Machine Learning

CS 165B

Prof. Matthew Turk

Wednesday, May 25, 2016

T
o
d
a
y

- Machine learning experiments (Ch. 12)
- Neural networks

Notes

- Homework assignment #5 available tomorrow, due Friday, June 3
- This Friday
  - Four Eyes Lab open house and Media Arts & Technology End of Year Show (Elings Hall) – 5-9pm
- Next week:
  - No class on Monday (Memorial Day holiday)
  - Tuesday’s discussion sessions
    - Final exam review
  - Wednesday’s class – will be recorded, posted by Wednesday a.m.
    - So don’t come to class on Wednesday either!
- I’ll be available for office hours this week and exam week

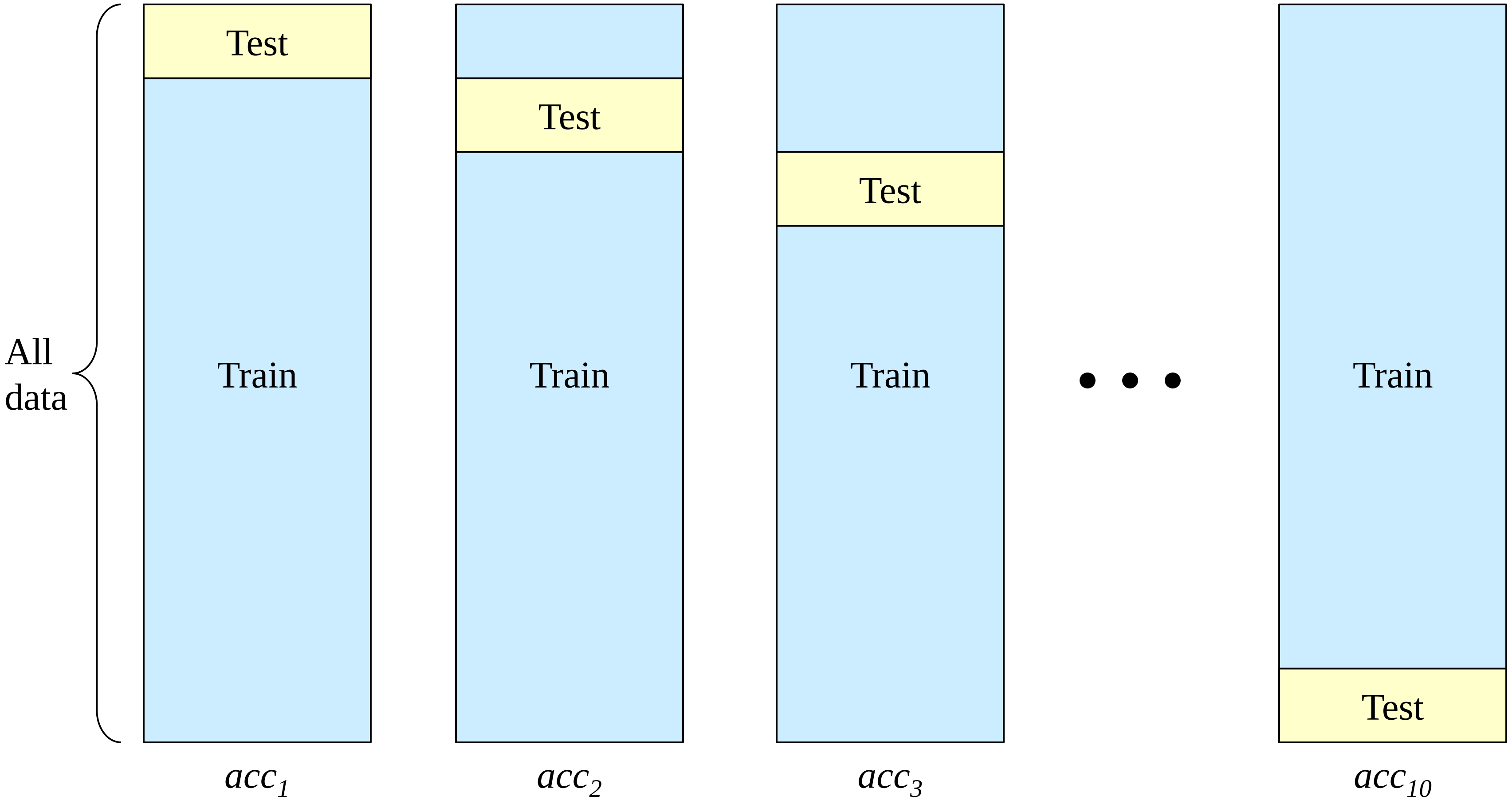
Notes

- Final exam (Wednesday, June 8, 8-11am, here)
  - Similar to the midterm in style – important to understand the concepts we've covered and how to apply them
  - Covers all material of the course
    - More weight on the material since the midterm
  - Practice exam will *probably* be provided
    - Good practice: midterm, homeworks
  - Closed book/notes
    - Exception: You may bring one 8.5"x11" sheet of paper with your notes (both sides)
    - I'll also provide some information, formulas, etc. (will be posted early or included with the practice exam)

Performance estimation (testing)

