349:Machine Learning

Fall 2024

Miscellaneous Topics

Neural Networks

Feed-forwards neural networks form the basis for a variety of specialized architectures:

- Basic components of the architectures generally remain the same: weights, biases, non-linearities, etc.
- Objective functions incorporate additional attributes and regularizes, but will generally be based upon cross-entropy, means squared error (MSE), etc.
- Output layers associated with regression and classification are still used
- Components that change are often the activation functions, connections among the various network components and the way in which data is fed into the network

Certain architectures are associated with certain domains

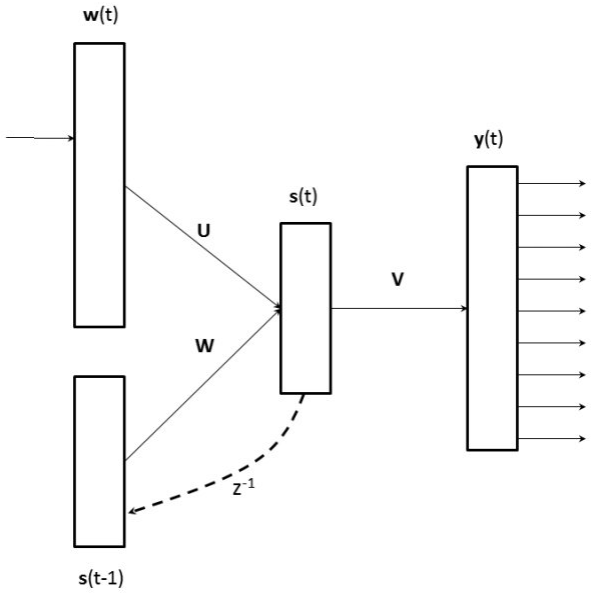
Recurrent Neural Networks

Feed forward networks are limited in handling long-term dependencies:

```
the clouds are in the [sky]
```

```
I grew up in France… I speak fluent [French]
```

Recurrent Neural Network language models were introduced in 2010 to better handle long-term dependencies:



