Perceptron

Different Definition

- **Fitting a function to data**

- Fitting: Optimization, what parameters can we change?
- Function: Model, loss function
- Data: Data/model assumptions? How we use data?

- ML Algorithms: minimize a function on some data

Setting

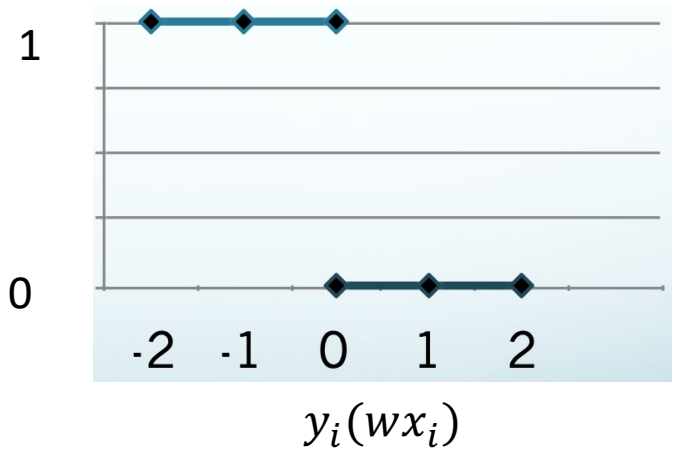
- Stochastic gradient descent
  - One example at a time
- Linear classifier

$$w \cdot x_i$$
$$y_i \in \{-1,1\}$$

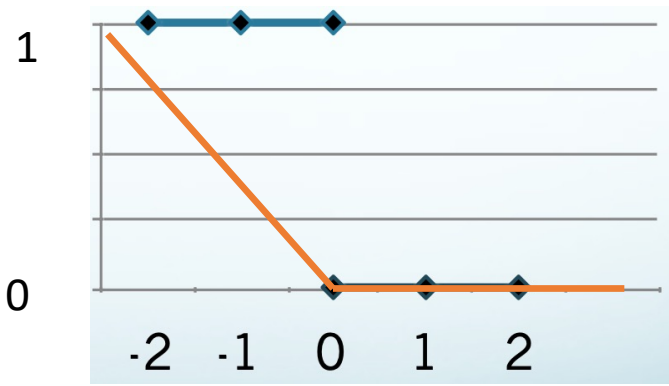
- Let's take the simplest (and most direct) loss function we can think of

0/1 Loss function

- If we are wrong: loss of 1
- If we are correct: loss of 0
  - Pretty much the simplest loss function around
- This mean:
  - Only make a change when we make a mistake



Hard to Minimize



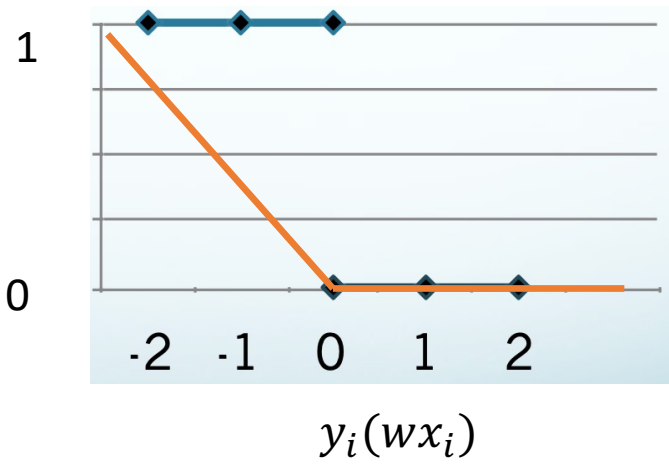
- Minimizing the 0/1 loss is difficult
  - Step function
- Replace with a similar form