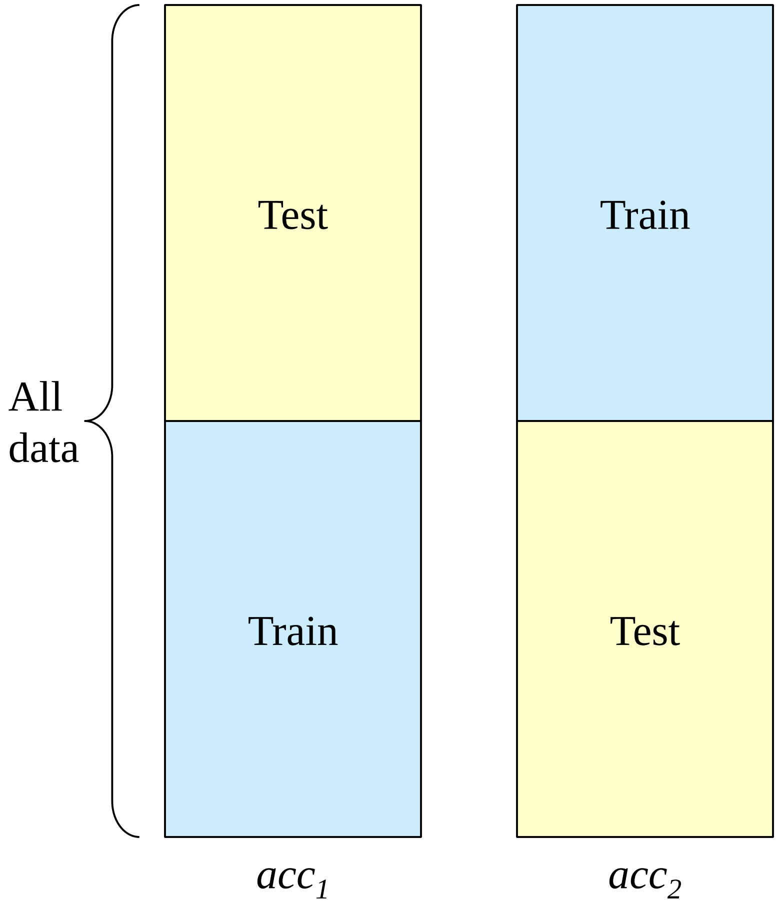
- We want to estimate the true error rate = the error rate on the entire population (not just on your data set)
- You cannot use your full dataset to train a model, estimate the model parameters, and estimate the performance (e.g., accuracy or error rate) of your model
  - This results in overfitting and overly optimistic results!
- Instead, we split the data into disjoint sets
  - Holdout method
    - Keep some data separate (typically ~30%) for testing
  - k-fold cross-validation: average the *k* different performance estimates
  - Leave-one-out cross-validation
    - Use *k* = the number of data points

Example: 10-fold cross-validation



$$acc = \text{Average}(acc_i)$$

Example: 2-fold cross-validation

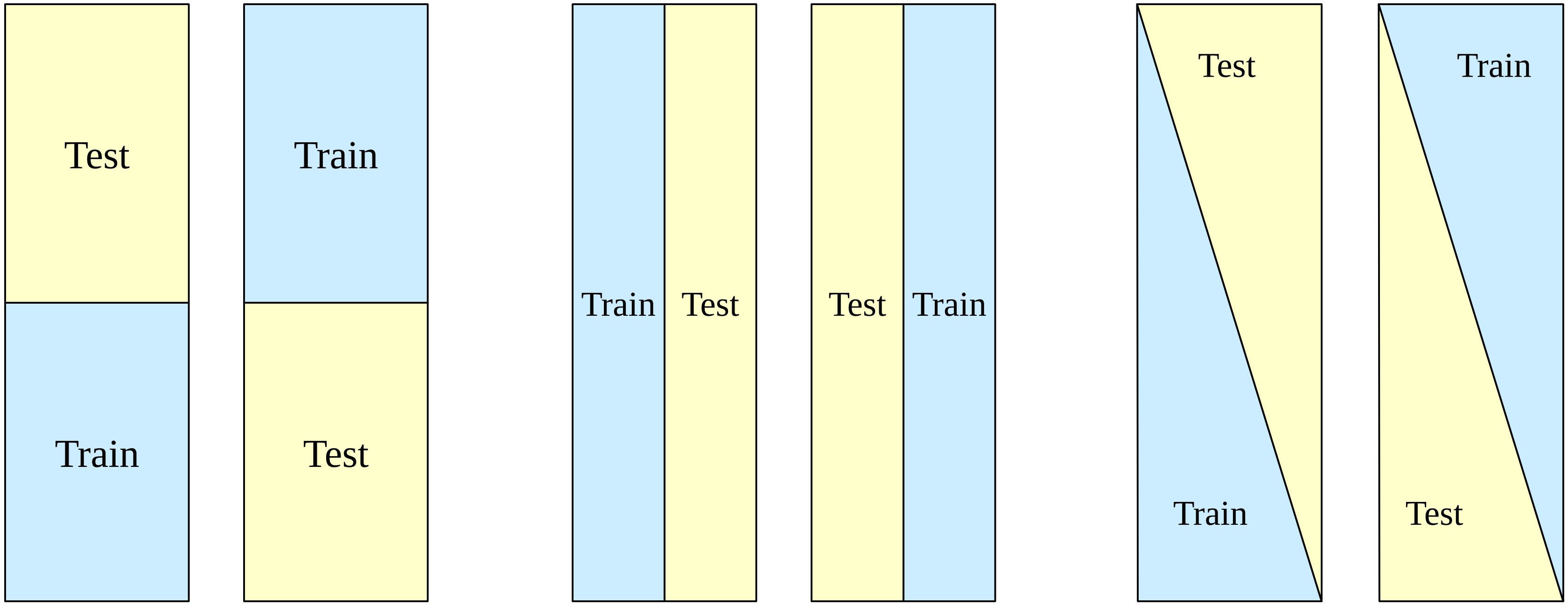


$$acc = ½\,(acc_1 + acc_2)$$

Repeated random sub-sampling validation

Randomly split the data into k folds N times, and then average the N results of k-fold cross-validation

- Typically, $k = 2$



acc_1 acc_2 acc_1 acc_2 acc_1 acc_2

$acc_a = ½\,(acc_1 + acc_2)$ $acc_b = ½\,(acc_1 + acc_2)$ $acc_c = ½\,(acc_1 + acc_2)$