$$x(t) = w(t) + s(t-1)$$

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right)$$

$$y_k(t) = g\left(\sum_j s_j(t) v_{kj}\right)$$

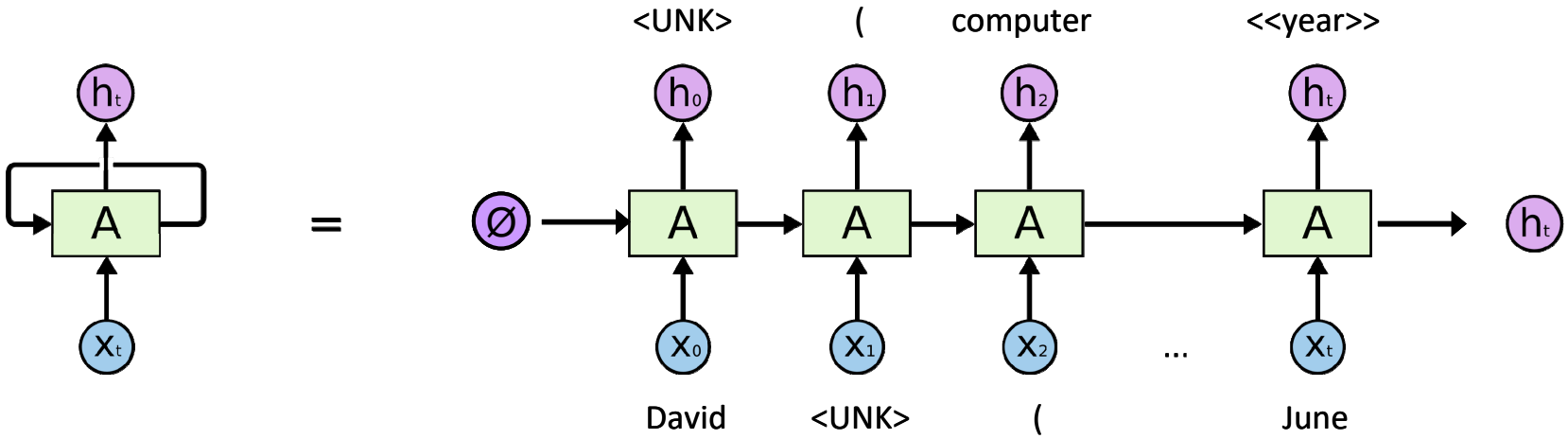
$$f(z) = \frac{1}{1+e^{-z}}$$

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

Recurrent Neural Networks

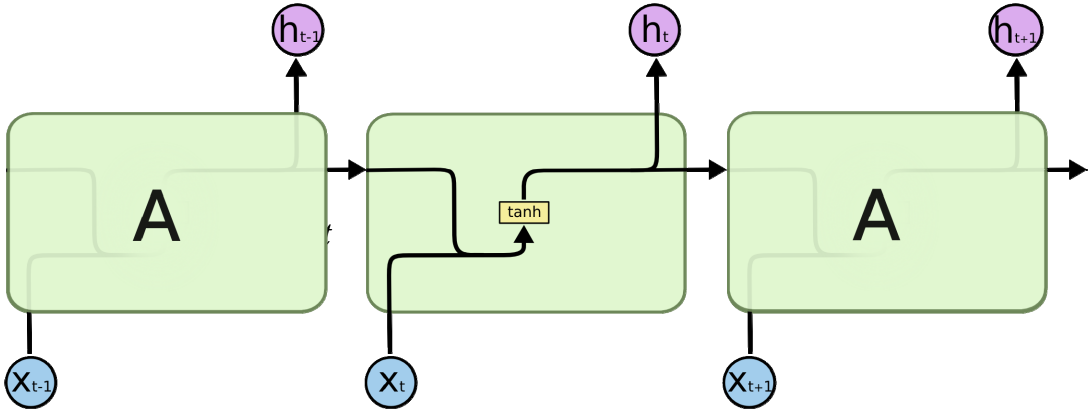
RNNs have loops, which are used to achieve recurrency:

- A RNN can be unrolled to look like a series of feed forward neural networks with some information flow between networks
- The number of time steps quantifies the length of the RNN when unrolled
- The hidden state h_t carries all information from preceding steps and in theory provides an unlimited context
- This results in observations being fed into a RNN differently than a feed forward language model

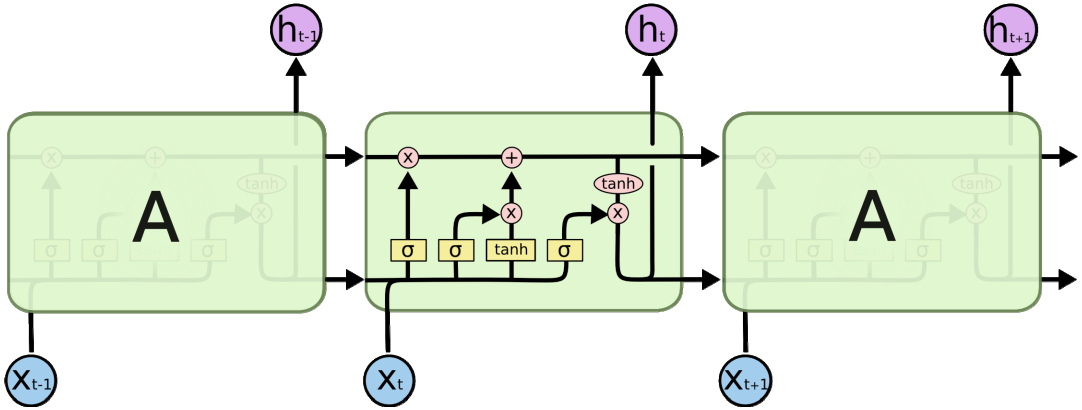


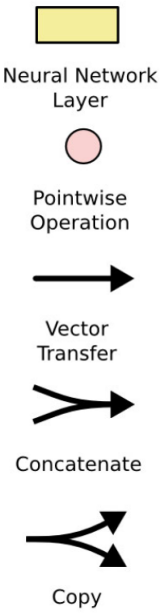
Long Short-Term Memory (LSTM) Cells

All RNN are formed by a chain of repeating modules:

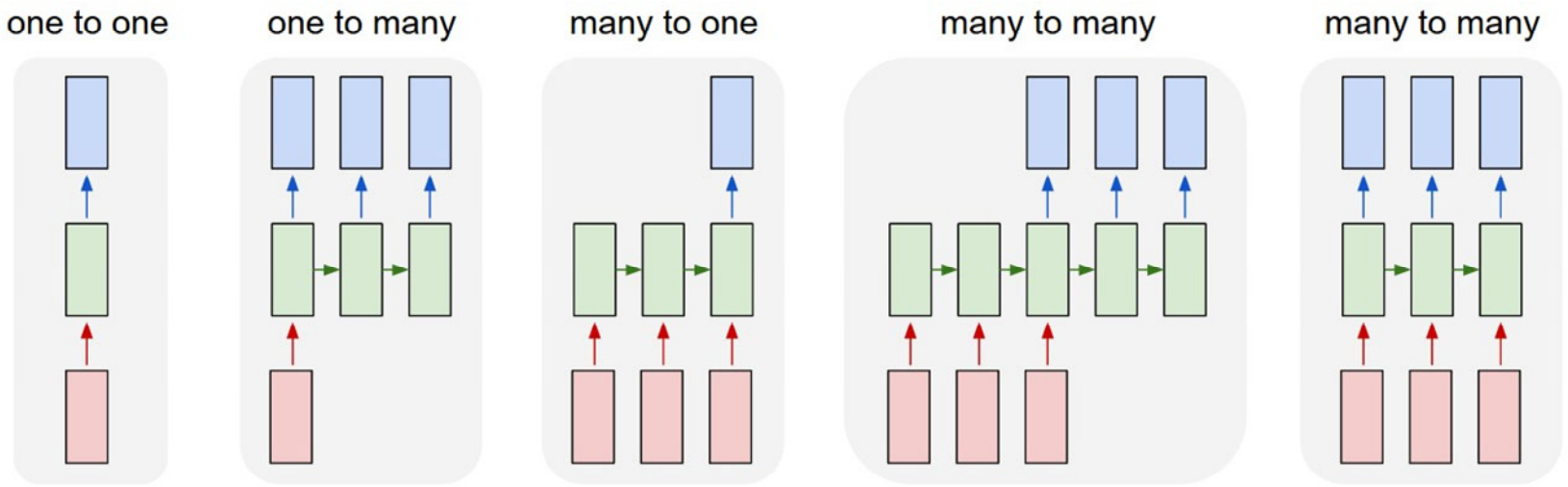


LSTMs are a special kind of RNN with four interacting gates





Various RNN (and LSTM) Configurations



No RNN (Vanilla)

Image Captioning

Sentiment Analysis

Machine Translation

Language Modeling

Generative Adversarial Networks

Consists of a *generator* network and a *discriminator* network:

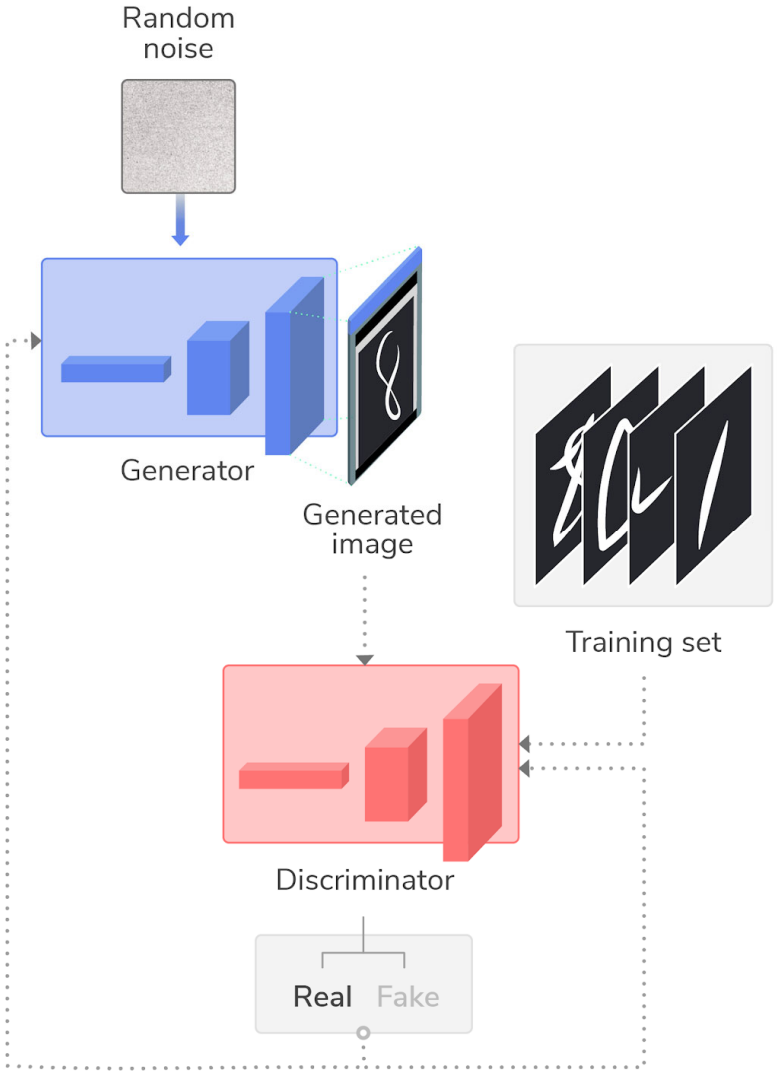
- generator learns how to make data comparable to *real world* training data
- discriminator learns how to distinguish ***real*** data from ***generated*** data
- adversarial aspect is a mechanism for *self improvement* by inducing competition

Typically used to generate images, video, audio, etc.

