Introduction to Machine Learning

Stephen Scott, Dept of CSE

University of Nebraska Lincoln

What is Machine Learning?

- Building machines that automatically **learn** from experience
  - Sub-area of artificial intelligence
- (Very) small sampling of applications:
  - Detection of fraudulent credit card transactions
  - Filtering spam email
  - Autonomous vehicles driving on public highways
  - Self-customizing programs: Web browser that learns what you like/where you are and adjusts; autocorrect
  - Applications we can't program by hand: E.g., speech recognition
- You've used it today already ☺

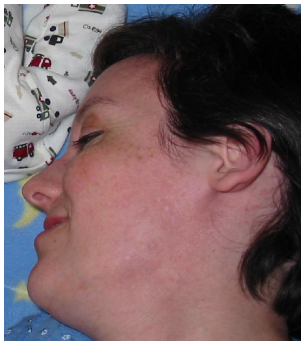
What is Learning?

- Many different answers, depending on the field you're considering and whom you ask
  - Artificial intelligence vs. psychology vs. education vs. neurobiology vs. …

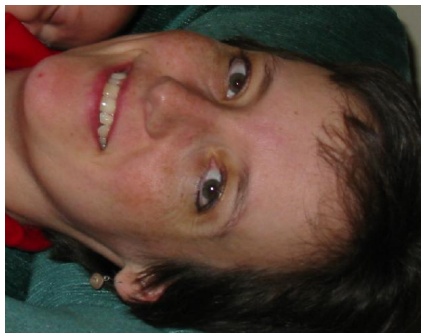
Does Memorization = Learning?

- Test #1: Thomas learns his mother's face

Sees:



But will he recognize:



Thus he can generalize beyond what he’s seen!

Does Memorization = Learning? (cont'd)

- Test #2: Nicholas learns about trucks

Sees:

But will he recognize others?

- So learning involves **ability to generalize** from labeled examples
- In contrast, memorization is trivial, especially for a computer

What is Machine Learning? (cont'd)

- When do we use machine learning?
  - Human expertise does not exist (navigating on Mars)
  - Humans are unable to explain their expertise (speech recognition; face recognition; driving)
  - Solution changes in time (routing on a computer network; browsing history; driving)
  - Solution needs to be adapted to particular cases (biometrics; speech recognition; spam filtering)
- In short, when one needs to generalize from experience in a non-obvious way

What is Machine Learning? (cont'd)

- When do we **not** use machine learning?
  - Calculating payroll
  - Sorting a list of words
  - Web server
  - Word processing
  - Monitoring CPU usage
  - Querying a database
- When we can definitively specify how all cases should be handled

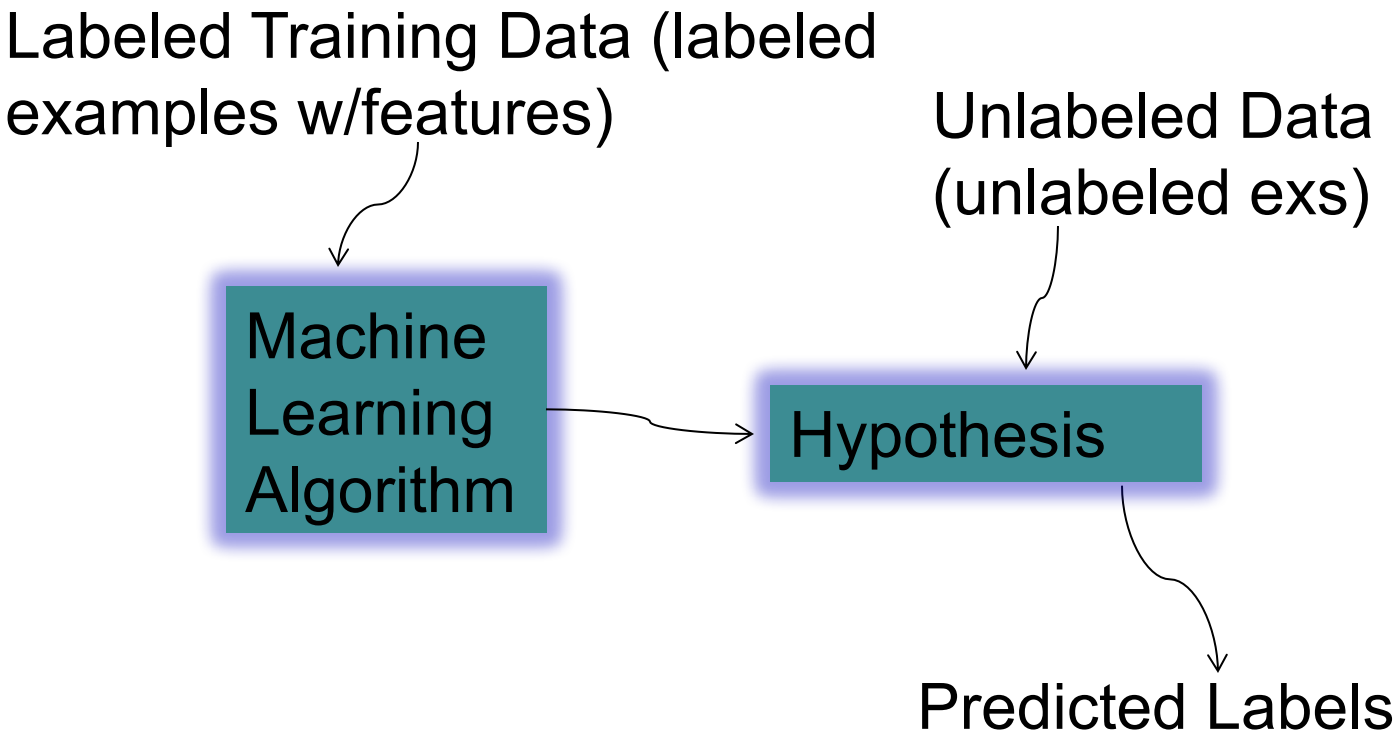
More Formal Definition

- From Tom Mitchell's 1997 textbook:
  - *"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."*
  - Wide variations of how *T*, *P*, and *E* manifest

