ECE M146 Introduction to Machine Learning

Lecture 1 - Spring 2021

Prof. Lara Dolecek
ECE Department, UCLA

Course Logistics

- Weekly homework -- 45%
- Exam 1 – 15% in Week 4
- Exam 2 – 15% in Week 8
- Final exam (comprehensive) – 25% in Week 11.

- Homework is due on Gradescope before the lecture that day.
- Homework has a mix of math and coding.
- Exams format: have 2 hours in a 24-hr window. (3 hours for final).
  - Two exams instead of one to alleviate the pressure of a single day performance.

Today's Lecture

- Overview of the ML terminology

- Math review

Why Take Intro to Machine Learning Course ?

- Goals of this course:

1. Learn about most popular ML algorithms and frameworks
2. Develop mathematical understanding as well as intuition for these techniques
3. Test and implement these methods on various examples

What is Machine Learning ?

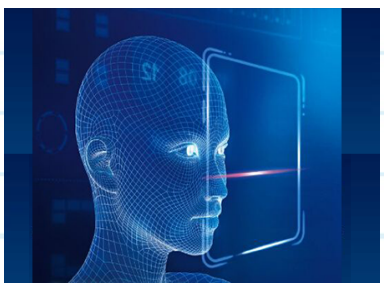
- Build a system (model/algorithm) based on training data to make inferences on testing data.

- “Machine” part is what specifies this system (model/algorithm). Optimize parameters of the system with respect to the training data.

Real Life Examples and Applications Abound!

- Medicine – from symptoms to diagnosis
- Transportation – automation of cars
- Privacy and authentication -- facial recognition
- Stock market -- roboinvestors
- Consumer behavior – e-commerce
- …and many more!






Human vs. Machine Learning

- Human learning – uses complex hypotheses set, context, generative modeling.

- Machine learning – pattern recognition equipped with a mathematical model for it. Can need a lot of training data for tasks that are “simple” for humans.

Conceptual Overview

General Procedure

- Training $\tilde{x} \xrightarrow{g} \tilde{y}$ g: unknown
- Build a Model f: serves as a proxy to g
- Testing $x \xrightarrow{f} y$ y: output

Classification of ML Algorithms

Broad classification of ML algorithms, based on what is available at the training time:

- Supervised Learning: at training time, we have access both to inputs and their labeled outputs.

  $(\tilde{x}, \tilde{y})$

- Unsupervised Learning: at training time, we have access only to input data, but not their labeled outputs.

  $(\tilde{x})$ but not $\tilde{y}$

Supervised Learning

1. Classification
   - Example: automated digit sorter for zip codes in hand written mail

     - $k = 2$: binary classification
     - $k = 10$: multiclass classification

2. Regression
   - Example: given the recent housing sales in zip code 90210, predict the sale value of a 4-bedroom house there.

Classification vs. Regression – Key Difference

- In classification, determine the value among a finite set of choices that appeared in the training set.
- In regression, predict/assign a (possibly new) real value to the test point, based on the input-output relationship in the training data.

Algorithms for Supervised Learning

- Perceptron
- Logistic Regression
- Decision Trees
- Linear-least squares
- K-Nearest Neighbors
- Support Vector Machines
- Naïve Bayes Classifier
- Gaussian Discriminant Analysis

- Linear vs. non-linear decision boundary
- Probabilistic vs. non-probabilistic setting
- Discriminative vs. generative
- On-line vs. batch updates

Unsupervised Learning

- At training time, we do not have access to labels, only the data.
- Main settings:

1. Clustering
   - K-means clustering
2. Projections/dimensionality reduction
   - Principal Component Analysis (PCA)
3. Density Estimation
   - MLE estimation

Reinforcement Learning

- Instead of the x → y relationship, we observe (x,z), where z is partial information about y.
- Instead of providing "good" training examples, as in supervised learning, algorithm needs to discover suitable actions by trial and error for maximum reward.
- There is a tradeoff between exploration (attempt to discover new actions) and exploitation (maximize rewards based on available actions).

Polynomial curve fitting

- Let's use it to solve a regression problem; will highlight important issues as well.
- Suppose we want to fit a polynomial to a sin(∏x) function (allow for random noise).
- Polynomial function:

- Degree M polynomial.

