Machine Learning (CSE 446): Learning as Minimizing Loss; Least Squares

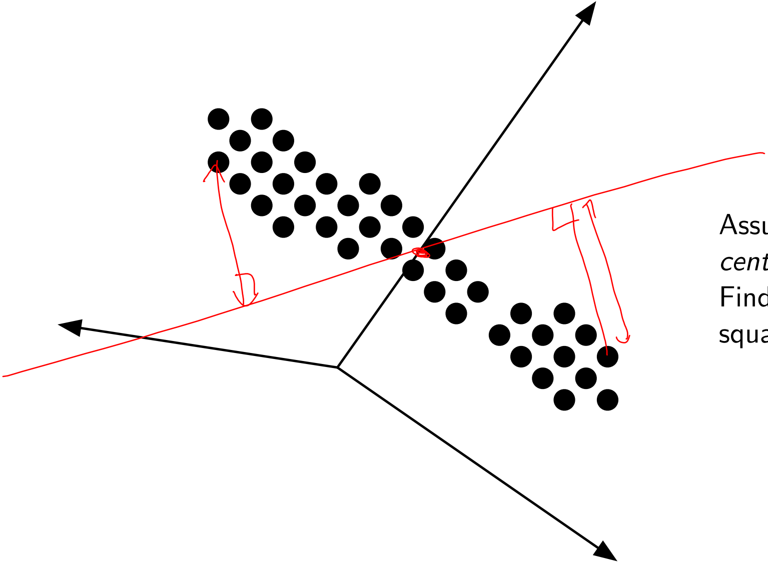
Sham M Kakade

University of Washington
cse446-staff@cs.washington.edu

HW2 due Thurs!

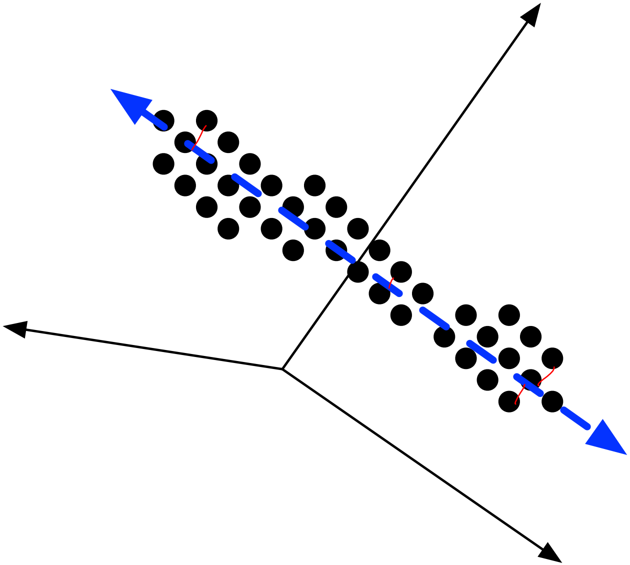
Review

Alternate View of PCA: Minimizing Reconstruction Error



Assume that the data are *centered*.
Find a line which minimizes the squared reconstruction error.

Alternate View of PCA: Minimizing Reconstruction Error



Assume that the data are *centered*.
Find a line which minimizes the squared reconstruction error.

Alternate View: Minimizing Reconstruction Error with K-dim subspace.

Equivalent ("dual") formulation of PCA: find an "orthonormal basis" $\mathbf{u_1}, \mathbf{u_2}, \ldots \mathbf{u_K}$ which minimizes the total reconstruction error on the data:

$$\underset{\text{orthonormal basis:}\mathbf{u_1},\mathbf{u_2},\ldots\mathbf{u_K}}{\operatorname{argmin}} \quad \frac{1}{N}\sum_i (\mathbf{x}_i - \text{Proj}_{\mathbf{u_1},\ldots\mathbf{u_K}}(\mathbf{x}_i))^2$$

