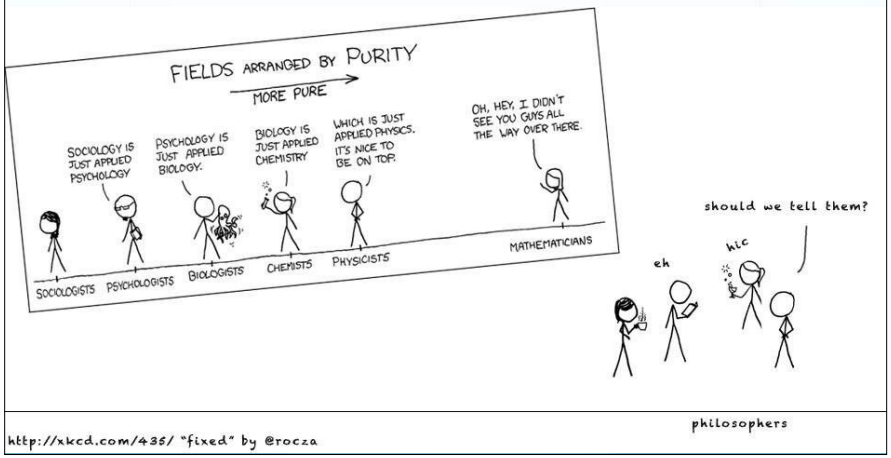
Perceptron

Mark Dredze

Machine Learning
CS 601.475



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

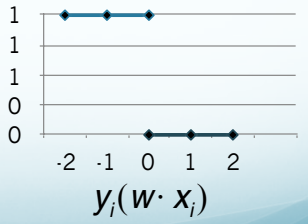
- Stochastic gradient descent
  - One example at a time
- Linear classifier

$$w \cdot x_i$$

- Output: binary classification
- Let's use the simplest loss function we can think of

0/1 Loss function

- If we are wrong: loss of 1
- If we are correct: loss of 0
- Simplest loss function we can imagine
- Implication:
  - Only make a change when we make a mistake



Hard to Minimize

- Minimizing the 0/1 loss is difficult
  - Step function without a gradient
- Replace with a similar form

$$L_w(y) = \sum_i^N max(0, -y_i w \cdot x_i)$$

- Single example

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

Gradient

- Calculate the gradient of our loss function

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

- What about when $w \cdot x_i = 0$?
  - We'll ignore this case, unlikely to occur

Update Rule

$$w^{i+1} = w^i + \eta \partial L_w(y_i)$$

$$w^{i+1} = w^i + \eta(y_i - \hat{y}_i)x_i$$

$$\hat{y}_i = \text{sign}(w \cdot x_i)$$

Update

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

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

- 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 $\hat{y} = \text{sign}(w \cdot x)$
  - Receive correct label $y \in \{+1, -1\}$
  - Suffer loss $\ell_{0/1}(y, \hat{y})$
  - Update w: $w^{j+1} = w^j + \eta\, y_i\, x_i$

Perceptron

Fitting a function to data

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

Perceptron Mark 1 (1960)

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

- A convenient way of representing data

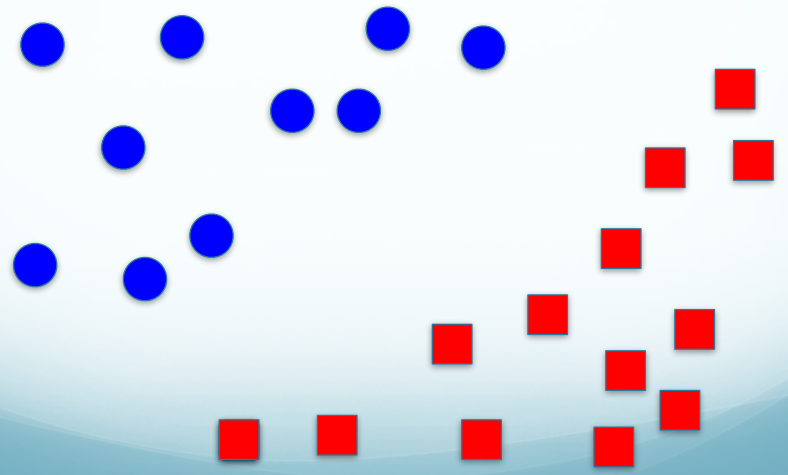
$$\{(x_i, y_i)\}_{i=1}^{N} \qquad x_i \in \Re^M \qquad y_i \in L$$

