  - *Generalize* to **unseen data**
  - As opposed to the optimization problem of doing well on **training data**
- Remember why we build models:
  - To understand the process that generated the data
  - To make predictions about out-of-sample data
- Do you think minimizing the MSE on the training data helps us achieve any of those two goals?

Underfitting, Overfitting

- Underfitting
  - When a model cannot reduce the *training error*
- Overfitting
  - A model achieves low *training error* but high *test error*
- Ideally, we want low training error and small gap between training and test error
  - That’s a model that explains the data generation process
  - That’s a model that helps us predict out-of-sample data

Underfitting, Overfitting…



So, What Is Machine Learning?

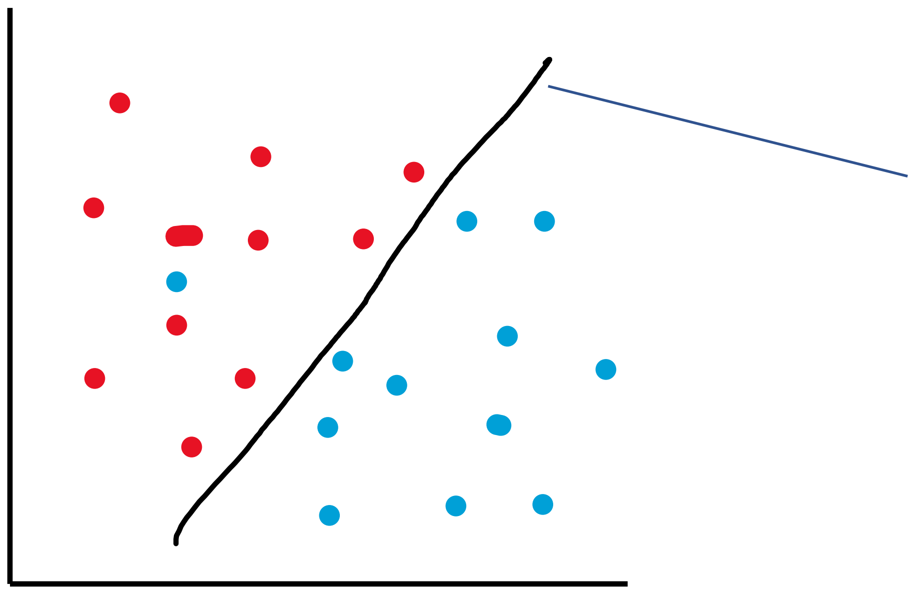
- A model
  - Linear regression, logistic regression, …
- Parameters
- A performance metric
  - MSE
- A training objective
  - Loss function
- A strategy to learn/fit the model parameters

One Common Task Formulation: Classification

[Some of the slides in this section were cannibalized from Elena Zheleva at UIC, and by the transitive property from the Berkeley DS 100 team, Marine Carpuat, Lise Getoor, Brian Ziebart. Please do not further distribute. Mistakes are my own.]

Classification Problem

- Given an input vector *x*, predict a class *c*
  - Binary classification problems
    - Spam vs. not spam
    - Give loan vs. don't
    - Admit student vs. don't
    - Will reoffend vs. won't

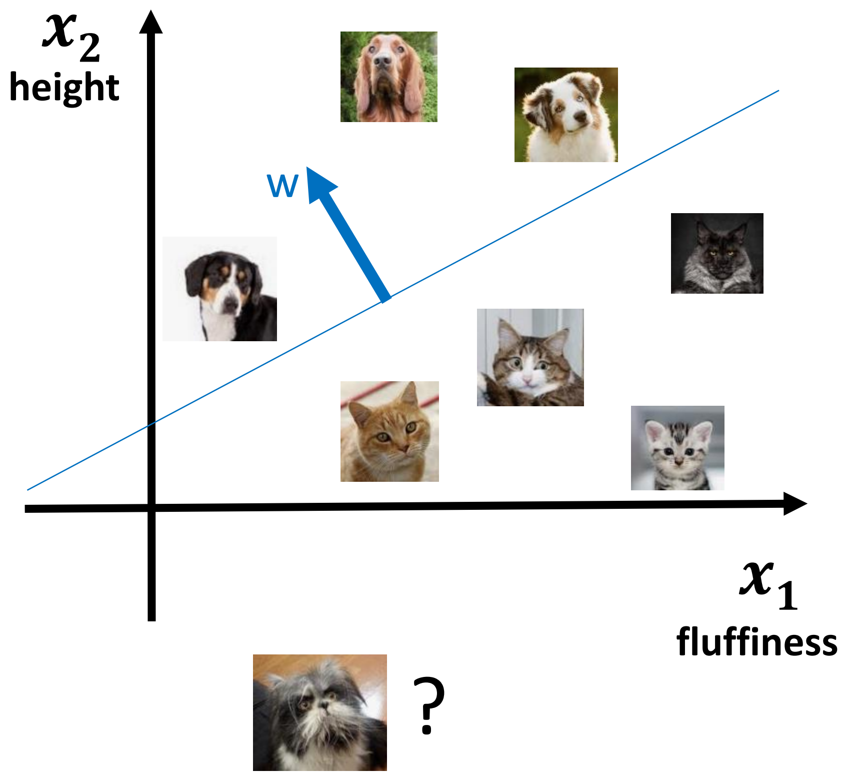


Find a hyperplane that separates the space of positive and negative samples

- How do you evaluate this?
  - Accuracy, false positives/negatives, …

Classification Problem

- Build a model that can predict the categorical value of an unseen object
- Problem setting
  - **X** – set of possible instances with features x_i
  - Y – target class
  - Unknown target function f: **X** →Y
  - Set of function hypotheses H={h|h: **X** →Y}
- Input
  - Training examples $\{(\boldsymbol{x}^{(1)}, y^{(1)}),\ldots(\boldsymbol{x}^{(N)}, y^{(N)})\}$ of unknown distribution
- Output
  - Hypothesis $h \in \mathrm{H}$ that best approximates target function f