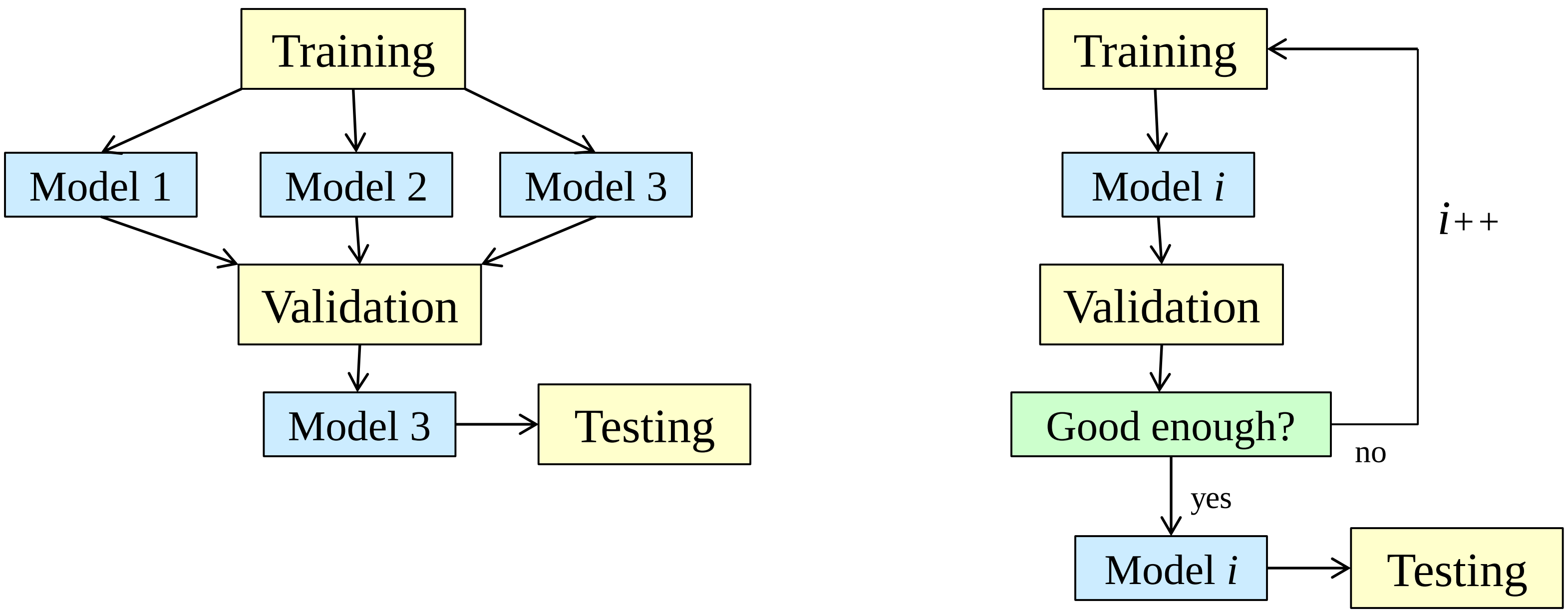
$$\boxed{\hat{a} = (acc_a + acc_b + acc_c)/3}$$

Cross-validation

- In practice, the choice of the number of folds depends on the size of the data set
  - For large data sets, fewer folds are needed
  - For sparse data sets, leave-one-out may be best
- We should have some notion of acceptable variance for the cross-validation performance measure
  - If the variance is high, we probably need more data!
- If we are satisfied with the performance of our learning algorithm, we than run it over the entire data set (i.e., train on the complete data set) to produce the final model
  - Hence the term cross-validation rather than cross-testing!

Training, validation, and test datasets

- We typically divide the total dataset into three subsets:
  - Training data is used for learning the parameters of the models
  - Validation data is used to decide which model to employ
  - Test data is used to get a final, unbiased estimate of how well the model works



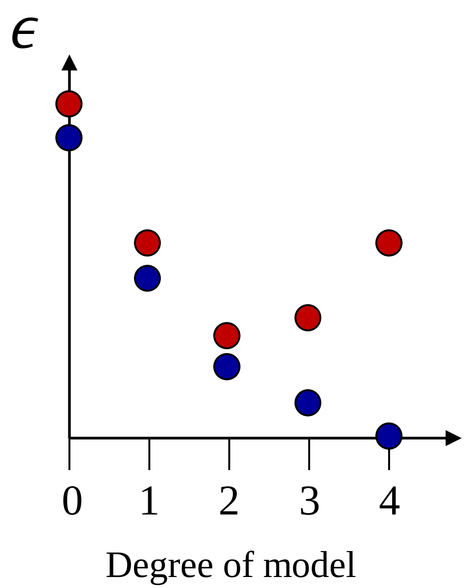
Often reduced to training and testing
a single mode (no validation step)

Training, validation, testing, and deployment

- Testing data is used to predict the performance of your ML model on the real problem
  - Applied after the model is finalized; it is not used to modify the model
  - In other words, testing is just a substitute for deploying your model in a product and gathering data from millions of users
  - Only training and validation data can change your model
- Testing on the training data is cheating!
  - Training/validation data and testing data must be completely separate
  - The test data is (conceptually) stored in a vault until ready to be used for testing the completed model
  - You cannot modify the model after assessing it with the testing data
    - At least, you can't modify it and re-test
- Generally, $\epsilon_{training} < \epsilon_{validation} < \epsilon_{testing}$

Typical training/validation/testing example

- For a 1D regression problem, let's train different models (different regression functions):
  - 0-degree polynomial (a constant), 1-degree polynomial (a line), 2nd-degree polynomial, 3rd-degree polynomial, 4th-degree polynomial
- Errors on the training data:



Which is the best model?

Answer: We don't know!

