Machine Learning

CS 165B

Prof. Matthew Turk

Wednesday, June 1, 2016

Today

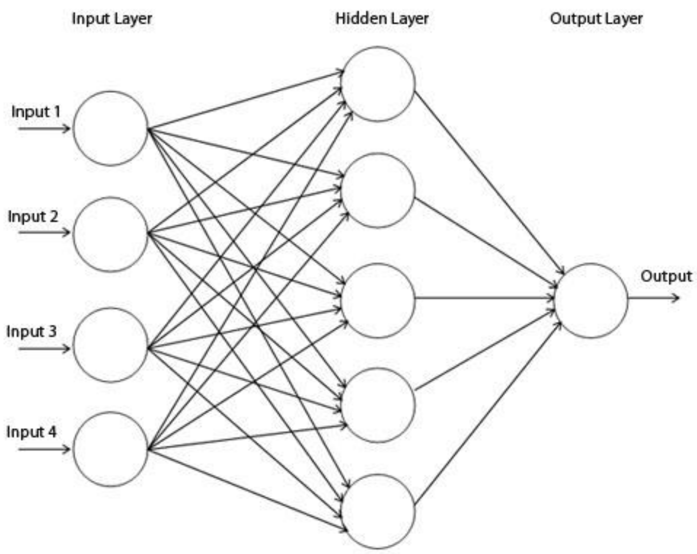
- Neural networks (cont.)

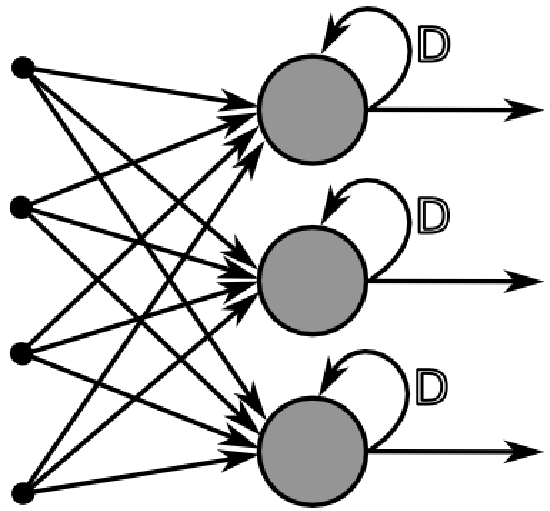
Recorded lecture

Notes

- This lecture is being recorded, so be sure to use the PowerPoint version to play the lecture (slides and audio)!
- Homework assignment #5 due Friday at 4:30pm
- Final exam
  - Practice exam posted soon
  - Pages of equations, etc. already posted
  - Wednesday 8-11am in the regular classroom
    - Just bring a calculator and something to write with

Neural networks





Feedforward network

- Information only moves forward, from input to output
- A.k.a. multi-layer perceptron

Recurrent network

- Directed cycles exist in the network

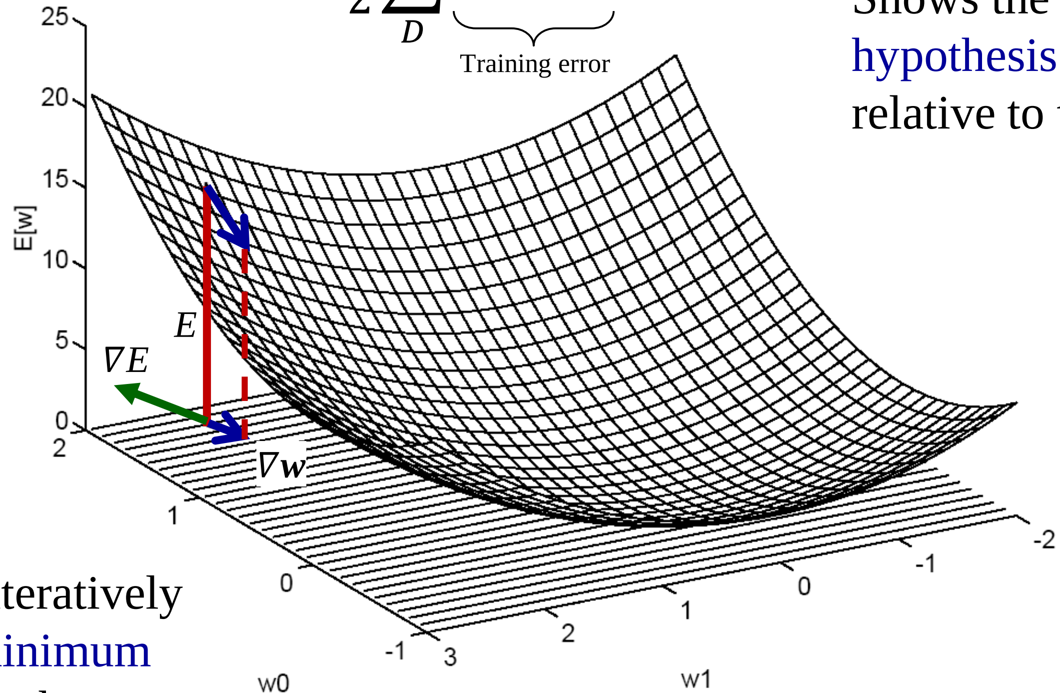
Typical neural network learning

Review

- The target function can be discrete-valued, real-valued, or a vector of several real- or discrete-valued attributes
- Training data: attribute-value pairs $(\boldsymbol{x}_i, y_i)$
  - E.g., for ALVINN, $\boldsymbol{x}_i$ is the input (30x32) image, y_i is the steering direction
- The training data may contain errors (i.e., noisy)
- Long training time, fast execution (evaluation) time
  - E.g., real-time steering response for ALVINN
- In training, use gradient descent to search the hypothesis space of possible weight vectors to find the $\boldsymbol{w}$ that best fits the training examples