One Type of Task *T*: Classification

- Given several **labeled examples** of a **concept**
  - E.g., trucks vs. non-trucks (binary); height (real)
  - This is the experience *E*
- Examples are described by **features**
  - E.g., number-of-wheels (int), relative-height (height divided by width), hauls-cargo (yes/no)
- A machine learning algorithm uses these examples to create a **hypothesis** (or **model**) that will **predict** the label of new (previously unseen) examples

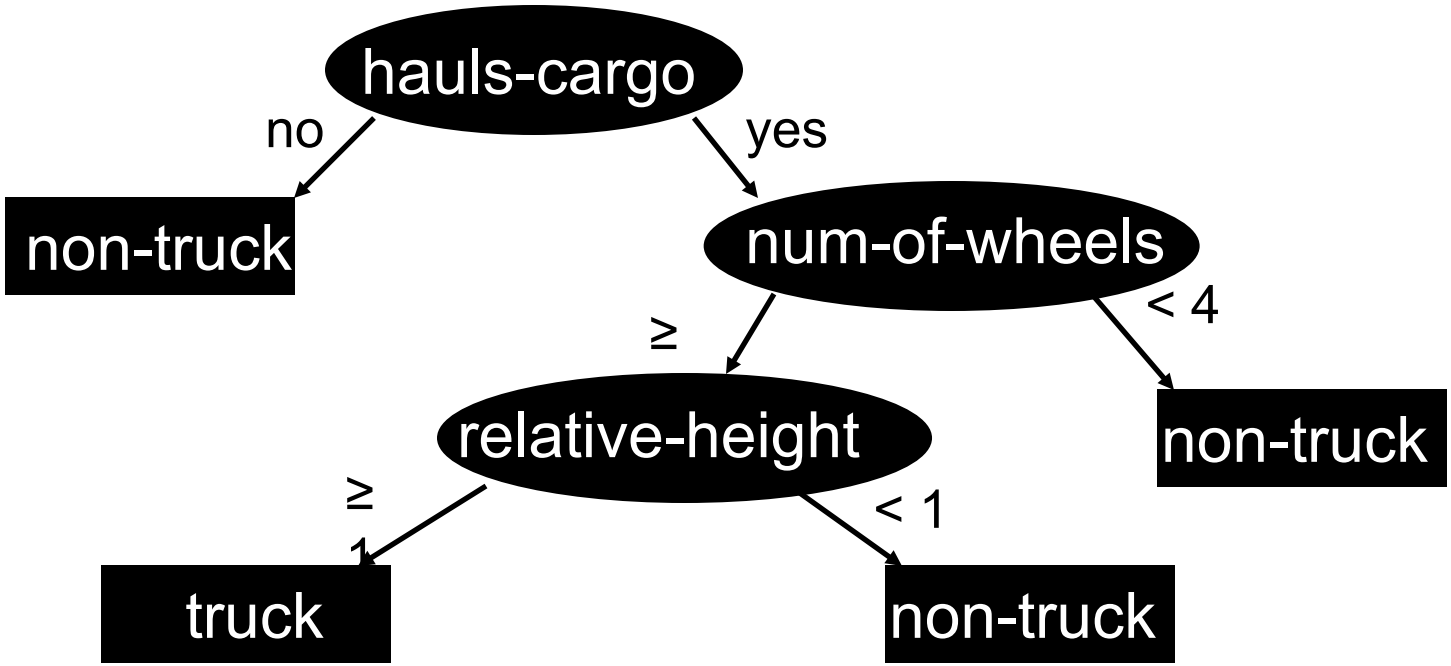
Classification (cont'd)