$$L_w(y) = \sum_{i=1}^{N} \max(0, -y_i w \cdot x_i)$$

- Single example

$$L_w(y_i) = \max(0, -y_i w \cdot x_i)$$

Gradient

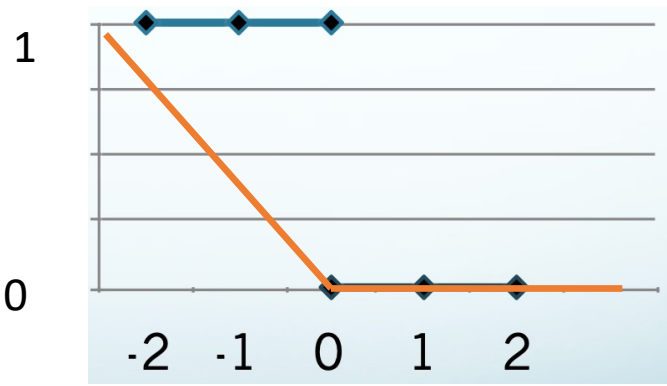


$$L_w(y_i) = \max(0, -y_i w \cdot x_i)$$

$$\nabla_w L_w(y_i) = \begin{cases} 0 & y_i w \cdot x_i > 0 \\ -y_i x_i & y_i w \cdot x_i < 0 \end{cases}$$

- Ignore case when $w \cdot x_i$=0

Update Rule



$$w^{t+1} = w^t + \eta \nabla_w L_w(y_i)$$

$$\nabla_w L_w(y_i) = \begin{cases} 0 & y_i w \cdot x_i > 0 \\ -y_i x_i & y_i w \cdot x_i < 0 \end{cases}$$

$$w^{t+1} = w^t + \eta(y_i - sign(w \cdot x_i))x_i$$

$$sign(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$

- Recall in logistic regression:
  - $w^{t+1} = w^t + \gamma \sum_{i=1}^{N} (y_i - p(y_i = 1 | x_i, w))x_i$

Update

- Why is this a good update? $w^{t+1} = w^t + \eta y_i x_i$

$$
\begin{aligned}
w^{t+1} x_i y_i &= w^t x_i y_i + \eta y_i x_i \, x_i y_i \\
(w^{t+1} x_i) y_i &= (w^t x_i) y_i + \eta y_i y_i x_i x_i \\
&= (w^t x_i) y_i + \eta |x_i|^2 \\
&> (w^t x_i) y_i
\end{aligned}
$$

- Our prediction has improved
  - This says nothing about seeing the example in the future
  - The prediction may still be incorrect
  - We are just moving in the right direction

Algorithm: Perceptron

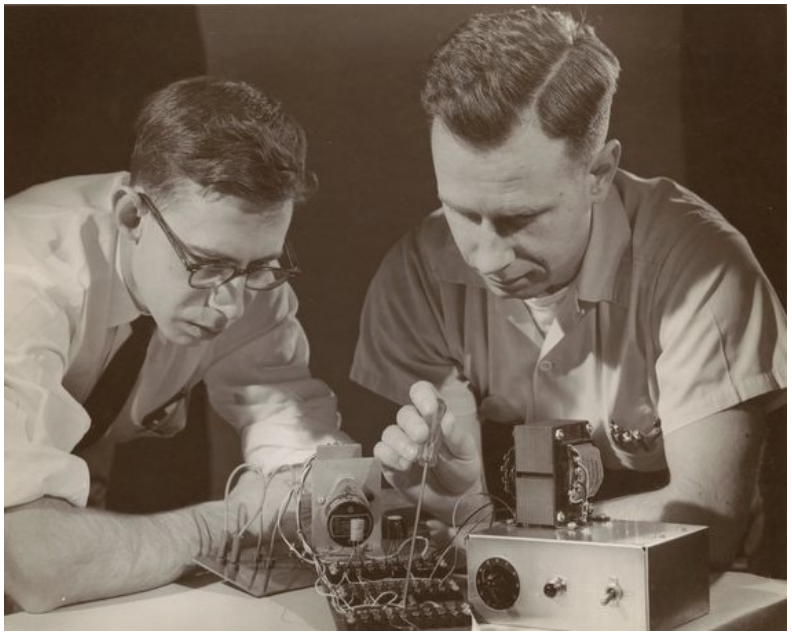
- Initialize w and η
- On each round
  - Receive example x
  - Predict $\hat{y} = sign(w \cdot x)$
  - Receive correct label $y \in \{-1, +1\}$
  - Suffer loss $\ell_{0/1}(\hat{y}, y)$
  - Update w: $w^{t+1} = w^t + \eta y_i x_i$