- 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 (M-1 dimensions)

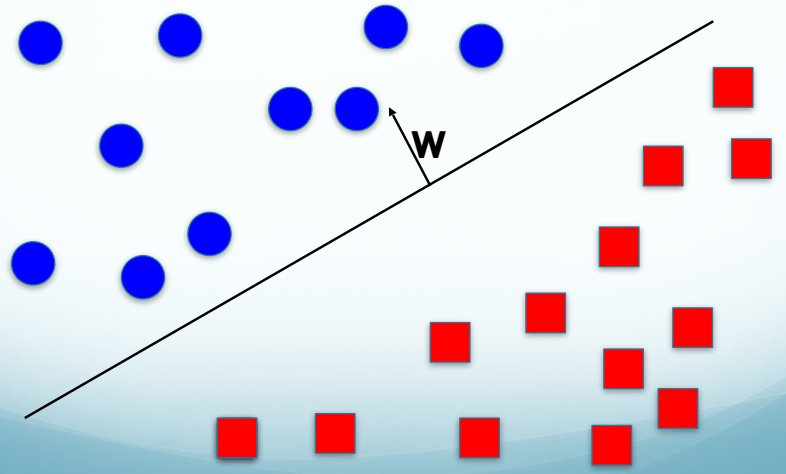
Geometric Representation



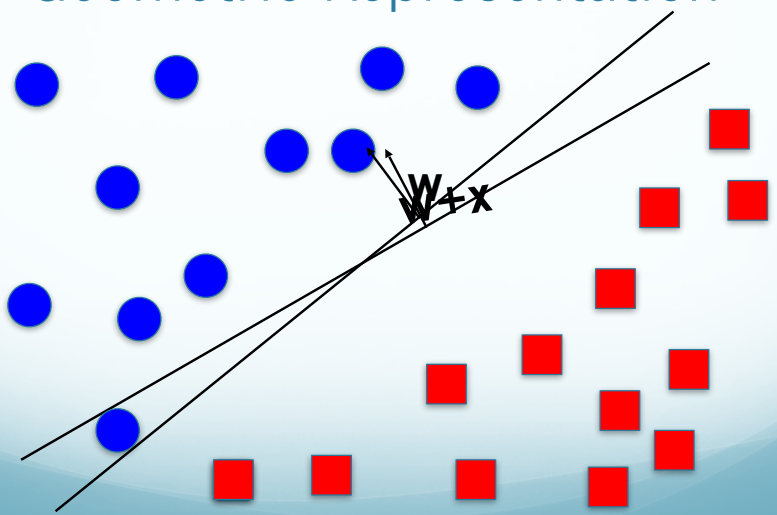
Discriminant Linear Classifiers

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

Geometric Representation



Geometric Representation



Generalized Linear Function

- This is a linear function
  - $w \cdot x$
- Pass the output through a non-linear function
  - $\hat{y} = \text{sign}(w \cdot x)$
- Generalized linear function!
- The output is either +1 (positive) or -1 (negative)

Discriminant Linear Classifiers

- Magnitude of $w \cdot x$ does not matter
  - Relative magnitudes matter (we'll discuss soon)
- Many representations of the same 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 a previously 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 of the data
  - Hypothesis class to describe it
- Online
  - Model learning interacts with a data stream or a human
  - Predictions needed after each 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 processes one example 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

1. Receive a single example x
2. Predict $\hat{y}$ for x using stored hypothesis
3. Receive correct label y
4. Suffer loss $\ell(y, \hat{y})$
5. Create new hypothesis using current hypothesis, x and y

How to Update?

- We know when to update
- Change w so it is *less wrong* on the given example
  - Less wrong = 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?

