Artificial Intelligence Methods for Social Good

Lecture 4:

Basics of Machine Learning

17-537 (9-unit) and 17-737 (12-unit)

Fei Fang

feifang@cmu.edu

Reminder

- Paper Reading Assignment 1 (PRA1)
  - Due 1/25, 10 pm
- Project proposal
  - Due 1/30, 10pm
- HW1
  - Due 2/1, 10pm

Outline

- Overview of Machine Learning
- Regression
  - Linear Regression
- Classification
  - Decision Tree
  - Ensemble Method
- Exercise: Social Bot Detection
- Feedforward Neural Network
- Discussion

Learning Objectives

- Understand the concept of
  - Supervised learning
  - Train/test set, Cross validation
  - Regression, Classification
  - Decision Tree, Ensemble, Bagging, Random Forest
  - Feedforward Neural Network, Backpropagation
- Briefly describe Stochastic Gradient Descent
- Describe evaluation criteria for classification
  - Error, Accuracy, Precision, Recall, F1, AUC
- Know how to apply the algorithm/solver/package for classification

What is Machine Learning (ML)

- Data → Intelligence
- Tom Mitchell: "A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

$$\begin{tikzcd} \overset{\text{data}}{X} \arrow[r, "f(\cdot)"] & \overset{\text{label}}{Y} \end{tikzcd}$$

Fei / Not Fei

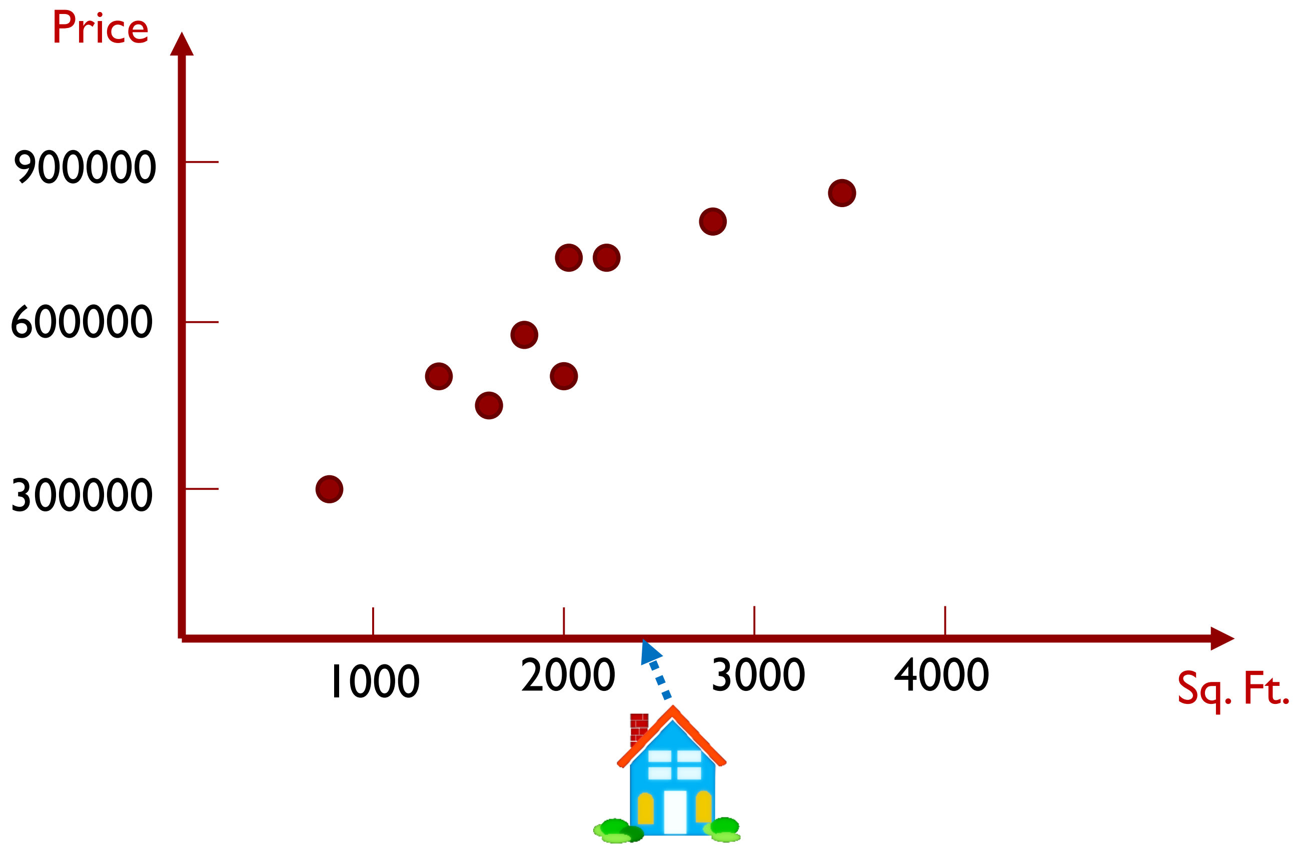
"Hello World"

Different Types of ML Problems

- Supervised learning
  - Learning from a set of labeled data
  - Regression (numerical, continuous-valued label)
  - Classification (categorical, discrete-valued label)

Regression Example

- Predict house price



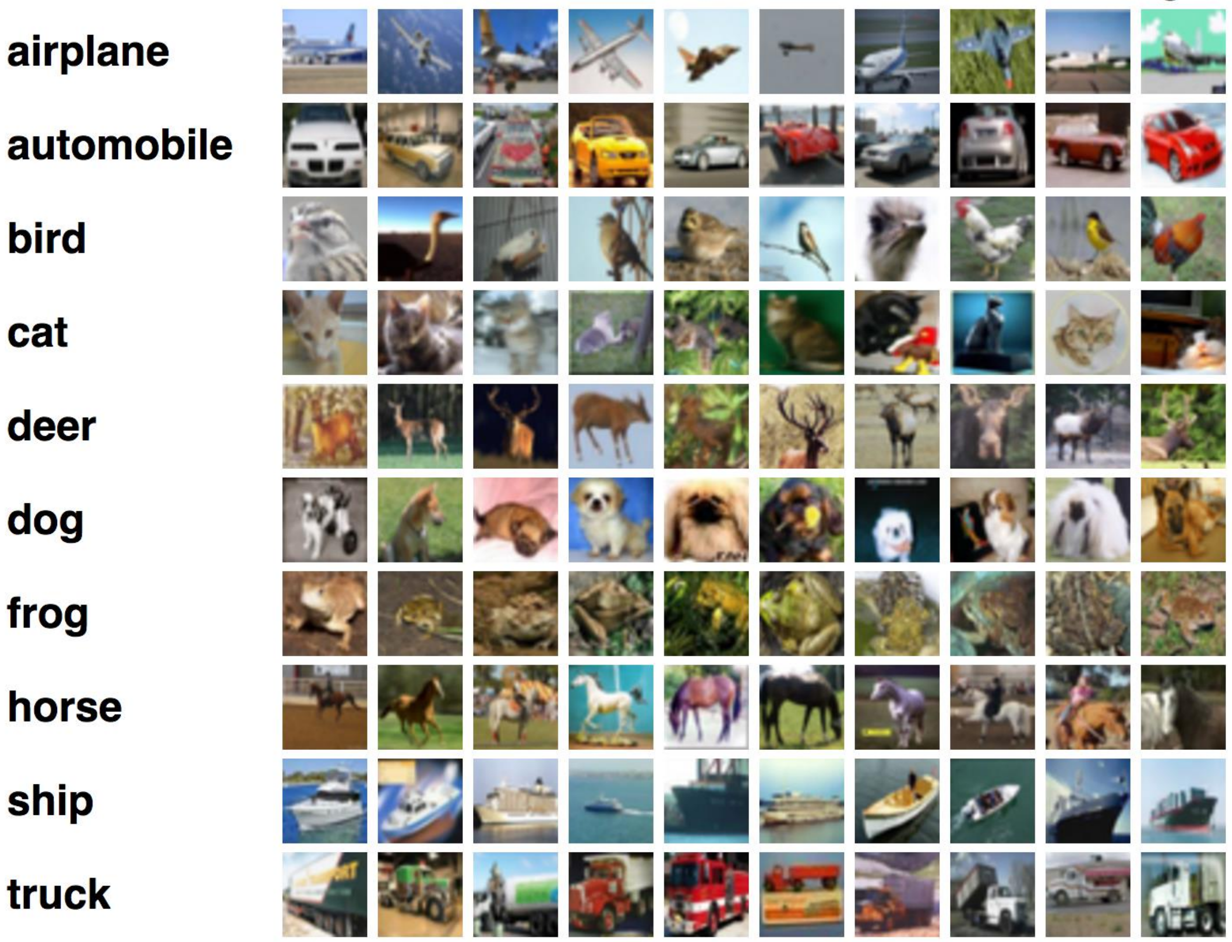
Classification Example

Cat or Dog?

Chihuahua or Muffin?

Classification Example

- More than two classes



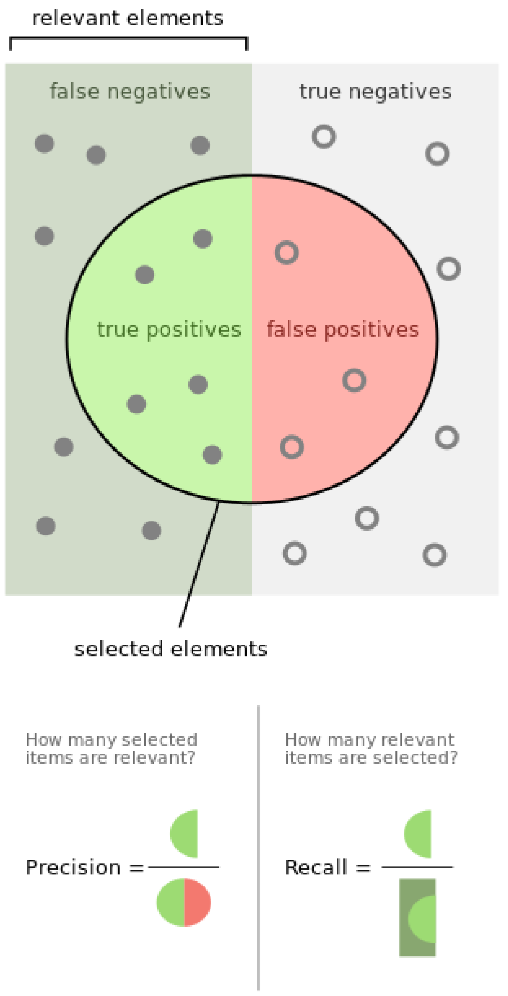
Classification

- Performance Measure P
  - Accuracy = #correct predictions / #instances
  - Error = #mistakes / #instances
- Alice has a rarely used email address and she has found that 90% of the emails sent to this address are spam. She asks you to design a spam filter for her. You designed one that labels every email as spam. What is the expected accuracy of this classifier?