Use the validation data to determine:

The second-degree polynomial has the lowest error on this validation data

Now run cross-validation on the test data to estimate the performance of your model

Produce the final model by using all your data

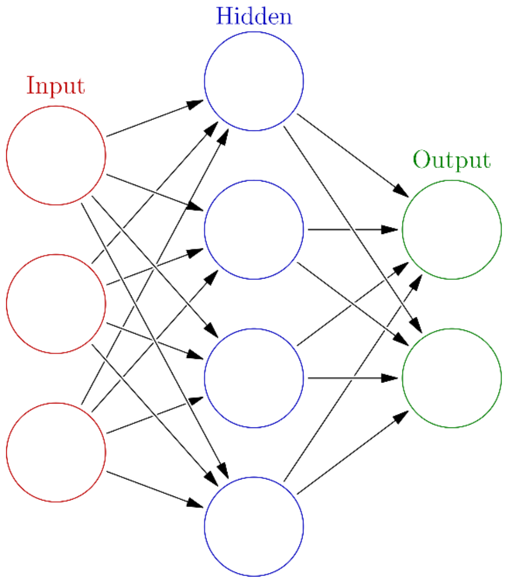
Ship it!

Neural Networks

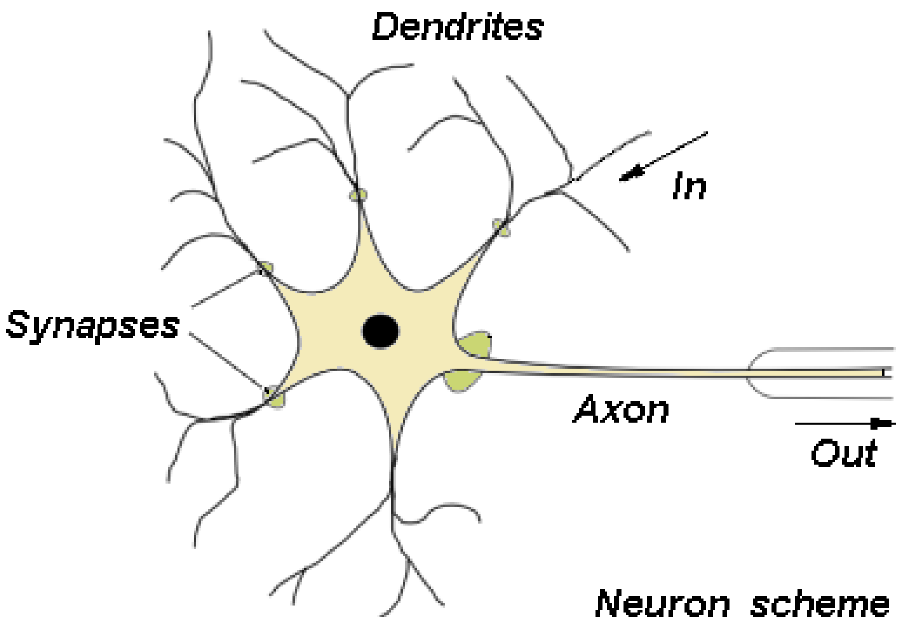
Not covered in the textbook

Neural networks

- A.k.a. *connectionism*
  - Alternative to *symbolism* in AI
- Networks of processing units (neurons) with connections (synapses) between them
  - Learning by tuning weights
  - Highly parallel, distributed processing
- Inspired by the brain:
  - Large number of neurons: $\sim 10^{11}$
  - Large connectitivity: $\sim 10^{4}$
  - High degree of parallel processing
  - Distributed computation/memory
  - Robust to noise, failures



Biological neurons



Dendrites

- Nerve fibers carrying electrical signals to the cell

Cell body (soma)

- "Computes" a non-linear function of its inputs

Axon

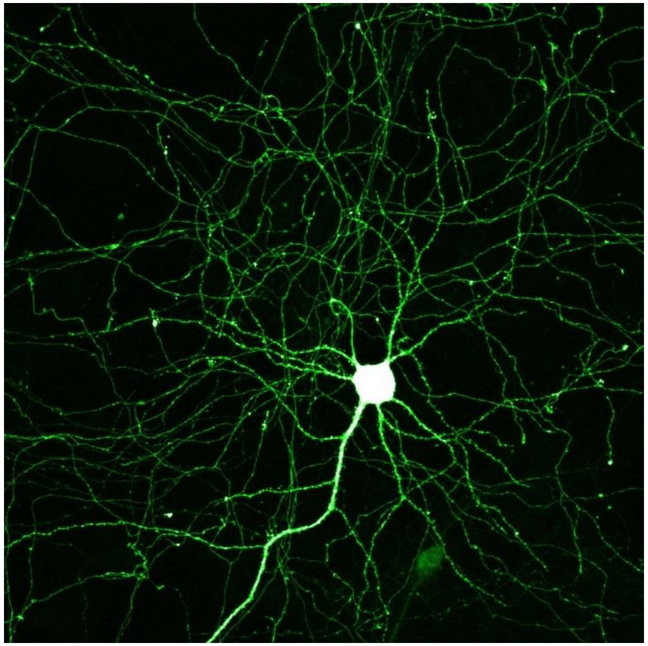
- Single long fiber that carries the electrical signal from the cell body to other neurons