Logistic Regression

- Widely used models for **binary classification:**

$x =$ "Get a FREE sample ..."

$$\phi(x) = [2.0, 0.0, \ldots, 1.0, 0.5]$$

➡ $y = 1$

1 = "Spam"
0 = "Not spam"

- Models P(y=1|x), the probability of *y=1* given *x*

$$\hat{\mathbf{P}}_\theta\left(y = 1 \mid x\right) = \sigma(\phi(x)^T\theta) = \frac{1}{1 + \exp\left(-\phi(x)^T\theta\right)}$$

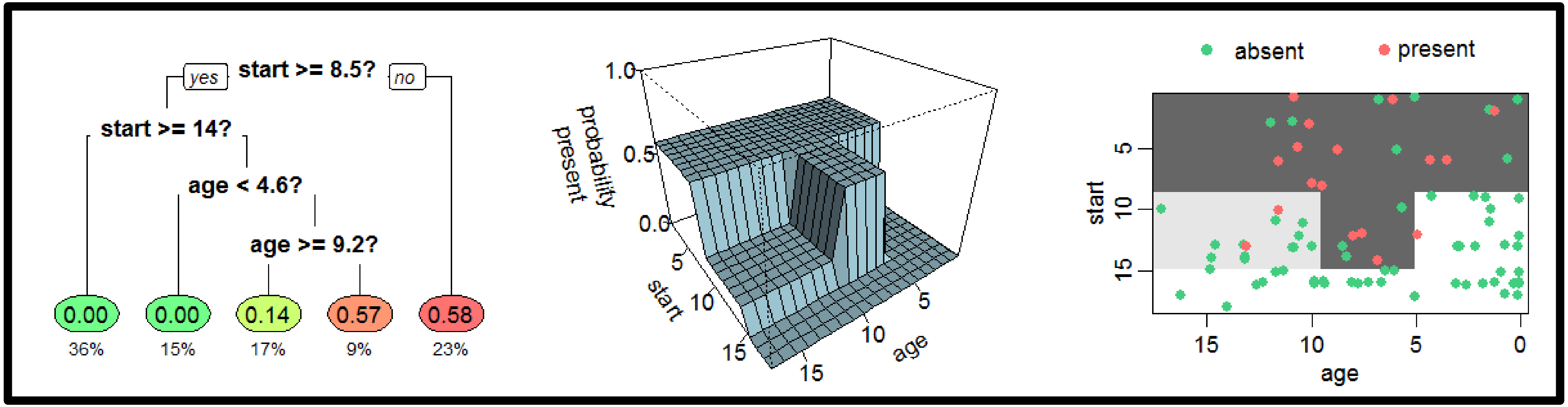
Model Architectures

[Some of the slides in this section were cannibalized from Elena Zheleva at UIC, and by the transitive property from the Berkeley DS 100 team, Marine Carpuat, Lise Getoor, Brian Ziebart. Please do not further distribute. Mistakes are my own.]

Some ML Model Architectures

- Regression models
- Decision trees
- Support Vector Machines (SVMs)
- Deep neural networks
- Many, many others:
  - PGM, genetic algorithms…

Example Decision Tree



Taken from https://en.wikipedia.org/wiki/Decision_tree_learning#/media/File:Cart_tree_kyphosis.png

The Fast Fashion of Model Architectures

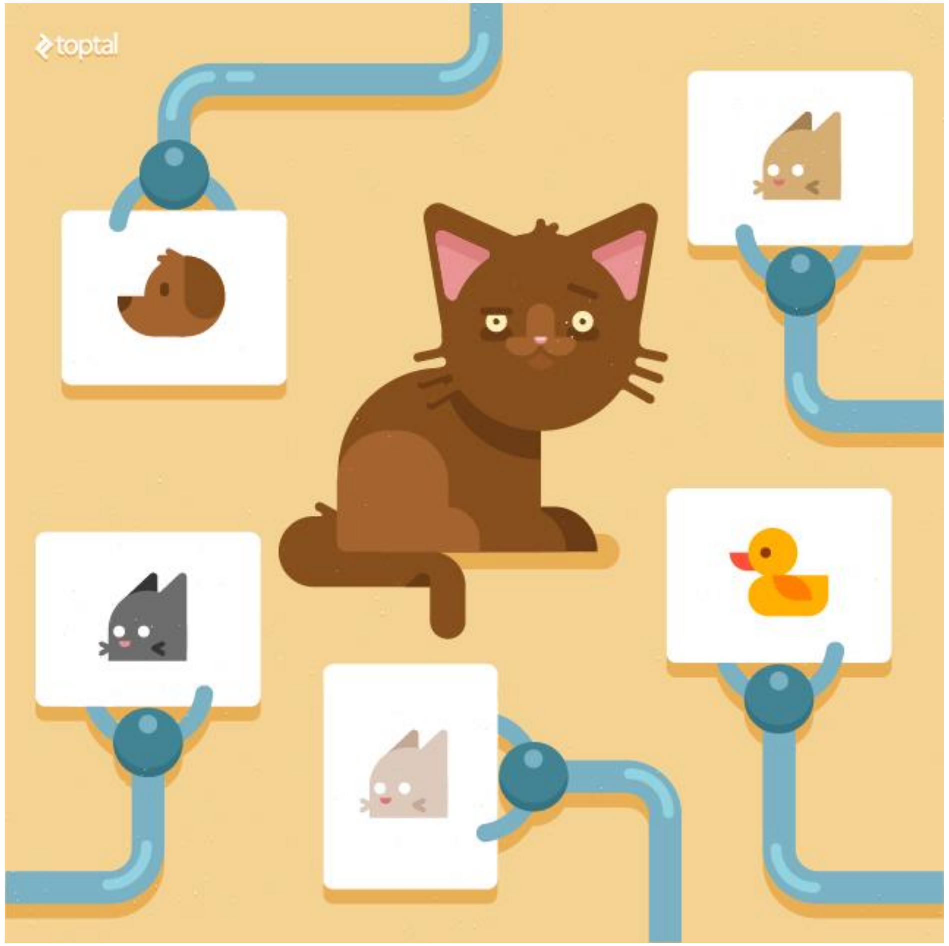
- **Support vector machine** (Boser et al. 1992)
  - Learning is **convex** (globally optimal weights)
  - Research shifted away from neural networks to SVMs / Kernel Methods
- SVMs are good for medium-large data.
- What about **REALLY BIG** data?