- Hypotheses can take on many forms

Hypothesis Type: Decision Tree

- Very easy to comprehend by humans
- Compactly represents if-then rules

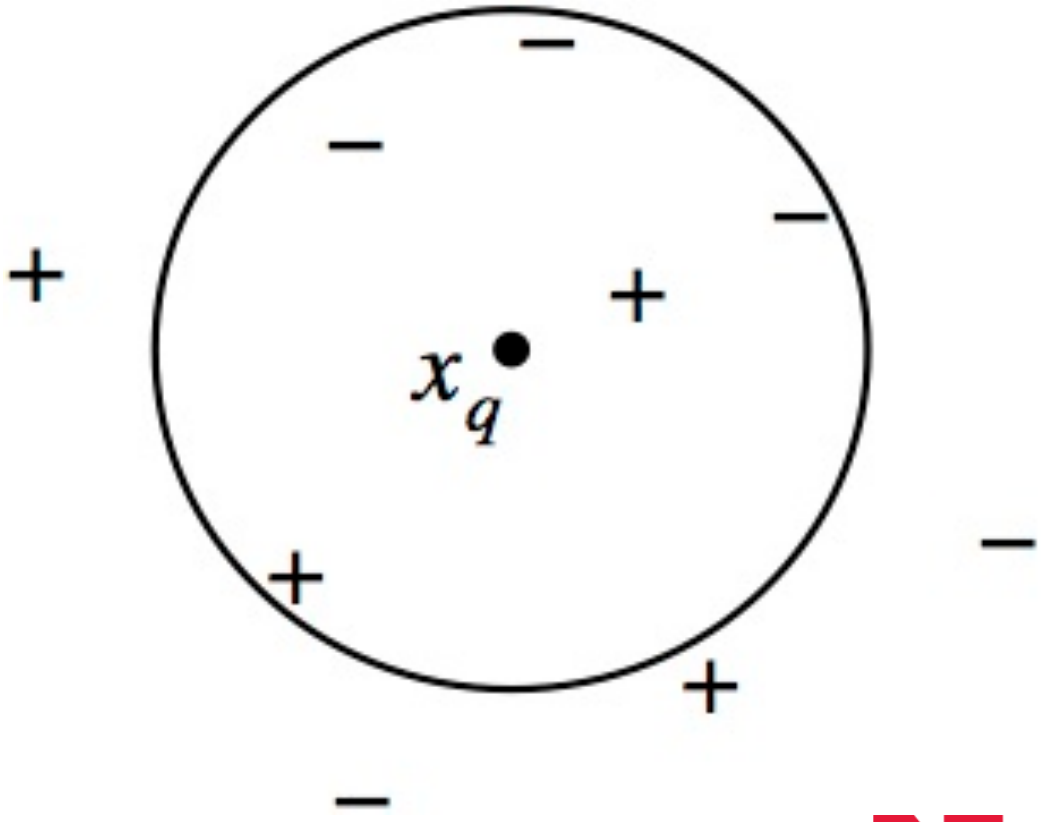


Learning Decision Trees

- While not done
  - Choose an attribute *A* to test at the current node
    - Choice based on which one results in most homogenous (in label) subsets of training data
      - Often measured based on **entropy** or **Gini index**
  - Partition the training set into subsets based on value of *A*
  - Recursively process subsets to choose attributes for subtrees

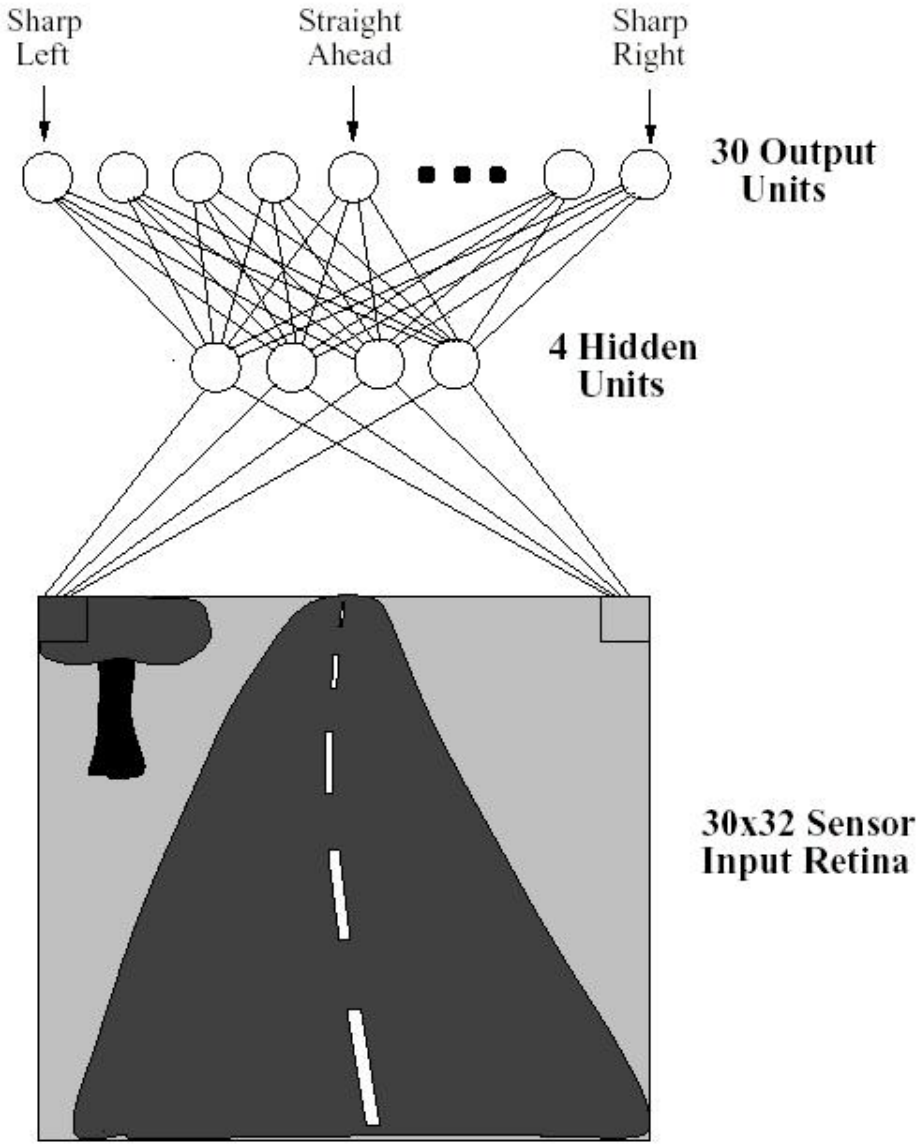
Hypothesis Type: *k*-Nearest Neighbor

- Compare new (unlabeled) example x_q with training examples
- Find k training examples most similar to x_q
- Predict label as majority vote