Recall the projection of x onto K-orthonormal basis is:

$$\text{Proj}_{\mathbf{u_1},\ldots\mathbf{u_K}}(\mathbf{x}) = \sum_{j=1}^{K} (\mathbf{u_i} \cdot \mathbf{x})\mathbf{u_i}$$

The SVD "simultaneously" finds all $\mathbf{u_1}, \mathbf{u_2}, \ldots \mathbf{u_K}$

Projection and Reconstruction: the one dimensional case

- Take out mean μ:
- Find the "top" eigenvector u of the covariance matrix.
- What are your projections?
- What are your reconstructions, $\widehat{\mathbf{X}} = [\widehat{\mathbf{x}}_1 | \widehat{\mathbf{x}}_2 | \cdots | \widehat{\mathbf{x}}_N]^\top$?
- What is your reconstruction error of doing nothing $(K = 0)$ and using $K = 1$?

$$\frac{1}{N} \sum_i (\mathbf{x}_i - \mu)^2 = \qquad \frac{1}{N} \sum_i (\mathbf{x}_i - \widehat{\mathbf{x}}_i)^2 =$$

- **Reduction in error** by using a k-dim PCA projection:

PCA vs. Clustering

Summarize your data with **fewer points or fewer dimensions**?

Loss functions

Today

Perceptron

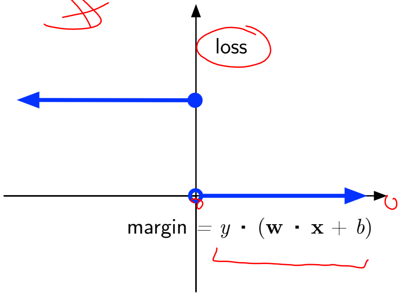
PERCEPTRON ALGORITHM: A model and an algorithm, rolled into one.

Isn't there a more principled methodology to derive algorithms?

What we ("naively") want:

"Minimize training-set error rate":

$$\min_{\mathbf{w},b} \frac{1}{N} \sum_{n=1}^{N} \underbrace{[\![y_n(\mathbf{w} \cdot \mathbf{x}_n + b) \leq 0]\!]}_{\text{zero-one loss on a point } n}$$



This problem is NP-hard; even for a (multiplicative) approximation.

Why is this loss function so unwieldy?

Relax!

- The mis-classification optimization problem:

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} [\![y_n(\mathbf{w} \cdot \mathbf{x}_n) \leq 0]\!]$$

- Instead, let's try to choose a "reasonable" loss function $\ell(y_n, \mathbf{w} \cdot \mathbf{x})$ and then try to solve the **relaxation:**

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \mathbf{w} \cdot \mathbf{x}_n)$$

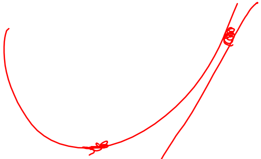
What is a good "relaxation"?

- Want that minimizing our **surrogate** loss helps with minimizing the mis-classification loss.
  - idea: try to use a (sharp) upper bound of the zero-one loss by ℓ:

$$[\![y(\mathbf{w} \cdot \mathbf{x}) \leq 0]\!] \leq \ell(y, \mathbf{w} \cdot \mathbf{x})$$

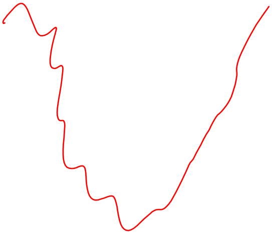
- want our **relaxed** optimization problem to be easy to solve.
  What properties might we want for $\ell(\cdot)$?

What is a good "relaxation"?