Polynomial curve fitting

- How to design this function ?
- It depends on what is available.

- **Idea #1: Given the training set, find the best fit.**

- Ok, let's see how. Specify the degree of the polynomial, M.

Polynomial curve fitting – idea #1

- Degree M = 0
  - Fit a horizonal line
- Degree M = 1
  - Fit a line
- Degree M = 2
  - Fit a quadratic

Polynomial curve fitting – idea #1

- Fitting a best curve means to find a polynomial of the given degree such that the error on the training set is minimized

$$\text{Error} = \sum_{i=1}^{N} |\tilde{y}_i - f(\tilde{x}_i, \vec{w})|^2$$

- Here, N denotes the number of training examples.

- Can we get the error to be zero ?

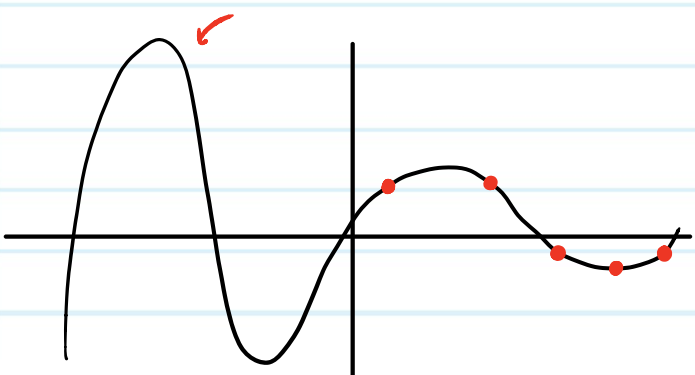
Yes: N = 2 data points → straight line (error = 0)

N data points → polynomial of degree N − 1 (error = 0)

Not so fast.

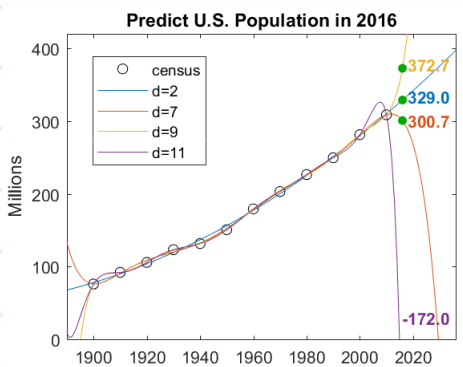
- Observe what we really want is a model that **generalizes well to unseen (testing) data**.

- Why is M = N-1 a bad idea ?



Not so fast.

- Observe what we really want is a model that **generalizes well to unseen (testing) data**.
- Why is M = N-1 a bad idea ?
- Totally unpredictable behavior outside the training set!
- This is known as **overfitting**.



Source: https://blogs.mathworks.com/cleve/2017/01/05/fitting-and-extrapolating-us-census-data/

Solution: Regularization

- This is **idea #2**
- Error formula:

$$\text{Error} = \sum_{i=1}^{n} |\tilde{y}_i - f(\tilde{x}_i, w)|^2 + \lambda \cdot \|\vec{w}\|^2$$

- Term with the parameter lambda Λ penalizes large magnitude values; value of lambda specifies by how much (non-negative parameter).

$\|\vec{w}\|^2$

$\lambda > 0$

$\vec{w} = (w_0, w_1, \ldots w_M)$

Today's Lecture

- Overview of ML terminology

- Math review
  - Probability
  - Linear Algebra
  - Optimization (later)

Probability

- Random Variables

a function that maps each outcome to a real value

$X : \Omega \rightarrow \mathbb{R}$

Probability

- Random Variables $X : \Omega \rightarrow \mathbb{R}$

  functions that map each outcome to a real value

- Conditional Probability and Bayes Rule

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} \qquad P(A) = \sum P(A|B_i)P(B_i)$$

- Examples of Important Random Variables: Bernoulli, Uniform, Gaussian

- Maximum Likelihood Estimation (MLE)

Example: Drawing balls from boxes

- Set up: box #1 box #2

- Pick a box at random. Draw a ball from this box.
- Suppose the drawn ball is red.

