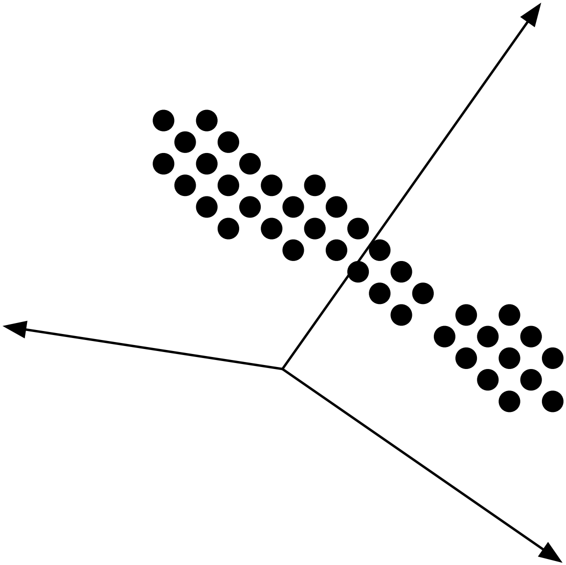
Machine Learning (CSE 446): PCA (continued) ~~and Learning as Minimizing Loss~~

Sham M Kakade

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

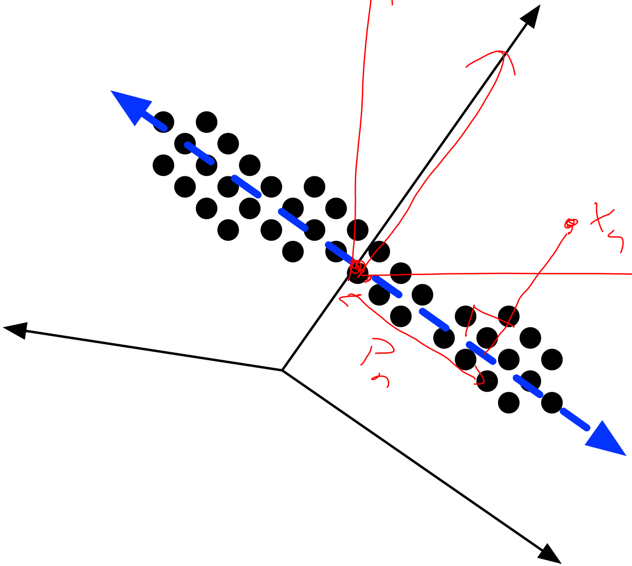
PCA: continuing on...

Dimension of Greatest Variance



Assume that the data are *centered*,
i.e., that
mean $\left(\langle \mathbf{x}_n \rangle_{n=1}^{N}\right) = \mathbf{0}$.

Dimension of Greatest Variance



Assume that the data are *centered*,
i.e., that
mean $\left(\langle \mathbf{x}_n \rangle_{n=1}^{N}\right) = \mathbf{0}$.

Projection into One Dimension

Let $\mathbf{u}$ be the dimension of greatest variance, where $\|\mathbf{u}\|^2 = 1$.

$p_n = \mathbf{x}_n \cdot \mathbf{u}$ is the projection of the nth example onto $\mathbf{u}$.

Since the mean of the data is $\mathbf{0}$, the mean of $\langle p_1, \ldots, p_N \rangle$ is also 0.

This implies that the variance of $\langle p_1, \ldots, p_N \rangle$ is $\frac{1}{N} \sum_{n=1}^{N} p_n^2$.

The $\mathbf{u}$ that gives the greatest variance, then, is:

$$\underset{\mathbf{u}}{\operatorname{argmax}} \sum_{n=1}^{N} (\mathbf{x}_n \cdot \mathbf{u})^2$$

Finding the Maximum-Variance Direction

$$\operatorname*{argmax}_{\mathbf{u}} \sum_{n=1}^{N} (\mathbf{x}_n \cdot \mathbf{u})^2$$

$$\text{s.t. } \|\mathbf{u}\|^2 = 1$$

(Why do we constrain $\mathbf{u}$ to have length 1?)