Hypothesis Type: Artificial Neural Network

- Designed to simulate brains
- "Neurons" (processing units) communicate via connections, each with a numeric weight
- Learning comes from adjusting the weights



Artificial Neural Networks (cont'd)

- ANNs are basis of **deep learning**
- "Deep" refers to depth of the architecture
  - More layers => more processing of inputs
- Each input to a node is multiplied by a weight
- Weighted sum *S* sent through **activation function:**
  - **Rectified linear:** max(0, *S*)
  - **Convolutional + pooling:** Weights represent a (e.g.) 3x3 **convolutional kernel** to identify features in (e.g.) images that are **translation invariant**
  - **Sigmoid:** tanh(*S*) or 1/(1+exp(-*S*))
- Often trained via **stochastic gradient descent**

Other Hypothesis Types for Classification

- Support vector machines
  - A major variation on artificial neural networks
- Bagging and boosting
  - Performance enhancers for learning algorithms via re-sampling training data
- Bayesian methods
  - Build probabilistic models of the data
- Many more
- Variations on the task *T*: **regression** (real-valued labels) and **predicting probabilities**

Example Performance Measures *P*

- Let X be a set of labeled instances
- **Classification error:** number of instances of X hypothesis h predicts correctly, divided by $|X|$
- **Squared error:** Sum $(y_i - h(x_i))^2$ over all x_i
  - If labels from {0,1}, same as classification error
  - Useful when labels are real-valued
- **Cross-entropy:** Sum over all x_i from X:

  $y_i \ln h(x_i) + (1 - y_i) \ln (1 - h(x_i))$
  - Generalizes to > 2 classes
  - Effective when h predicts probabilities

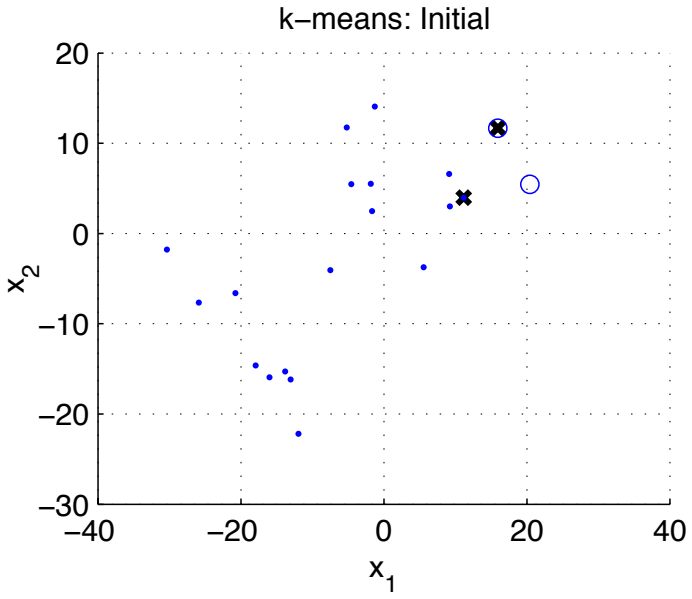
Another Type of Task *T*: Unsupervised Learning

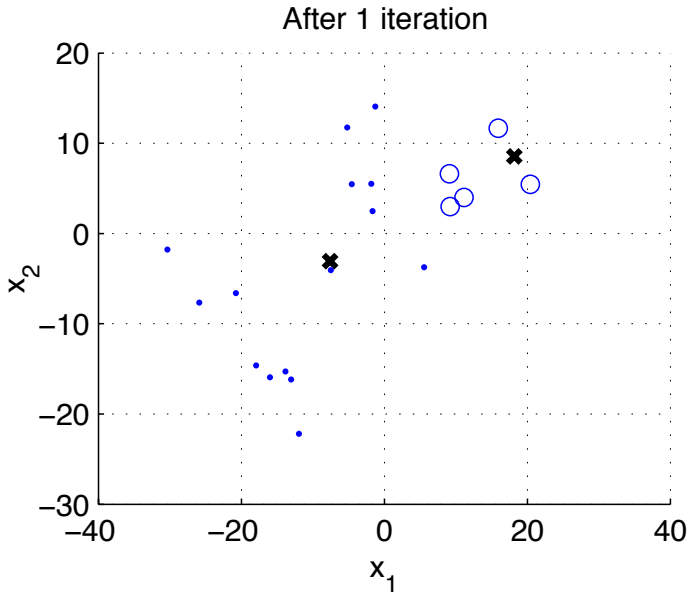
- *E* is now a set of **unlabeled examples**
- Examples are still described by **features**
- Still want to infer a model of the data, but instead of predicting labels, want to understand its **structure**
- E.g., **clustering, density estimation, feature extraction**

