

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

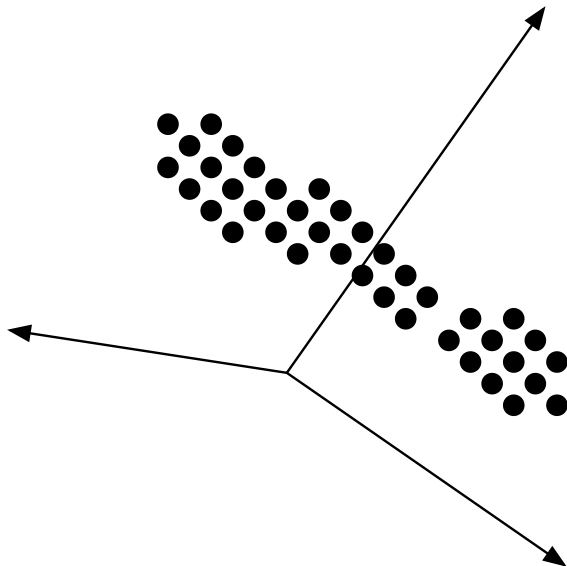
Sham M Kakade

© 2018

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

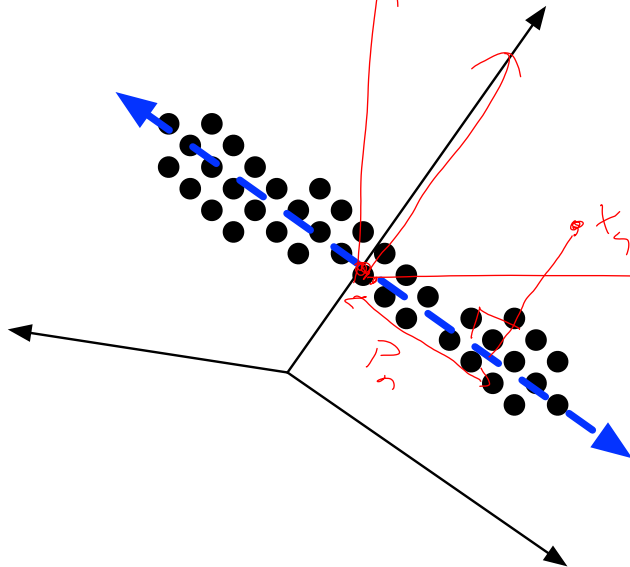
PCA: continuing on...

## Dimension of Greatest Variance



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

## Dimension of Greatest Variance



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

# 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  $n$ th example onto  $\mathbf{u}$ .



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

This implies that the variance of  $\langle p_1, \dots, p_N \rangle$  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

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

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

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

*n x d*

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

*(Handwritten red annotations: a bracket above the matrix  $\mathbf{X}$  labeled  $\mathbf{x}_i^\top$ , and a bracket below the matrix  $\mathbf{X}$  labeled  $\mathbf{x}_n^\top$ )*

~~2~~ 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  $\mathbf{Xu}$ ?

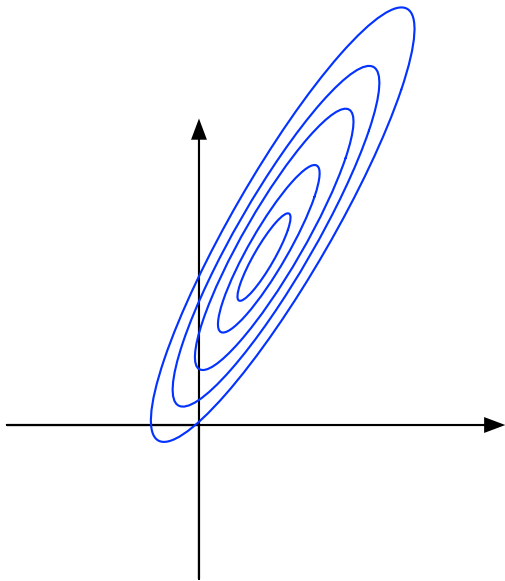
$$\mathbf{Xu} = \begin{bmatrix} x_{1 \cdot u} \\ \vdots \\ x_{i \cdot u} \\ \vdots \\ x_{n \cdot u} \end{bmatrix} \leftarrow i\text{-th component}$$

► Also, note:  $\|\mathbf{u}\|^2 = \mathbf{u}^\top \mathbf{u}$

► So what is  $\|\mathbf{Xu}\|^2$ ?

$$\|\mathbf{Xu}\|^2 = \mathbf{u}^\top \mathbf{X}^\top \mathbf{X} \mathbf{u} = \sum_i (x_{i \cdot u})^2$$

# 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.

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

- 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.

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

- ▶ 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 ( $\lambda$ ) for the matrix  $\mathbf{X}^\top \mathbf{X}$ .
- ▶ We take the first (largest) eigenvalue.