Synapse

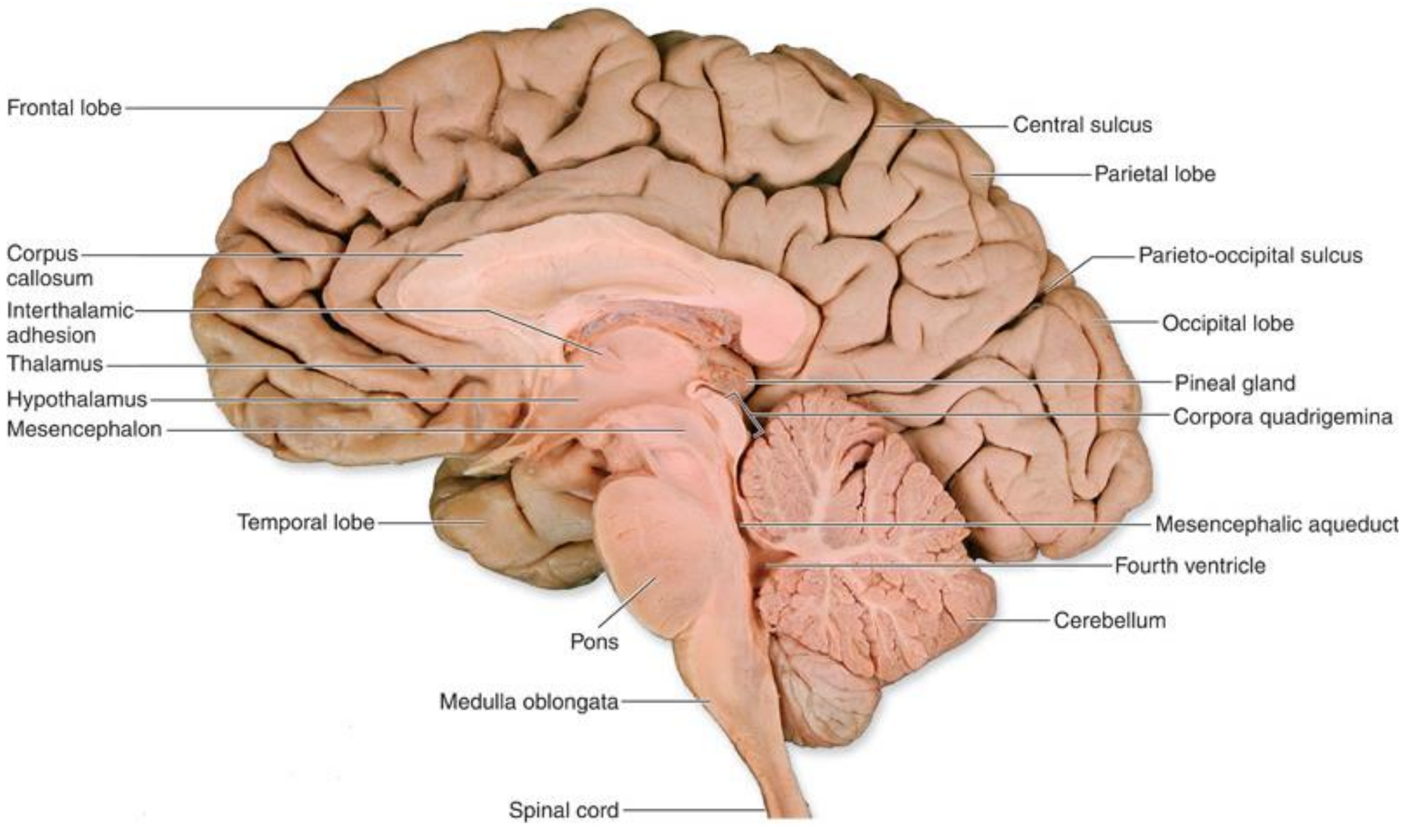
- The point of contact between the axon of one cell and the dendrite of another, regulating a chemical connection whose strength affects the input to the cell

Biological neurons

- A variety of different neurons exist (motor neurons, various visual neurons, etc.), with different branching structures
- The connections of the network and the strengths of the individual synapses establish the function of the network.