Other Evaluation Criteria for Classification Task

- Confusion matrix

		Predicted Label	
		+	-
True Label	+	True Positive (TP)	False Negative (FN)
	-	False Positive (FP)	True Negative (TN)



- Precision = TP / (TP+FP)
- Recall = TP / (TP+FN)
- F1 score: harmonic mean of precision and recall $= \frac{1}{\frac{\frac{1}{p}+\frac{1}{r}}{2}} = \frac{2pr}{p+r}$

Poll 1: Evaluation Criteria for Classification Task

- Alice has a rarely used email address and she has found that 90% of the emails sent to this address are spam. She asks you to design a spam filter for her. You designed one that labels every email as spam. What is the F1 score of this classifier (keep two digits after the decimal point)?
  - A: 0.90
  - B: 0.10
  - C: 0.95
  - D: 0.85
  - E: I don't know

Supervised Learning Workflow

- A typical problem: Given a set of labeled data, train a model and evaluate its performance

- Workflow
  - Split the dataset into training set and test set
  - Train a model using training set
  - Test the model using test set
  - Report performance based on evaluation metrics

Q: If your evaluation shows the performance is good, and you want to deploy the model in the field (to make future predictions), what should you do?

Supervised Learning Workflow

- A typical problem: Given a set of labeled data, train a model and evaluate its performance

- Workflow
  - Split the dataset into training set and test set
  - Train a model using training set
  - Test the model using test set
  - Report performance based on evaluation metrics

Q: If your evaluation shows the performance is good, and you want to deploy the model in the field (to make future predictions), what should you do? Use all training data to train a model and use the trained for deployment

Different Types of ML Problems

- Supervised learning
  - Learning from a set of labeled data
  - Regression (numerical, continuous-valued label)
  - Classification (categorical, discrete-valued label)
- Unsupervised learning
  - Learning from a set of unlabeled data
- Semi-supervised learning
- Active Learning
- Reinforcement Learning

Outline

- Overview of Machine Learning
- Regression
  - Linear Regression
- Classification
  - Decision Tree
  - Ensemble Method
- Exercise: Social Bot Detection
- Feedforward Neural Network
- Discussion

Regression

- Given a set of labeled data $\{(\mathbf{x}_i, y_i)\}$, predict numerical, continuous-valued label for a new data point

x_i	1.0	2.0	3.5
y_i	4.1	5.98	9.0

If $x = 3$, what should be the label y?

Parametric Regression

x_i	1.0	2.0	3.5
y_i	4.1	5.98	9.0

- Build a model with a set of parameters to represent the relationship between data (input) and label (output)

  Example: $y = f(x) + \epsilon$ where
  $f(x) = ax + b$ and ϵ is a small noise
  Parameters: a and b

  Linear Regression

- Find the parameters that can "best" fit the data
- Predict $y = f(x)$ for a new data point x

Parametric Regression

- How to learn the best model (i.e., find the best parameters) to fit the data?
  - Solve an optimization problem that minimizes loss function
  - Example loss function - Residual sum of squares (RSS):
    - $RSS = \sum_i (y_i - f(x_i))^2$

x_i	1.0	2.0	3.5
y_i	4.1	5.98	9.0

Linear Regression $f(x) = ax + b$

$$\min_{a,b} \sum_{i=1}^{3} (y_i - (ax_i + b))^2$$

s.t. $a, b \in \mathbb{R}$

Is this a convex optimization problem?

Approach 1: set gradient = 0

Approach 2: use gradient descent

In practice: Call the solver

Linear Regression with Multiple Features

- Predict house price

$$f(\text{housesize}) = w_0 + w_1 \cdot \text{housesize} + w_2 \cdot \text{housesize}^2$$

$f(\text{housesize}, \#\text{bedrooms}, \#\text{bathrooms}) = a \cdot \text{housesize} + b \cdot \#\text{bedrooms} + \text{c} \cdot \#\text{bathrooms} + \text{d}$

$f(\text{housesize}, \#\text{bedrooms}, \#\text{bathrooms}) = a \cdot \log(\text{housesize}) + b \cdot \#\text{bedrooms}^2 + \text{c} \cdot \#\text{bathrooms}^2 \cdot \#\text{bedrooms}$

- Generalized linear regression model
  - $x = (x[1], x[2], \dots, x[d])$: a vector of d dimensions
  - $h_j(x)$: a known function of x (no unknown parameters)
  - $f(x) = \sum_j w_j h_j(x)$
  - $y_i = f(x_i) + \epsilon_i$

Linear Regression with Multiple Features

- Learn the best model (Find the best parameters)

$$\min_{w} \sum_{i=1}^{N} \left(y_i - \sum_{j} w_j h_j(x_i) \right)^2$$
$$\text{s.t. } w_j \in \mathbb{R}, \forall j$$

Is this a convex optimization problem?

- Approach 1: Set gradient = 0 → closed form solution
- Approach 2: Gradient descent
  - More general, Can be more efficient

Stochastic Gradient Descent

$$\min_w L(w) = \sum_{i=1}^{N} \left(y_i - \sum_j w_j h_j(x) \right)^2$$
$$\text{s.t. } w_j \in \mathbb{R}, \forall j$$

- Gradient Descent
  - $w \leftarrow w - \eta \nabla L(w)$
  - $w_k \leftarrow w_k + 2\eta h_k(x) \sum_{i=1}^{N} (y_i - \sum_j w_j h_j(x_i))$
- What if N is too large?
  - $L(w) = \sum_{i=1}^{N} L_i(w)$
- Stochastic Gradient Descent
  - $w \leftarrow w - \eta \nabla L_i(w)$
  - $w_k \leftarrow w_k + 2\eta h_k(x_i)(y_i - \sum_j w_j h_j(x_i))$

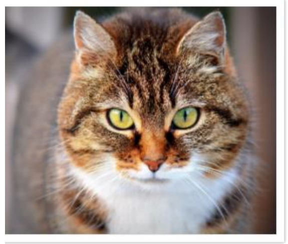
Outline

- Overview of Machine Learning
- Regression
  - Linear Regression
- Classification
  - Decision Tree
  - Ensemble Method
- Exercise: Social Bot Detection
- Feedforward Neural Network
- Discussion

Classification

- Given a set of labeled data $\{(\mathbf{x}_i, y_i)\}$, predict categorical, discrete-valued label for a new data point

$\mathbf{x}_i$			
y_i	Cat	Dog	Dog



Cat or Dog?

Decision Tree

- Widely used in the real world
- Example: Predict risk of loan (Safe vs Risky)
  - Credit score
  - Current income
  - Amount of loan
  - Type of loan
  - …

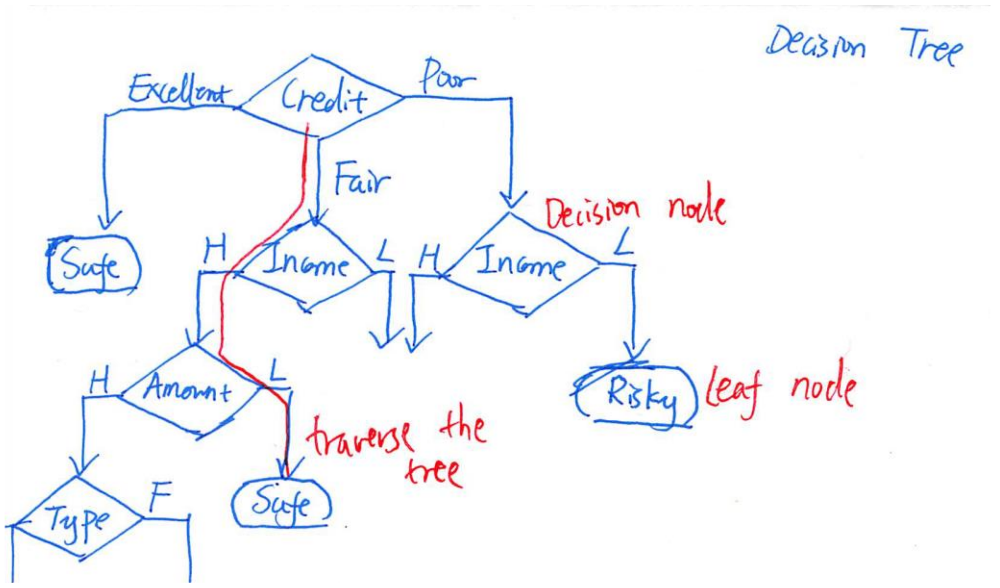
Decision Tree

- Widely used in the real world
- Example: Predict risk of loan (Safe vs Risky)
  - Credit score
  - Current income
  - Amount of loan
  - Type of loan
  - …

- A decision tree consists of
  - Decision nodes: Root node + Intermediate nodes
  - Leaf nodes / End nodes

Decision Tree

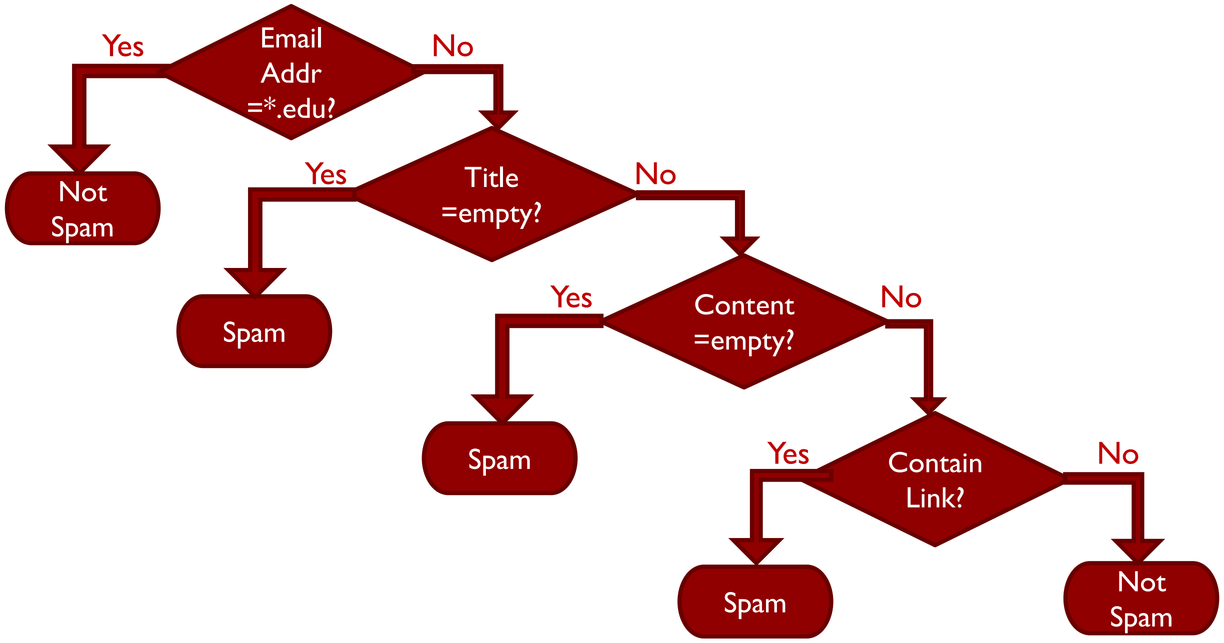
- Widely used in the real world
- Example: Predict risk of loan (Safe vs Risky)
  - Credit score
  - Current income
  - Amount of loan
  - Type of loan
  - …