The hypothesis space and gradient descent

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{D}\underbrace{(y_i - o_i)^2}_{\text{Training error}}$$



Shows the error E of the hypothesis $\boldsymbol{w} = (w_0, w_1)$ relative to the training data

Gradient descent iteratively searches for the minimum error by moving in the direction $(\delta w_0, \delta w_1)$ that most reduces the error over the whole data set

So $\Delta\boldsymbol{w} = -\eta\nabla E(\boldsymbol{w})$

where $\nabla E(\boldsymbol{w}) = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_n}\right)$ Gradient of E with respect to $\boldsymbol{w}$

The hypothesis space and gradient descent

- Gradient descent is an important general paradigm for learning
- Can be applied whenever
  - The hypothesis space contains continuously parameterized hypotheses
    - E.g., the weights in a linear unit
  - The error on training data can be computed with respect to these hypotheses
- This will converge to a solution even with noisy, non-separable training data
- Practical difficulties in applying gradient descent:
  - Convergence can be slow (e.g., can require thousands of steps)
  - Converges to a local minimum – no global guarantee
- A common variation is incremental gradient descent (a.k.a. stochastic gradient descent) that updates the weights incrementally after each training example

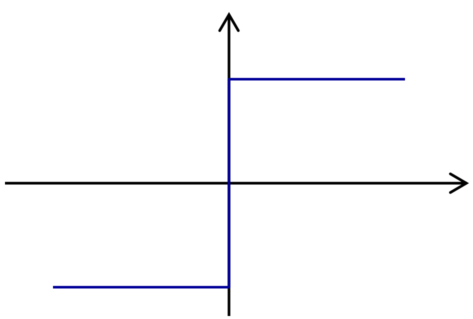
Network output

- Unlike the perceptron, most neural networks output one or more weights (rather than a binary classification)
- So we replace the thresholding unit in the perceptron with the sigmoid (or logistic) function $\sigma(x)$ or the tanh(x) function
  - Typicaly tanh in hidden layers and sigmoid for output nodes

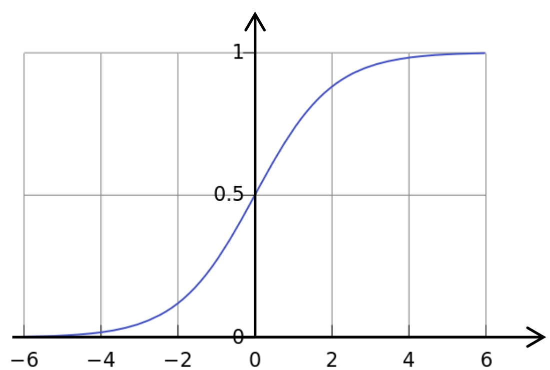
$$f(x) = \begin{cases} 1 & x > 0 \\ -1 & \text{otherwise} \end{cases}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

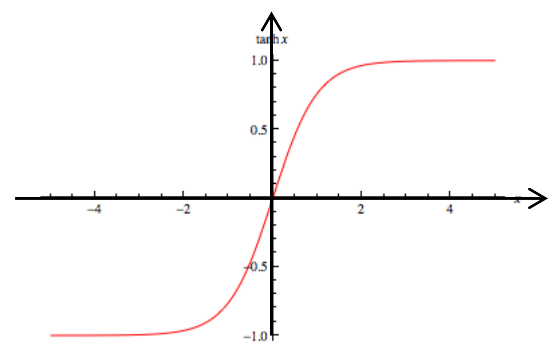
$$\sigma(x) = \tanh(x)$$



$f(x) \in \{-1,1\}$



$f(x) \in (0,1)$

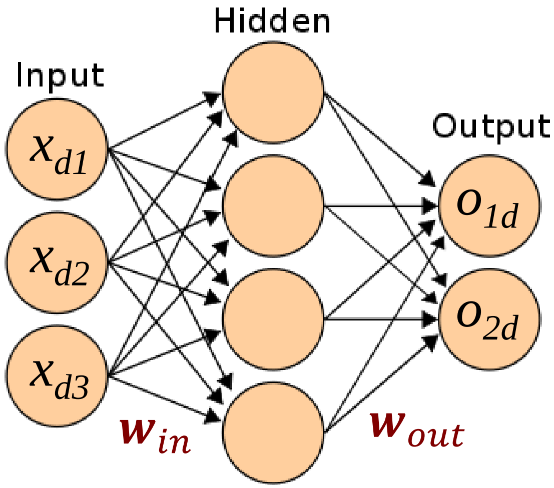


$f(x) \in (-1,1)$

- Nonlinear
- Differentiable

Backpropagation

- The backpropagation algorithm learns weights for a multilayer network (with fixed structure)
- It uses gradient descent to (attempt to) minimize the squared error between the target values and the network output values (for the training data)