$n \times d$

$$\begin{bmatrix} \text{———} \; x_1^\top \; \text{———} \\ \vdots \\ \text{———} \; x_n^\top \; \text{———} \end{bmatrix}$$

If we let $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix}$, then we want: $\operatorname*{argmax}_{\mathbf{u}} \|\mathbf{X}\mathbf{u}\|^2$, s.t. $\|\mathbf{u}\|^2 = 1$.

This is PCA in one dimension!

Linear algebra review: things to understand

$\|x\|_2 = \sqrt{\sum x_i^2}$

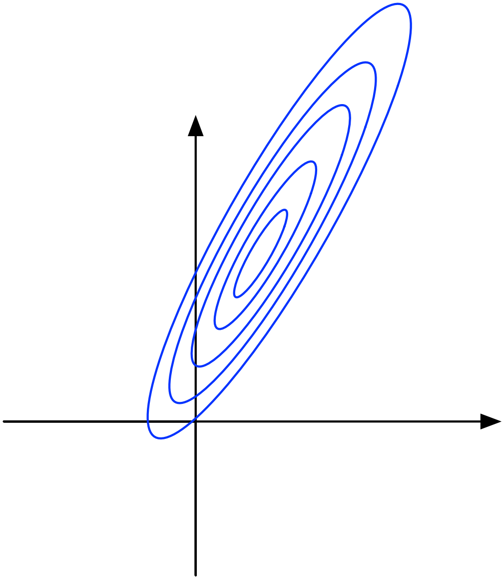
- $\|x\|_2$ is the **Euclidean** norm.
- What is the dimension of $\mathbf{Xu}$? – n-dim. vector
- What is i-th component of $\mathbf{Xu}$?

$$Xu = \begin{bmatrix} x_1 \cdot u \\ \vdots \\ x_i \cdot u \\ \vdots \\ x_n \cdot u \end{bmatrix} \longleftarrow \text{i-th component.}$$

- Also, note: $\|\mathbf{u}\|^2 = \mathbf{u}^\top \mathbf{u}$
- So what is $\|\mathbf{Xu}\|^2$?

$$\|Xu\|^2 = u^T X^T X u = \sum_i (x_i \cdot u)^2$$

Constrained Optimization



The blue lines represent *contours*: all points on a blue line have the same objective function value.

Deriving the Solution

Don't panic.

$$\underset{\mathbf{u}}{\operatorname{argmax}} \|\mathbf{X}\mathbf{u}\|^2, \text{ s.t. } \|\mathbf{u}\|^2 = 1$$

- The Lagrangian encoding of the problem moves the constraint into the objective:

$$\max_{\mathbf{u}} \min_{\lambda} \|\mathbf{X}\mathbf{u}\|^2 - \lambda(\|\mathbf{u}\|^2 - 1) \quad \Rightarrow \quad \min_{\lambda} \max_{\mathbf{u}} \|\mathbf{X}\mathbf{u}\|^2 - \lambda(\|\mathbf{u}\|^2 - 1)$$

Deriving the Solution

Don't panic.

$$\underset{\mathbf{u}}{\operatorname{argmax}} \|\mathbf{X}\mathbf{u}\|^2, \text{ s.t. } \|\mathbf{u}\|^2 = 1$$

- The Lagrangian encoding of the problem moves the constraint into the objective:

$$\max_{\mathbf{u}} \min_{\lambda} \|\mathbf{X}\mathbf{u}\|^2 - \lambda(\|\mathbf{u}\|^2 - 1) \quad \Rightarrow \quad \min_{\lambda} \max_{\mathbf{u}} \|\mathbf{X}\mathbf{u}\|^2 - \lambda(\|\mathbf{u}\|^2 - 1)$$

- Gradient (first derivatives with respect to $\mathbf{u}$): $2\mathbf{X}^\top\mathbf{X}\mathbf{u} - 2\lambda\mathbf{u}$
- Setting equal to $\mathbf{0}$ leads to: $\lambda\mathbf{u} = \mathbf{X}^\top\mathbf{X}\mathbf{u}$
- You may recognize this as the definition of an eigenvector ($\mathbf{u}$) and eigenvalue (λ) for the matrix $\mathbf{X}^\top\mathbf{X}$.
- We take the first (largest) eigenvalue.