Ensemble Methods

[Some of the slides in this section were cannibalized from Elena Zheleva at UIC, and by the transitive property from the Berkeley DS 100 team, Marine Carpuat, Lise Getoor, Brian Ziebart. Please do not further distribute. Mistakes are my own.]

Ensemble Methods

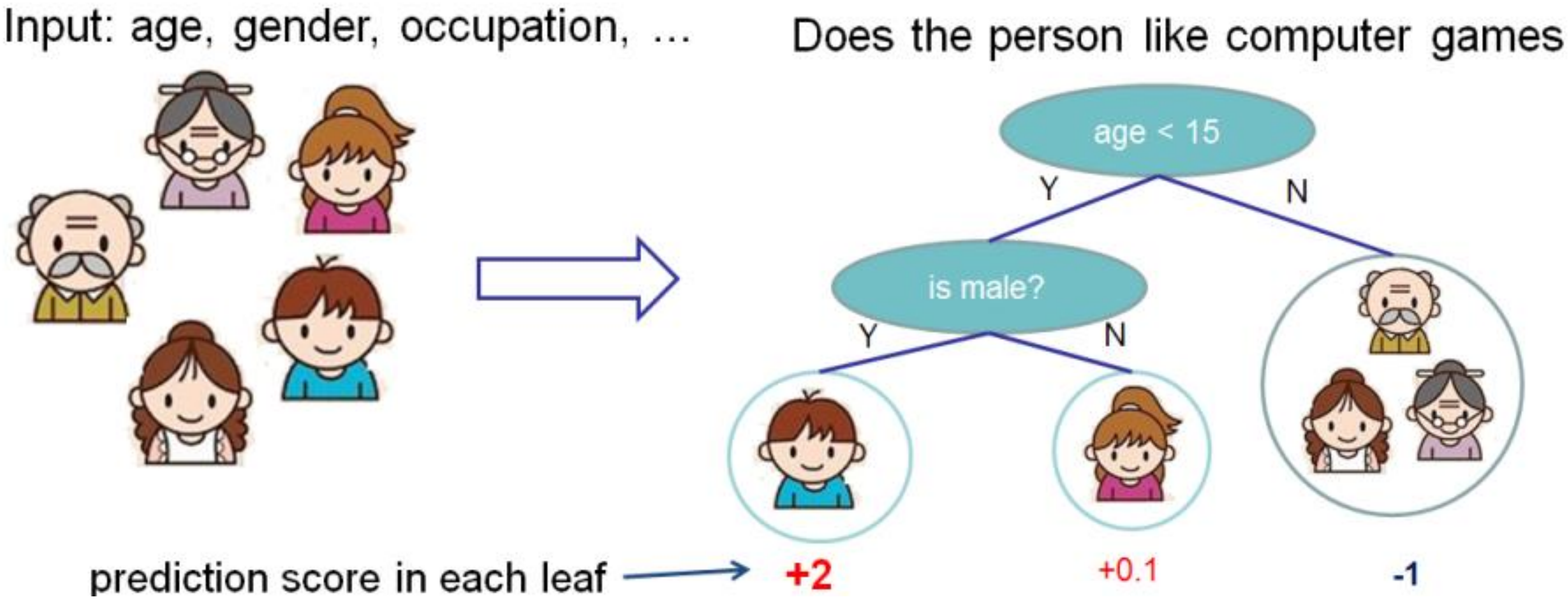
- Simplest approach:
  1. Generate **multiple classifiers**
  2. Each votes on test instance
  3. Take majority as classification
- Classifiers can be different due to
  - different sampling of training data
  - randomized parameters within the classification algorithm
  - inductive bias (e.g, decision tree + perceptron + kNN)