- Want that minimizing our **surrogate** loss helps with minimizing the mis-classification loss.
  - idea: try to use a (sharp) upper bound of the zero-one loss by ℓ:

$$[\![y(\mathbf{w}\cdot\mathbf{x}) \leq 0]\!] \leq \ell(y, \mathbf{w}\cdot\mathbf{x})$$



- want our **relaxed** optimization problem to be easy to solve. What properties might we want for $\ell(\cdot)$?
  - differentiable? sensitive to changes in w?
  - **convex?**

The square loss! (and linear regression)

- The square loss: $\ell(y, \mathbf{w} \cdot \mathbf{x}) = (y - \mathbf{w} \cdot \mathbf{x})^2$.
- The relaxed optimization problem:

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} (y_n - \mathbf{w} \cdot \mathbf{x}_n)^2$$

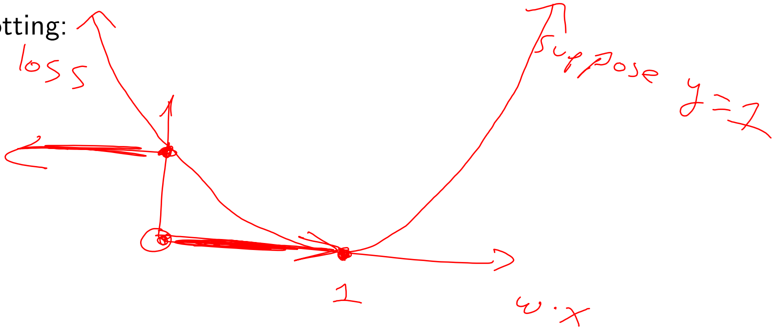
- nice properties:
  - for binary classification, it is a an upper bound on the zero-one loss.
  - It makes sense more generally, e.g. if we want to predict real valued y.
  - We have a convex optimization problem.
- For classification, what is your decision rule using a $\mathbf{w}$?

The square loss as an upper bound

- We have:

$$[\![y(\mathbf{w}\cdot\mathbf{x}) \le 0]\!] \le (y - \mathbf{w}\cdot\mathbf{x})^2$$

- Easy to see, by plotting:



Remember this problem?

Data derived from `https://archive.ics.uci.edu/ml/datasets/Auto+MPG`

mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
18.0	8	307.0	130.0	3504.	12.0	70	1
15.0	8	350.0	165.0	3693.	11.5	70	1
18.0	8	318.0	150.0	3436.	11.0	70	1
16.0	8	304.0	150.0	3433.	12.0	70	1
17.0	8	302.0	140.0	3449.	10.5	70	1
15.0	8	429.0	198.0	4341.	10.0	70	1
14.0	8	454.0	220.0	4354.	9.0	70	1
14.0	8	440.0	215.0	4312.	8.5	70	1
14.0	8	455.0	225.0	4425.	10.0	70	1
15.0	8	390.0	190.0	3850.	8.5	70	1
15.0	8	383.0	170.0	3563.	10.0	70	1
14.0	8	340.0	160.0	3609.	8.0	70	1
15.0	8	400.0	150.0	3761.	9.5	70	1
14.0	8	455.0	225.0	3086.	10.0	70	1
24.0	4	113.0	95.00	2372.	15.0	70	3
22.0	6	198.0	95.00	2833.	15.5	70	1
18.0	6	199.0	97.00	2774.	15.5	70	1
21.0	6	200.0	85.00	2587.	16.0	70	1
27.0	4	97.00	88.00	2130.	14.5	70	3
26.0	4	97.00	46.00	1835.	20.5	70	2
25.0	4	110.0	87.00	2672.	17.5	70	2
24.0	4	107.0	90.00	2430.	14.5	70	2

Input: a row in this table.

Goal: predict whether mpg is < 23 ("bad" $= 0$) or above ("good" $=$ 1) given the input row.

Remember this problem?