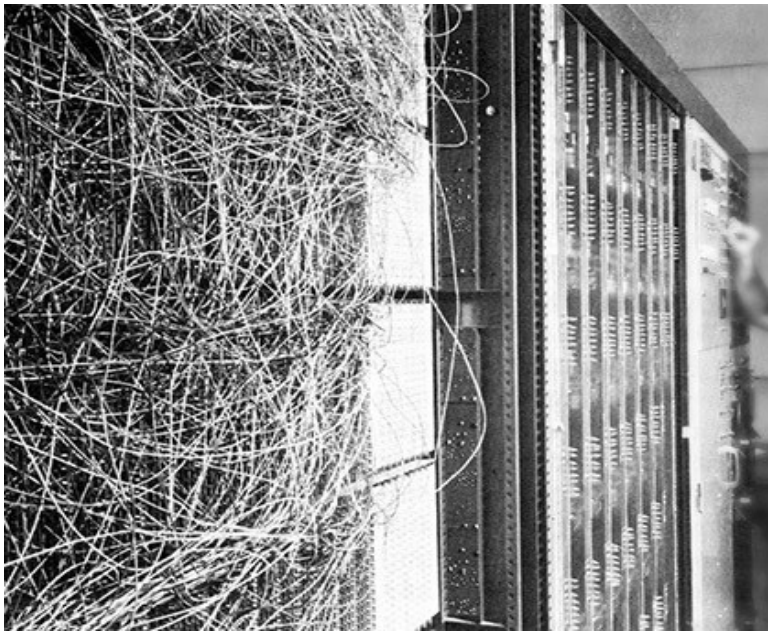
Different Definition

- **Fitting a function to data**

- Fitting: Stochastic gradient decent
- Function: 0/1 loss with linear function
- Data: Update using a single example at a time

Perceptron Mark 1 (1960)





Frank Rosenblatt, left, and Charles W. Wightman work on part of the unit that became the first perceptron in December 1958.

Why Perceptron?

- The first learning algorithm

- A way of thinking about data: Geometric interpretation of data
- A way of using data: online learning algorithms

A way of thinking about data

How We Represent Data

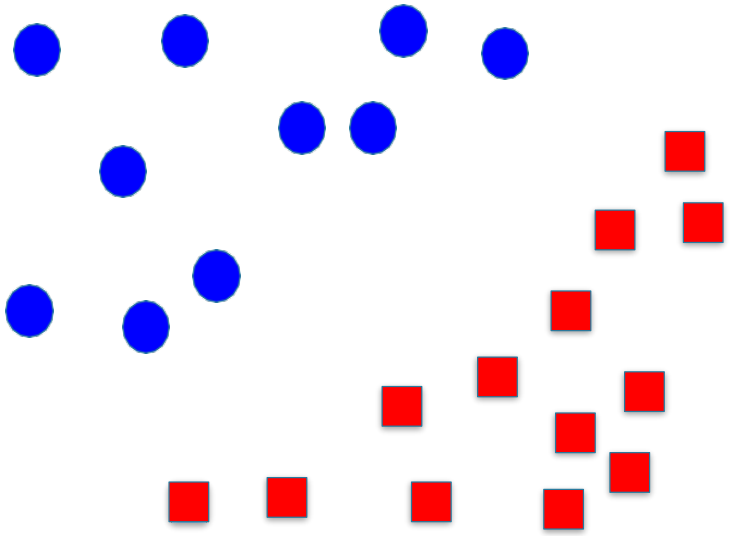
$$\{(x_i, y_i)\}_{i=1}^{N}, x_i \in \Re^M, y_i \in L$$

- A convenient way of representing data
- Also: a convenient way of thinking about data

Geometric Representations

- Each example $x_i \in \Re^M$ represents a point in an M dimensional space
- Classification: divides space into 2 parts
- Examples labeled according to where they are
- Linear classification: a linear decision boundary
  - A hyper-plane (flat high dimensional line)