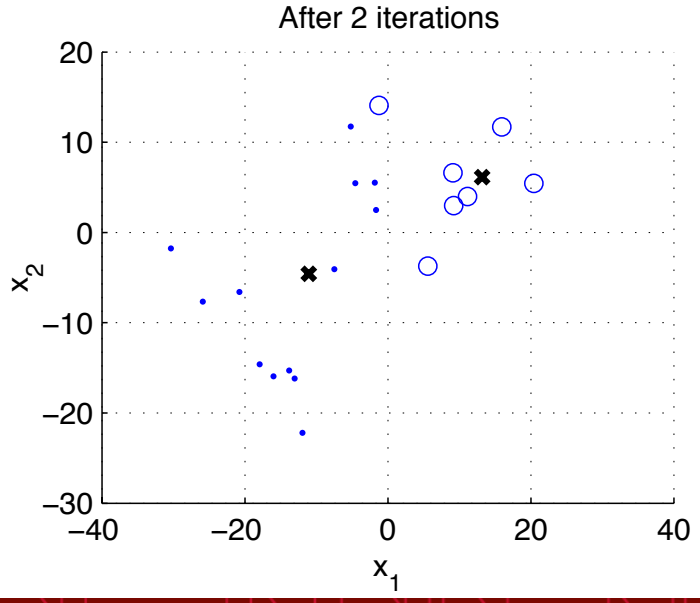
k-Means Clustering

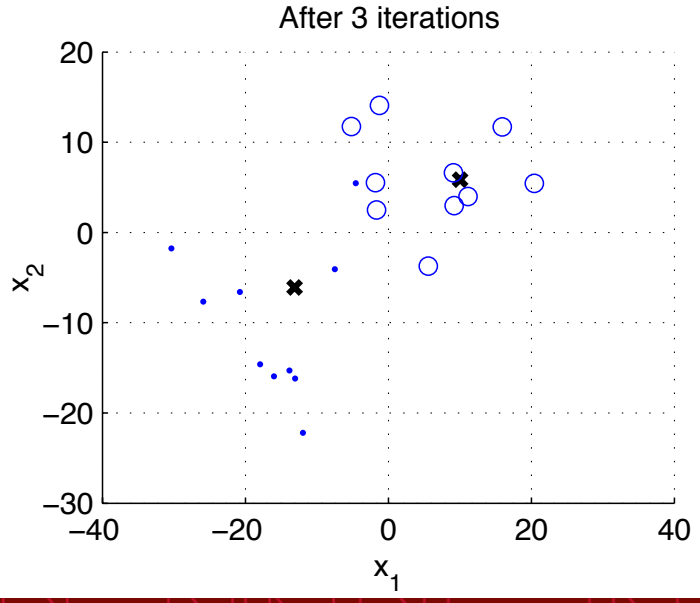
- Randomly choose k cluster **centers** $m_1,\ldots,m_k$
- Assign each instance x in X to its nearest center
- While not done
  - Re-compute m_i to be the center of cluster i
  - Re-assign each x to cluster of its nearest center