- What is the probability that the drawn ball is drawn from box #2?

Example: Drawing balls from boxes

- Let X be the random variable denoting the index of the box.
- Values of X ?

- Conditional probability:

Review: Bayes Rule and Total Probability Law

- P(A|B)

Review: Bernoulli Random Variable

- X is a Bernoulli RV if it takes on value “1” with probability p, and value “0” with probability (1-p).
- Boxes and balls example, revisited.
- Pick box #1 with probability q, and pick box #2 with probability (1-q). Draw a ball from this box.
- Suppose the drawn ball is red.

- What is the probability that the drawn ball is drawn from box #2?

Example, continued

- Conditional probability:

Review: Uniform Random Variable

- Discrete

- Continuous

Review: Gaussian Random Variable

- A random variable X is said to be Gaussian if its pdf has the following format:

- Interpretation for the mean and variance

Review: Mean and Variance of a RV

- Discrete RV

- Continuous RV

Multivariate Gaussian RV

- A vector RV X is said to be multivariate jointly Gaussian if its pdf has the following format:

$$f_X(x) = \frac{1}{(2\pi)^{D/2}\sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

$1\times D$ $D\times D$ $D\times 1$ — Σ is $D\times D$ matrix

- Here, D is the dimension of X.
- Interpretation for the mean and covariance matrix.

$$\mu = \begin{bmatrix} E[x_1] \\ E[x_2] \\ \vdots \\ E[x_D] \end{bmatrix}$$

$D\times 1$

$$\Sigma_{ij} = \text{cov}(x_i, x_j)$$

$$\text{cov}(x_i, x_j) = E[(x_i - \mu_i)(x_j - \mu_j)]$$

More on Variance and Covariance

$$\Sigma = \begin{bmatrix} \text{var}(X_1) & \text{cov}(X_1, X_2) & \cdots & \text{cov}(X_1, X_D) \\ \text{cov}(X_1, X_2) & \text{var}(X_2) & \cdots & \text{cov}(X_2, X_D) \\ \vdots & \vdots & & \vdots \\ \text{cov}(X_1, X_D) & \text{cov}(X_2, X_D) & \cdots & \text{var}(X_D) \end{bmatrix}$$

$$\text{cov}(X_i, X_j) = \text{cov}(X_j, X_i)$$

$$\Rightarrow \Sigma \text{ is symmetric}$$

$$\Sigma_{ii} = \text{var}(X_i)$$

$$= E[(X_i - \mu_i)(X_i - \mu_i)]$$

Linear Algebra and Gaussian RV

- Covariance matrix $\sum$. Σ is $D \times D$ symmetric
- The inverse of $\sum$ is $\sum^{-1}$ $\Sigma \cdot \Sigma^{-1} = I$
- Determinant of $\sum$ is det($\sum$).
- For jointly Gaussian, matrix $\sum$ is positive definite. Inverse exists and determinant is positive.

$\forall\ z \neq 0, \quad z^T \cdot \Sigma \cdot z > 0$

Special case: Covariance Matrix is Diagonal

$$f_x(x) = \frac{1}{(2\pi)^{p/2}(\sigma^2)^{p/2}}$$

$$\Sigma = \begin{bmatrix} \sigma^2 & & 0 \\ & \sigma^2 & \\ 0 & & \ddots \\ & & & \sigma^2 \end{bmatrix}$$

$$\Sigma^{-1} = \begin{bmatrix} \frac{1}{\sigma^2} & & 0 \\ & \frac{1}{\sigma^2} & \\ 0 & & \ddots \\ & & & \frac{1}{\sigma^2} \end{bmatrix}$$

+

all diagonal entries are the same

$$\exp\left(-\frac{1}{2} \cdot \frac{1}{\sigma^2} \sum_{i=1}^{p} (x_i - \mu_i)^2\right)$$

$$\det(\Sigma) = (\sigma^2)^p$$

$$\downarrow$$

$$f_x(x) = \prod_{i=1}^{p} f_x(x_i)$$

$$x_i \sim \mathcal{N}(\mu_i, \sigma^2)$$

Review: Linear Algebra

- Recall rules for taking the transposes

$$x \cdot A = y$$
$$A^T \cdot x^T = y^T$$

Review: Linear Algebra

- Vector projections.
- Consider vector x of dimension D.
- We write || x || for vector norm

- L1 norm $\sum_{i=1}^{D} |x_i|$

- L2 norm $\left(\sum_{i=1}^{D} x_i^2\right)^{1/2}$

Review: Linear Algebra

- Inner product of vectors x and y.