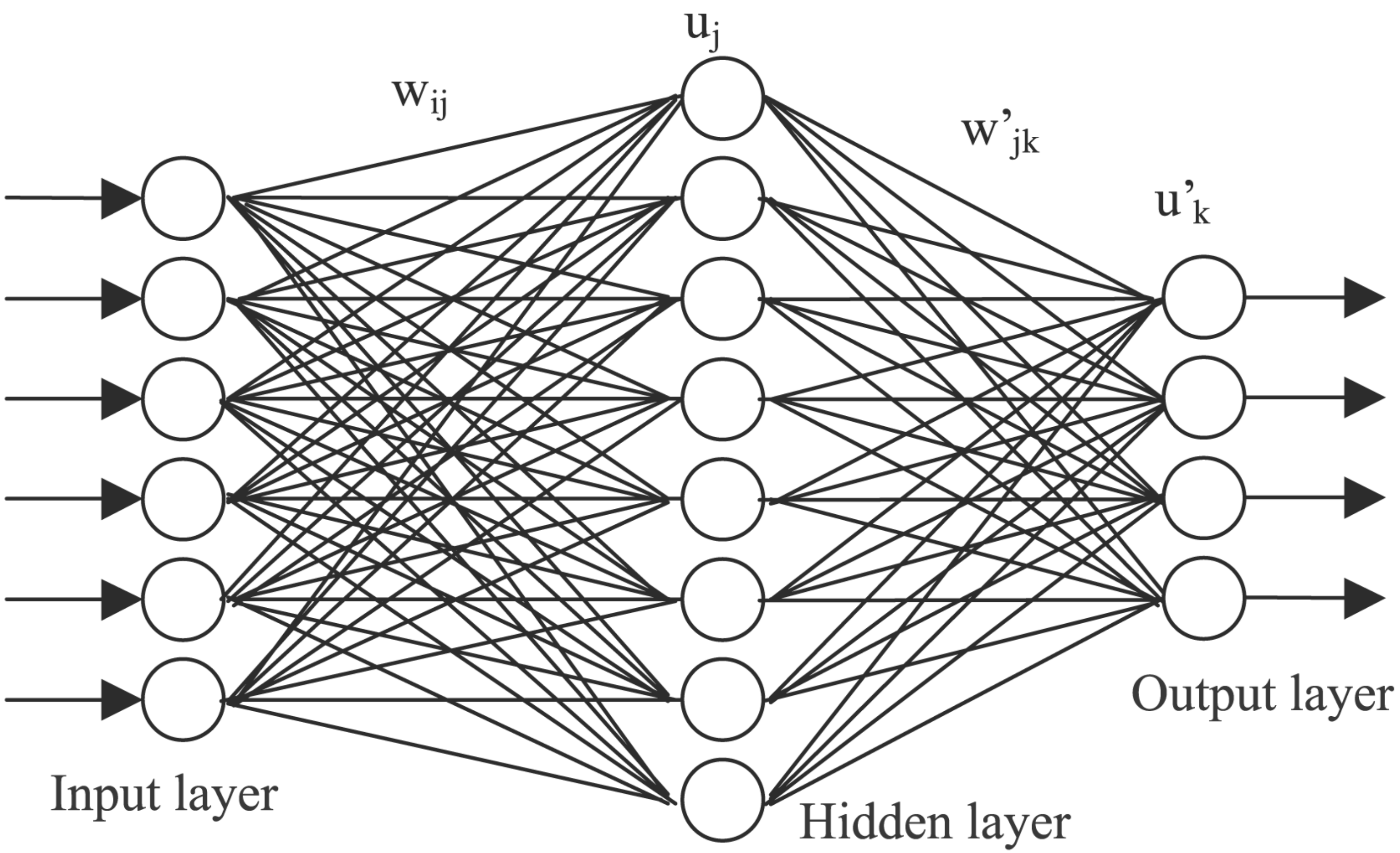


Brains = Neural networks??



Thoughts	Emotions	Language	Consciousness	Balance
Muscle control	Sensing	Reasoning	Planning	Timing
Feelings	Perception	Decision making	Memory	Coordination

Brains = Neural networks??



Note: This is a two-layer (not three-layer) neural network. The input layer does not count.

So… what do neural networks do?

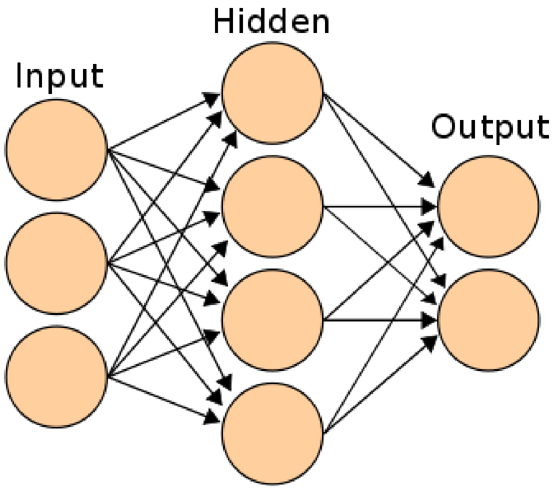
- Neural networks provide a way to learn functions, which are used for the types of machine learning problems we've been discussing: classification, clustering, regression, etc.
- NNs provide a family of techniques to address ML problems:
  - Feedforward neural network
  - Convolutional neural network
  - Radial basis function (RBF) network
  - Kohonen self-organizing network
  - Learning Vector Quantization
  - Recurrent neural network
  - Hopfield network
  - Boltzmann machine
  - Associative neural network
  - Cascading neural network
  - Etc., etc.

Keep in mind:
A neural network is a tool, not a solution!

Artificial neural networks

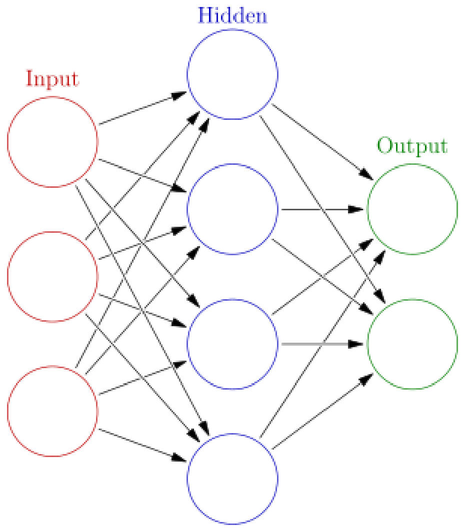
- Inspired by our understanding of the brain, an ANN is a set of simple processing elements (nodes or neurons) connected together to form a network that shares some properties with connected networks of neurons in the brain
- Neural networks typically have these characteristics:
  - Node connections with adaptive (learnable) weights
  - Can approximate nonlinear functions of the inputs
  - Highly parallel (conceptually – may be implemented serially)