Feedforward network with two layers of sigmoid units including 4 hidden units

$$\boldsymbol{w} = (\boldsymbol{w}_{in}, \boldsymbol{w}_{out})$$

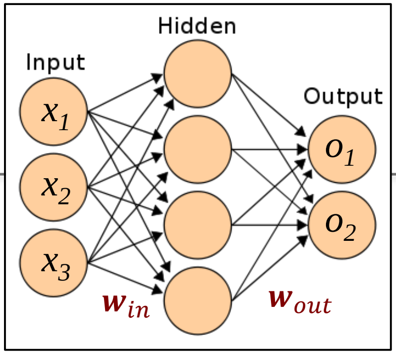
$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{d \in D}\sum_{k \in outputs}(y_{kd} - o_{kd})^2$$

Target (label) Network output

… of the k^{th} output unit for the d^{th} training example

Backprop trains the network by iteratively propagating errors backwards from output units

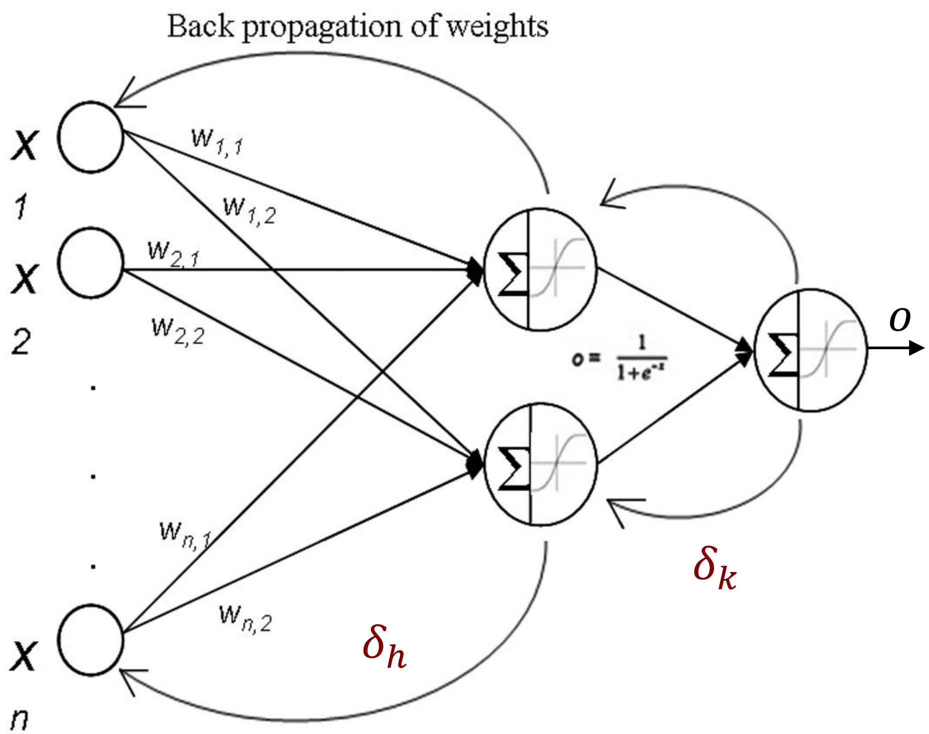
The Backpropagation algorithm

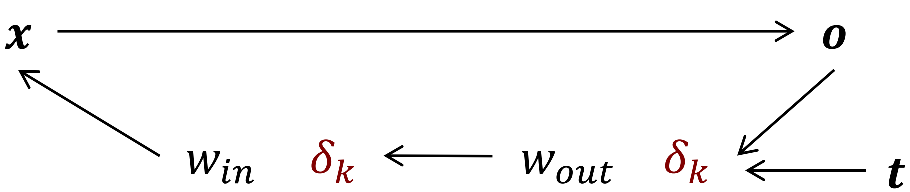


- Initialize all network weights $\boldsymbol{w}$ to small random numbers
- Until termination condition is met, do
  - For each training example $(\boldsymbol{x}, y)$, do
    - Propagate the input forward through the network
      - Input the training instance $\boldsymbol{x}$ and compute the outputs o_k
        - » Using $\boldsymbol{x}$, $\boldsymbol{w}$, and sigmoid functions
    - Propagate the errors backward through the network
      - For each output unit o_k, calculate its error term δ_k
        - » $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)^2$
      - For each hidden unit h, calculate its error term δ_h
        - » $\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh} \delta_k$
      - Update each network weight w_{ji}
        - » $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$

          where $\Delta w_{ji} = \eta \delta_j z_{ji}$, and z_{ji} is the input from node i to node j

The Backpropagation algorithm





Updating weights

Iterative procedure:

- Apply training data $\boldsymbol{x}$ and feed forward to outputs
- Use training labels $\boldsymbol{t}$ to determine output errors
- Propagate errors backwards, updating all weights
- Lather, rinse, repeat…

The Backpropagation algorithm

- Implements a gradient descent search through the hypothesis space
  - Acts linearly early on, when the weights are small, since the sigmoid function for is approximately linear for small inputs
  - When the weights grow, the network starts to learn nonlinearly
- Errors propagate backwards
- Same process for more layers
- Training can be quite slow
- Converges to local minimum (if given enough time)
- Methods to avoid local minimum trap problem include:
  - Run multiple times with different initial weights
  - Use a weight momentum term in the weight update rule
  - Stochastic gradient descent
  - Etc….