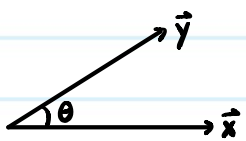
$$x^T \cdot y = \sum_{i=1}^{p} x_i y_i$$

- If $x^T y = 0$, x and y are perpendicular.

- When is the projection maximized? When y = a*x.

$a > 0$

$$x^T \cdot y = \|x\| \|y\| \cdot \cos\theta$$



Parameter Estimation

indep. & identically distributed

- Suppose we sample from an iid Gaussian with known variance and unknown mean.

$$f_x(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

- Goal is to provide the best estimate of the mean based on the sampled data.

Parameter Estimation – ctd.

N data points

$$f_x(x) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

one μ, one σ^2

↑ unknown

$$\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \prod_{i=1}^{N} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$$F = \log\left(\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \prod_{i=1}^{N} e^{\frac{(x_i-\mu)^2}{2\sigma^2}}\right)$$

$$= N \log\frac{1}{\sqrt{2\pi}\sigma} + (-1)\sum_{i=1}^{N}\frac{(x_i-\mu)^2}{2\sigma^2}$$

$$\frac{\partial F}{\partial \mu} = 0 \qquad \frac{\partial}{\partial \mu}\left(N \log\frac{1}{\sqrt{2\pi}\sigma}\right) = 0$$

$$-1 \cdot \frac{1}{2\sigma^2} \cdot 2\sum_{i=1}^{N}(x_i-\mu) = 0$$

$$\sum_{i=1}^{N} x_i - N\cdot\mu = 0$$

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$ sample mean

Lecture Summary

- Overview of ML terminology: supervised vs. unsupervised learning; regression and classification; means of method comparisons, etc.
- Polynomial curve fitting. Overfitting and regularization.
- Math review. Probability and linear algebra.

ECE M146 Introduction to Machine Learning

Lecture 2 - Spring 2021

Prof. Lara Dolecek

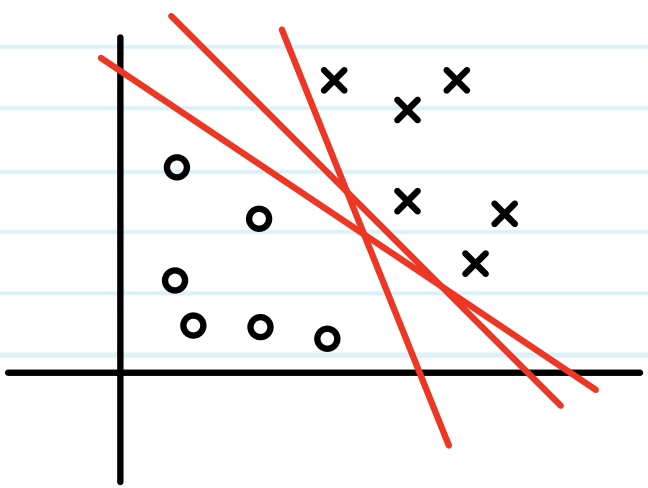
ECE Department, UCLA

Today

- High-level overview of machine learning methods
  - PERCEPTRON
- Math review
  - NORMS AND PROJECTIONS

Perceptron Algorithm

- Is an algorithm used to perform **binary classification**
- It produces values **on-line**
- Makes no assumptions on the statistics of the underlying training set except that the data has to be **linearly separable**



Linearly separating hyperplane

- Goal of the perceptron is to find a decision boundary as a **linearly separating hyperplane**
- This hyperplane is not necessarily the "best" one but it does separate the training data
- This decision boundary produced by perceptron algorithm is found in a finite number of steps
  - We will see a proof for this as well.