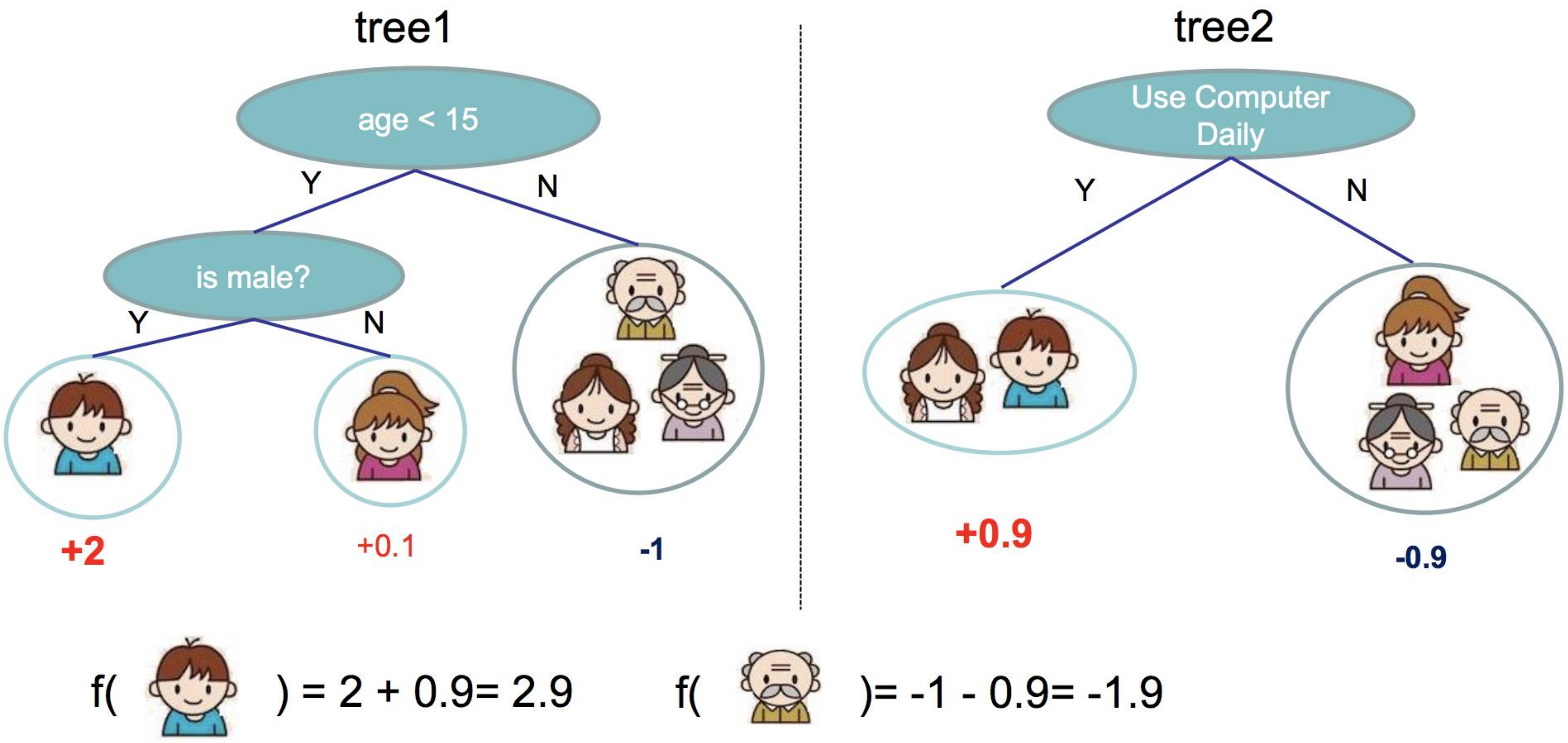
Random Forests

- Definition: Ensemble of decision trees
- Algorithm:
  - Divide training examples into multiple training sets (bagging)
  - Train a decision tree on each set
    - randomly select subset of variables to consider
  - Aggregate the predictions of each tree to make classification decision
    - e.g., can choose mode (most often) vote