Deriving the Solution: Scratch space

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

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

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

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

$$\left( \sum_i x_i x_i^T \right) u = \cancel{2\lambda} u$$

## 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:

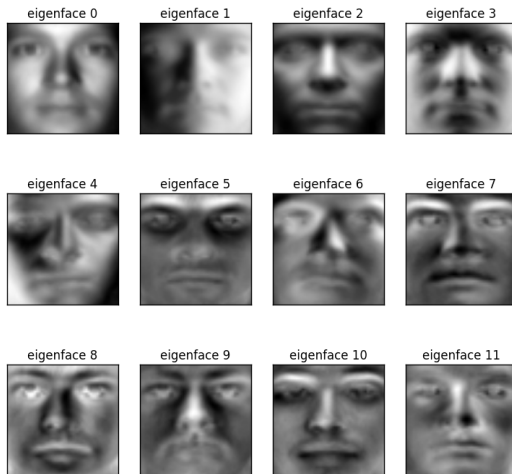
$$\operatorname{argmax}_{\mathbf{v}} \|\mathbf{X}\mathbf{v}\|^2, \text{ s.t. } \|\mathbf{v}\|^2 = 1 \text{ and } \boxed{\mathbf{u} \cdot \mathbf{v} = 0}$$

(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  $\mu$
  2. compute the **covariance matrix**:

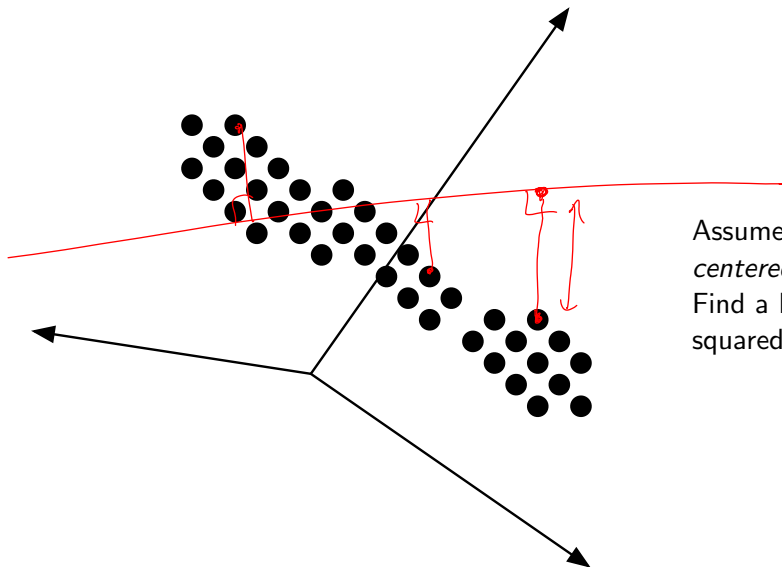
$$\Sigma = \frac{1}{N} \sum_i (\mathbf{x}_i - \mu) (\mathbf{x}_i - \mu)^\top$$

3. let  $\langle \lambda_1, \dots, \lambda_K \rangle$  be the top  $K$  eigenvalues of  $\Sigma$  and  $\langle \mathbf{u}_1, \dots, \mathbf{u}_K \rangle$  be the corresponding eigenvectors
- ▶ Let  $\tilde{\mathbf{U}} = [\mathbf{u}_1 | \mathbf{u}_2 | \cdots | \mathbf{u}_K]$   
**Return**  $\tilde{\mathbf{U}}$

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

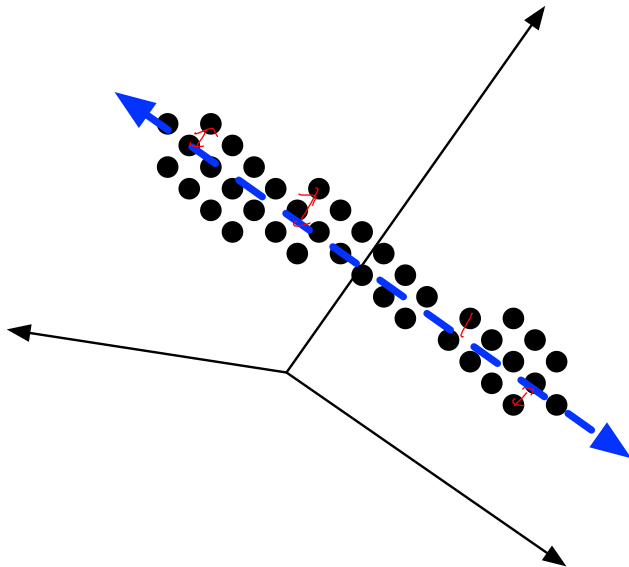


## Alternate View of PCA: Minimizing Reconstruction Error



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  $\mu$ :  $x \leftarrow x - \mu$
- ▶ Find the “top” eigenvector  $u$  of the covariance matrix.
- ▶ What are your projections?

$$(X \cdot u)$$

- ▶ What are your reconstructions,  $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1 | \hat{\mathbf{x}}_2 | \dots | \hat{\mathbf{x}}_N]^T$ ?