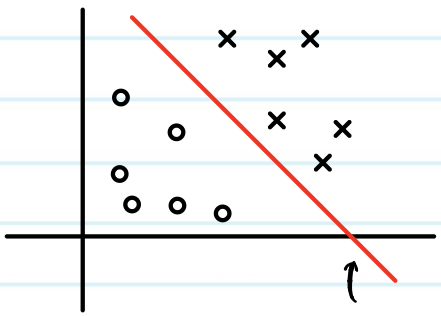
```
Suppose our data is of dimension D.
Linearly separating hyperplane is of
dimension D-1. (As a special example,
think of a 2D graph where a straight
line is the separator.
```

Key idea

- Perceptron processes labeled training data one at the time, until there are no more misclassified points left with respect to the current classification rule.

Illustrative example: two features

- Picture:



- Line: $D = 2$ D is the separating hyperplane

$$w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0$$

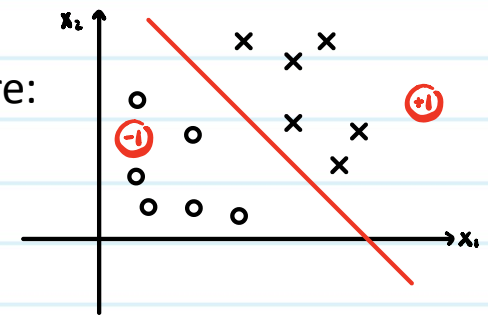
- Input data and their labels:

$$\tilde{x}_i = \begin{bmatrix} \tilde{x}_{i1} \\ \tilde{x}_{i2} \end{bmatrix} \qquad \tilde{y}_i \in \{+1, -1\} \qquad 1 \le i \le N$$

($\tilde{y}_i$: label; N: data pts)

Illustrative example: two features

- Picture:



- One side has: $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 > 0$
- The other side has: $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 < 0$
- Points on the decision boundary satisfy:

$$w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0$$

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Convenient transformation

- Instead of 2D, view the data points in 3D.
- Mapping: $\vec{x}_{old} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $\vec{x}_{new} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$
- Decision rule now becomes: $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$

  $\text{sign}(w_0 \cdot 1 + w_1 x_1 + w_2 x_2)$

  $\vec{w}^T \cdot \vec{x} > 0$: class +1

  $\vec{w}^T \cdot \vec{x} < 0$: class −1
- Convenience is that we no longer deal with the intercept term separately.
- Note that now in 3D, the decision boundary passes through the origin.

Perceptron Algorithm

Initialize

- Set k=1
- Set the vector $w^{(k)}$ to be the all zero vector.

- While there exists a misclassified point, i.e., there exists index j s.t.

$$y_j \ (w^{(k)T} * x_j) \ < 0$$

Update:

- $w^{(k+1)T} = w^{(k)T} + y_j x_j$ and increment iteration k by one.

Return $w^{(k)}$.

Perceptron Algorithm

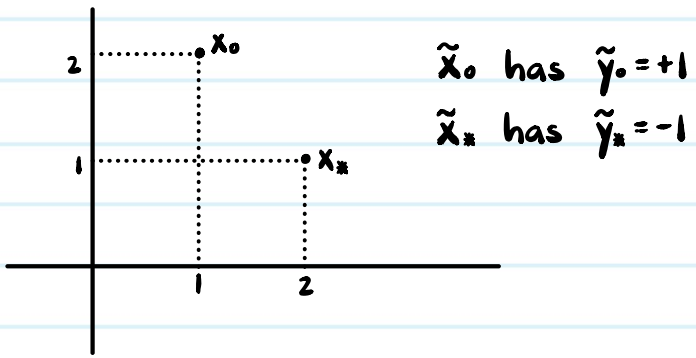
Interpretation:

- While there are misclassified data points, tilt $w^{(k)}$ towards the right classification rule.