Deriving the Solution: Scratch space

$\sum (x_i \cdot u)^2$

$$f_\lambda(u) = \|Xu\|^2 - \lambda \|u\|^2$$

$(x_i^T \cdot u)$

$$0 = \frac{\partial f_\lambda(u)}{\partial u} = 2 \sum_i \underbrace{(x_i \cdot u)}\, x_i - 2\lambda \vec{u} = 0$$

$$= 2 \sum_i x_i (x_i^T \cdot u) - 2\lambda \vec{u}$$

$$= 2 \left(\sum_i x_i x_i^T \right) u - 2\lambda u = 0$$

$$\left(\sum_i x_i x_i^T \right) u = \lambda u$$

Deriving the Solution: Scratch space

Deriving the Solution: Scratch space

Variance in Multiple Dimensions

So far, we've projected each $\mathbf{x}_n$ into one dimension.

To get a second direction $\mathbf{v}$, we solve the same problem again, but this time with another constraint:

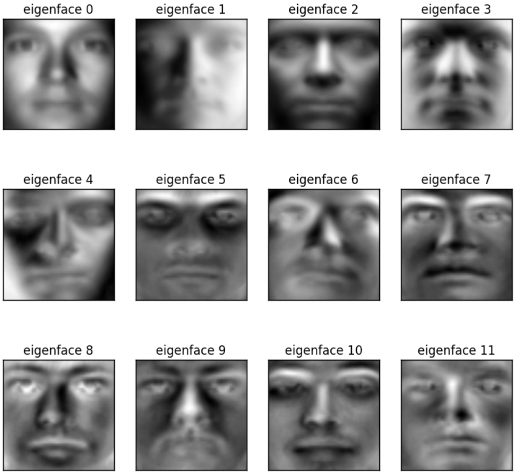
$$\operatorname*{argmax}_{\mathbf{v}} \|\mathbf{X}\mathbf{v}\|^2, \text{ s.t. } \|\mathbf{v}\|^2 = 1 \text{ and } \boxed{\mathbf{u} \cdot \mathbf{v} = 0}$$

(That is, we want a dimension that's orthogonal to the $\mathbf{u}$ that we found earlier.)

Following the same steps we had for $\mathbf{u}$, **the solution will be the second eigenvector.**

“Eigenfaces”

Fig. from `https://github.com/AlexOuyang/RealTimeFaceRecognition`



Principal Components Analysis

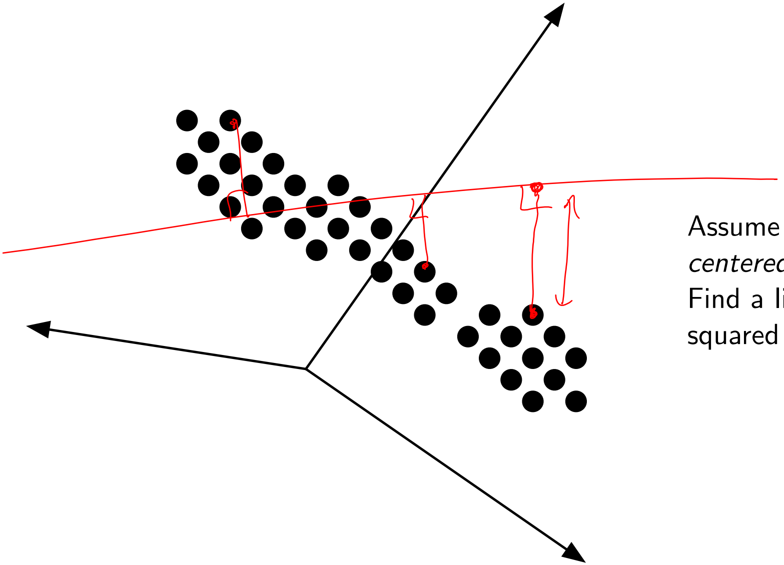
- Input: unlabeled data $\mathbf{X} = [\mathbf{x}_1 | \mathbf{x}_2 | \cdots | \mathbf{x}_N]^\top$; dimensionality $K < d$
- Output: K-dimensional "subspace".
- Algorithm:
  1. Compute the mean μ
  2. compute the **covariance matrix**:

$$\Sigma = \frac{1}{N} \sum_i (\mathbf{x}_i - \mu)^\top (\mathbf{x}_i - \mu)$$

  3. let $\langle \lambda_1, \ldots, \lambda_K \rangle$ be the top K eigenvalues of Σ and $\langle \mathbf{u}_1, \ldots, \mathbf{u}_K \rangle$ be the corresponding eigenvectors
- Let $\widetilde{\mathbf{U}} = [\mathbf{u}_1 | \mathbf{u} | \cdots | \mathbf{u}_K]$
  Return $\widetilde{\mathbf{U}}$

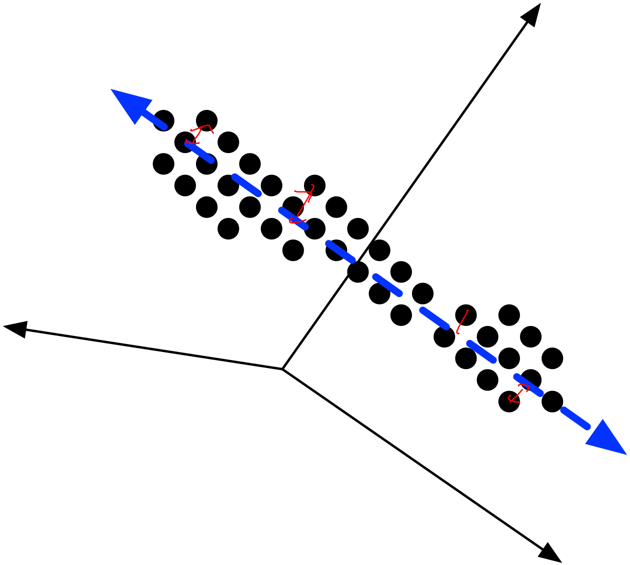
You can read about many algorithms for finding eigendecompositions of a matrix.

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.

Projection and Reconstruction: the one dimensional case

- Take out mean μ: $x \leftarrow x - \mu$
- Find the "top" eigenvector u of the covariance matrix.
- What are your projections?

  $(x \cdot u)$

- What are your reconstructions, $\widehat{\mathbf{X}} = [\widehat{\mathbf{x}}_1 | \widehat{\mathbf{x}}_2 | \cdots | \widehat{\mathbf{x}}_N]^\top$?

  $(x \cdot u)\vec{u} + \vec{\mu}$

- Whis is your reconstruction error?

if k

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

$\lambda_2 + \cdots + \lambda_d$

$\lambda_{k+1} + \cdots + \lambda_d$

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:

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

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

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

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

Choosing K (Hyperparameter Tuning)

How do you select K for PCA?

Read CIML (similar methods for K-means)

PCA and Clustering

There's a unified view of both PCA and clustering.

- K-Means chooses cluster-means so that squared distances to data are small.
- PCA chooses a basis so that reconstruction error of data is small.

Both attempt to find a "simple" way to summarize the data:
fewer points or fewer dimensions.

Both could be used to create new features for supervised learning

Loss functions

Perceptron

A model and an algorithm, rolled into one.

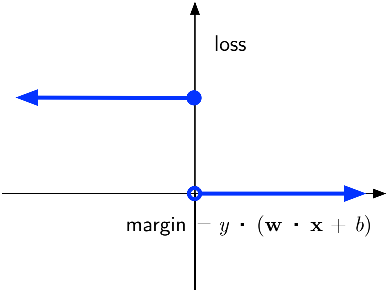
Model: $f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$, known as **linear**, visualized by a (hopefully) separating hyperplane in feature-space.

Algorithm: $\textsc{PerceptronTrain}$, an error-driven, iterative updating algorithm.