Regression Tree Ensemble



Regression Tree Ensemble



Prediction of is sum of scores predicted by each of the tree

XGBoost

- Developed by Chen and Guestrin (2016)
- Relies on gradient boosting

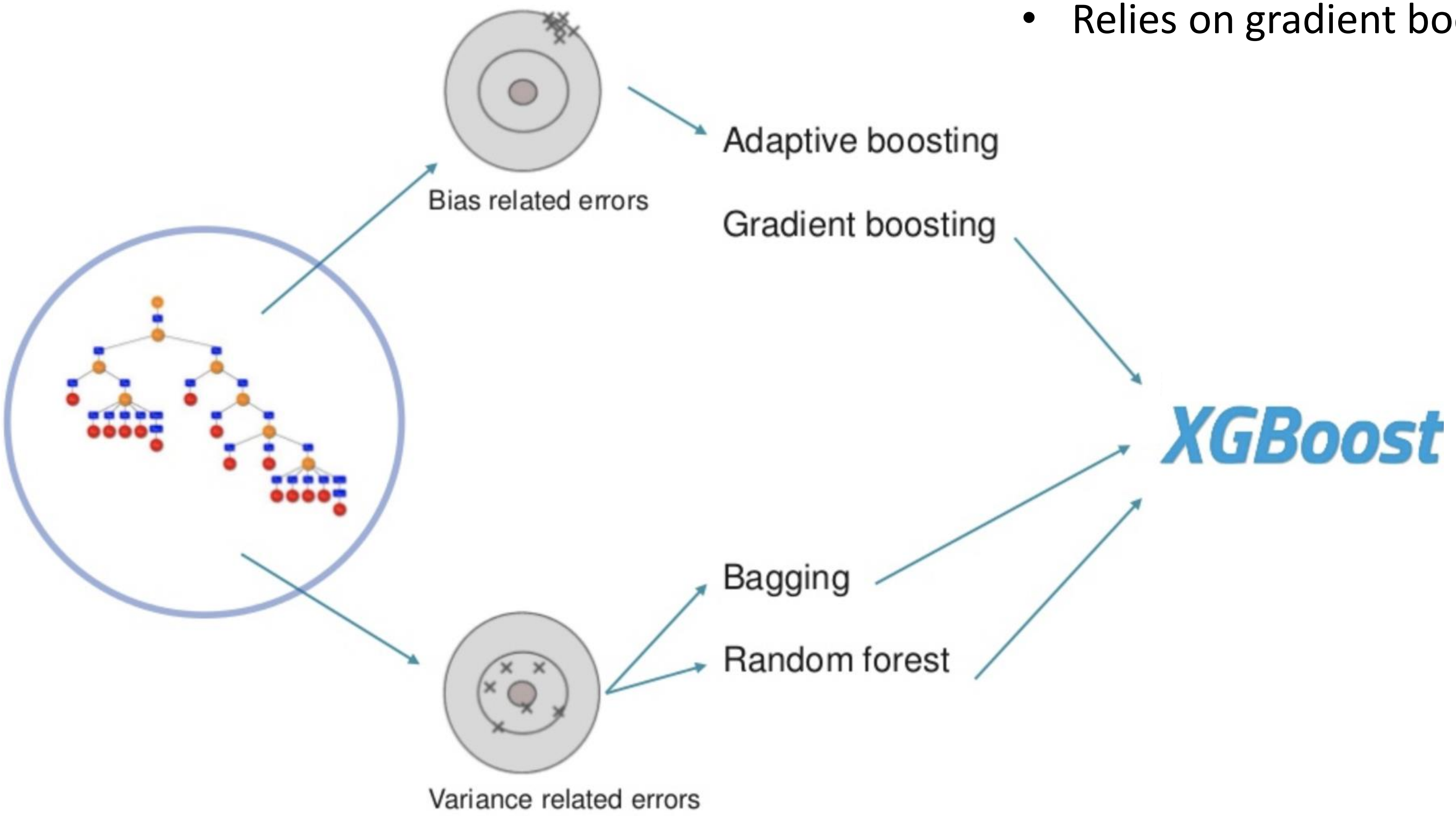


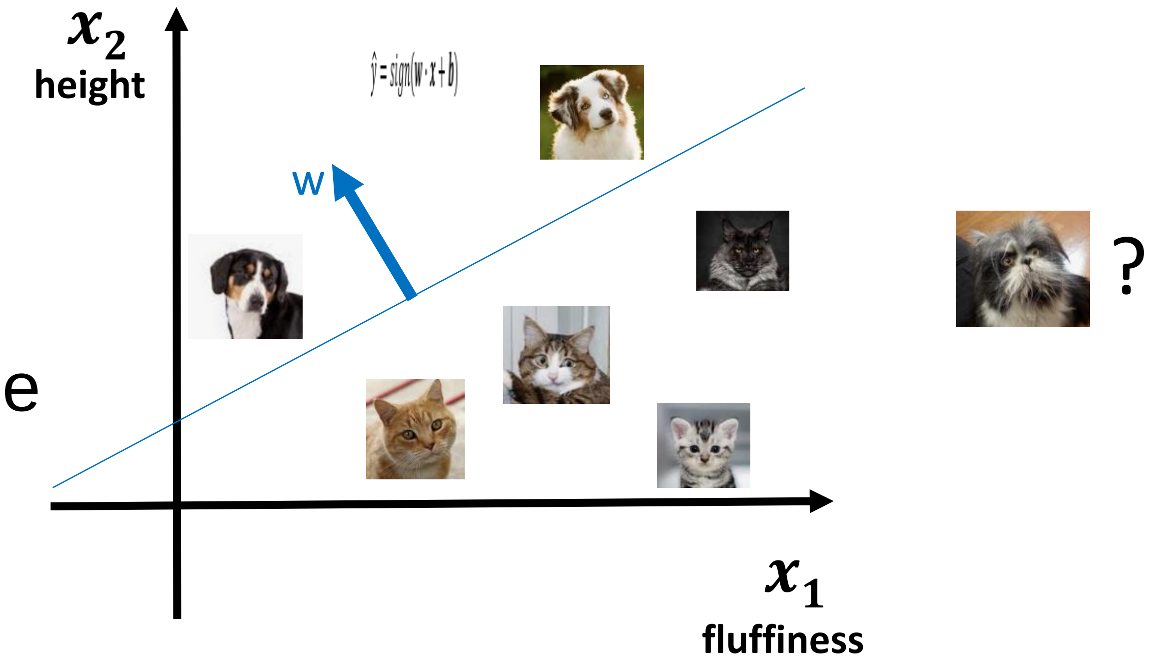
Figure: https://www.slideshare.net/JaroslawSzymczak1/xgboost-the-algorithm-that-wins-every-competition

Neural Networks

[Some of the slides in this section were cannibalized from Elena Zheleva at UIC, and by the transitive property from the Berkeley DS 100 team, Marine Carpuat, Lise Getoor, Brian Ziebart. Please do not further distribute. Mistakes are my own.]

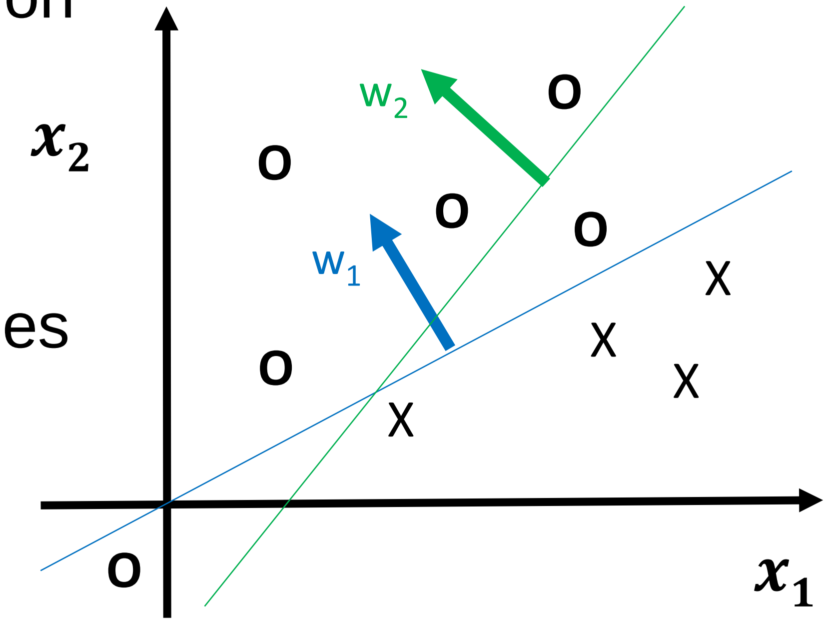
Predecessor: Perceptron (1958)

- Assume decision boundary is a hyperplane
- Training = find a hyperplane w that separates positive from negative examples
- Testing = check on which side of the hyperplane examples fall
- Classifier = hyperplane that separates positive from negative examples
- See https://en.wikipedia.org/wiki/Perceptron