k-Means Clustering Example



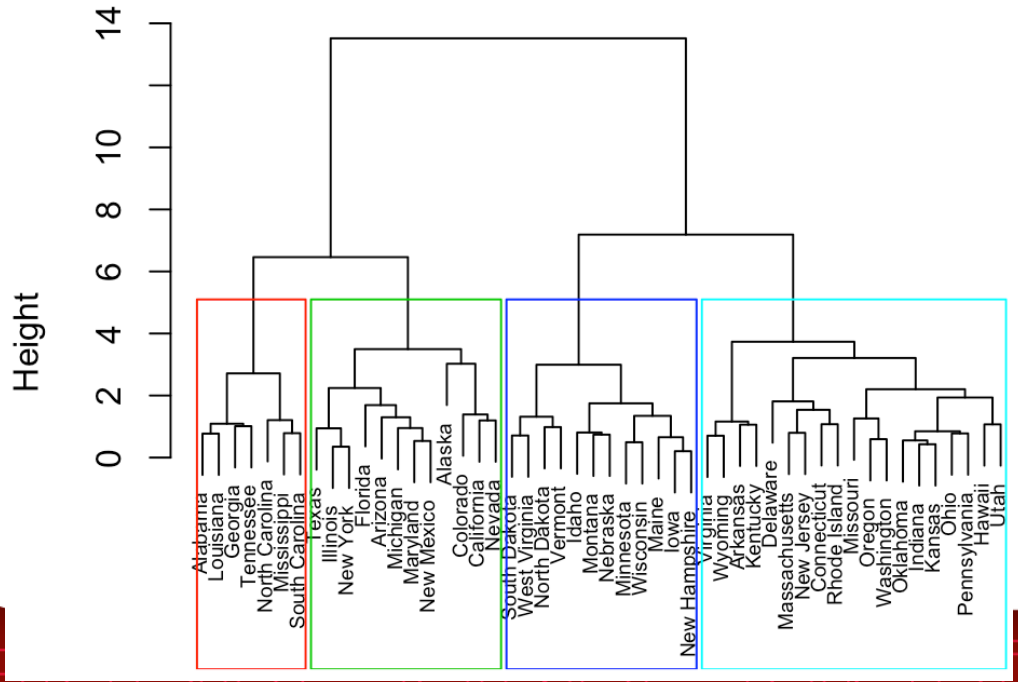






Hierarchical Clustering

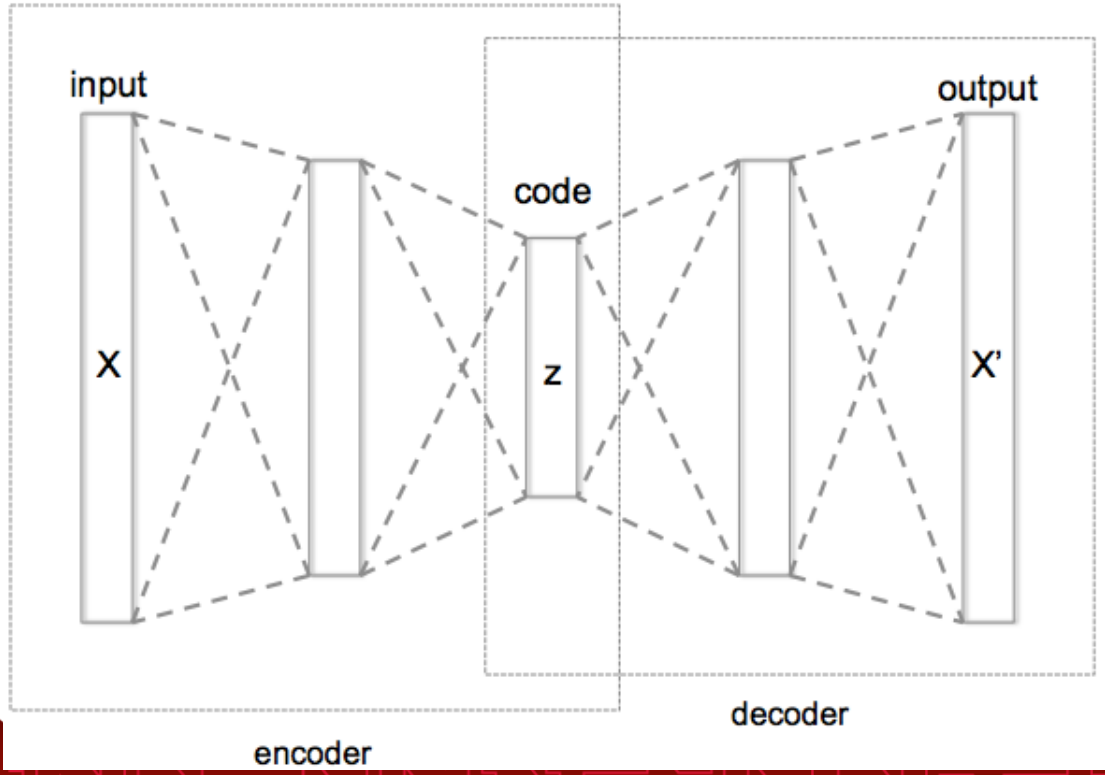
- Group instances into clusters, then group clusters into larger ones
- As with *k*-means, requires a **similarity measure**
  - E.g., Euclidean distance, Manhattan distance, dot product, biological sequence similarity



- Common in bioinformatics

Feature Extraction via Autoencoding

- Can train an ANN with unlabeled data
- **Goal:** have output *x'* match input *x*
- Results in **embedding** *z* of input *x*
- Can **pre-train** network to identify features



- Later, replace decoder with classifier
- **Semi-supervised** learning

Another Type of Task *T*: Reinforcement Learning

- An **agent** *A* interacts with its **environment**
- At each step, *A* perceives the **state** *s* of its environment and takes **action** *a*
- Action *a* results in some **reward** *r* and changes state to *s'*
  - **Markov decision process (MDP)**
- Goal is to maximize **expected long-term reward**
- Applications: Backgammon, Go, video games, self-driving cars

Reinforcement Learning (cont'd)

- RL differs from previous tasks in that the feedback (reward) is typically delayed
  - Often takes several actions before reward received
  - E.g., no reward in checkers until game ends
  - Need to decide how much each action contributed to final reward
    - **Credit assignment problem**
- Also, limited sensing ability makes distinct states look the same
  - **Partially observable MDP (POMDP)**

Q-Learning

- Popular RL algorithm estimates the value of taking action *a* in state *s*
  - *Q(s, a)* is total reward from taking action *a* in state *s* and acting optimally from then on
  - If we know this, then in state *s* we can choose the action *a* that maximizes *Q*
- **Algorithm to learn *Q*:** Each iteration, observe *s*, take action *a*, receive immediate reward *r*, observe new state *s'*
- Update $Q(s, a) = r + g \max_{a'} Q(s', a')$

  (*g* is a parameter)

Q-Learning (cont'd)

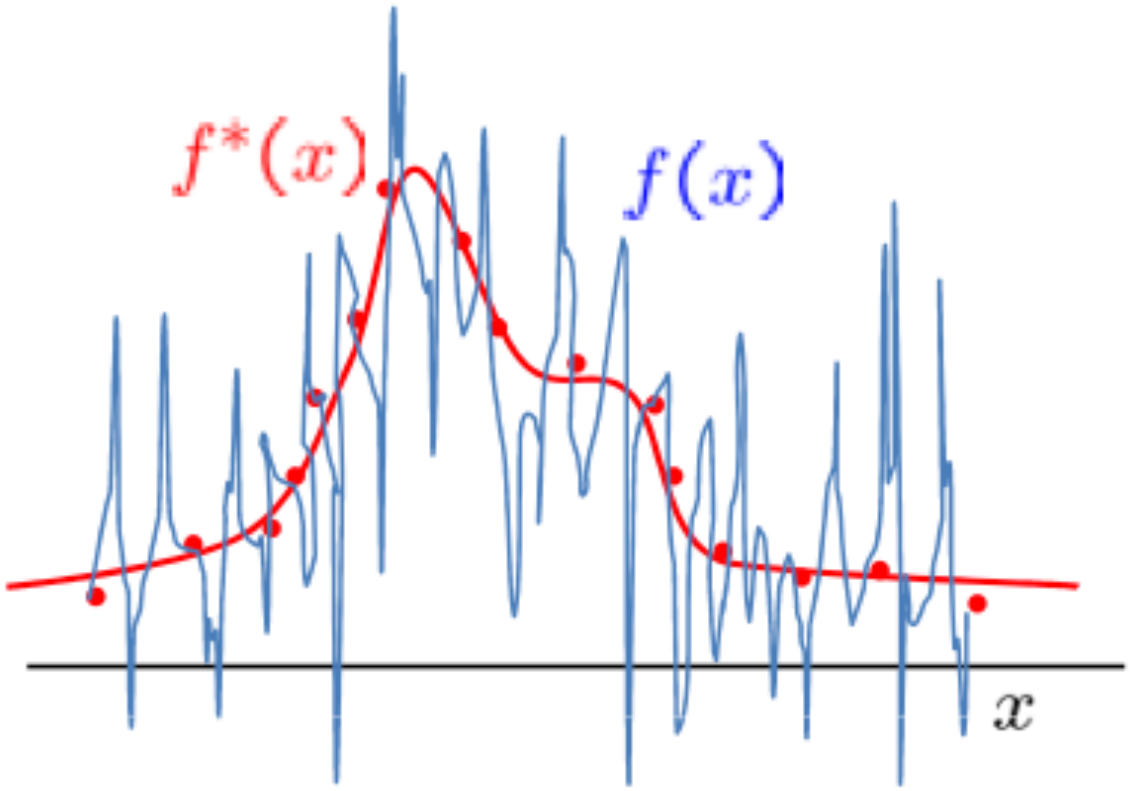
- Q-Learning algorithm guaranteed to converge to correct values if every state visited infinitely often
  - Good to know, but not very practical
  - Cannot expect to even visit every state **once**
  - Chess has about 10^{45} states, Go has 10^{170}
- Need to generalize beyond states already seen
  - Where have we heard this before?
- Machine learning (esp. ANNs) very effective in learning *Q* functions (or other value functions)
  - Deep learning very effective recently: Atari games and Go at better-than-human levels