Ethical questions have become more prominent as quality has improved

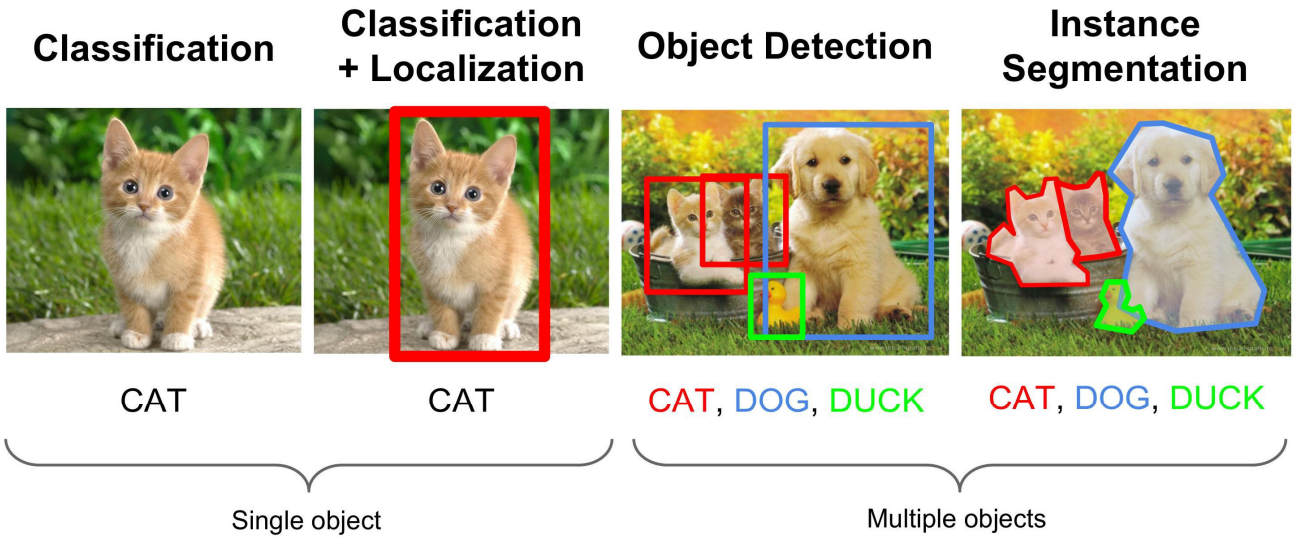
Generative Adversarial Networks

Each network will have it's own architecture and objective function



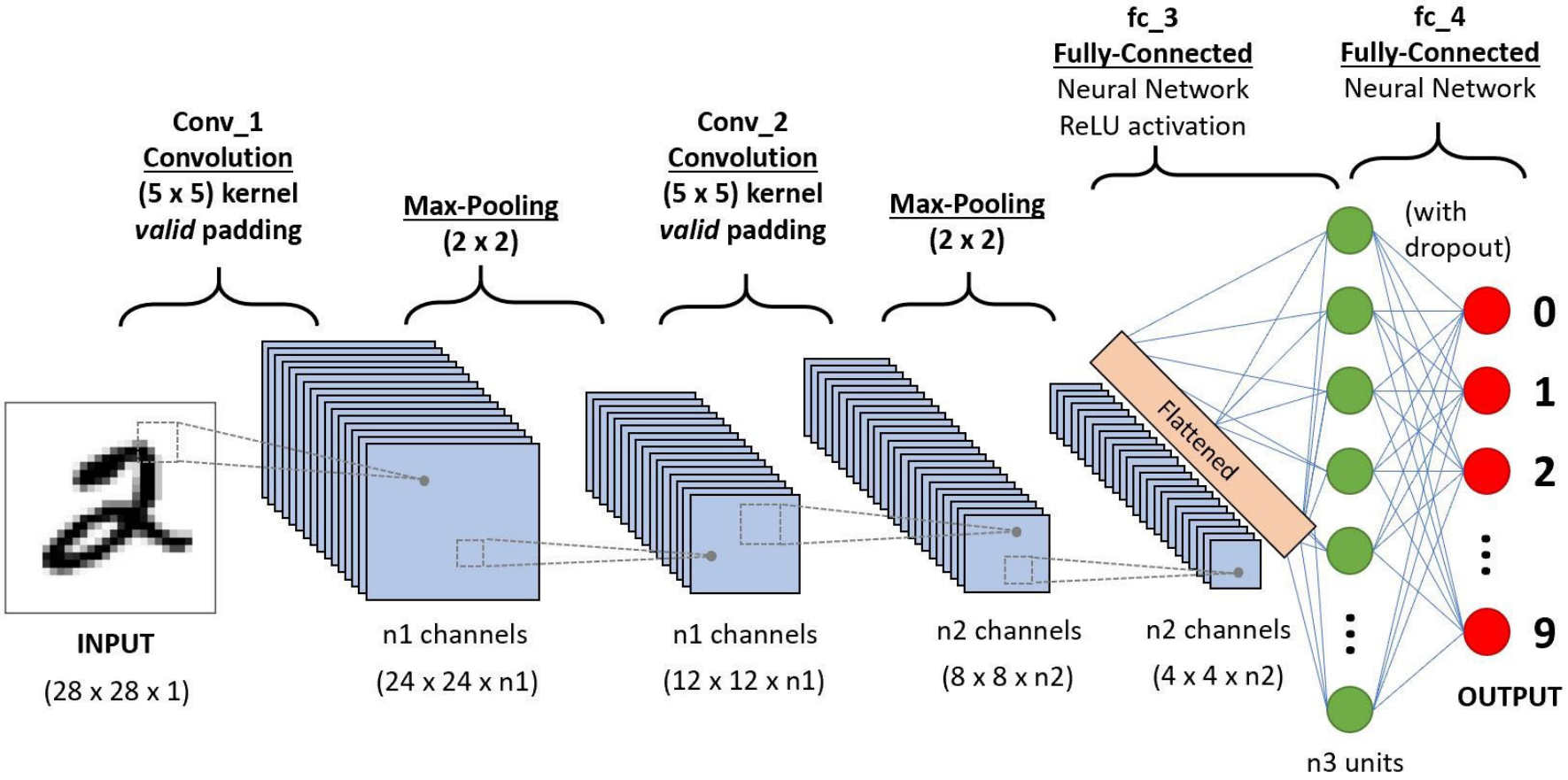
Convolutional Neural Networks

Convolution Neural Networks (CNNs) are used for image processing