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} + y_j \cdot \vec{x}_j$$

Illustrative example: two data points

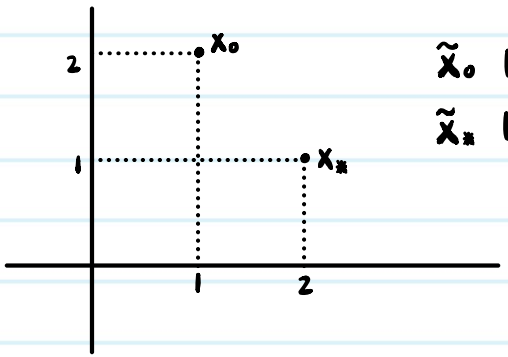
- Picture:



- Goal is to find vector w such that w produces correct labels for the training data, and as such can be used to predict labels on the testing data.

Illustrative example: two data points

- Picture:



$\tilde{x}_0$ has $\tilde{y}_0 = +1$

$\tilde{x}_*$ has $\tilde{y}_* = -1$

- First, perform transformation from 2D to 3D so that the intercept term is absorbed.

$\vec{x}_0 = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$ $\vec{x}_* = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ $\text{sign}(\vec{w}^T \cdot \vec{x})$ is the correct label

Illustrative example: two data points

- Initialize: $\vec{w} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ $k = 1$
- Evaluate at x_o

$$\text{sign}(\vec{w}^T \cdot \vec{x}_o) = \text{sign}(0) = +1$$

- Evaluate at x_*

$$\text{sign}(\vec{w}^T \cdot \vec{x}_*) = \text{sign}(0) = +1$$

- Point x_* is misclassified. Pick this point for the update.
  - because its label is -1

Illustrative example: two data points

- Perform the update:

$$\vec{w}^{(2)} = \vec{w}^{(1)} + y_* \vec{x}_*$$
$$= \vec{0} + (-1)\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \\ -1 \end{bmatrix}$$

- Point x_* is now correctly classified.

$$\text{sign}(\vec{w}^{(2)T} \cdot \vec{x}_*) = \text{sign}\left(\begin{bmatrix} -1 & -2 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}\right) = -1$$

- What about point x_o ?

Illustrative example: two data points

- Check point x_o :

$$\text{sign}(\vec{w}^{(2)T} \cdot \vec{x}_o) = -1$$

$$= \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$$

- But this point is now misclassified! So we need to do another update:

$$\vec{w}^{(3)} = \vec{w}^{(2)} + (+1) \cdot \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

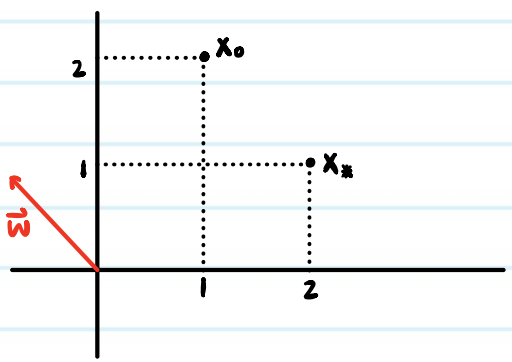
Illustrative example: two data points

- Evaluate at data point x_0:

$$\text{sign}\left(\vec{w}^{(3)T} \cdot \vec{x}_0\right) = +1$$

- Evaluate at data point x_*:

$$\text{sign}\left(\vec{w}^{(3)T} \cdot \vec{x}_*\right) = -1$$



Recap

- In words: What the update does is that it improves on a misclassified point, and may even make it correctly classified in one step – as in our example.
- In math:

Suppose we have a misclassified point (x_j, y_j)

$y_j = +1$

$\vec{w}^{(k)T} \cdot \vec{x}_j < 0 \qquad y_j(\vec{w}^{(k)T} \cdot \vec{x}_j) < 0$

$\vec{w}^{(k+1)} = \vec{w}^{(k)} + y_j \cdot \vec{x}_j$

$\vec{w}^{(k+1)T} \cdot \vec{x}_j = \vec{w}^{(k)T} \cdot \vec{x}_j + y_j \cdot \vec{x}_j^T \cdot \vec{x}_j$

less negative $\quad \ominus \quad \oplus \quad = \|\vec{x}_j\|^2$

What about previously correctly classified points ?