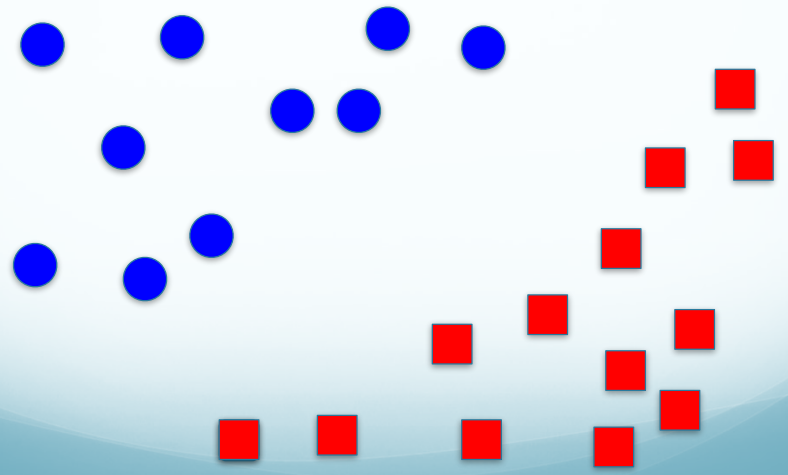
Expected Behavior

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

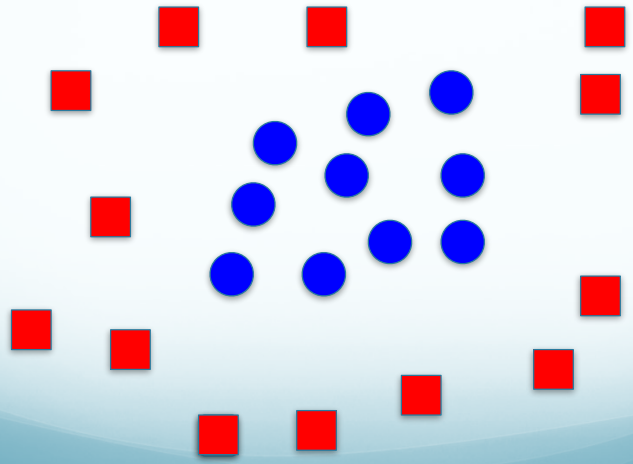
Linearly Separable

- Question: Is there a linear boundary (hyper-plane) 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 but consistent

Linearly Separable



Not Linearly Separable



Convergence

- Assume the data are linearly separable
- Will the Perceptron algorithm 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

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
  - $$\hat{y} = \text{sign}\left(\sum_i \text{sign}(\mathbf{w}_i \cdot \mathbf{x})\right)$$
- Averaging
  - Take the average of w at each round and average
  - $$\hat{y} = \text{sign}\left(\left(\sum_i \mathbf{w}_i\right) \cdot \mathbf{x}\right)$$

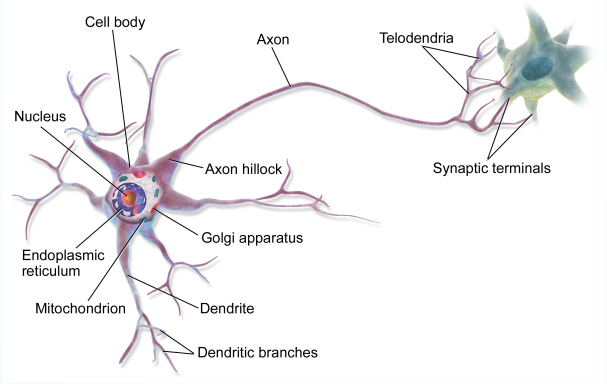
Online Algorithms on the Rise

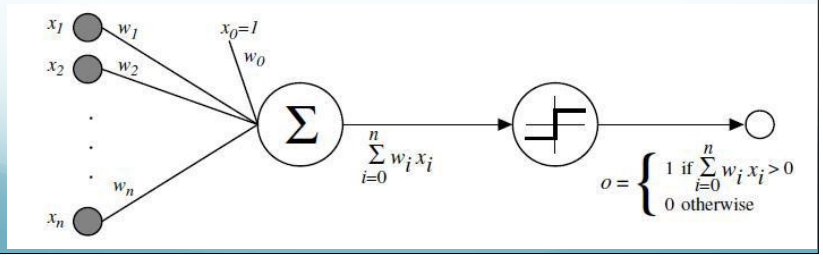
- Online algorithms are widely applied to large scale data problems
- In practice, many traditional algorithms now have stochastic optimizers
  - Faster for lots of data
  - Better for GPU hardware (mini-batch)

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
  - There are many
  - Which one is the best?
- Our solution for non-linear data was pretty silly
  - What can we do for non-linearly separable data?

One More Thing: The Neural View





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

Next Time

Support Vector Machines