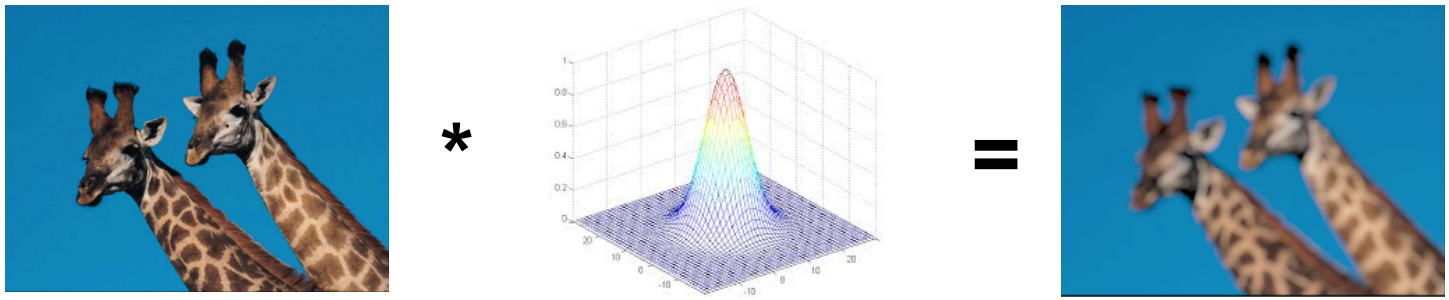
Convolutional Neural Networks

Large portion of the network is dedicated to feature extraction

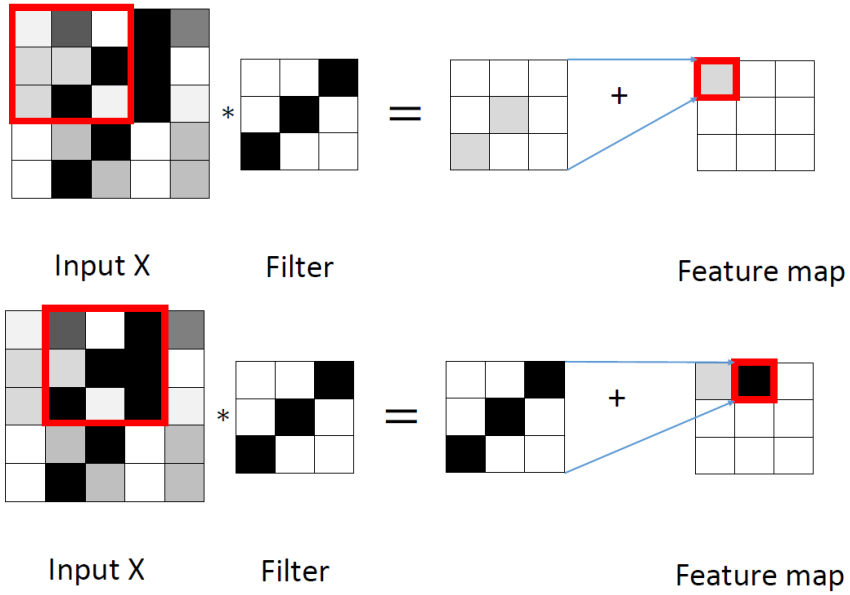


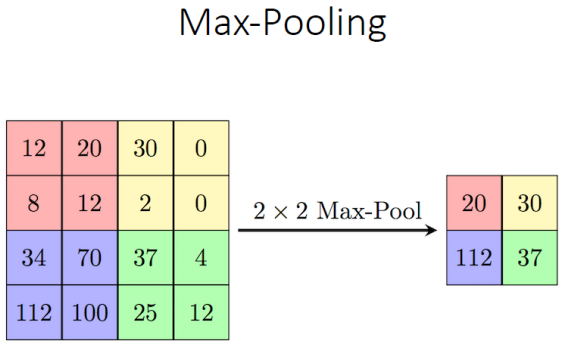
Convolutional Neural Networks

Blurry images are examples of convolutions:



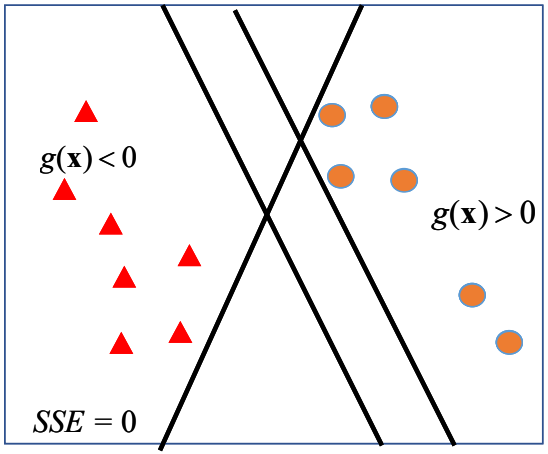
Serves as a form of dimensionality reduction:





Support Vector Machines (SVM)

Perceptrons do not differentiate between the quality of linear discriminants



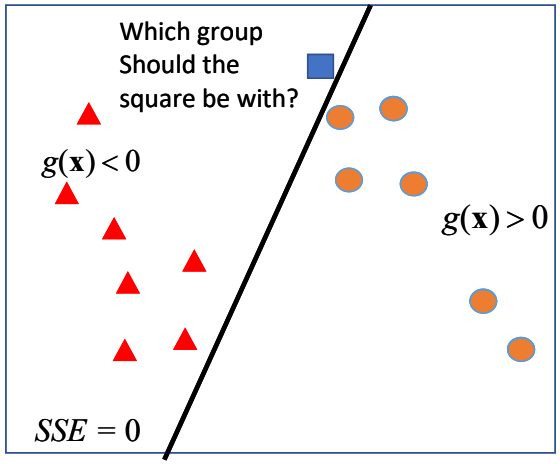
Are all 3 lines equally good?

$$g(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 = 0$$ Decision surface

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 0 \\ -1 & \text{otherwise} \end{cases}$$ Hypothesis function

$$SSE = \sum_i^n (y_i - h(\mathbf{x}_i))^2$$ 0-1 loss function

But what if there is noise?



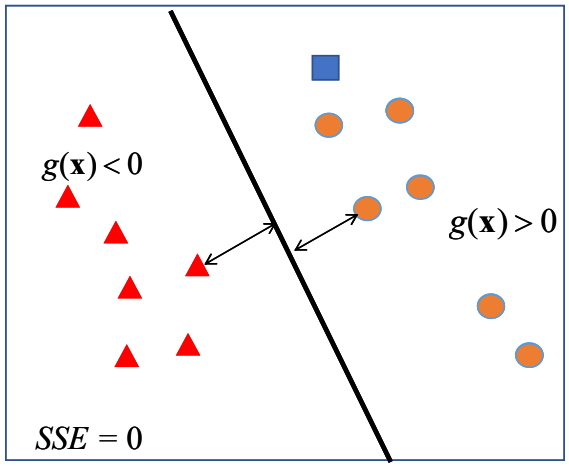
$$g(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 = 0$$ Decision surface

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 0 \\ -1 & \text{otherwise} \end{cases}$$ Hypothesis function

$$SSE = \sum_i^n (y_i - h(\mathbf{x}_i))^2$$ 0-1 loss function

Support Vector Machines (SVM)

A large-margin classifier tends to be more “robust”

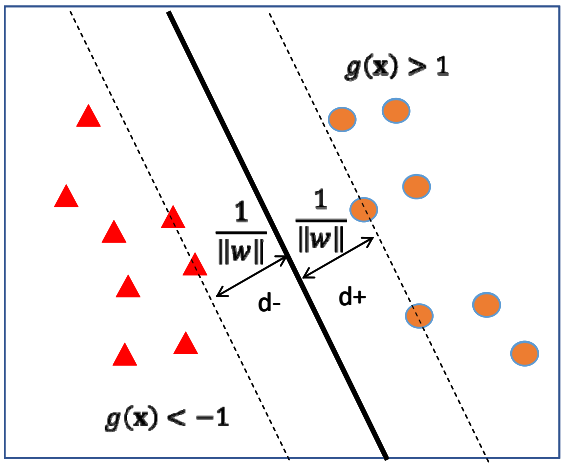


$$g(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 = 0$$ Decision surface

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 0 \\ -1 & \text{otherwise} \end{cases}$$ Hypothesis function

$$SSE = \sum_{i}^{n} (y_i - h(\mathbf{x}_i))^2$$ 0-1 loss function

Solve for the maximum margin classifier



- There is some scaling of the data where...

$$d^+ = d^- = \frac{1}{\| w \|}$$

- Now, the decision boundary function will output a value with magnitude 1 or greater..

$$g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

Learned weights — Learned offset from origin

- Maximize the margin $\frac{2}{\|\mathbf{w}\|}$
- ...such that, for every data point, the following equation holds.

$$y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1,$$

True label drawn from {+1,-1} — Learned weights of decision boundary — Learned offset from origin