- It is indeed possible that previously correctly classified points become misclassified with an update in vector w.
- If so, can this process continue on forever without ever reaching the state of having all points correctly classified ?
- Fortunately, no.

Convergence proof of the perceptron algorithm

- Assumption 1: Suppose there exists some w_{opt} that separates the two classes. (It must exist by the linear separability condition).
- Formally: $\exists\ \vec{w}_{opt} \qquad \|\vec{w}_{opt}\| = 1$

  $y_i(\vec{w}_{opt}^T \cdot \vec{x}_i) > u \quad$ for some $u > 0$

  $\forall\ i,\ 1 \le i \le N$
- Assumption 2: Bounded coordinates.

  $\exists\ R$ s.t. $\|\vec{x}_i\| < R \quad \forall\ i,\ 1 \le i \le N$
- Recall that the algorithm does not know w_{opt}, u, or R!

Convergence proof of the perceptron algorithm

- Theorem: Perceptron makes at most R^2/u^2 updates until convergence i.e., there are no misclassified points.

- Proof:
- At k=1 $\vec{w}^{(1)} = \vec{0}$
- For any $k \geq 1$, suppose x_j is the misclassified point at that step.
- We build the vector w.

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} + y_j \vec{x}_j$$

Convergence proof of the perceptron algorithm

- Proof, continued.

$$\vec{w}^{(k+1)T} \cdot \vec{w}_{opt} = (\vec{w}^{(k)} + y_j \vec{x}_j)^T \cdot \vec{w}_{opt}$$
$$= \vec{w}^{(k)T} \vec{w}_{opt} + y_j \vec{x}_j^T \vec{w}_{opt}$$
$$> \vec{w}^{(k)T} \vec{w}_{opt} + u$$

$$\vec{w}^{(k)T} \cdot \vec{w}_{opt} > \vec{w}^{(k-1)T} \vec{w}_{opt} + u \quad \nearrow ku$$
$$\vdots$$
$$\vec{w}^{(3)T} \cdot \vec{w}_{opt} > \vec{w}^{(2)T} \vec{w}_{opt} + u \quad \nearrow 2u$$
$$\vec{w}^{(2)T} \cdot \vec{w}_{opt} > \underbrace{\vec{w}^{(1)T} \vec{w}_{opt}}_{0} + u = u$$

- Result 1:

$$\vec{w}^{(k+1)T} \cdot \vec{w}_{opt} > k \cdot u$$

Convergence proof of the perceptron algorithm

- Proof, continued.
- Result 1: $\vec{w}^{(k+1)T} \cdot \vec{w}_{opt} > k \cdot u$

- But, the projection could be getting bigger because $w^{(k+1)}$ is getting bigger, not closer.

Convergence proof of the perceptron algorithm

- Next, consider the inner product.

$$\vec{w}^{(k+1)T} \cdot \vec{w}_{opt} \leq \|\vec{w}^{(k+1)}\| \cdot \|\vec{w}_{opt}\|$$

(Recall: $\vec{x} \cdot \vec{y} \leq \|\vec{x}\|\|\vec{y}\| \cos\theta$)

$$\vec{x}^T \vec{y} \leq \|\vec{x}\|\|\vec{y}\|$$

- This gives the lower bound on the norm of the vector $w^{(k+1)}$

$$\|\vec{w}^{(k+1)}\| \geq \vec{w}^{(k+1)T} \cdot \vec{w}_{opt} > k \cdot u$$

Auxiliary result – squared norm of a sum

$$\|\vec{x}+\vec{y}\|^2 = \|\vec{x}\|^2 + \|\vec{y}\|^2 + 2\vec{x}^T\vec{y}$$

More generally:

$$\|\vec{x}_1+\vec{x}_2+\ldots+\vec{x}_M\| = \|\vec{x}_1\|^2 + \|\vec{x}_2\|^2 + \ldots + \|\vec{x}_M\|^2 + \sum_{i=1}^{M-1}\sum_{j=i+1}^{M} \vec{x}_i^T\vec{x}_j$$

Convergence proof of the perceptron algorithm

- Now, expand the quadratic norm.