- A decision tree consists of
  - Decision nodes: Root node + Intermediate nodes
  - Leaf nodes / End nodes

Poll 2: Decision Tree

- Given a spam filter represented by the example decision tree below, if I send an email using email address xxx@123.com with title "Hello" and content "I love Pittsburgh", will it be classified as spam?
  - A: Yes
  - B: No
  - C: I don't know



Decision Tree

- Predict output given a hard-coded decision tree
  - Input → Traverse the tree → Output

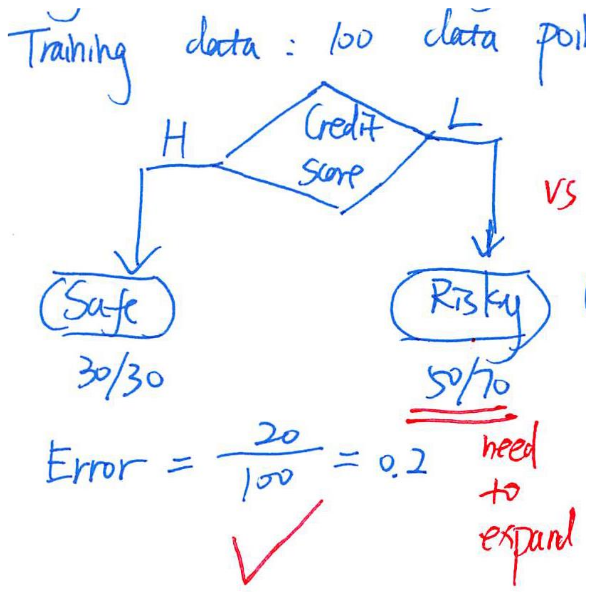
- Learn/Train a decision tree
  - Given a set of input data and output label, find a proper decision tree

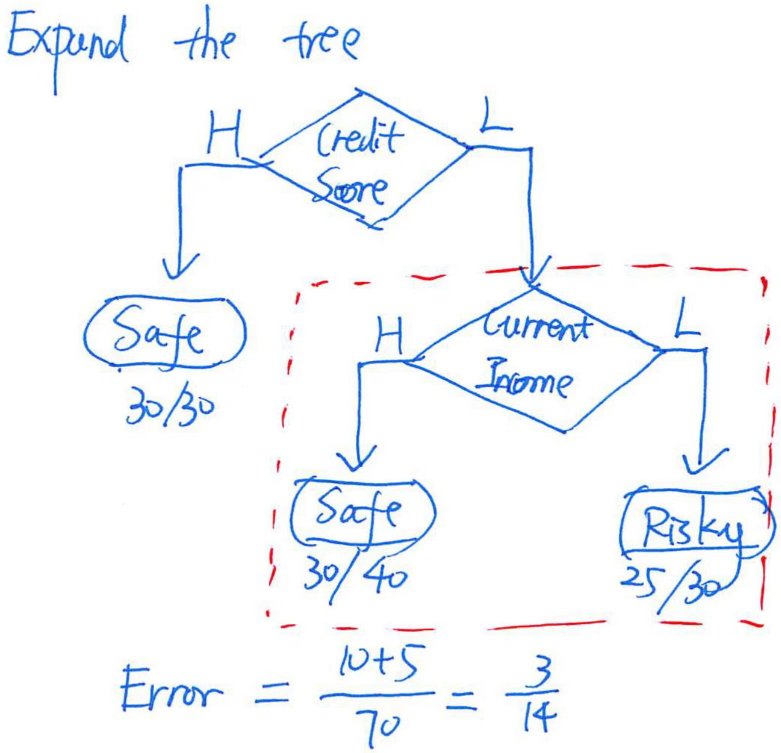
Algorithm: Greedy Decision Tree Learning

- Step 1: Start with an empty tree
- Step 2: Choose a feature and split on the feature
- Step 3: Making predictions
- Go back to Step 2

Algorithm: Greedy Decision Tree Learning

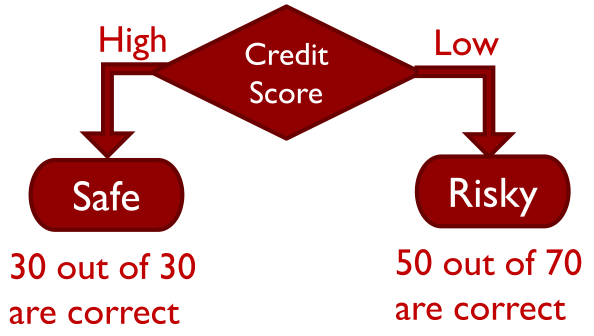
- Step 1: Start with an empty tree
- Step 2: Choose a feature and split on the feature
- Step 3: Making predictions
- Go back to Step 2

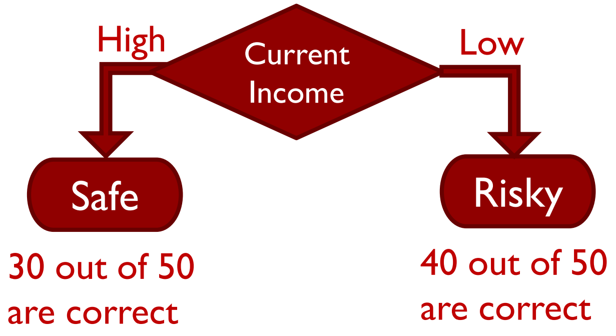




Algorithm: Greedy Decision Tree Learning

- Problem 1: Feature split selection
  - Greedily choose the feature to minimize error when assigning majority label

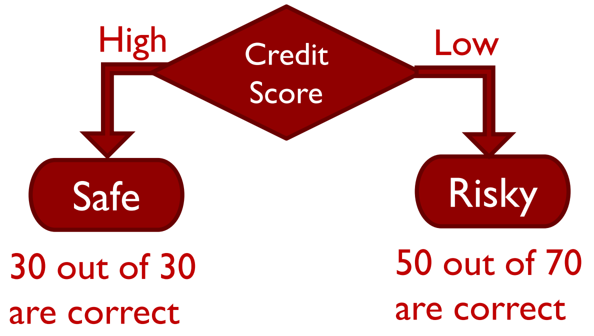




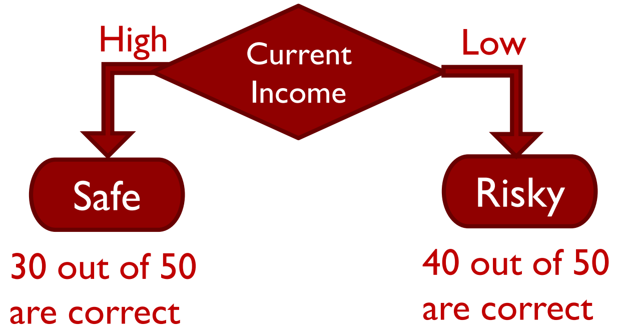
Q: Which feature split should we choose?
What is the corresponding error?

Algorithm: Greedy Decision Tree Learning

- Problem 1: Feature split selection
  - Greedily choose the feature to minimize error when assigning majority label



Error = 20/100



Error = (20+10)/100

Q: Which feature split should we choose? What is the corresponding error?

Algorithm: Greedy Decision Tree Learning

- Problem 1: Feature split selection
  - Greedily choose the feature to minimize error when assigning majority label

- Problem 2: Stopping condition
  - 1) All data in the node have same y value
  - 2) Already split on all features

Q1: How to deal with real-valued features?

Q2: How to deal with multi-class labels?

Q3: Can Decision Trees output probability distribution over classes?

Algorithm: Greedy Decision Tree Learning

- Problem 1: Feature split selection
  - Greedily choose the feature to minimize error when assigning majority label

- Problem 2: Stopping condition
  - 1) All data in the node have same y value
  - 2) Already split on all features

Q1: How to deal with real-valued features?

Q2: How to deal with multi-class labels?

Q3: Can Decision Trees output probability distribution over classes?

But prob. dist. output from a single decision tree is not so useful. Why?

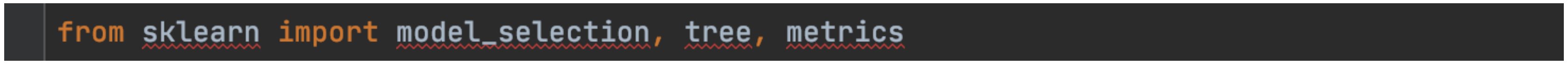
Decision Tree

- Pro
  - Intuitive
  - Simple to implement
  - Interpretable prediction
- Con
  - Hard to represent complex decision boundary

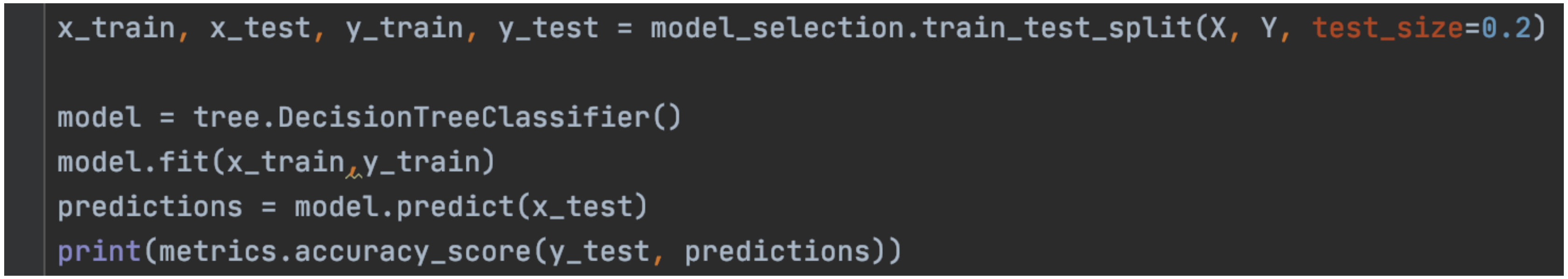
Decision Tree for Classification in Practice

- Code packages: scikit-learn (Python), fitctree (MATLAB)

With scikit-learn 0.23



```
from sklearn import model_selection, tree, metrics
```



```
x_train, x_test, y_train, y_test = model_selection.train_test_split(X, Y, test_size=0.2)

model = tree.DecisionTreeClassifier()
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(metrics.accuracy_score(y_test, predictions))
```

Note: decision tree model in scikit-learn requires all inputs to be numeric, so we need to convert categorical features into numeric by encoding the categories