Genetic Algorithms

Harnesses evolutionary processes to solve combinatorial[1] problems

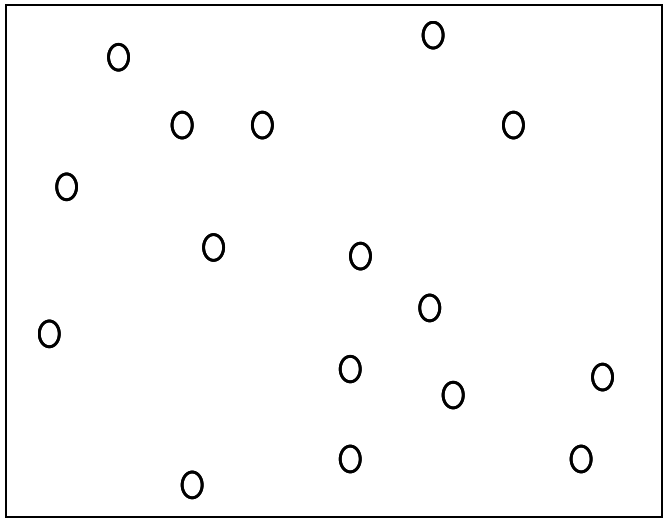
- Need to define a problem with a string (genotype) solution encoding
- Define a fitness function (to rank performance), how inheritance and mutations work and a lifespan for each solution
- String encoding describes parameters of a solution, without *being* the solution
- Genetic algorithms are (i) slow, (ii) fun (see boxcar2D.com), and challenging to code (finding the right representation)
- Dominant technique in symbolic regression and circuit board design

Combinatorial problems involve finding a grouping, ordering, or assignment of a discrete, finite set of objects that satisfies given conditions.

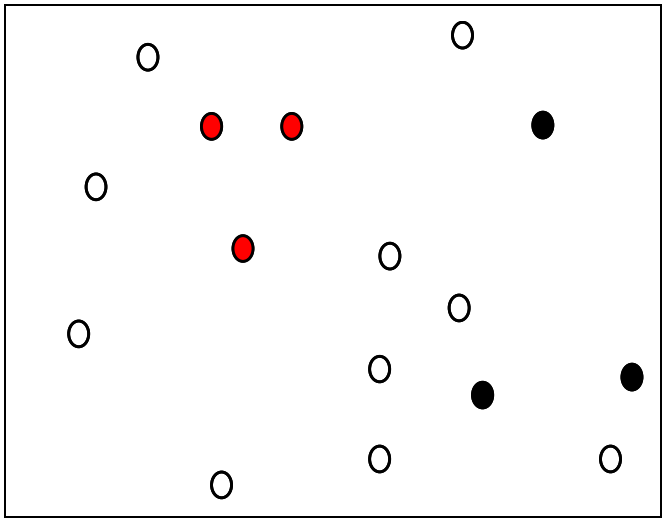
Active Learning

C*oncept learning* -- acquiring general concepts from specific training examples (or a subset of the data)

- Labeling an entire dataset is expensive
- If we just pick the RIGHT examples to label, we can learn the concept from only a few labeled examples (it's like 20 questions)