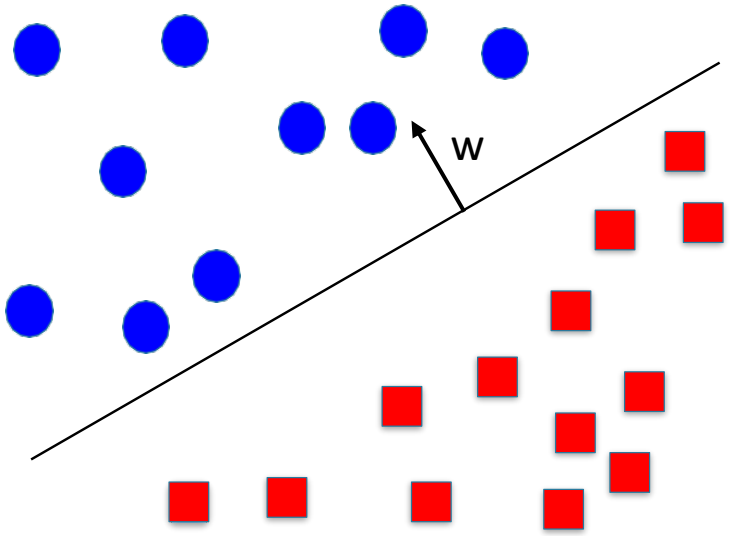
Geometric Representation



Discriminant Linear Classifiers

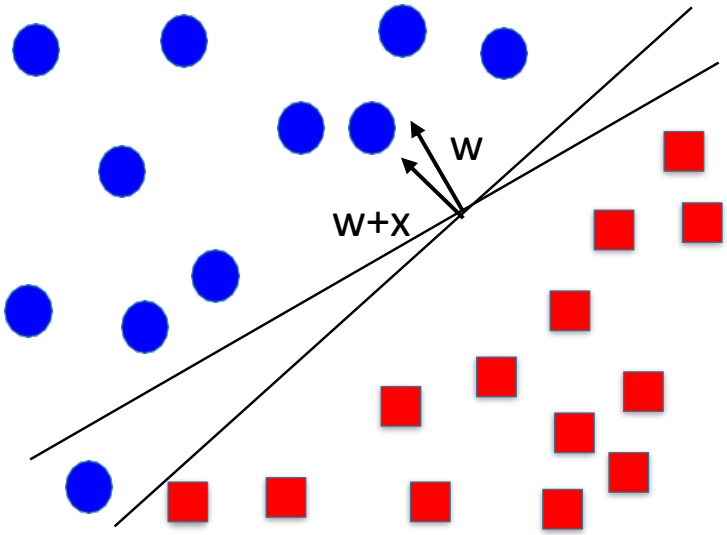
- Previously: we forced prediction by thresholding output
- Now: output either 0 or 1 directly
- Classification boundary represented by w
  - w is a vector that is orthogonal (normal) to the decision boundary $\hat{y} = sign(w \cdot x)$
- Prediction
  - The sign of the prediction indicates which side of the boundary
- Assume decision boundary passes through origin

Geometric Representation



Geometric Representation

$$w^{t+1} = w^t + \eta y_i x_i$$



Generalized Linear Function

- This is a linear function

$$w \cdot x_i$$

- Pass the output through a non-linear function

$$\hat{y} = sign(w \cdot x)$$

- Generalized linear function!

- The output is either +1 (positive) or -1 (negative)

Discriminant Linear Classifiers

- Magnitude of $w \cdot x_i$ does not matter
  - Relative magnitudes matter (we'll discuss soon)
- There are many representations of the same decision boundary
  - We can scale w without changing prediction

A way of using data

Batch Algorithms

- Train
  - Given training data {X,Y}
  - Learn the parameters of the model
- Test
  - Given an unseen unlabeled example x
  - Assign a label according to learned parameters
- Batch algorithms
  - Takes a batch of examples at once
- Idea: to improve speed, use a stochastic optimization algorithm

Assumptions Behind Batch

- The data is labeled with a consistent hypothesis
  - There is one optimal hypothesis that fits all of the data
    - This hypothesis is not necessarily in our hypothesis class
    - There may be some noise in the labels
- Examples are drawn from a common distribution IID (independent and identically distributed)
  - Identically- each draw is from the same distribution
  - Independent- each draw is independent
  - This allowed us to decompose the data likelihood in Logistic Regression

Removing Assumptions

- No train/test data
  - Instead access to a data stream
- Concept drift
  - No consistency in hypothesis used to label each example
- Examples not IID
  - Data distribution may change
  - Examples may be dependent
- Adversary model
  - An adversary controls the stream

Online Learning Algorithms

- Online learning handles all of these cases
- Make no assumptions about the data
  - Distribution
  - Hypothesis
- Online
  - They interact with the stream or a human
  - Predictions needed after every example
- Do the best you can on each single example

Why Online Learning?