Backpropagation: comments

- What is the hypothesis space of Backpropagation?
  - The N-dimensional Euclidian space of the N network weights
  - High-dimensional!
- Since the hypothesis space is continuous and the error function E is differentiable with respect to the weights, it makes an efficient gradient descent approach possible

- What is the inductive bias of Backpropagation?
  (The way generalization is enforced beyond the data points)
  - It's hard to state precisely, but roughly it's *smooth interpolation between data points* or *convexity*
  - I.e., if two positive training example have no negative example between them, backprop tends to label the points in between as positive as well

Backpropagation: comments (cont.)

- There are multiple choices for the termination condition for updating weights
  - A fixed number of iterations (but how many?)
  - Until the error E falls below some predetermined threshold
- Backpropagation is susceptible to overfitting the training data, thus decreasing generalization to unseen examples
- Validation data comes in very useful here
- Typical strategy: Use the training data to train the network, but after every iteration measure error on the validation set
  - Choose the model (weights) that give the smallest error on the validation set
  - This is a cross-validation approach

Feedforward neural networks

- Every Boolean function can be represented by a two-layer network (one hidden unit layer)
  - Though the number of hidden units required may be very large
- Every bounded continuous function can be approximated with arbitrarily small error by a two-layer network
  - Using sigmoid units in the hidden layer and linear (unthresholded) units at the output layer
- Any function can be approximated to arbitrary accuracy by a three-layer network
  - Using sigmoid units in the hidden layers and linear (unthresholded) units at the output layer

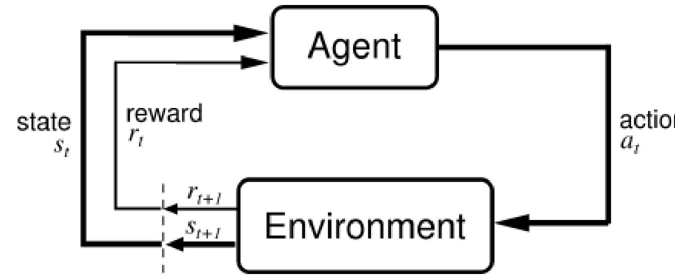
Some other Machine Learning topics

Miscellaneous…

Reinforcement learning

- There are many situations where we don't know the correct answers that supervised learning requires
- Reinforcement learning is the problem faced by an agent that learns behavior (to achieve a goal) through trial-and-error interactions with a dynamic environment based on a reward signal (rather than through supervision)
  - Discover which actions yield the most (cumulative) reward by trying them; i.e., learning by experience
- Examples:
  - Game playing (e.g., chess): player knows whether it wins or loses, but not know how to move at each step
  - Control: a robot juggler
  - Mobile robot navigation
  - Teaching CS courses

Reinforcement learning



- Rewards from sequences of actions
  - Learn action to maximize payoff
    - Not much information in a payoff signal
    - Payoff is often delayed
- How do we learn to choose the actions to maximize reward? This is the problem addressed by RL
- In contrast to supervised learning, the reward only tells us whether the action we chose was good or bad (or led to a good or bad result), not what would have been the "correct" action
- Also, the actions we take and the reward can be separated temporally, so that the problem arises how to assign the reward signal to actions
  - Thus, reinforcement learning has some aspects of supervised learning, but with a very "poor" teacher!

Reinforcement learning: Q-learning

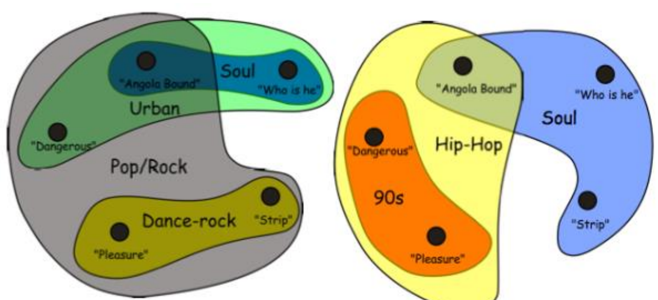
- Q-learning (1989) is a reinforcement learning technique that finds an optimal action-selection policy for any given (finite) Markov decision process (defining states, actions, and rewards)
- It learns an action-value function that gives the expected utility of taking a given action in a given state and following the optimal policy (the action with the highest value in each state) thereafter
- Convergence may be slow…
- There are many variations…

- Reinforcement learning is a very active area of machine learning!