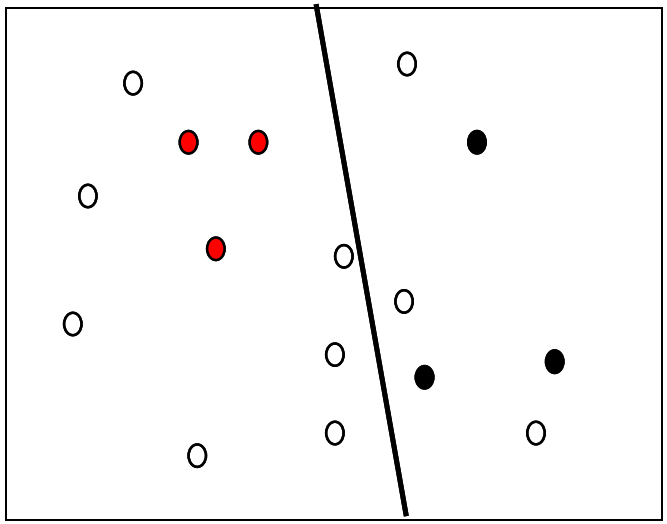


Start with a pool of unlabeled data

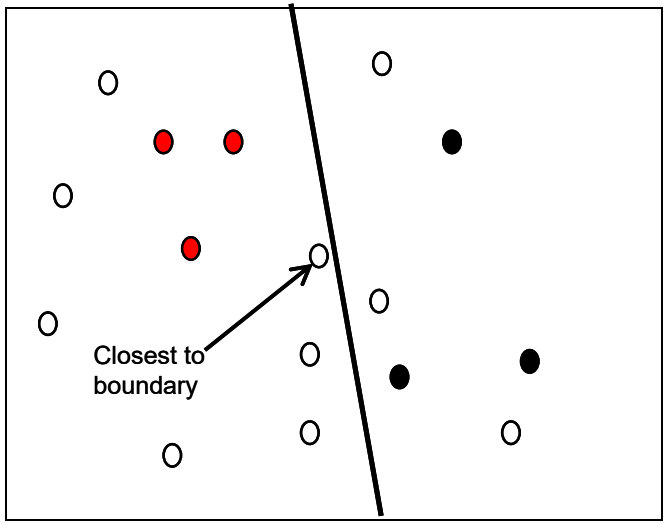


Label a random subset of the data

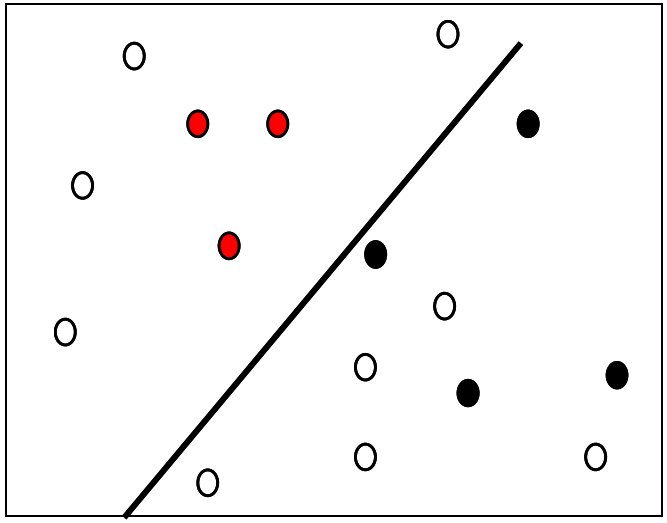
Active Learning



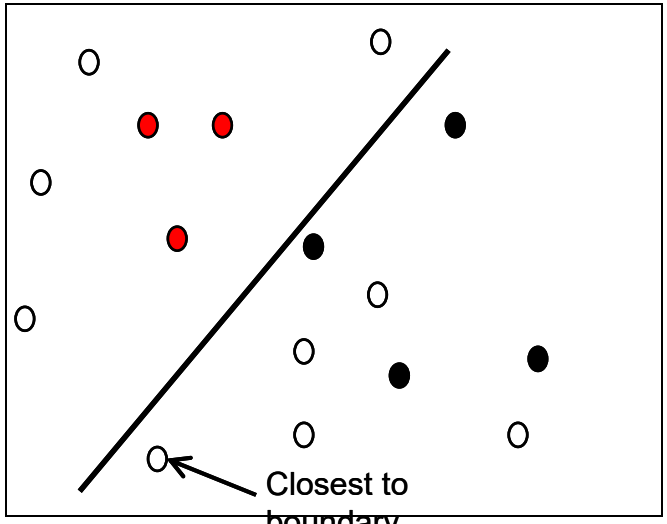
Fit a classifier to the data



Pick the “best” point to label next



Fit a classifier to the data



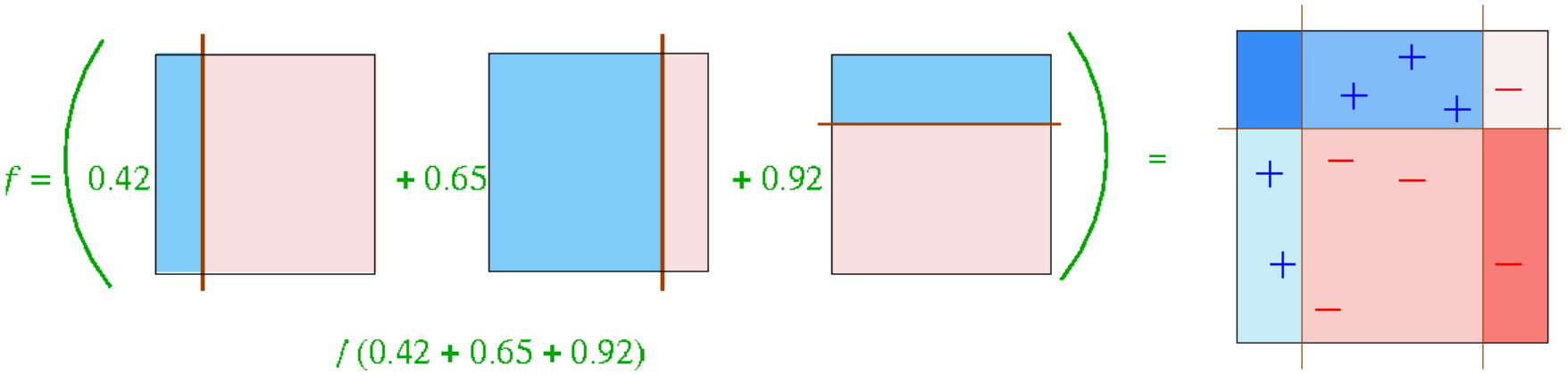
Pick the next “best” point to label next

Boosting

General idea is to (i) ask an expert for rule-of-thumb, (ii) assemble set of cases where rule-of-thumb fails (hard cases), (iii) ask another expert for a rule-of-thumb to deal with the hard cases, (iv) repeat second step

- Combine all rules-of-thumb
- Expert could be “weak” learning algorithm

Final hypothesis may be a “learned” combination of weak hypotheses



AdaBoost is a popular optimization algorithm to perform boosting