Outline

- Overview of Machine Learning
- Regression
  - Linear Regression
- Classification
  - Decision Tree
  - Ensemble Method
- Exercise: Social Bot Detection
- Feedforward Neural Network
- Discussion

Ensemble Method

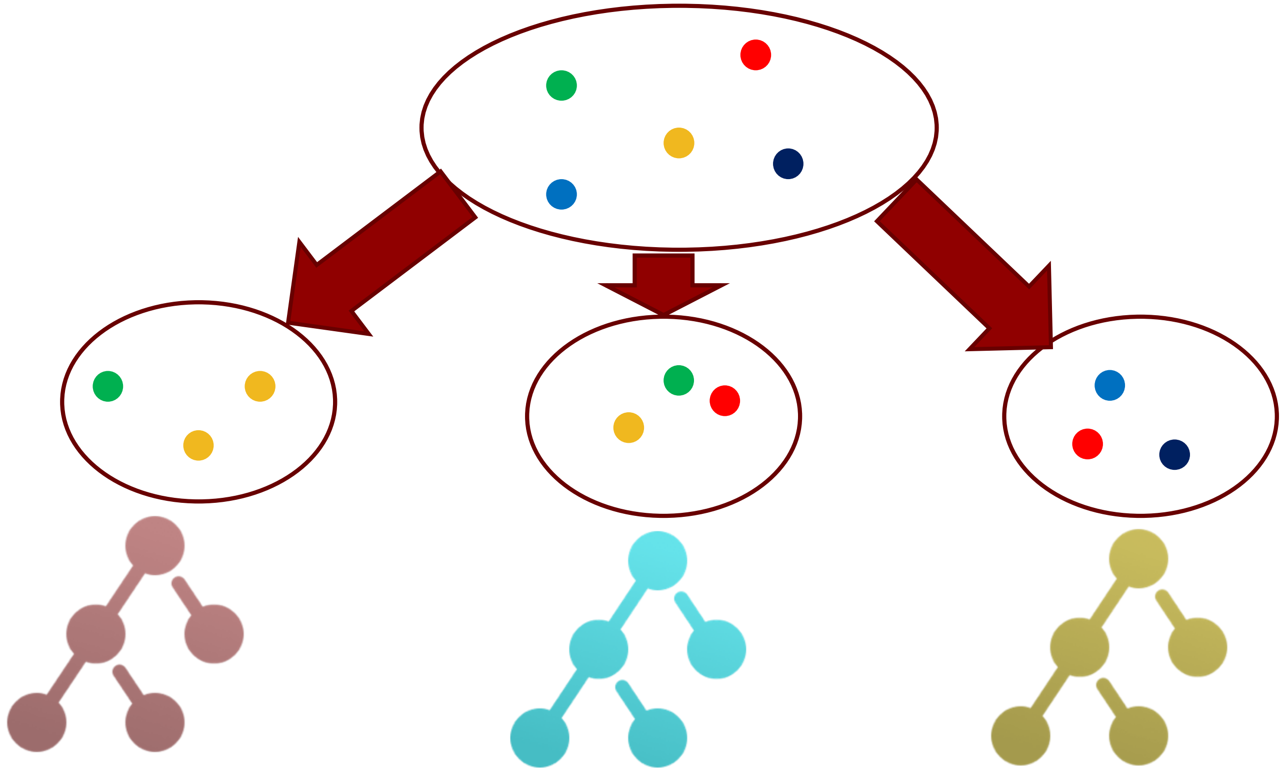
- Combine the predictions of several base estimators
- Improve generalizability / robustness over a single estimator

- Averaging methods
  - Build several estimators independently and then to average their predictions
  - Example: Bagging Decision Tree, Random Forest

- Boosting methods
  - Base estimators are built sequentially
  - Combine several weak models to produce a powerful ensemble

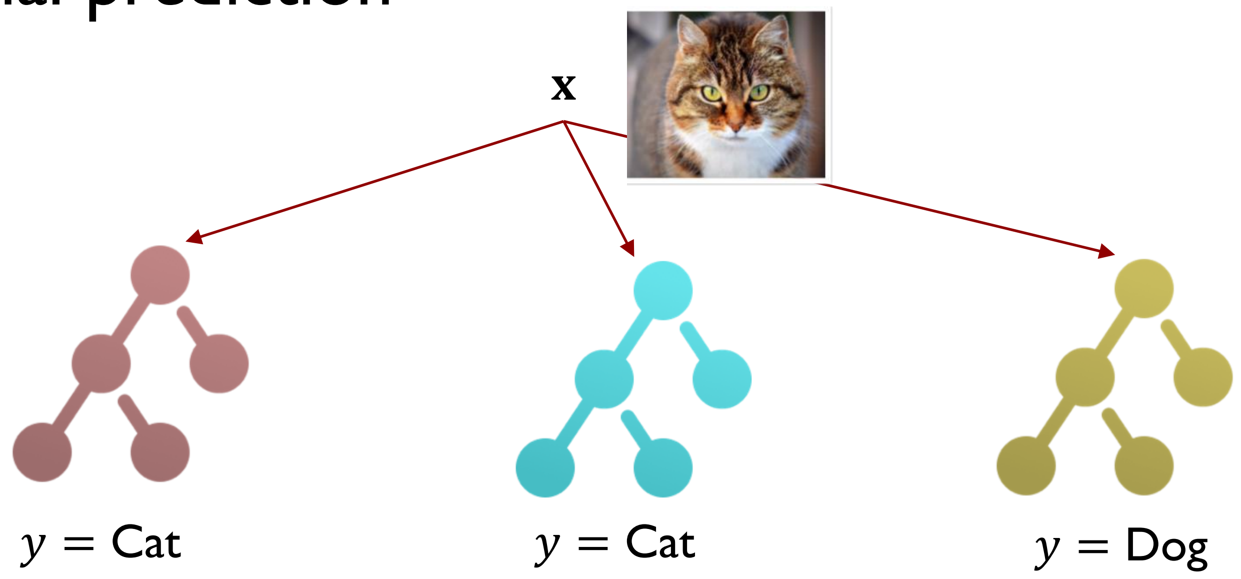
Bagging Decision Tree

- Step 1: Train on random subsets of the original training set to get multiple decision trees
  - Samples are drawn with replacement



Bagging Decision Tree

- Step 2: Aggregate their individual predictions to form a final prediction



Final prediction=? Voting!

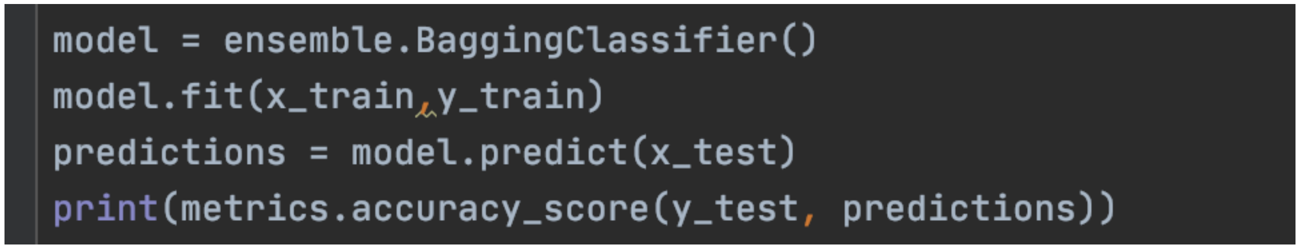
Q1: Can the final prediction be prob. dist?

Q2: What if the base estimators output prob. dist?

Q3: Can we use other voting rules?

Bagging Decision Tree

- A very simple way to improve performance
- Work best with strong and complex models as base estimators (e.g., fully developed decision trees)
- Code packages: scikit-learn (Python), TreeBagger (MATLAB)

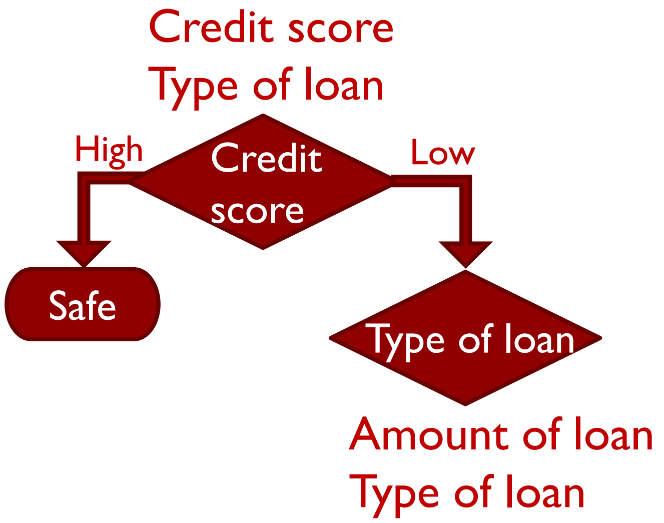


```
model = ensemble.BaggingClassifier()
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(metrics.accuracy_score(y_test, predictions))
```

Random Forest

- Similar to bagging decision tree
- The only difference is, when splitting each node during the construction of a tree, the best split is found using a random subset of features
  - Typically $\sqrt{p}$ features are used in each split

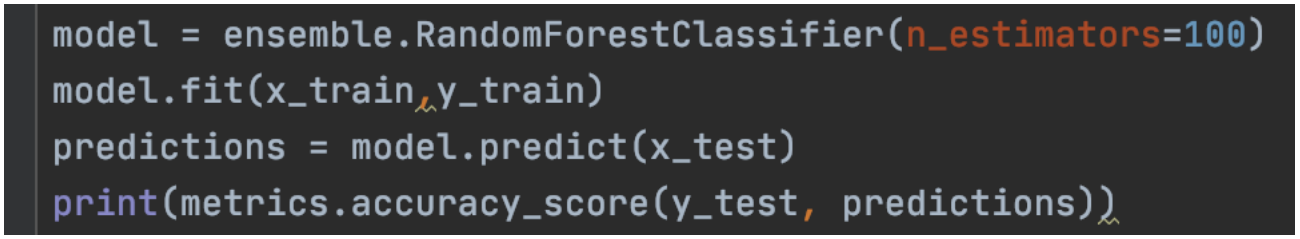
Credit score
Current income
Amount of loan
Type of loan



Random Forest

- Typically output a prob. dist. calculated as the mean of base estimators

- Code packages: scikit-learn (Python), fitcensemble (MATLAB)



```
model = ensemble.RandomForestClassifier(n_estimators=100)
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(metrics.accuracy_score(y_test, predictions))
```