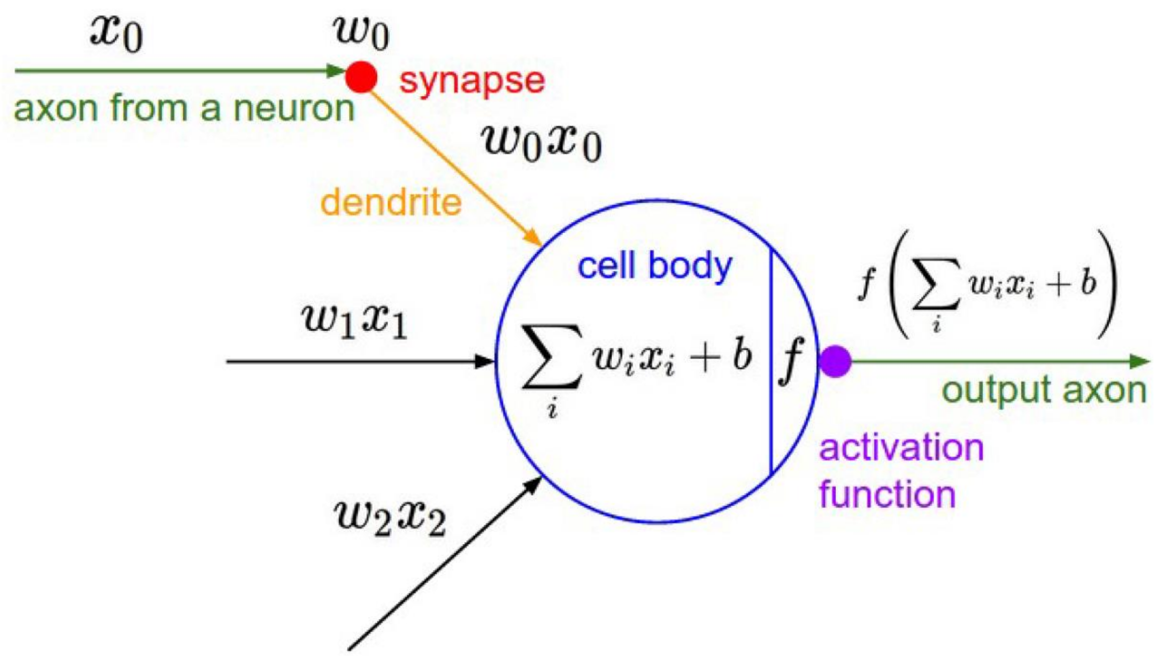
Neural Networks

- We can think of neural networks as combination of multiple linear models (perceptrons)
  - Multilayer perceptron
- Why would we want to do that?
  - Discover more complex decision boundaries
  - Learn combinations of features



Mathematical Model of a Neuron

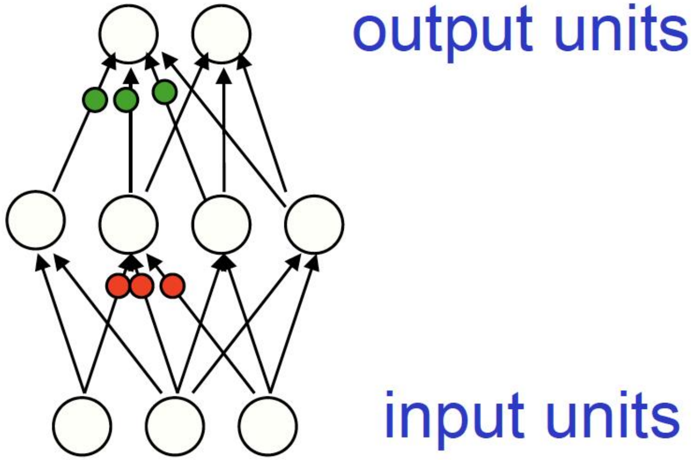
- We can think of neural networks as combination of multiple perceptrons
  - Hidden features define functions of the inputs, computed by neurons
  - Artificial neurons are called units
  - Vanilla perceptron: activation function is sign(z)

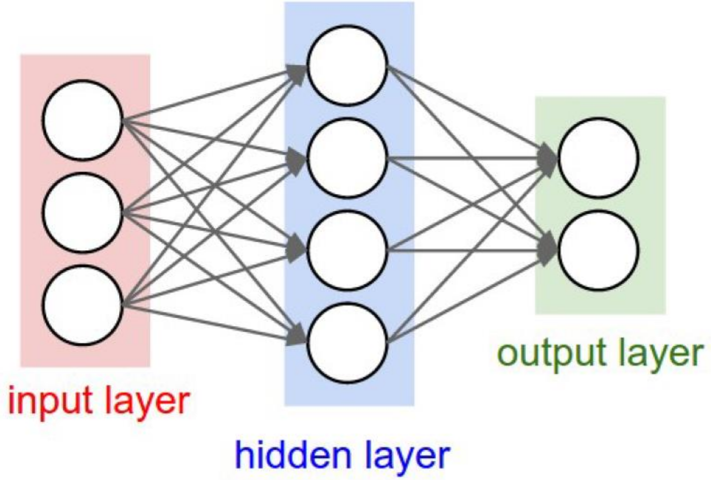


$$z = \sum_{d=1}^{D} w_d \, x_d + b$$

Neural Network Architecture

- Neural network with one layer of four hidden units:





  - Figure: Two different visualizations of a 2-layer neural network. In this example: 3 input units, 4 hidden units (layer 1) and 2 output units (layer 2)
- Each unit computes its value based on linear combination of values of units that point into it, and an activation function

Figure: http://cs231n.github.io/neural-networks-1/

Neural Network Architecture

- Going deeper: a 3-layer neural network with two layers of hidden units
- N-layer neural network:
  - N-1 layers of hidden units
  - One output layer