$$\|\vec{w}^{(k+1)}\|^2 = \|\vec{w}^{(k)} + y_j \vec{x}_j\|^2$$
$$= \|\vec{w}^{(k)}\|^2 + \|y_j \vec{x}_j\|^2 + 2(\vec{w}^{(k)T} \cdot \vec{x}_j) \cdot y_j$$
$$= \|\vec{w}^{(k)}\|^2 + \|\vec{x}_j\|^2 + 2(\vec{w}^{(k)T} \cdot \vec{x}_j) \cdot y_j$$
$$< \|\vec{w}^{(k)}\|^2 + \|\vec{x}_j\|^2 \qquad 2(\vec{w}^{(k)T} \cdot \vec{x}_j) \cdot y_j < 0 \text{ (misclassified)}$$
$$\leq \|\vec{w}^{(k)}\|^2 + R^2 \quad \text{(bounded coords)}$$

Convergence proof of the perceptron algorithm

- Now, expand the quadratic norm.

$$\|\vec{w}^{(k+1)}\|^2 < \|\vec{w}^{(k)}\|^2 + R^2 \quad \uparrow kR^2$$
$$\|\vec{w}^{(k)}\|^2 < \|\vec{w}^{(k-1)}\|^2 + R^2$$
$$\vdots \qquad\qquad \vdots$$
$$\|\vec{w}^{(2)}\|^2 < \|\vec{w}^{(1)}\|^2 + R^2 \quad \uparrow 2R^2$$
$$\|\vec{w}^{(1)}\|^2 < \|\vec{w}^{(0)}\|^2 + R^2 = R^2$$

- Result 2:

$$\|\vec{w}^{(k+1)}\|^2 < kR^2$$

Combining the two results

$$\vec{w}^{(k+1)T} \cdot \vec{w}_{opt} > k \cdot u$$

$$\|\vec{w}^{(k+1)}\|^2 < kR^2$$

$$k^2 \cdot u^2 < (\vec{w}^{(k+1)T} \cdot \vec{w}_{opt})^2 \leq \|\vec{w}^{(k+1)}\|^2 < k \cdot R^2 \quad \Rightarrow$$

- Conclusion: $k < \frac{R^2}{u^2}$

- Established lower and upper bound on the squared norm of the weight vector and expressed it in terms of k.

upper bound on the # of updates

Further comments on the perceptron algorithm

- In practice, add/design features to make the data set linearly separable in a higher dimensional space.

- The discussion and examples thus far were for the binary classification.

- This can be extended to the multi-class classification problem as well.
- Procedure: L classes

  $L \geq 3 \quad 1 \leq \ell \leq L$

  Train on ℓ vs. not ℓ: get $\vec{w}_\ell$ from each such choice

  at test time $\max_\ell (w_\ell^T \cdot \vec{x})$

Further comments on the perceptron algorithm

- Observe that perceptron is an **on-line algorithm**: it processes one data point at the time and immediately updates the current value of the vector w.

- One consequence is that the **order** in which data is processed matters! We could get different final w's depending on which point was processed first, second, etc.

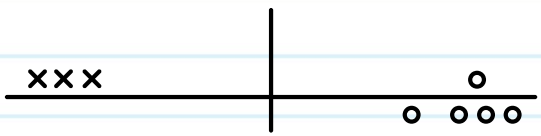
shuffle data so there is no implicit (forced) ordering

Further comments on the perceptron algorithm

- Additionally, perceptron weighs later examples more.
- A weight vector could be good for 1000's of points then trip up on one.
- In practice, can save older w's and weigh them by a committee vote i.e., how many steps each has survived. This is known as **voted** perceptron.
- Alternatively, can use the **averaged** perceptron (keep the running average of the weight vector but in a computationally efficient way)
- Better generalization.

Further comments on the perceptron algorithm

- **Perceptron does not optimize for the margin**
- Example:



- For this optimization, we will study the support vector machine (SVM)

Further comments on the perceptron algorithm

- **Perceptron cannot process a simple non-linear function**
- Example:

XOR function

0	1
1	0

- This requires a non-linear classifier. Perceptron generalizes to neural nets.

Summary

- Today's lecture: Perceptron for binary classification

- Next lecture: Linear least squares regression