Multi-label classification

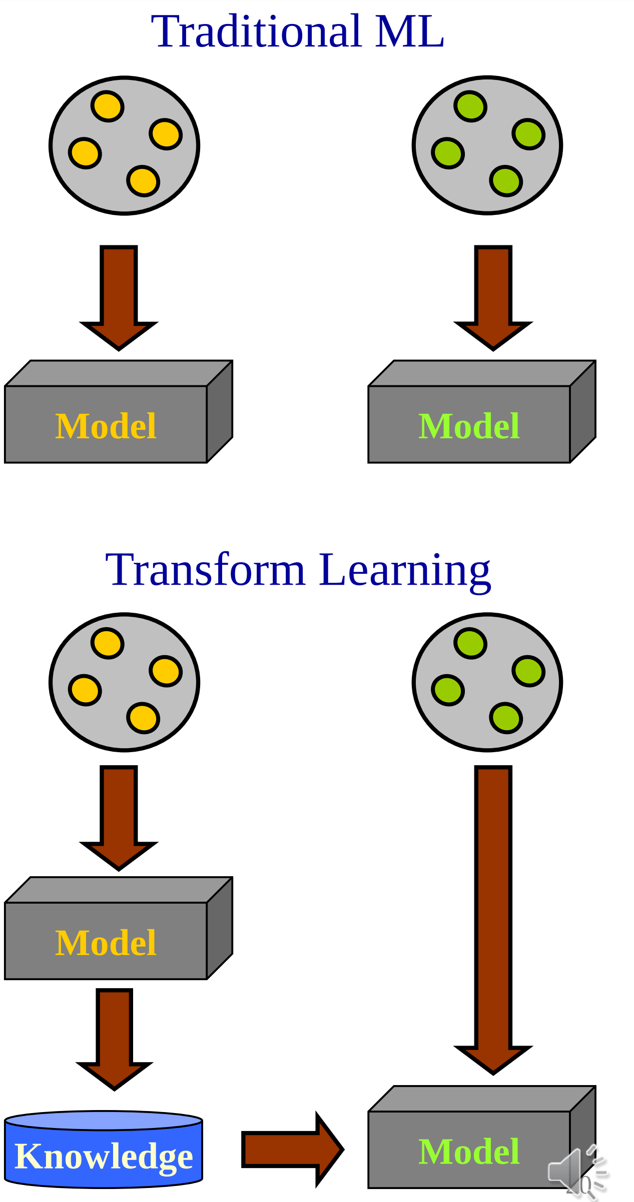
Images			
Labels	tree water black picture drawing sea art blue boat green city	man woman people hair girl picture smile group photo kid family	sky tree water white house window wood sea ocean cloud blue door

- In some problems, class labels are not mutually exclusive
  - E.g., tagging a blog post, labeling an outdoor scene, categorizing a document, medical diagnosis, music categorization
- In multi-label classification, the dependence between labels (classes) is learned, as well as the feature-to-class mapping



Transfer learning (a.k.a. inductive transfer)

- Transfer learning focuses on applying knowledge and skills from one problem to a different but related problem
  - I.e., transferring learning across domains
- In some domains, labeled data may be scarce
- For example:
  - Learning to walk → learning to run
  - Playing the trumpet → playing the French horn
  - Playing the trumpet → playing the guitar
  - Learning French → learning to speak Italian
  - WiFi localization in different spaces
  - Sentiment classification



Online (incremental) learning

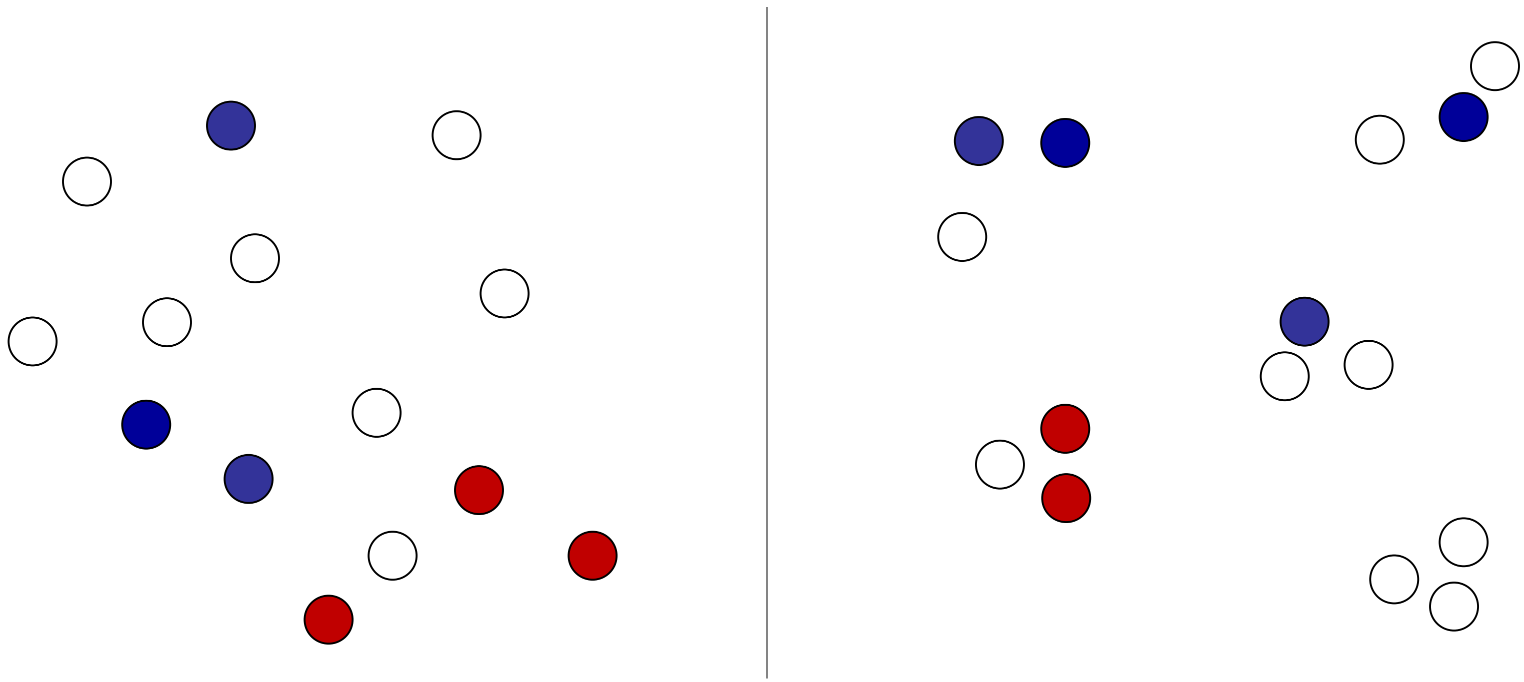
- Sometimes data is not available all at once; it may become available sequentially over time
- Online learning (incremental learning) updates the model each time a new data point arrives
  - Learning takes place continuously, rather than one-shot (batch) learning then applying the learned model
- For example:
  - Visual tracking
  - User modeling
  - Intelligent agents
  - Sequence prediction
- Many traditional ML techniques have incremental versions
  - SVM, PCA, GMM, regression methods, …