A simple 2-layer neural network



The NN learning problem: from training data, learn the weights

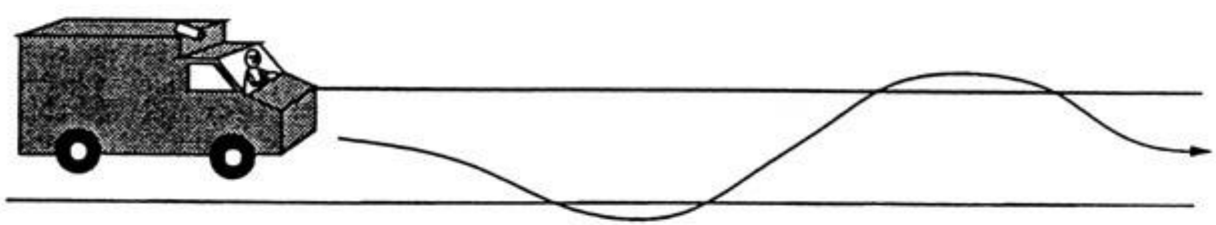
When to consider using an ANN

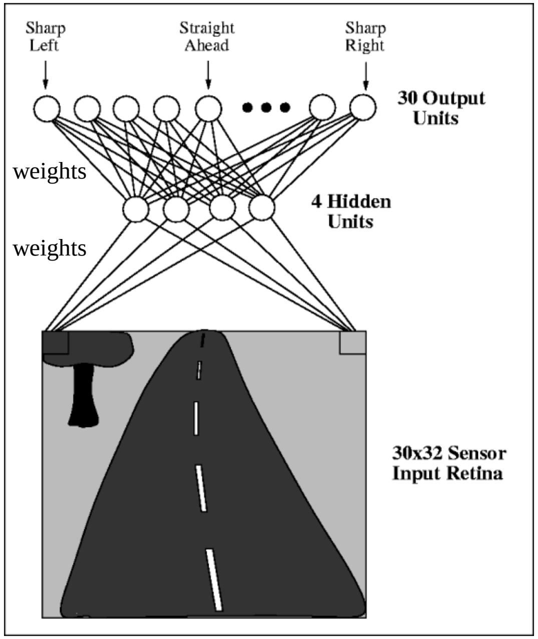


- When the input is
  - High-dimensional
  - Not well understood
  - Noisy
- Form of target function is unknown
- Long training times are acceptable
- Human readability is unimportant
  - Don't necessarily need to understand what's "under the hood"
- Especially good for complex pattern recognition problems, such as:
  - Speech recognition
  - Image classification
  - Financial prediction

Problems “too hard to program”

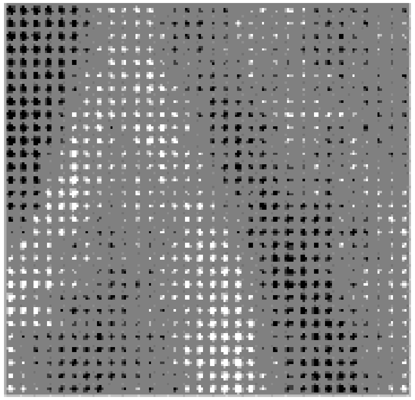
ALVINN: a NN perception system which learns to control the CMU NAVLAB vehicles, trained by observing a person drive