- Few assumptions means widely applicable
- Strong theoretical foundation
- Scales to large amounts of data
  - Streaming metaphor processes examples one at a time
- Updates model without retraining
  - Spam filter: a single new spam email can update the model
- Handles a changing world
  - No assumptions about how data should behave

Online Learning Framework

- On each round
- Receive a single example x
- Predict $\hat{y}$ for x using stored hypothesis
- Receive correct label y
- Suffer loss $\ell(y, \hat{y})$
- Create new hypothesis using current hypothesis, x and y

How to Update?

- We know when to update
- How do we update?
- Change w so it is less wrong on the given example
  - Less wrong = has smaller loss
  - A gradient step in the right direction

- We need to answer:
  - What is our model (e.g. linear classifiers)
  - What is our loss function/objective function?

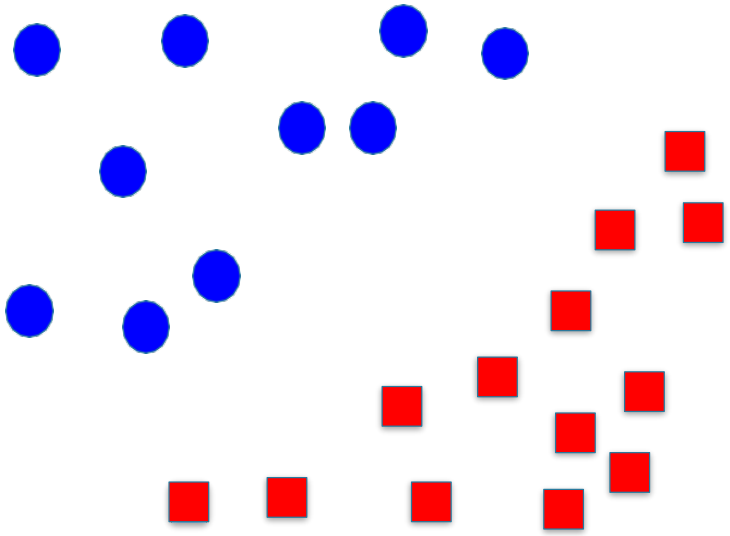
Expected Behavior

- We know that we improve with each update
- What can be said about the expected number of mistakes over a data stream?
  - Is it a lot?
  - A little?
  - Infinite?

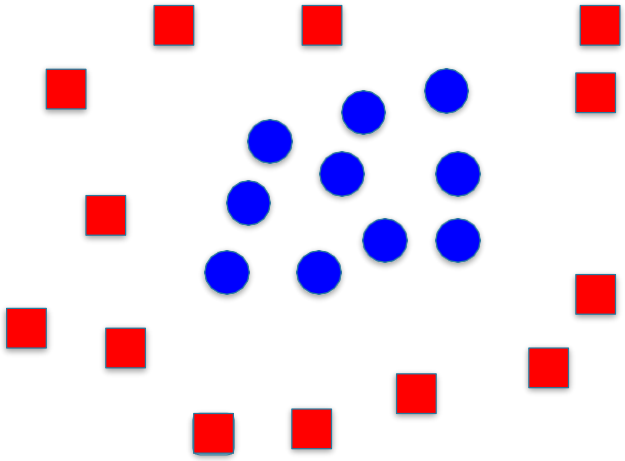
Linearly Separable

- Question: Is there a linear boundary that correctly separates all of the examples?
  - Yes: the examples are linearly separable
  - No: the examples are not linearly separable
- This is a separate issue from a consistent or optimal hypothesis
  - Could be not linearly separable and consistent hypothesis