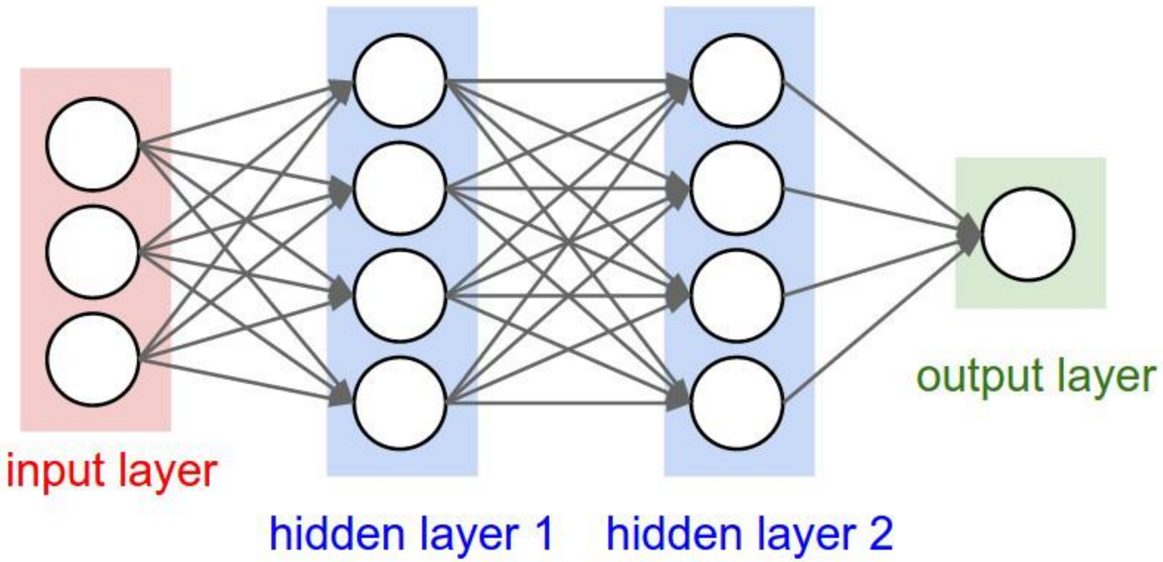
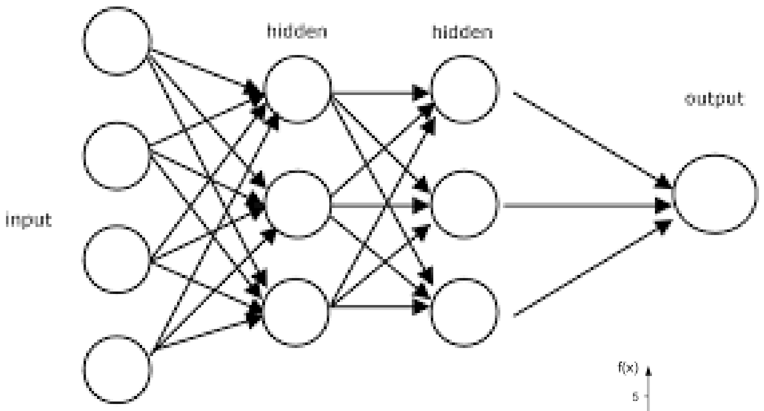


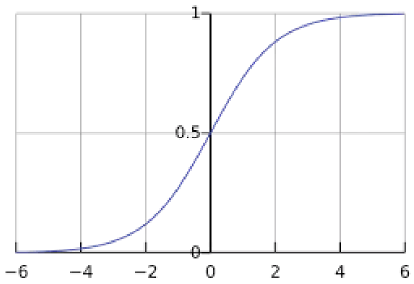
Figure : A 3-layer neural net with 3 input units, 4 hidden units in the first and second hidden layer and 1 output unit

Figure: http://cs231n.github.io/neural-networks-1/

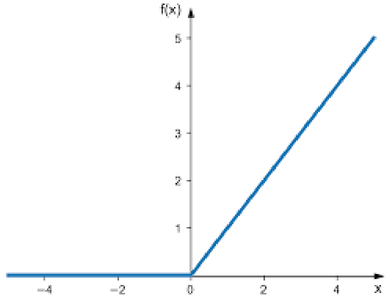
Neural Networks at 10,000 Feet

- Y = f(X)
  - F may be constructed by combining different functions
    - $\mathbf{h}^1 = g^1 (W^1 \mathbf{x} + b^1)$
    - $h^2 = g^2 (W^2 \mathbf{h}^1 + b^2)$
    - …
- Activation functions
  - Softmax
  - Relu
  - And many many more…
- Optimizers