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

- ▶ What is your reconstruction error?

if  $K$   
then  
 $\lambda_{K+1} + \dots + \lambda_d$

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

$$\lambda_2 + \dots + \lambda_d$$

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

Equivalent (“dual”) formulation of PCA: find an “orthonormal basis”  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K$  which minimizes the total reconstruction error on the data:

$$\underset{\text{orthonormal basis: } \mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K}{\operatorname{argmin}} \quad \frac{1}{N} \sum_i (\mathbf{x}_i - \operatorname{Proj}_{\mathbf{u}_1, \dots, \mathbf{u}_K}(\mathbf{x}_i))^2$$

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

$$\operatorname{Proj}_{\mathbf{u}_1, \dots, \mathbf{u}_K}(\mathbf{x}) = \sum_{j=1}^K (\mathbf{u}_j \cdot \mathbf{x}) \mathbf{u}_j$$

The SVD “simultaneously” finds all  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K$

# 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.

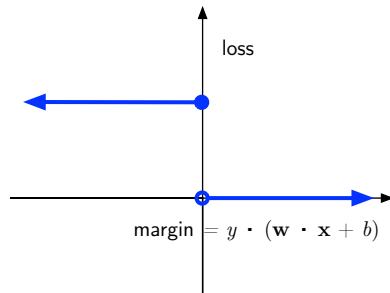
Algorithm: PERCEPTRONTRAIN, an error-driven, iterative updating algorithm.



# A Different View of PERCEPTRONTRAIN: Optimization

“Minimize training-set error rate”:

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

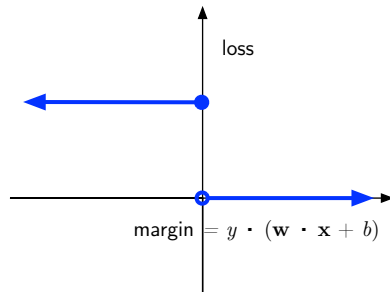


# A Different View of PERCEPTRON TRAIN: Optimization

“Minimize training-set error rate”:

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

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



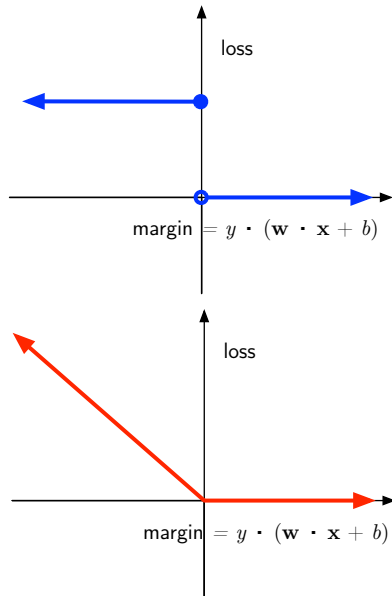
# A Different View of PERCEPTRONTRAIN: Optimization

“Minimize training-set error rate”:

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

What the perceptron does:

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



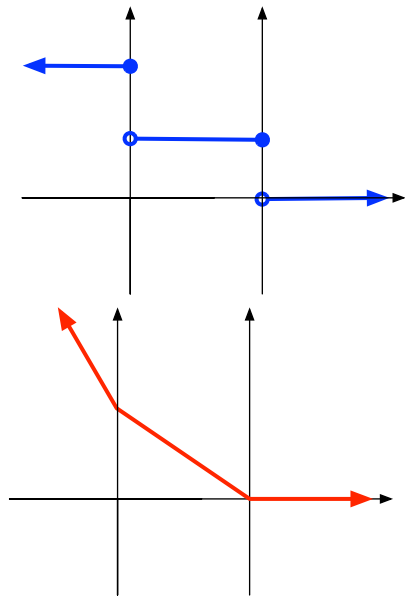
# A Different View of PERCEPTRONTRAIN: Optimization

“Minimize training-set error rate”:

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

What the perceptron does:

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



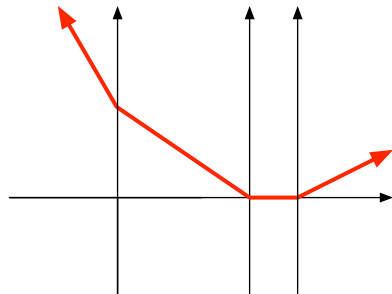
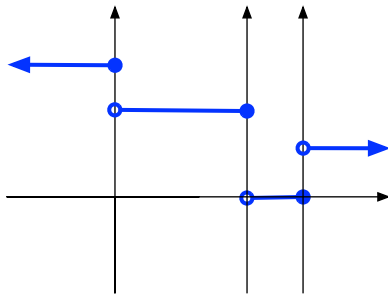
# A Different View of PERCEPTRONTRAIN: Optimization

“Minimize training-set error rate”:

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

What the perceptron does:

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



## Smooth out the Loss?