Linearly Separable



Not Linearly Separable



Convergence

- Assume the data are linearly separable
- Will the Perceptron converge?
  - ie. Will it stop making updates?
- Yes! The Perceptron is guaranteed to converge on linearly separable data
- If it converges, then updates stop
  - If updates stop, then it makes a finite number of mistakes

Convergence

- Theorem: if the provided data are linearly separable with margin γ, then Perceptron will terminate in iterations linear with respect to the number of examples
  - Different theoretical frameworks for analyzing online algorithms
    - E.g. mistake bounds

- There are many versions of this theorem and proof
- This is a general version

Not Linearly Separable

- Data not linearly separable then will not converge
- Trivial to make data linearly separable
  - Add a unique feature to each example
  - Not very useful in practice
  - We'll learn better ways to do this

Oscillating Models

- For non-separable data, w will oscillate
- For separable data, it will converge
  - But which decision boundary will it converge to?
- Both cases mean w is highly dependent on the order of the data in the stream

Model Combinations

- Solution: take the best model over time
  - A model that was good on many examples should be trusted more than the latest model
- Voting
  - Save w on each round and predict by voting

$$sign[\sum_t sign(w_t x)]$$

- Averaging
  - Take the average of w at each round and average

$$sign[\sum_t (w_t) x]$$

Online Algorithms on the Rise

- Online algorithms are widely applied to large scale data problems
- In practice, many traditional algorithms now have stochastic optimizers
  - Faster for lots of data
  - Better for GPU hardware (mini-batch)