Active learning

- Active learning is a type of semi-supervised learning in which the algorithm queries the information source (e.g., the user) to obtain the desired model
- Sub-problem: Given the labeled data points and tentative model, which data point should be labeled/explored next?
  - E.g., in robotics – where to point the camera or sensors in order to gain the most useful information?
- Typically, there are relatively few labeled data points and very many unlabeled data points (or things that can be explored) – but limited resources or time in which to do so
  - Labeling/exploration can be expensive and time-consuming
- Some methods exploit structure in the data to infer which unlabeled points would be most helpful to label

Active learning

Which of the unlabeled points would be most useful to label?



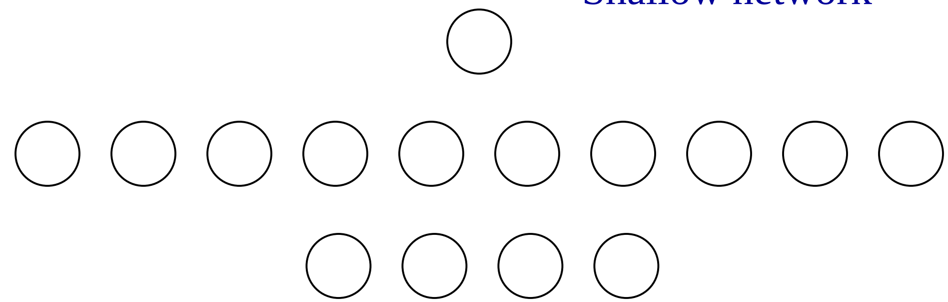
Deep learning

- Deep learning is about learning deep (many-layered) neural networks – multiple non-linear transformations
- Biological motivation: The human brain is a deep neural network, has many layers of neurons which acts as feature detectors, detecting more and more abstract features as you go up
- The more layers in a network, the more abstract features can be represented
- E.g. to classify or detect a cat in an image:
  - Bottom layers: Edge detectors, curves, corners straight lines
  - Middle layers: Fur patterns, eyes, ears
  - Higher layers: Body, head, legs
  - Top layer: Cat

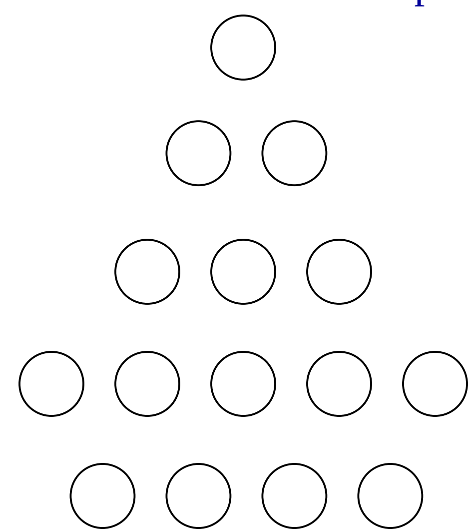
Deep network structure

- Although 2- and 3-layer neural networks have been shown to be able to approximate any function, they may require exponential size
  - I.e., very wide, shallow networks
- In a deep network, higher levels can express combinations between features learned at lower levels