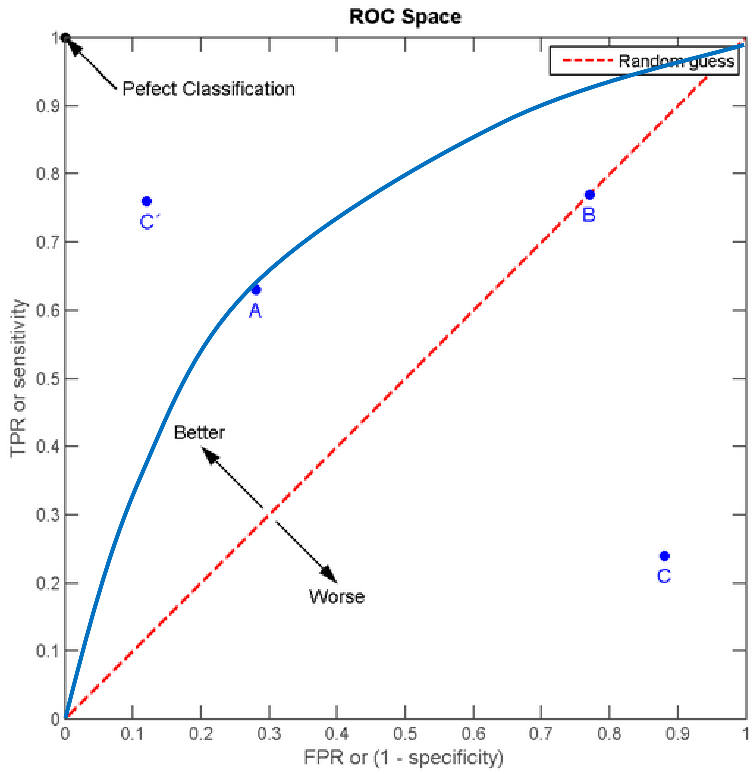
AUC (Area Under ROC Curve)

- When the final prediction is a prob. dist. over classes, how to evaluate the learned classification model?
  - Error, Accuracy, Precision, Recall, F1 all require binary predictions
  - Idea: use a threshold to convert the prob. dist. into binary predictions
  - Different threshold lead to different predictions

$$\begin{tikzcd}
 & \text{Threshold}= 0.5 \arrow[r] & y' = 1 \\
y = 0.6 \arrow[ur] \arrow[dr] & & \\
 & \text{Threshold}= 0.7 \arrow[r] & y' = 0
\end{tikzcd}$$

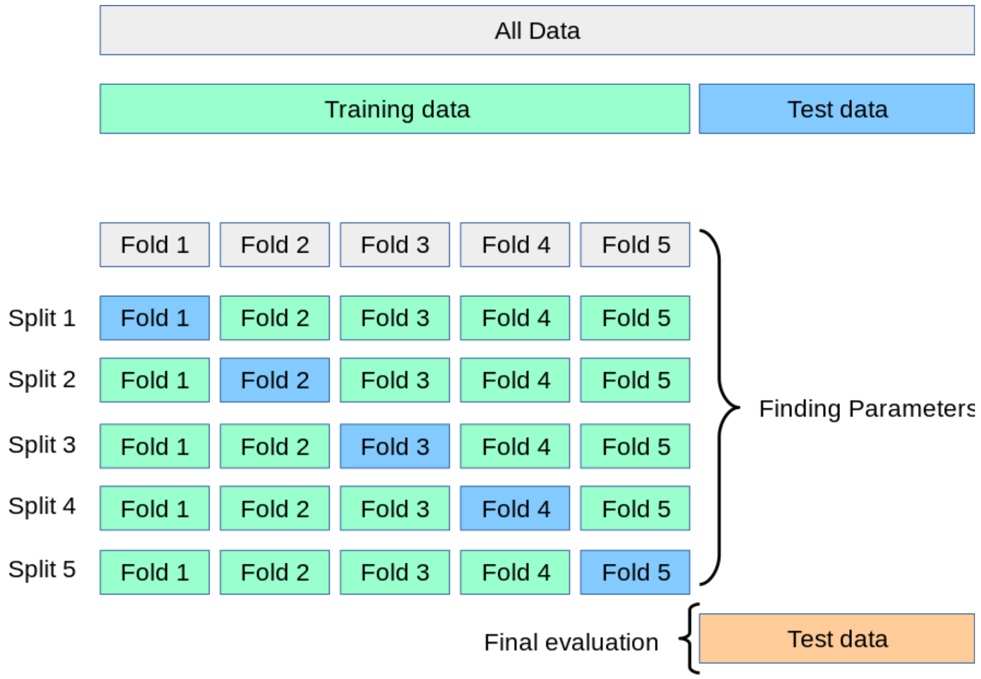
AUC (Area Under ROC Curve)

- ROC (Receiver Operating Characteristic) curve
  - Vary the threshold
  - False positive rate (FPR) vs True positive rate (TPR)
  - FPR=fall-out=probability of false alarm= FP / (FP+TN)
  - TPR=recall=sensitivity=probability of detection= TP / (TP+FN)
- AUC: area under the curve



Select Hyperparameters

- Hyperparameters
  - How many base decision trees should I use?
  - How many features should I sample at each node?
- Select hyperparameters using k-fold cross validation
  - Split the training dataset into k smaller sets (called k folds)
  - For each parameter value
    - For each of the fold
      - Train a model using the other k-1 folds
      - Test the model using this fold
    - Compute average performance
  - Choose best parameter value



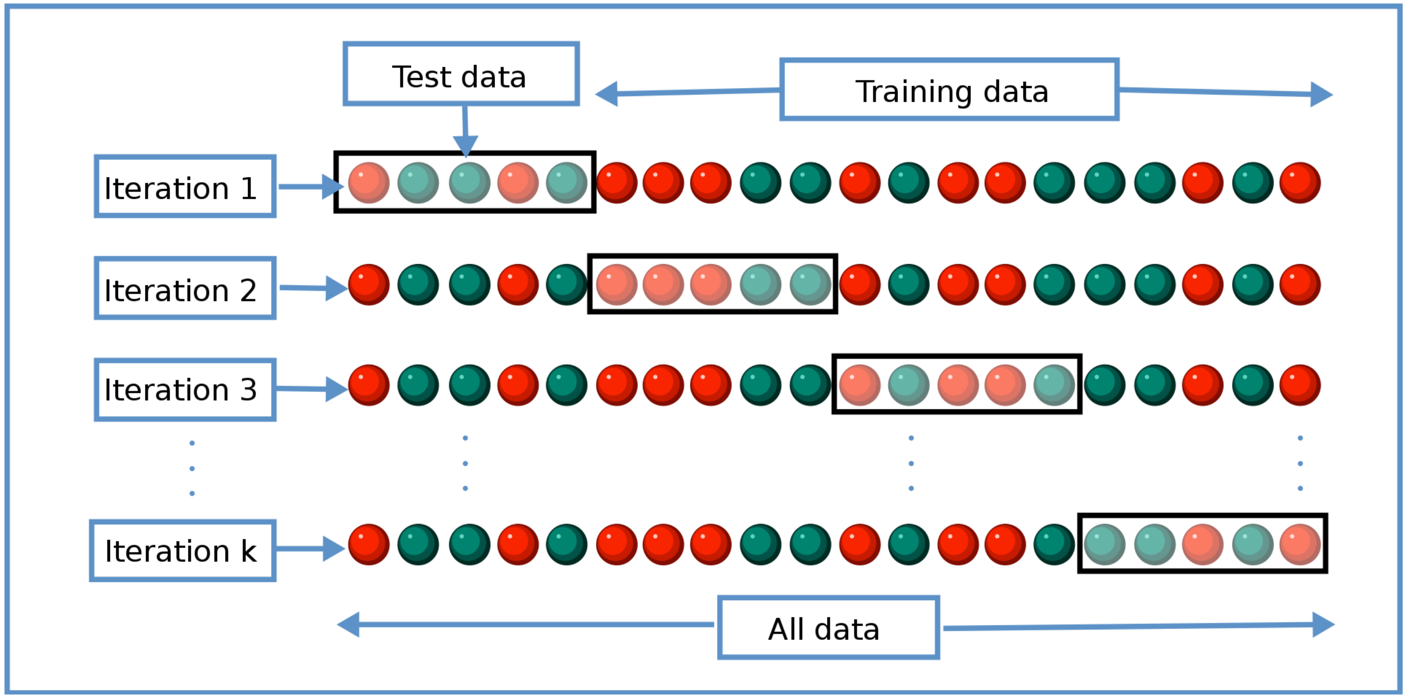
https://scikit-learn.org/stable/_images/grid_search_cross_validation.png

Recall: Supervised Learning Workflow

- A typical problem: Given a set of labeled data, train a model and evaluate its performance
- Workflow
  - Split the dataset into training set and test set
  - Train a model using training set
  - Test the model using test set
  - Report performance based on evaluation metrics

K-Fold Cross-Validation

- Different training/test set splits lead to varying performance
- K-Fold cross-validation can also better evaluate the learning algorithm



K-Fold Cross-Validation

- A typical problem: Given a set of labeled data, train a model and evaluate its performance

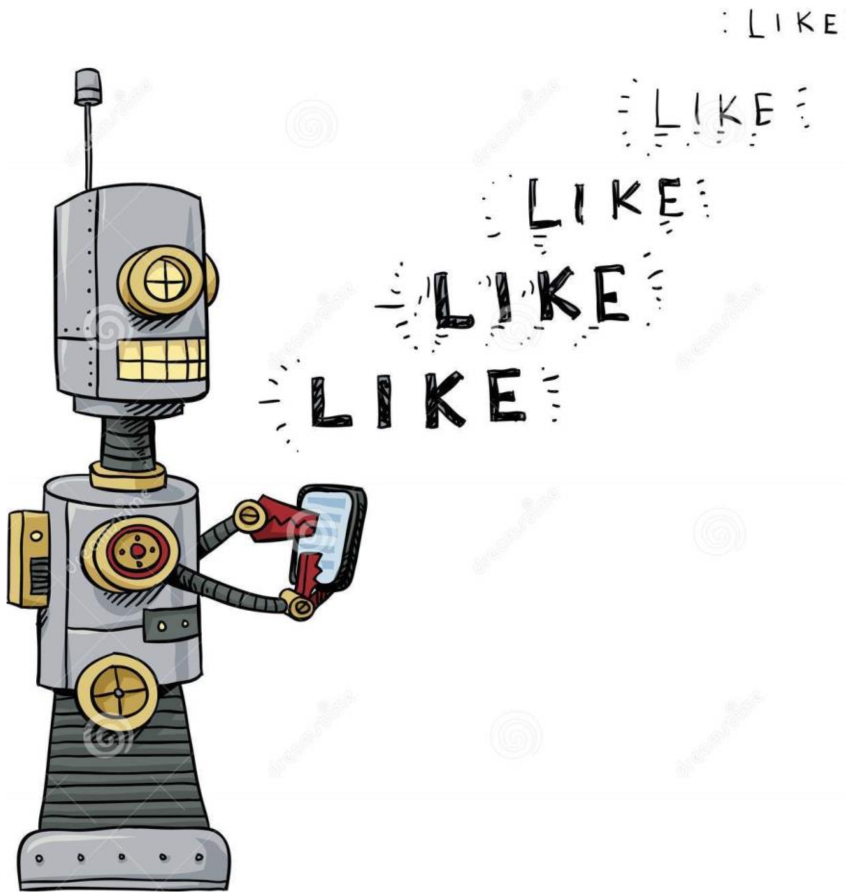
- Workflow
  - Split the dataset into k smaller sets (called k folds)
  - For each of the fold
    - Train a model using the other k-1 folds
    - Test the model using this fold
  - Report performance as the average of the values computed in the loop