Yet Other Variations

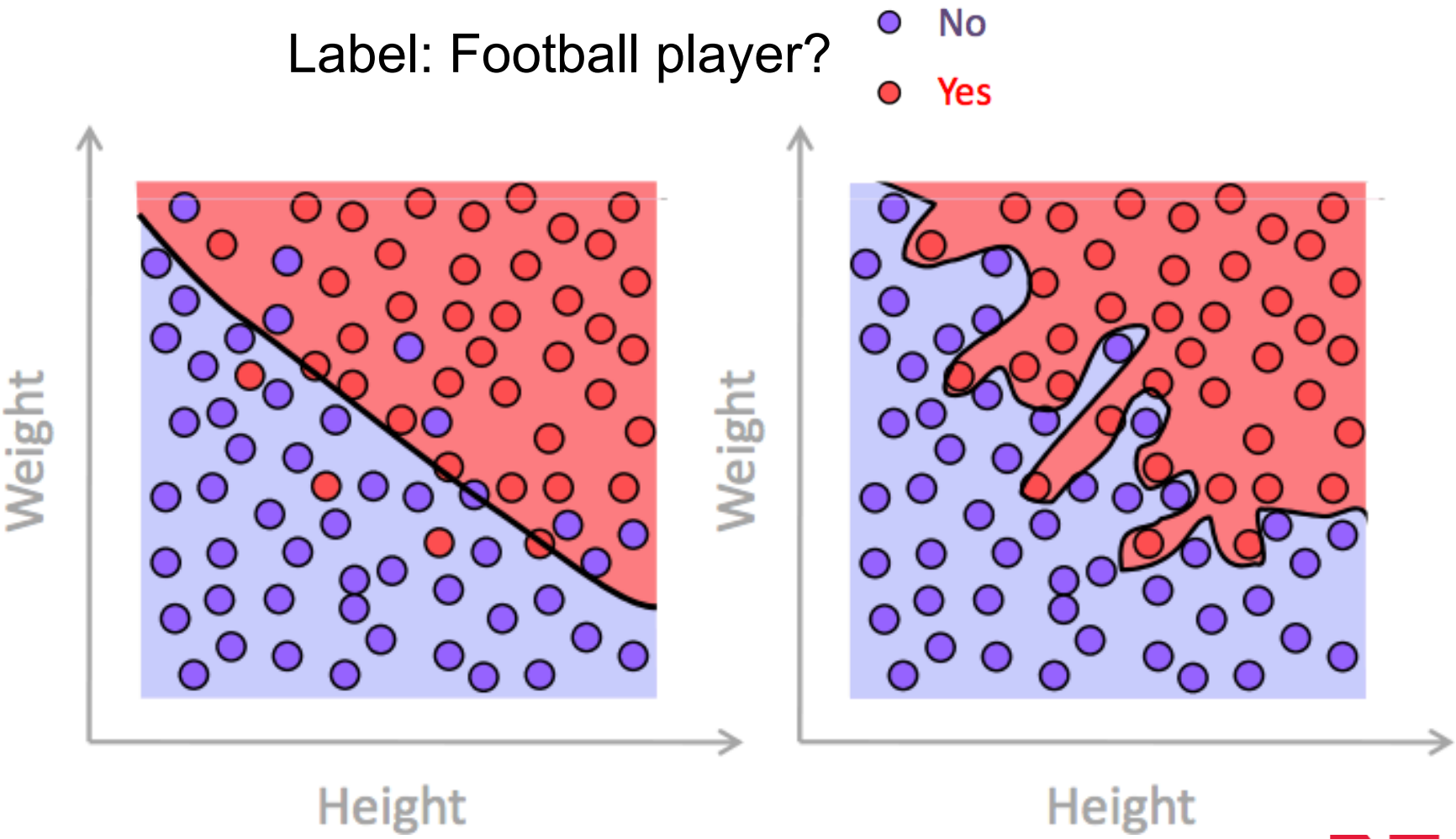
- Missing attributes
  - Must some how estimate values or tolerate them
- Sequential data, e.g., genomic sequences, speech
  - Hidden Markov models
  - Recurrent neural networks
- Have much unlabeled data and/or missing attributes, but can purchase some labels/attributes for a price
  - **Active learning** approaches try to minimize cost
- Outlier detection
  - E.g., intrusion detection in computer systems

Issue: Model Complexity

- In classification and regression, possible to find hypothesis that perfectly classifies all training data
  - But should we necessarily use it?