Data derived from `https://archive.ics.uci.edu/ml/datasets/Auto+MPG`

mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
18.0	8	307.0	130.0	3504.	12.0	70	1
15.0	8	350.0	165.0	3693.	11.5	70	1
18.0	8	318.0	150.0	3436.	11.0	70	1
16.0	8	304.0	150.0	3433.	12.0	70	1
17.0	8	302.0	140.0	3449.	10.5	70	1
15.0	8	429.0	198.0	4341.	10.0	70	1
14.0	8	454.0	220.0	4354.	9.0	70	1
14.0	8	440.0	215.0	4312.	8.5	70	1
14.0	8	455.0	225.0	4425.	10.0	70	1
15.0	8	390.0	190.0	3850.	8.5	70	1
15.0	8	383.0	170.0	3563.	10.0	70	1
14.0	8	340.0	160.0	3609.	8.0	70	1
15.0	8	400.0	150.0	3761.	9.5	70	1
14.0	8	455.0	225.0	3086.	10.0	70	1
24.0	4	113.0	95.00	2372.	15.0	70	3
22.0	6	198.0	95.00	2833.	15.5	70	1
18.0	6	199.0	97.00	2774.	15.5	70	1
21.0	6	200.0	85.00	2587.	16.0	70	1
27.0	4	97.00	88.00	2130.	14.5	70	3
26.0	4	97.00	46.00	1835.	20.5	70	2
25.0	4	110.0	87.00	2672.	17.5	70	2
24.0	4	107.0	90.00	2430.	14.5	70	2

Input: a row in this table.

Goal: predict whether mpg is < 23 ("bad" $= 0$) or above ("good" $=$ 1) given the input row.

Predicting a real y (often) makes more sense.

A better (convex) upper bound

- The logistic loss:

$$\ell^{\text{logistic}}(y, \mathbf{w} \cdot \mathbf{x}) = \log\left(1 + \exp(-y\mathbf{w} \cdot \mathbf{x})\right).$$

- We have:

$$[\![y(\mathbf{w} \cdot \mathbf{x}) \leq 0]\!] \leq \text{constant} * \ell^{\text{logistic}}(y, \mathbf{w} \cdot \mathbf{x})$$

- Again, easy to see, by plotting:

Least squares: let's minimize it!

Typo: this should be $X\mathbf{w}$ (X: $n \times d$, $\mathbf{w}$: d-vector)

- The optimization problem:

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} (y_n - \mathbf{w} \cdot \mathbf{x}_n)^2 =$$

$$\min_{\mathbf{w}} \|Y - X^\top \mathbf{w}\|^2 \cdot \frac{1}{2}$$

where Y is an n-vector and X is our $n \times d$ data matrix.

- How do we interpret $X^\top \mathbf{w}$?

$$[X\mathbf{w}]_n = \mathbf{w} \cdot \mathbf{x}_n$$

Least squares: let's minimize it!

- The optimization problem:

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} (y_n - \mathbf{w} \cdot \mathbf{x}_n)^2 =$$

$$\min_{\mathbf{w}} \|Y - X^\top \mathbf{w}\|^2$$

  where Y is an n-vector and X is our $n \times d$ data matrix.

- How do we interpret $X^\top \mathbf{w}$?

The solution is the **least squares estimator**:

$$\mathbf{w}^{\text{least squares}} = (X^\top X)^{-1} X^\top Y$$

Matrix calculus proof: scratch space

$$\|Y - Xw\|^2 = \|Y\|^2 - 2\, Y \cdot Xw + \|Xw\|^2 \qquad (Y \cdot Xw = Y^T X w)$$

$$\frac{\partial}{\partial w} \| \quad \|^2 = 0 - 2X^T Y + 2(X^T X) w$$

$$\Downarrow$$

$$(X^T X) \vec{w} = X^T Y$$

Matrix calculus proof: scratch space

Remember your linear system solving!

Lots of questions:

- What could go wrong with least squares?
  - Suppose we are in "high dimensions": more dimensions than data points.
  - Inductive bias: we need a way to control the complexity of the model.
- How do we minimize (sum) logistic loss?
- Optimization: how do we do this all quickly?