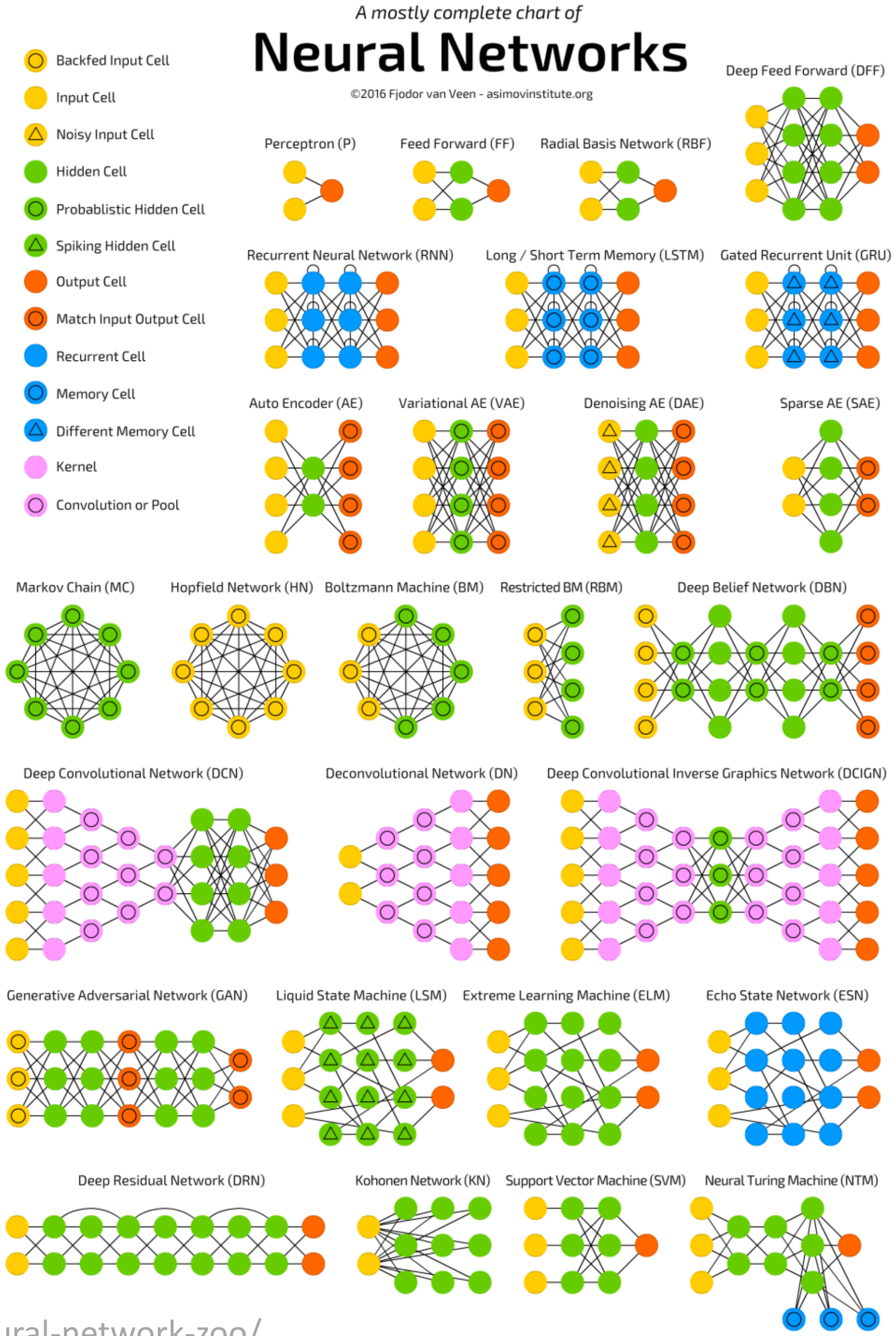
Softmax



Relu

Neural Networks: Backpropagation

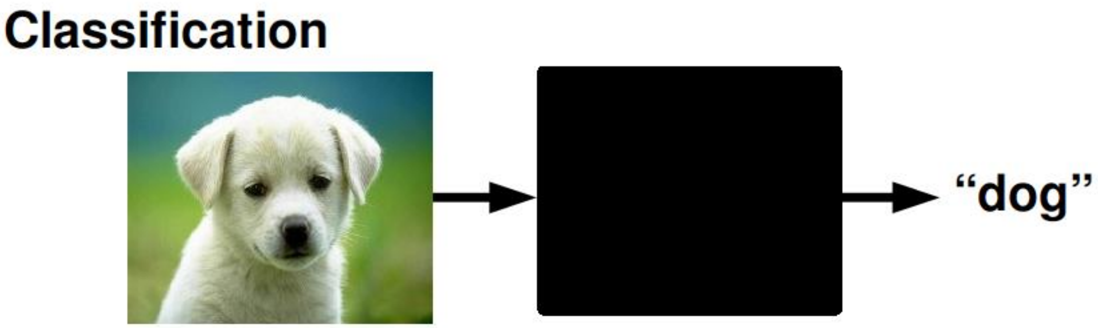
- Goal: learn the weights of each layer
- Using backpropagation algorithm
  - Forward pass = prediction/inference
  - Backward pass = learning
    - Convert discrepancy between each output and its target value into an error derivative
    - Compute error derivatives in each hidden layer from error derivatives in layer above
- The optimization function is non-convex



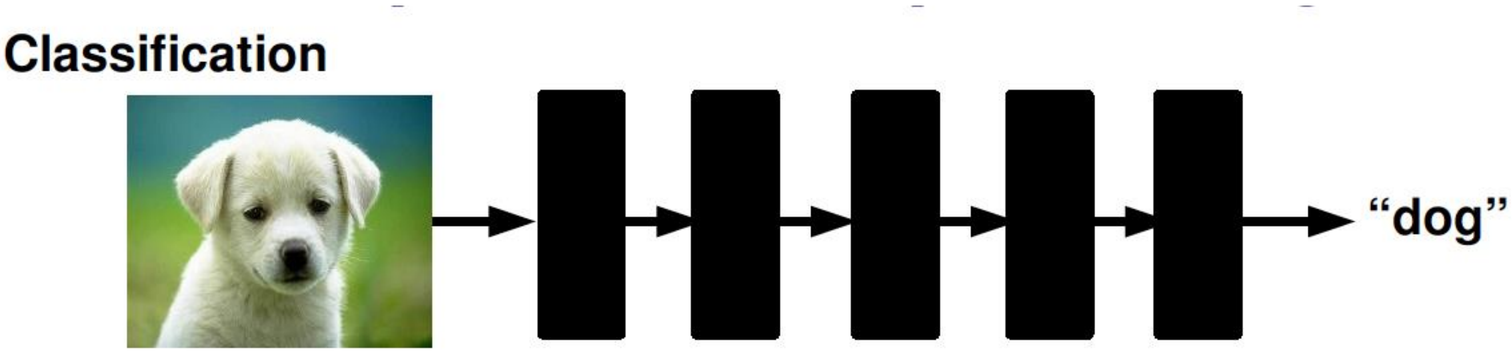
Taken from http://www.asimovinstitute.org/neural-network-zoo/

“DEEP” Learning?

- Supervised learning



- Supervised deep learning



Pictures: M. Ranzato