Shallow network

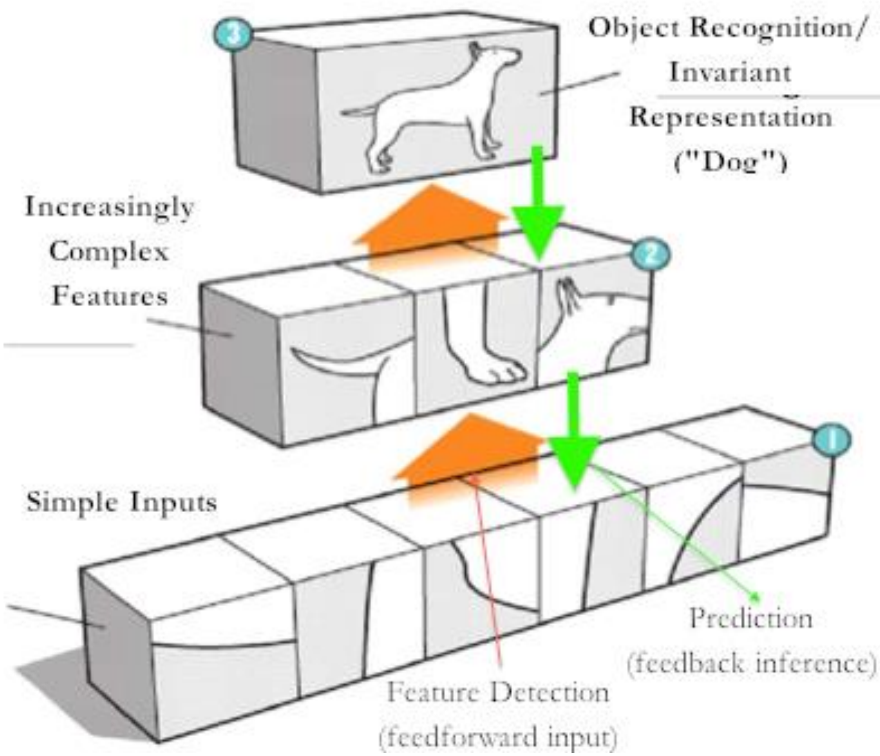


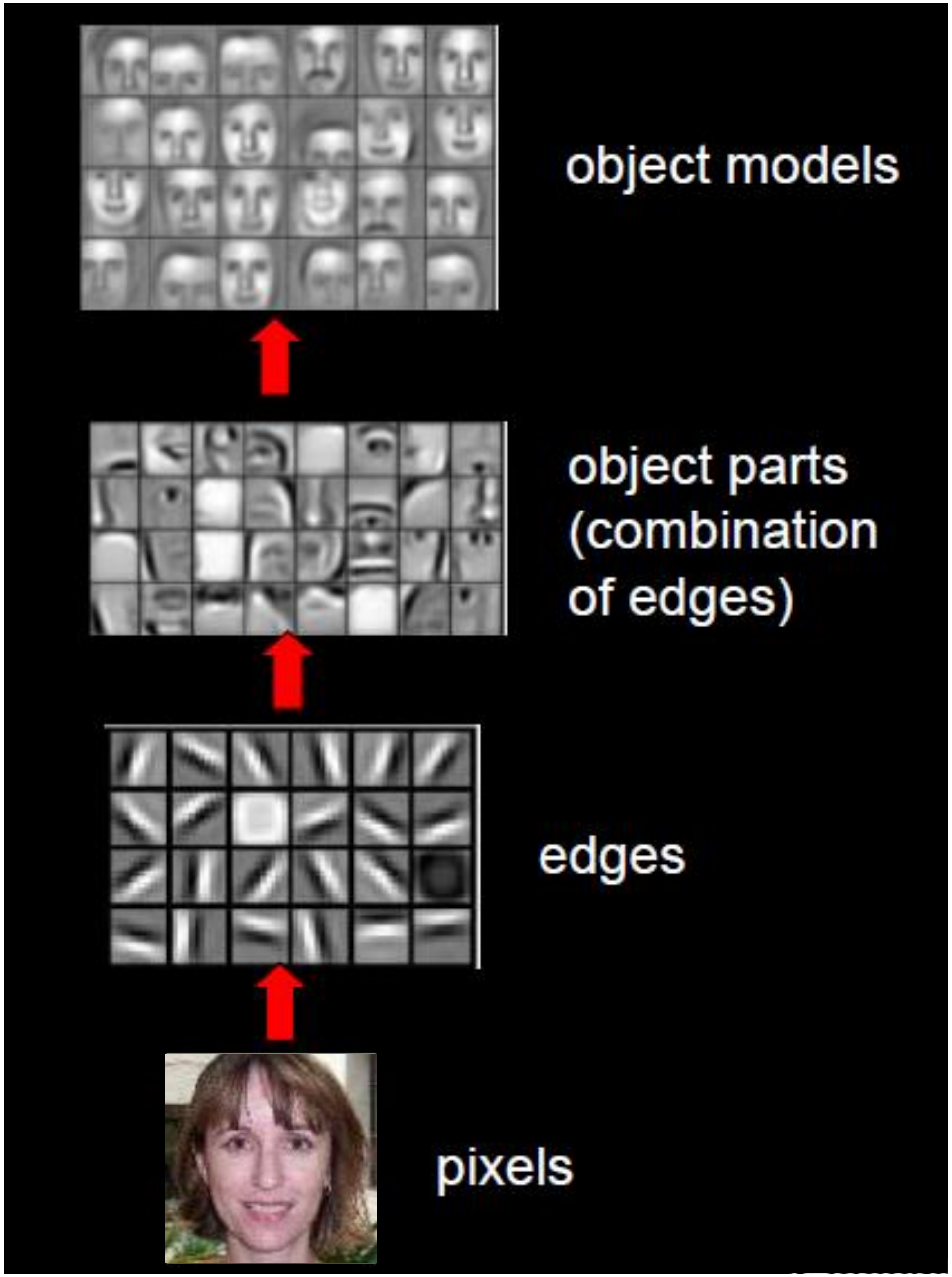
Deep network



Deep learning

- Each level creates new (more complex or abstract) features from combinations of features from the level below