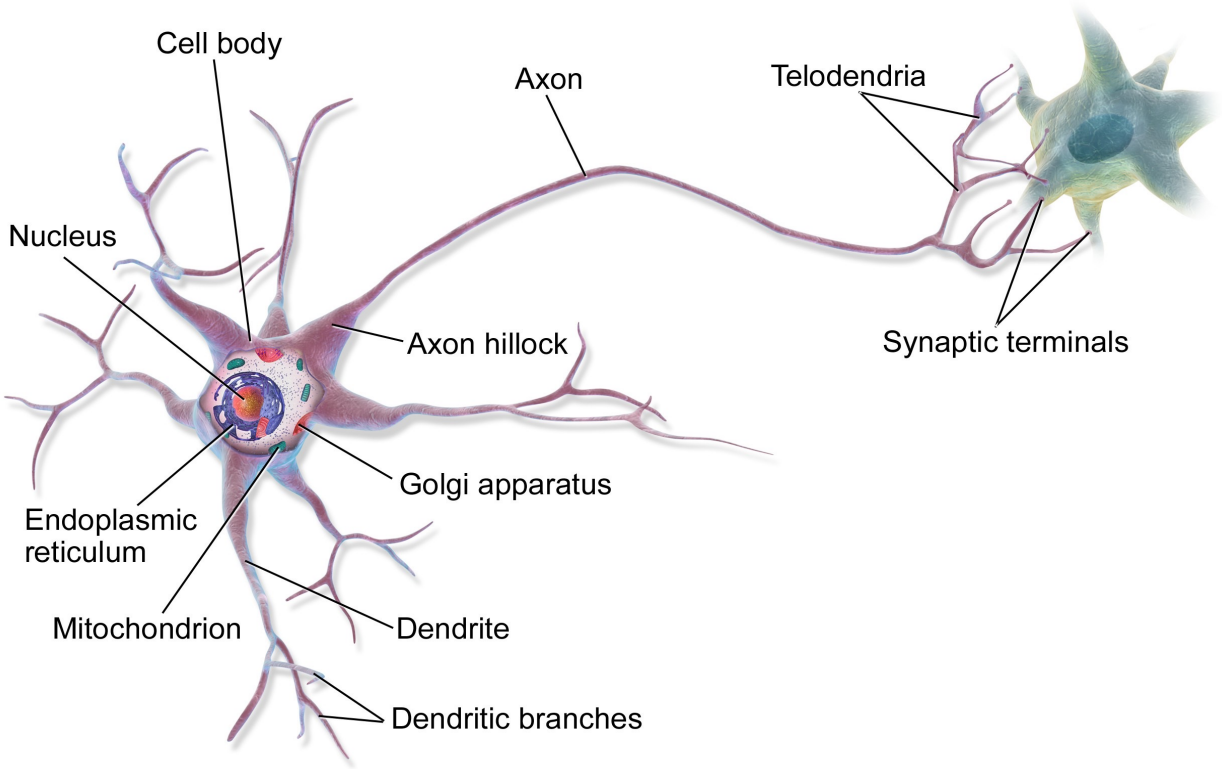
Lingering Questions

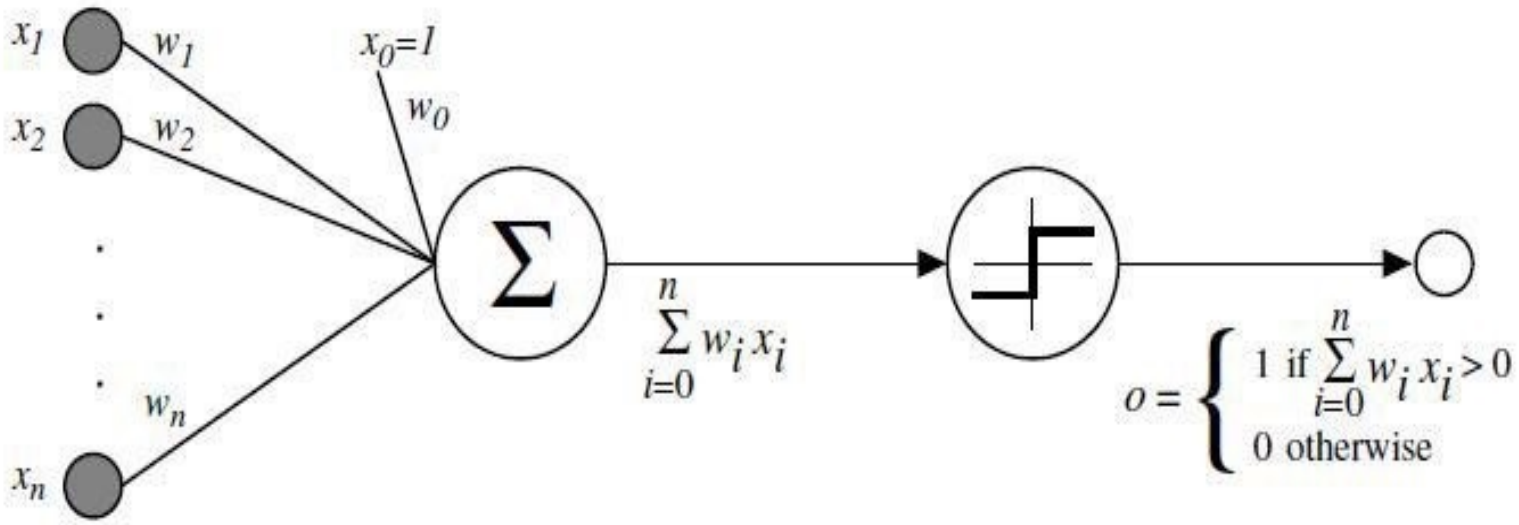
- We have discussed online training of discriminant functions for linear classifiers
  - What would we do if we saw all of the data (batch)?
- Perceptron converges to a separating hyperplane, but there are many
  - Which one is the best?
- Our solution for non-linear data was pretty silly
  - What can we do for non-linear data?

One More Thing: The Neural View

History

- Frank Rosenblatt: Psychologist at Cornell looking for a simple mathematical model of how neurons work in the brain to study how we learn
  - 1957: Starts work on Perceptron
  - 1960: Builds Perceptron Mark 1 machine
  - 1966: Minsky and Papert book show fundamental limitation of Perceptron
    - It's linear


Cell body
Axon
Telodendria
Nucleus
Axon hillock
Synaptic terminals
Golgi apparatus
Endoplasmic reticulum
Mitochondrion
Dendrite
Dendritic branches


x_1
x_2
x_n
w_1
w_2
w_n
$x_0=1$
w_0
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 \text{ otherwise} \end{cases}$

Why a Neural View?

- Coming up in a few weeks: Multi-layer Perceptrons
  - We can stack Perceptrons on top of each other
  - This creates a NON-linear function
  - Overcomes historical limitations
- Stacking of Perceptrons (neurons) creates a neural network