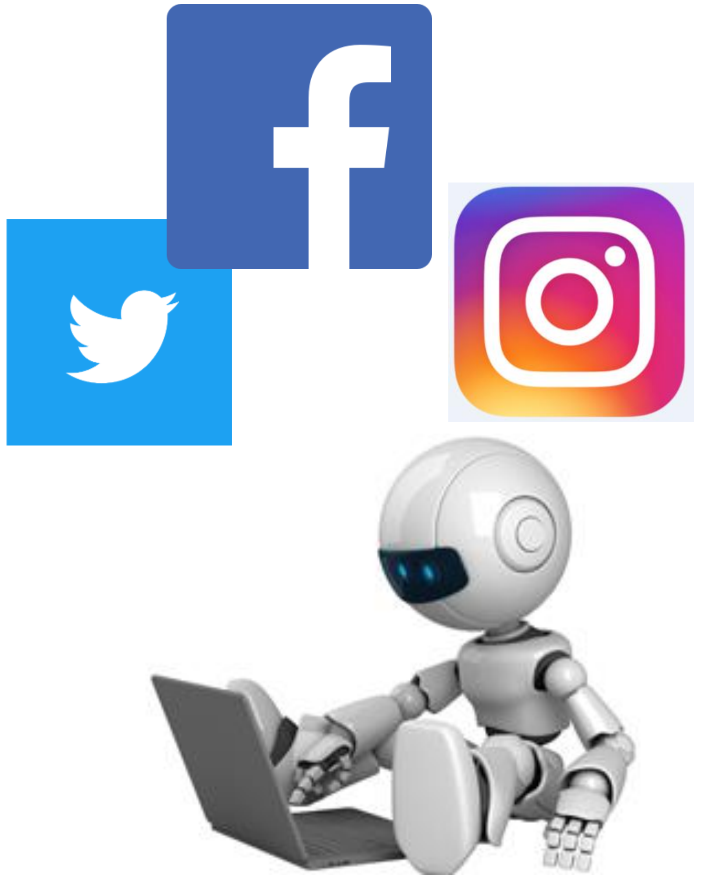
Outline

- Overview of Machine Learning
- Regression
  - Linear Regression
- Classification
  - Decision Tree
  - Ensemble Method
- Exercise: Social Bot Detection
- Feedforward Neural Network
- Discussion

Exercise: Social Bot Detection

- Human or Bot?





https://www.adweek.com/digital/social-bots-twitter-minor-nuisance/

Exercise: Social Bot Detection

- What type of ML problem is it?
- What method can be used?
- What features can be used?
- How to evaluate?

Outline

Recap: Linear Regression

- More generally, $y \approx f(x) = \sum_j w_j h_j(x)$
- $h_j(x)$ are (manually) engineered features of input x
- Find best parameters w with (Stochastic) Gradient Descent

$$\min_w L(w) = \sum_{i=1}^{N} \left(y_i - \sum_j w_j h_j(x_i) \right)^2 + ||w||_p$$

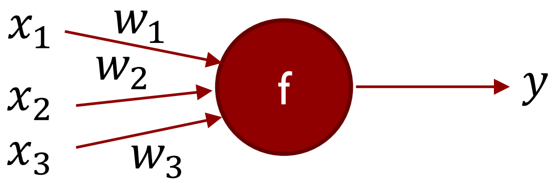
$$\text{s.t. } w_j \in \mathbb{R}, \forall j$$

Designing features $h_j(x)$ is challenging!
Domain-specific
Often need domain knowledge
Sometimes need decades of effort

A Basic Unit in Neural Networks

- A unit completes a simple operation of input variables: apply a non-linear function on a linear transformation of input

$$y = f(\boldsymbol{x}^T \boldsymbol{w} + b) \qquad \boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^n, b \in \mathbb{R}$$



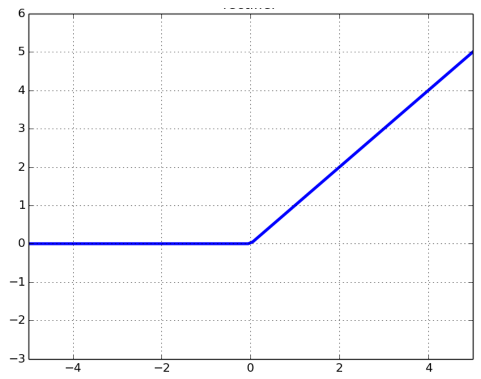
- f: activation function
- w and b are parameters / weights

A Basic Unit in Neural Networks

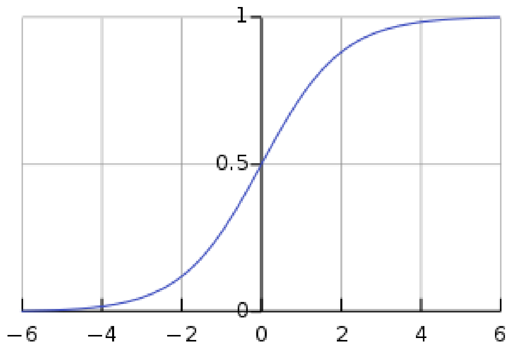
$$y = f(\boldsymbol{x}^T\boldsymbol{w} + b)$$

- Commonly used f
  - (default) ReLU (rectified linear unit): $f(z) = \max\{0, z\}$
  - Sigmoid: $f(z) = \sigma(z) = \frac{1}{1+e^{-z}}$
  - tanh: $f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

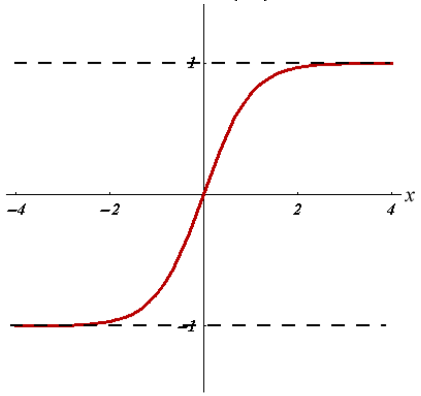
ReLU



$\sigma(z)$

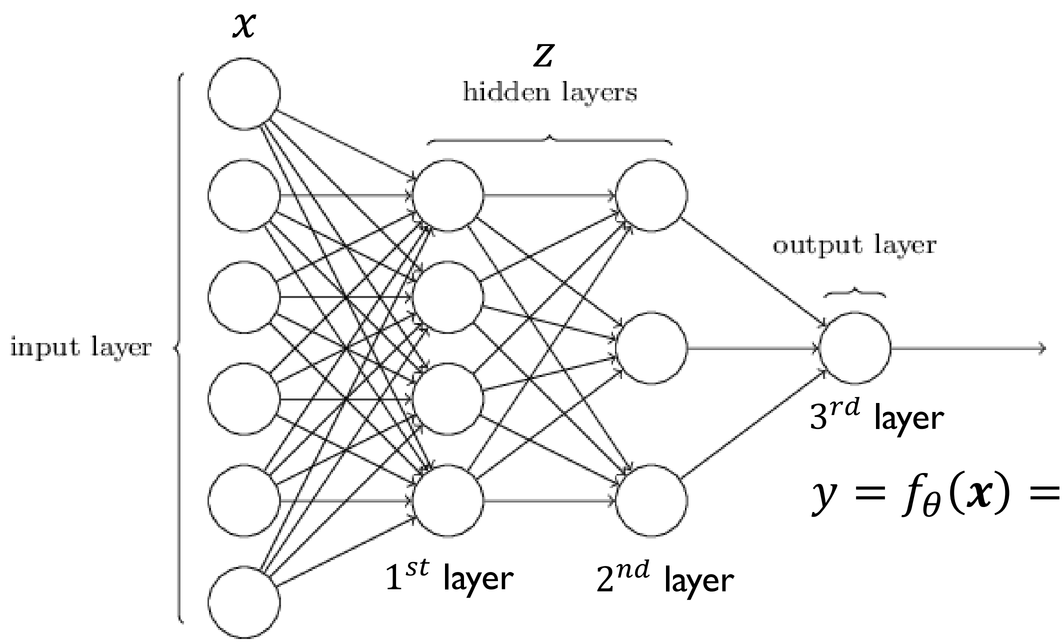


$\tanh(z)$



Feedforward Neural Network (FFN)

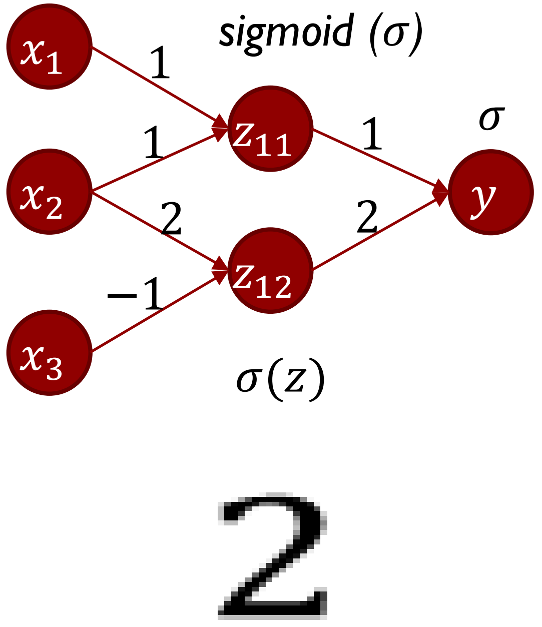
- Also known as multi-layer perceptron (MLP)
  - $\boldsymbol{x} \in \mathbb{R}^n \to$ layers of units $\to y = f_\theta(\boldsymbol{x}) \in \mathbb{R}$ or $\in \mathbb{R}^M$



$$y = f_\theta(\boldsymbol{x}) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(\boldsymbol{x})\right)\right)$$

Forward Pass: Compute Network Predictions

- Given a network with the parameters known, compute the output value of each unit
  - Compute layer-by-layer from first layer to last layer