object models
object parts
(combination
of edges)
edges
pixels

Deep learning

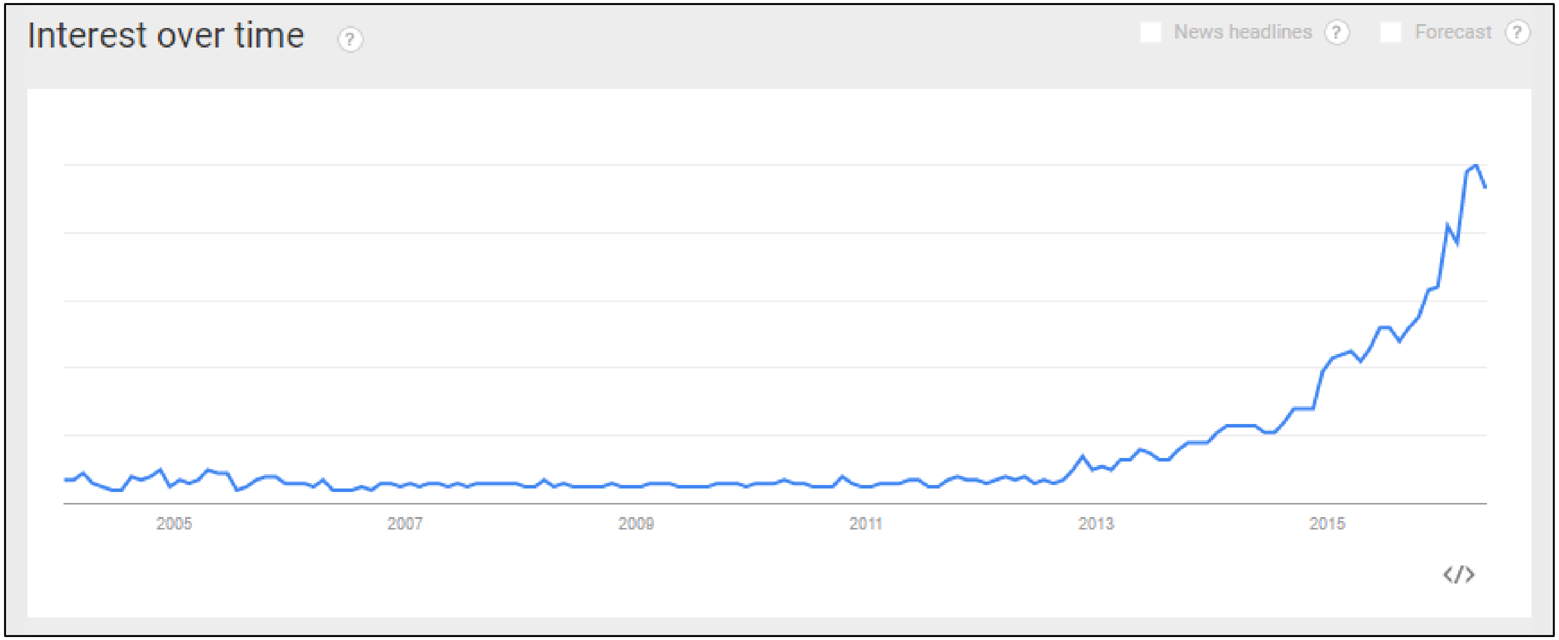
- The traditional neural network approach is to use back propagation (or similar methods) to train multiple layers
  - But back propagation does not work well over several layers, does not scale well, and cannot leverage unlabeled data
- Recent advances in deep learning attempt to address these short-comings
- Deep networks take advantage of unlabeled data by learning good representations of the data through unsupervised learning
  - Reduces the need for manual "feature engineering"
  - Latent (hidden) features are learned from the unlabeled data
- So deep learning is a semi-supervised approach that combines bottom-up, unsupervised training with backprop-like training on labeled data

Success of deep learning

- State-of-the-art performance in a wide range of different areas:
  - Language Modeling (2012, Mikolov et al.)
  - Image Recognition (2012 ImageNet competition, Krizhevsky)
  - Sentiment Classification (2011, Socher et al.)
  - Speech Recognition (2010, Dahl et al.)
  - MNIST hand-written digit recognition (2010, Ciresan et al.)
- What do these problems have in common?
  - Each are non-linear classification problems where the information is highly hierarchal in nature
  - Problems that humans excel at and machines do very poorly
- Andrew Ng – Machine Learning Professor, Stanford/Baidu:

  "I've worked all my life in Machine Learning, and I've never seen one algorithm knock over benchmarks like Deep Learning."

Popularity of deep learning



Google Trends for “Deep learning”

Some disadvantages of deep learning

- The models are very complex, with lots of parameters to choose and optimize:
  - Number of layers, size of layers, node functions
- Very slow to train
- Some problems more amenable to deep learning than other applications
  - Simpler models may be sufficient for many domains
- The learned models can be very hard to explain (e.g., compared with decision trees)
  - What does neuron 524 do?
- Is deep learning being over-hyped?

What's Hot in Machine Learning

- You mean besides deep learning???
- Did I mention deep learning?
- Deep convolutional networks
- Recurrent networks
- Tensor methods
- Big data, data science, data analytics
- Applications to… everything
- Personal agents (Siri, Google Now, Amazon Echo, etc.)
- And don't forget… deep learning!