Model Complexity (cont'd)



➔ To generalize well, need to balance **training accuracy** with **simplicity**

Relevant Disciplines

- Artificial intelligence: Learning as a search problem, using prior knowledge to guide learning
- Probability theory: computing probabilities of hypotheses
- Computational complexity theory: Bounds on inherent complexity of learning
- Control theory: Learning to control processes to optimize performance measures
- Philosophy: Occam's razor (everything else being equal, simplest explanation is best)
- Psychology and neurobiology: Practice improves performance, biological justification for artificial neural networks
- Statistics: Estimating generalization performance

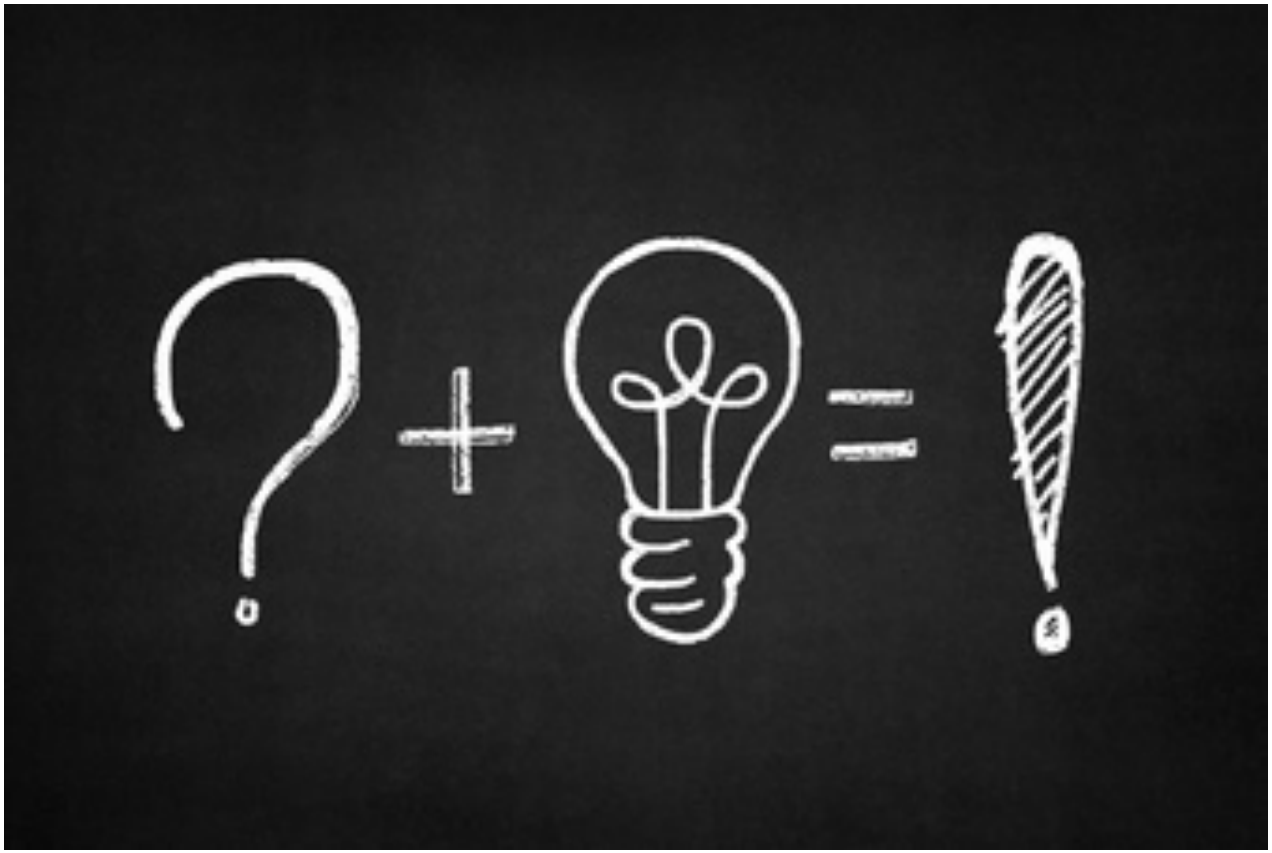
Conclusions

- Idea of intelligent machines has been around a long time
- Early on was primarily academic interest
- Past few decades, improvements in processing power plus very large data sets allows highly sophisticated (and successful!) approaches
- Prevalent in modern society
  - You've probably used it several times today
- No single "best" approach for any problem
  - Depends on requirements, type of data, volume of data

Example References

- *Hands-On Machine Learning with Scikit-Learn & TensorFlow* by Aurelien Geron, O'Reilly, 2017, ISBN 9781491962299
  - Good introduction for the practitioner
  - To be used in CSCE 496/896 Spring 2018
- *Introduction to Machine Learning, Third Edition* by Ethem Alpaydin, MIT Press, 2014, ISBN 9780262028189
  - Used in CSCE 478/878
- *Machine Learning*, by Tom Mitchell. McGraw-Hill, 1997, ISBN 0070428077
  - Excellent, highly accessible introduction, but dated

Thank you!



Any questions?

Any ideas?