30x32 = 960 inputs

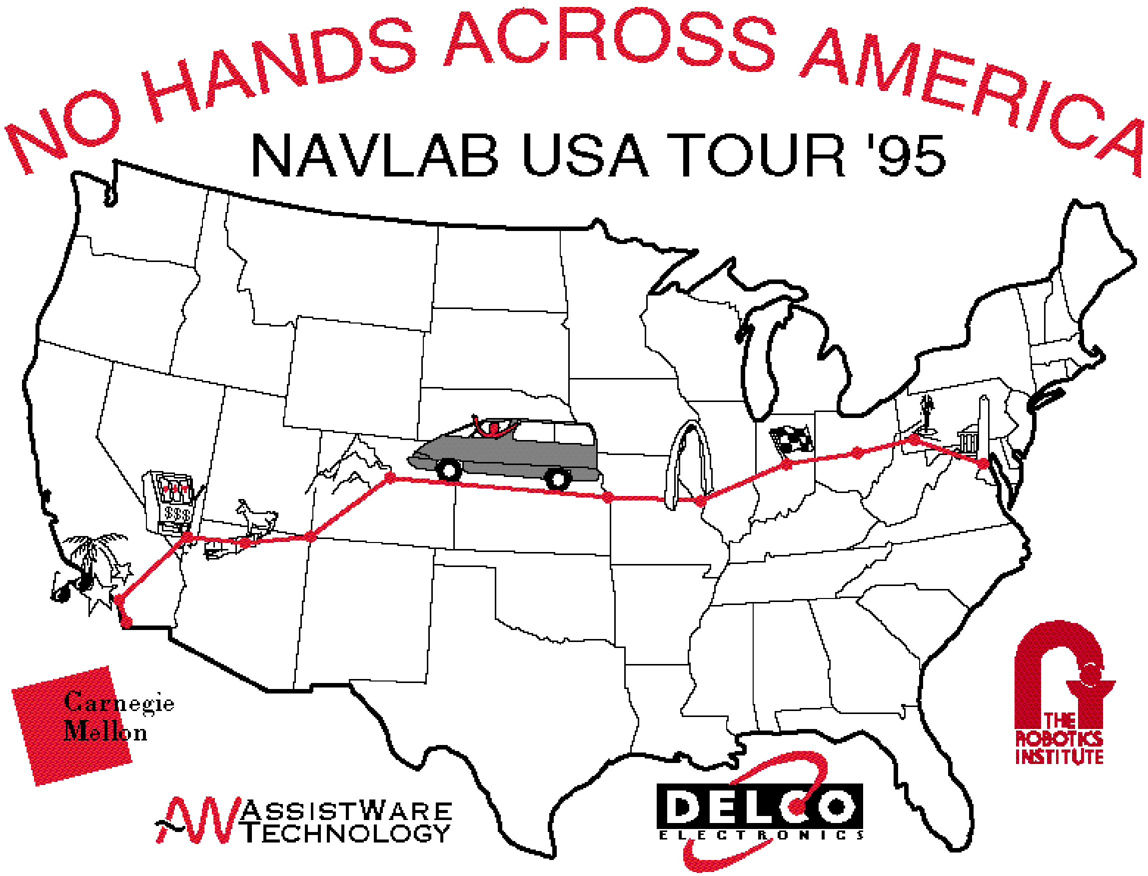
Outputs encode steering direction



w values for one of the hidden units

1995

2797/2849 miles
(98.2%)
Speeds up to 70 mph



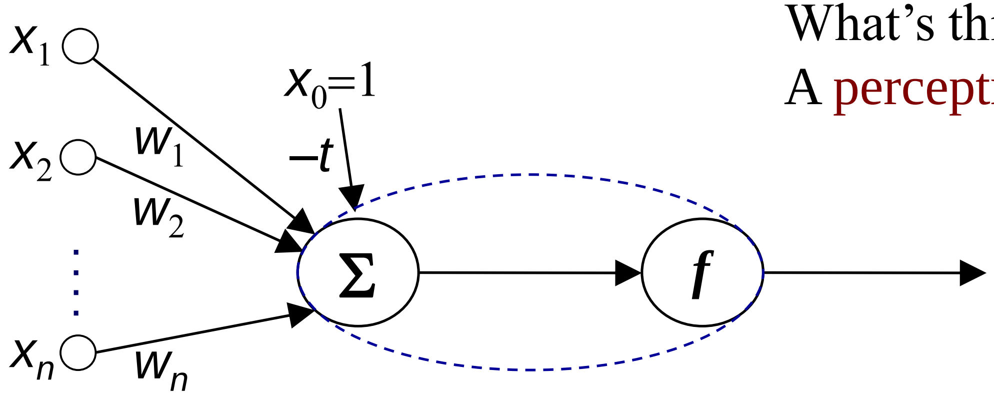
- Washington DC
- Pittsburgh PA
- Columbus OH
- Indianapolis IN
- Kokomo IN
- Saint Louis MO
- Kansas City KA
- Denver CO
- Four Corners
- Grand Canyon
- Las Vegas NV
- Los Angeles CA
- San Diego CA

https://www.youtube.com/watch?v=xkJVV1_4l8E

More details: https://www.youtube.com/watch?v=Tat70DqpKw8

A simple neural network

What's this neural network called?
A perceptron! (I.e., a linear classifier)



$$o = f\left(\sum_{i=0}^{n} w_i x_i\right)$$

Inputs **x** | Weights **w** | Sum of weighted inputs | (Threshold) function

$-t$: threshold value or bias

$$\underbrace{\left(\sum_{i=0}^{n} w_i x_i\right)}_{\text{Homogeneous}} = \underbrace{\left(\sum_{i=1}^{n} w_i x_i\right) - t = w^T x - t}_{\text{Non-homogeneous}}$$

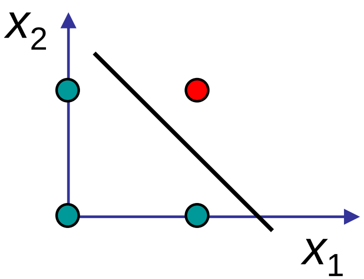
f : activation function – may be a thresholding unit (binary output):

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

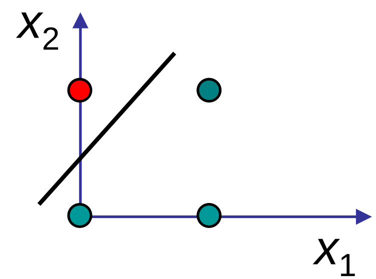
What can be decided by a perceptron?

- The decision surface is a hyperplane given by $\sum_{i=0}^{n} w_i x_i = 0$
  - 2D case: the decision surface is a line
  - 3D case: the decision surface is a plane
  - N-D case: the decision surface is a (N-1)D hyperplane
- Represents many useful functions: for example:

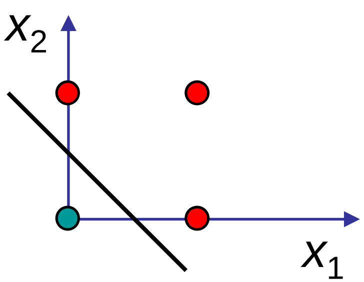
$x_1 \wedge x_2$



$\neg x_1 \wedge x_2$

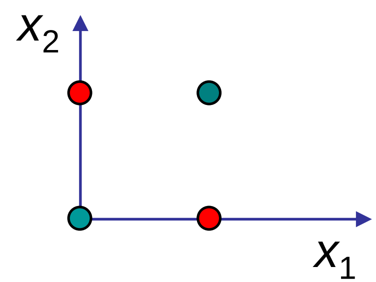


$x_1 \vee x_2$



x_1 XOR x_2

NO!

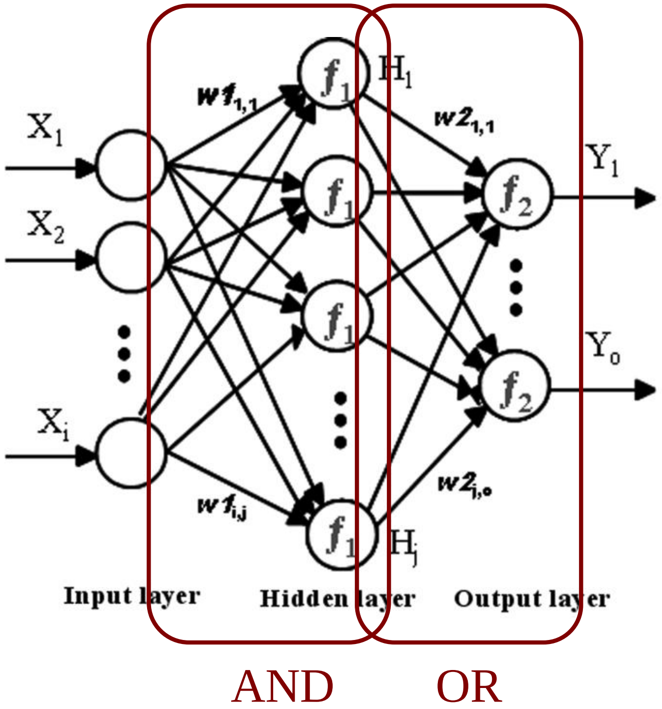


Implementing general Boolean functions

- Solution:
  - A network of perceptrons
  - Any Boolean function representable as disjunctive normal form (DNF)
    - 2 layers
    - Disjunction (layer 2) of conjunctions (layer 1)
- Example of XOR in DNF

  $x_1 \text{ XOR } x_2 = (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$

- Practical problem of representing high-dimensional functions



Feedforward network
(no cycles in the network)

As opposed to a **recurrent** network

Typical neural network learning

- 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