Assume all the constant parameters are zero.
If $\boldsymbol{x} = (0,0,1)$,

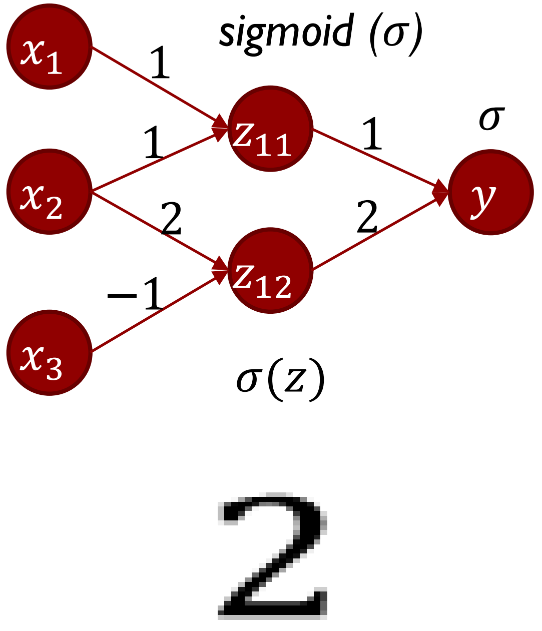
$z_{11} =$

$z_{12} =$

$y =$

Forward Pass: Compute Network Predictions

- Given a network with the parameters known, compute the output value of each unit
  - Compute layer-by-layer from first layer to last layer



Assume all the constant parameters are zero.
If $\boldsymbol{x} = (0,0,1)$,

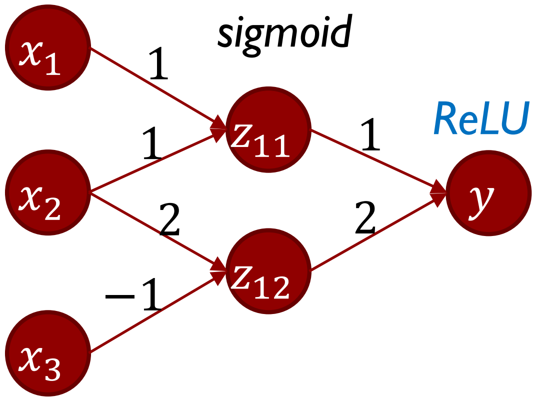
$$z_{11} = \sigma(x_1 + x_2) = \sigma(0) = 0.5$$

$$z_{12} = \sigma(2x_1 - x_3) = \sigma(-1) \approx 0.269$$

$$y = \sigma(z_1 + 2z_2) \approx \sigma(0.5 + 2 * 0.269)$$

Poll 3

- For the provided neural network, what is the value of y when $\boldsymbol{x} = (1,0,1)$? Hint: $\sigma(1) \approx 0.731, \sigma(-1) \approx 0.269$
  - A: 1.269
  - B: 1
  - C: 0.731
  - D: None of the above
  - E: I don't know

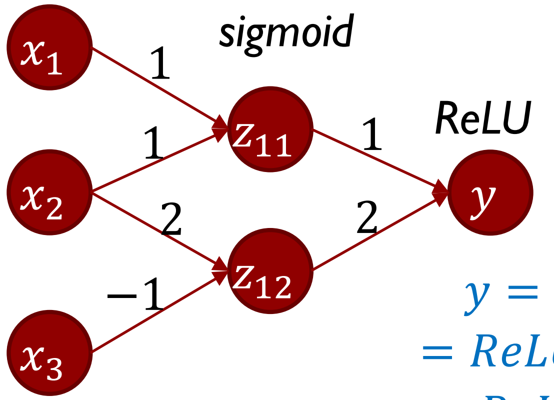


ReLU: $f(z) = \max\{0, z\}$

Poll 3

- For the provided neural network, what is the value of y when $\boldsymbol{x} = (1,0,1)$? Hint: $\sigma(1) \approx 0.731, \sigma(-1) \approx 0.269$
  - A: 1.269
  - B: 1
  - C: 0.731
  - D: None of the above
  - E: I don't know

$z_{11} = \sigma(x_1 + x_2) = \sigma(1)$

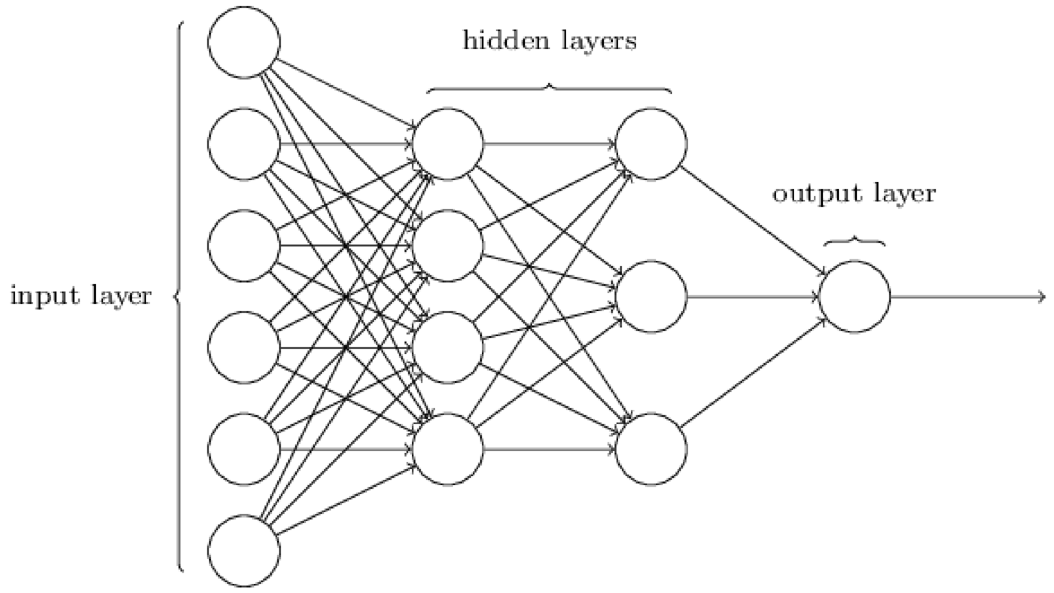


$$
\begin{aligned}
y &= ReLU(z_{11} + z_{12}) \\
&= ReLU\big(\sigma(1) + 2\sigma(-1)\big) \\
&= ReLU(1.269) = 1.269
\end{aligned}
$$

$z_{12} = \sigma(2x_1 - x_2) = \sigma(-1)$

Feedforward Neural Network (FFN)

- Why neural networks are powerful: even with a single hidden layer, an FFN can approximate any function



Train a FFN

- Find value of θ (parameters in the network) to minimize some loss function defined on observed value $\hat{y}$ and output of network $f_\theta(x)$ in training set

$$\min_\theta L(\theta) = \sum_{i=1}^{m} l(f_\theta(x^i), \hat{y}^i)$$

- Why training neural networks is challenging: $\sum_i l(f_\theta(x^i), \hat{y}^i)$ is often highly non-convex w.r.t. θ for multi-layer NN $\rightarrow$ can only get local optima

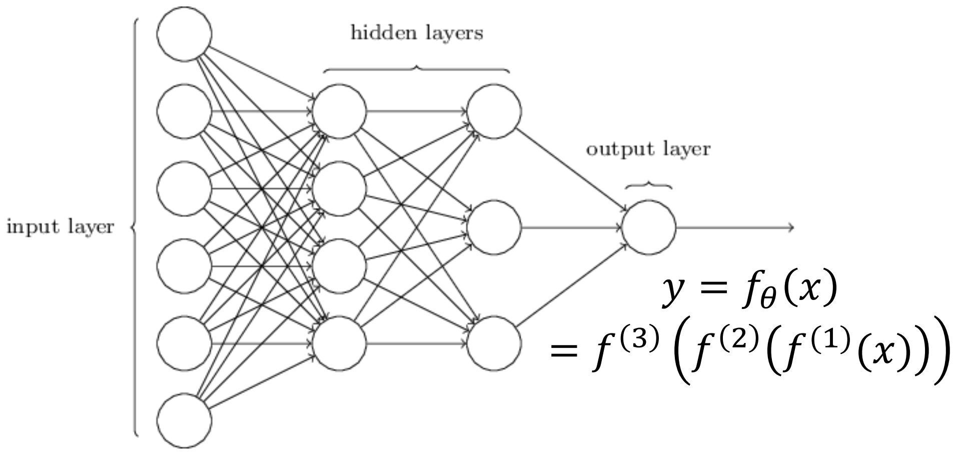
Train a FFN

- Stochastic gradient descent (SGD)
  - Initialize θ
  - Repeat until convergence
    - Randomly shuffle examples in the training set
    - For $i = 1 \dots m$
      - $\theta \leftarrow \theta - \alpha \boxed{\nabla_\theta l(f_\theta(x^i), \hat{y}^i)} \longrightarrow$ How to compute?

To improve efficiency: Use Mini-Batch, i.e., Compute gradient and update parameters for every batch of k data samples.

Back Propagation

- Back Propagation: An efficient way of computing $\nabla_\theta l(f_\theta(x^i), \hat{y}^i)$
  - (1) Forward pass: Compute network prediction layer-by-layer from first layer to last layer
    - Given x^i and current θ, compute z^i (value of all the units) and $f_\theta(x^i)$

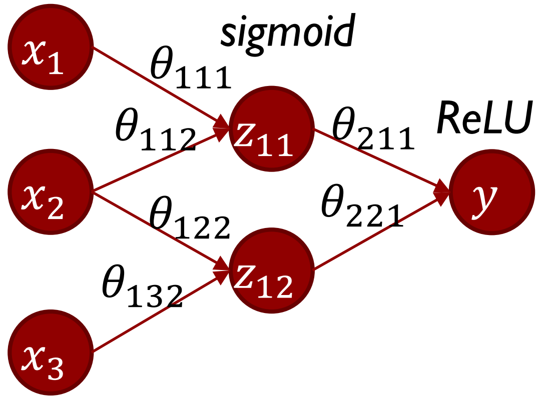


Back Propagation

- Back Propagation: An efficient way of computing $\nabla_\theta l(f_\theta(x^i), \hat{y}^i)$
  - (1) Forward pass: Compute network prediction layer-by-layer from first layer to last layer
  - (2) Backward pass: Compute gradient of the loss function w.r.t. network parameters layer-by-layer from last layer to first layer
    - Given $f_\theta(x^i)$ and y^i, compute $\nabla_{\theta_k} l(f_\theta(x^i), \hat{y}^i)$ where θ_k are parameters in layer k

Backward Pass: Intuition and Example

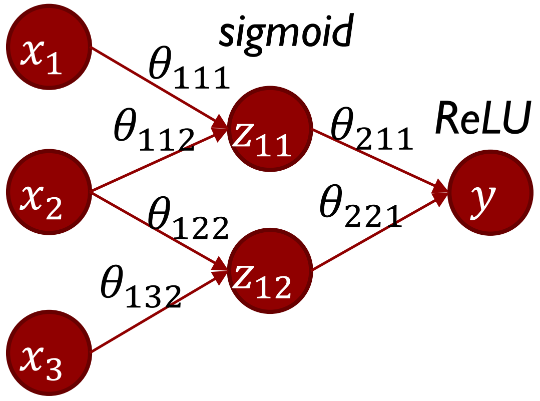
- To compute $\frac{\partial l(f_\theta(x^i), \hat{y}^i)}{\partial \theta_{kpq}}, \forall k, p, q$
  - First compute $\frac{\partial l(y, \hat{y}^i)}{\partial y}$