A Different View of PERCEPTRONTRAIN: Optimization

"Minimize training-set error rate":

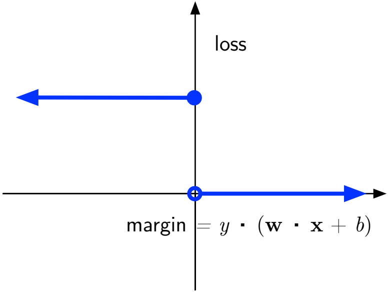
$$\min_{\mathbf{w},b} \underbrace{\frac{1}{N}\sum_{n=1}^{N} [\![y_n \cdot (\mathbf{w} \cdot \mathbf{x} + b) \leq 0]\!]}_{\epsilon^{\text{train}} \equiv \text{ zero-one loss}}$$



A Different View of PerceptronTrain: Optimization

"Minimize training-set error rate":

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

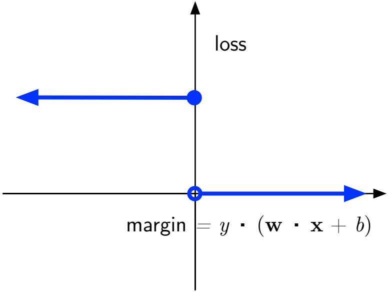


This problem is NP-hard; even solving trying to get a (multiplicaive) approximatation is NP-hard.

A Different View of PerceptronTrain: Optimization

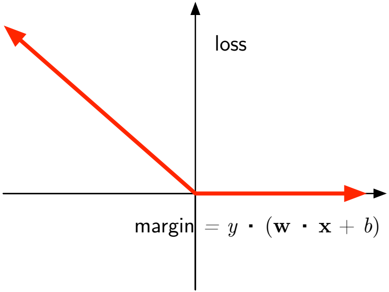
"Minimize training-set error rate":

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

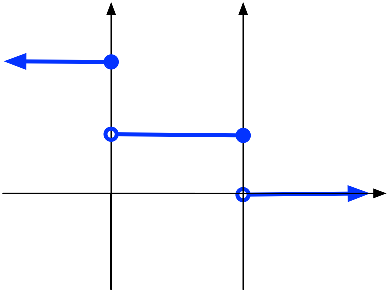


What the perceptron does:

$$\min_{\mathbf{w},b} \frac{1}{N}\sum_{n=1}^{N} \underbrace{\max(-y_n \cdot (\mathbf{w} \cdot \mathbf{x} + b), \quad 0)}_{\text{perceptron loss}}$$

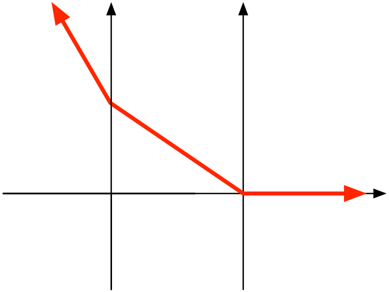


A Different View of PERCEPTRONTRAIN: Optimization



"Minimize training-set error rate":

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



What the perceptron does:

$$\min_{\mathbf{w},b} \frac{1}{N}\sum_{n=1}^{N} \underbrace{\max(-y_n \cdot (\mathbf{w} \cdot \mathbf{x} + b), \quad 0)}_{\text{perceptron loss}}$$

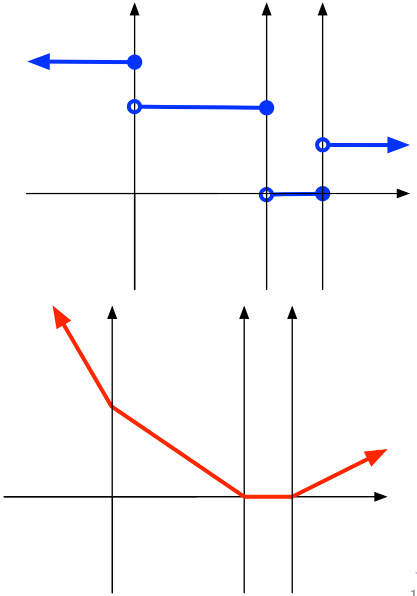
A Different View of PerceptronTrain: Optimization

"Minimize training-set error rate":

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

What the perceptron does:

$$\min_{\mathbf{w},b} \frac{1}{N}\sum_{n=1}^{N} \underbrace{\max(-y_n \cdot (\mathbf{w}\cdot\mathbf{x} + b), \quad 0)}_{\text{perceptron loss}}$$



Smooth out the Loss?