Backward Pass: Intuition and Example

- To compute $\frac{\partial l(f_\theta(x^i), \hat{y}^i)}{\partial \theta_{kpq}}, \forall k, p, q$
  - First compute $\frac{\partial l(y, \hat{y}^i)}{\partial y}$
  - Then compute $\frac{\partial l(y, \hat{y}^i)}{\partial \theta_{211}}, \frac{\partial l(y, \hat{y}^i)}{\partial \theta_{221}}, \frac{\partial l(y, \hat{y}^i)}{\partial z_{11}}, \frac{\partial l(y, \hat{y}^i)}{\partial z_{12}}$ using chain rule, e.g., $\frac{\partial l(y, \hat{y}^i)}{\partial z_{11}} = \frac{\partial l(y, \hat{y}^i)}{\partial y} \frac{\partial y}{\partial z_{11}}$
  - Then compute $\frac{\partial l(y, \hat{y}^i)}{\partial \theta_{1pq}}$ using chain rule,

e.g., $\frac{\partial l(y, \hat{y}^i)}{\partial \theta_{111}} = \frac{\partial l(y, \hat{y}^i)}{\partial z_{11}} \frac{\partial z_{11}}{\partial \theta_{111}}$



Apply Chain Rule in Backward Pass

- Chain Rule: $\frac{\partial f(g(x))}{\partial x} = \frac{\partial f(g(x))}{\partial g(x)} \frac{\partial g(x)}{\partial x}$
- Apply Chain Rule in Backward Pass: given $f_\theta(x^i)$ and y^i, compute $\nabla_{\theta_k} l(f_\theta(x^i), \widehat{y^i})$ starting from last layer to first layer

$$\frac{\partial l(f_\theta(x), \hat{y})}{\partial \theta_j} = \frac{\partial l(y, \hat{y})}{\partial y} \frac{\partial y}{\partial z_K} \frac{\partial z_K}{\partial z_{K-1}} \cdots \frac{\partial z_{j+1}}{\partial \theta_j}$$

θ_j are the parameters in layer j

z_k denote the "input" for k^{th} layer

Back Propagation

- In practice, network predictions (forward pass) and gradient (backward pass) can be calculated easily if you are using e.g., PyTorch, TensorFlow

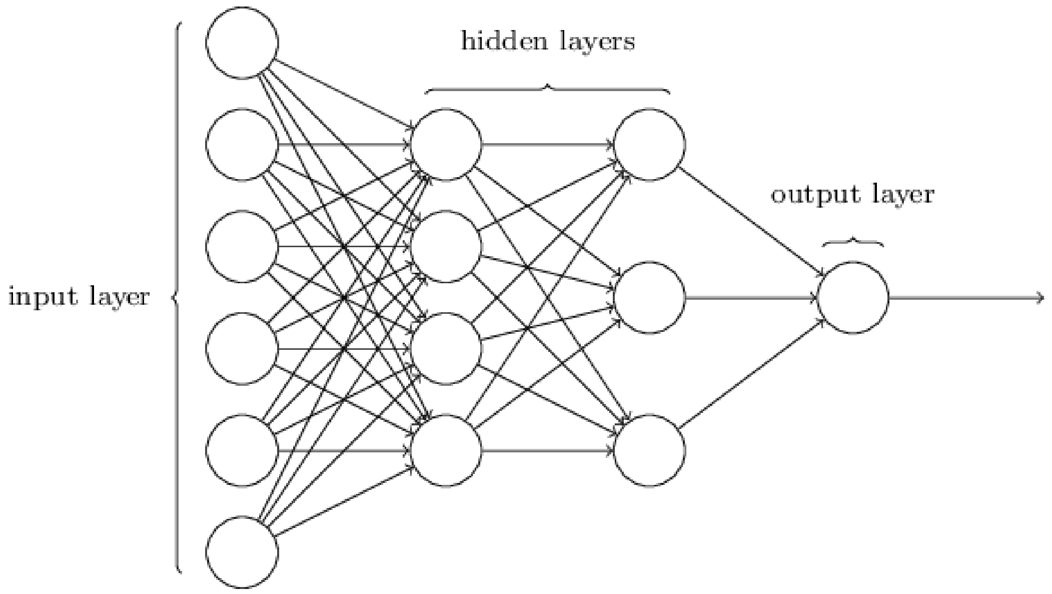
With PyTorch, after defining a model and a loss function, for one iteration with one data point:

```
## forward + backprop + loss
y = yourmodel(x)
loss = yourlossfunction(y, label)
optimizer.zero_grad()
loss.backward()
```

Hyperparameters for an FFN

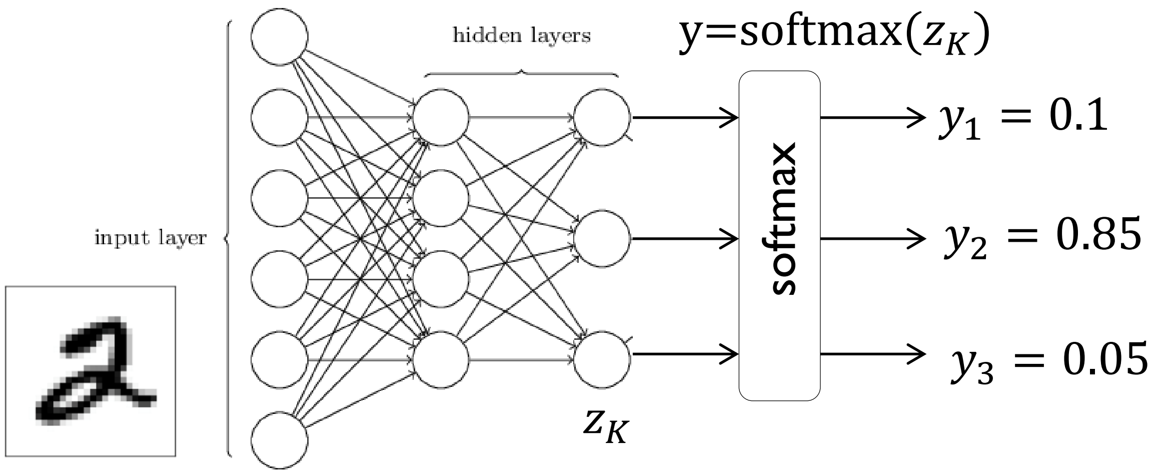
- Number of layers
- Number of hidden dimension for each layer

Q: How to determine these hyperparameters?



Special Units and Layers in FFN

- Fully connected layer / dense layer: every node in layer k is connected with every node in layer $k + 1$
- MaxPool unit: $y = \max_{i \in \rightarrow y} x_i$
- Softmax Layer: $y_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$



Outline

- Overview of Machine Learning
- Regression
  - Linear Regression
- Classification
  - Decision Tree
  - Ensemble Method
- Exercise: Social Bot Detection
- Feedforward Neural Network
- Discussion

Discussion

- If you trained a decision tree for predicting whether a bot is a social bot, but the F1 score is low (e.g., 0.55), what can you do to improve the prediction performance?

References and Additional Resources

Additional Resources

- Text book
  - *Pattern Recognition and Machine Learning, Chapters 3,6*
  - Christopher Bishop
- Online course
  - https://www.coursera.org/learn/ml-regression
  - https://www.coursera.org/learn/ml-classification
  - https://www.coursera.org/learn/machine-learning

Tutorials on ML for Classification

- Scikit-learn decision trees
- Example code

Backup Slides

Regression

- Task T: predict a numerical value given some input
  - The learning algorithm produces a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$
  - Given an input $x \in \mathbb{R}^n$, output $y = f(x)$
- Experience E: A dataset consists of labeled data points $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)$
  - Input for the learning algorithm, Train the model f
- Typically assume $x_i \sim D, x \sim D$

Tom Mitchell: "A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Regression

- Evaluate a learned model f
  - Make predictions with f at a test set of data (separated from the set of data used for training)
  - Ideally: $y = f(x)$
- Performance Measure P
  - Define a loss function, e.g.,
    - Absolute loss: $|y - f(x)|$; Squared loss: $\left(y - f(x)\right)^2$
  - Compute total loss on test set

Tom Mitchell: "A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Different Types of ML Problems

- Supervised learning
  - Learning from a set of labeled data
  - Regression (numerical, continuous-valued label)
  - Classification (categorical, discrete-valued label)

Classification

- Task T: assign a class/category given some input
  - The learning algorithm produces a function $f: \mathbb{R}^n \rightarrow \{1 \dots k\}$
  - Given an input $x \in \mathbb{R}^n$, output $y = f(x)$
  - Other variants: output a prob. distribution over classes
- Experience E: A dataset consists of labeled data points $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)$
- Typically assume $x_i \sim D, x \sim D$

Tom Mitchell: "A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Social Bot

- How They Can Help
- How They Can Be Trouble

Social Bot

- How They Can Help
  - Entertaining
  - Make people feel that there are people who cares about them / like them
- How They Can Be Trouble
  - Orchestrated campaign
  - Create illusion of artificial grassroots support for political aims
  - Create fake "buzz" about a company (and make money with automated stock trading algorithms acted on this chatter)

Social Bot Detection

- Act towards social bot campaign timely and effectively

How Political Campaigns Weaponize Social Media Bots

Analysis of computational propaganda in the 2016 U.S. presidential election reveals the reach of bots

By Philip N. Howard

Illustration: Jude Buffum

Data

- Collect data
  - Find list of social bots identified by people or in literature
  - Use Twitter Search API to collect up to 200 of their most recent tweets and up to 100 of the most recent tweets mentioning them

- Raw data from twitter: Twitter screen name, account meta-data, account's recent activity (tweet content and time, retweets, mentions, hashtag), followers, followees

- Label meaning: bot-likelihood score
- Assign label: based on known bot lists

Data

- 15k manually verified social bots and 16k legitimate (human) accounts
- More than 5.6 millions tweets in total

Data

- Features
  - Network features:
  - User features:
  - Friends features:
  - Temporal features:
  - Content features:
  - Sentiment features:

Data

- Features
  - Network features: retweets, mentions, and hashtag co-occurrence, and their statistical features, e.g. degree distribution, clustering coefficient, and centrality measures
  - User features: language, geographic locations, and account creation time
  - Friends features: statistics relative to an account's social contacts, such as the median, moments, entropy of the distributions of their number of followers, followees, posts
  - Temporal features: timing patterns of content generation and consumption, such as tweet rate and inter-tweet time distribution
  - Content features: linguistic cues computed through natural language processing, especially part-of-speech tagging
  - Sentiment features: using sentiment analysis algorithms, including happiness, arousal-dominance-valence, and emoticon scores

BotOrNot

- Use Random Forest

- Train seven different classifiers: one for each subclass of features and one for the overall score

Evaluation

- Ten-fold cross-validation
- AUC (Area Under ROC Curve): 0.95
  - likely to overestimate current performance