PCA: continuing on...
Assume that the data are *centered*, i.e., that
\[
\text{mean } \left( \langle x_n \rangle_{n=1}^N \right) = 0.
\]
Assume that the data are centered, i.e., that \( \text{mean} \left( \langle x_n \rangle_{n=1}^N \right) = 0 \).
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 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 $rac{1}{N} \sum_{n=1}^{N} p_n^2$.

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

$$\text{argmax}_{\mathbf{u}} \sum_{n=1}^{N} (\mathbf{x}_n \cdot \mathbf{u})^2$$
Finding the Maximum-Variance Direction

\[
\operatorname{argmax}_u \sum_{n=1}^{N} (x_n \cdot u)^2
\]

s.t. \( \|u\|^2 = 1 \)

(Why do we constrain \( u \) to have length 1?)

If we let \( X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_N^\top \end{bmatrix} \), then we want: \( \operatorname{argmax}_u \|Xu\|^2 \), s.t. \( \|u\|^2 = 1 \).

This is PCA in one dimension!
Linear algebra review: things to understand

- $\|x\|_2$ is the **Euclidean** norm.
- What is the dimension of $Xu$?
- What is $i$-th component of $Xu$?

\[ Xu = \begin{bmatrix} x_1 \cdot u \\ \vdots \\ x_n \cdot u \end{bmatrix} \quad \text{\textit{i-th component}} \]

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

\[ \|Xu\|^2 = u^\top X^\top Xu = \sum_i (x_i \cdot u)^2 \]
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.

\[
\text{argmax } \|Xu\|^2, \text{ s.t. } \|u\|^2 = 1
\]

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

\[
\max_u \min_\lambda \|Xu\|^2 - \lambda(\|u\|^2 - 1) \implies \min_\lambda \max_u \|Xu\|^2 - \lambda(\|u\|^2 - 1)
\]

\[
\text{Gradient (first derivatives with respect to } u) : 2X^\top Xu - 2\lambda u
\]

\[
\text{Setting equal to } 0 \text{ leads to: } \lambda u = X^\top Xu
\]

You may recognize this as the definition of an eigenvector \((u)\) and eigenvalue \((\lambda)\) for the matrix \(X^\top X\).

We take the first (largest) eigenvalue.
Deriving the Solution

Don’t panic.

\[
\arg\max_u \| Xu \|^2, \text{ s.t. } \| u \|^2 = 1
\]

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

\[
\max_u \min_\lambda \| Xu \|^2 - \lambda (\| u \|^2 - 1) \Rightarrow \min_\lambda \max_u \| Xu \|^2 - \lambda (\| u \|^2 - 1)
\]

- Gradient (first derivatives with respect to \( u \)): \( 2X^TXu - 2\lambda u \)

- Setting equal to 0 leads to: \( \lambda u = X^TXu \)

- You may recognize this as the definition of an eigenvector (\( u \)) and eigenvalue (\( \lambda \)) for the matrix \( X^TX \).

- We take the first (largest) eigenvalue.
Deriving the Solution: Scratch space

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

\[
0 = \frac{\partial f_\lambda(u)}{\partial u} = 2 \sum (X_i^T u)x_i - 2x\tilde{u} = 0
\]

\[
= 2 \sum (X_i^T u)x_i - 2x\tilde{u} = 0
\]

\[
= 2 \left( \sum x_i x_i^T \right) u - 2xu = 0
\]

\[
(\sum x_i x_i^T) u = \lambda u
\]
Deriving the Solution: Scratch space
Deriving the Solution: Scratch space
Variance in Multiple Dimensions

So far, we’ve projected each $x_n$ into one dimension.

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

$$\arg\max_v \|Xv\|^2, \text{ s.t. } \|v\|^2 = 1 \text{ and } u \cdot v = 0$$

(That is, we want a dimension that’s orthogonal to the $u$ that we found earlier.)

Following the same steps we had for $u$, the solution will be the second eigenvector.
“Eigenfaces”

Fig. from https://github.com/AlexOuyang/RealTimeFaceRecognition
Principal Components Analysis

- Input: unlabeled data $X = [x_1 | x_2 | \cdots | x_N]^\top$; dimensionality $K < d$
- Output: $K$-dimensional “subspace”.
- Algorithm:
  1. Compute the mean $\mu$
  2. Compute the covariance matrix:

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

  3. Let $\langle \lambda_1, \ldots, \lambda_K \rangle$ be the top $K$ eigenvalues of $\Sigma$ and $\langle u_1, \ldots, u_K \rangle$ be the corresponding eigenvectors
- Let $\tilde{U} = [u_1 | u| \cdots | u_K]$
- Return $\tilde{U}$

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 $\mu$: $\mathbf{x} \in \mathbf{x} - \mu$
- Find the "top" eigenvector $\mathbf{u}$ of the covariance matrix.
- What are your projections?

$$\left(\mathbf{x} \cdot \mathbf{u}\right)$$

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

$$\left(\mathbf{x} \cdot \mathbf{u}\right)\mathbf{u}^\top + \mathbf{u}$$

- What is your reconstruction error?

$$\left\| \mathbf{x}_{\mathbf{k}+1} + \cdots + \mathbf{x}_d \right\|^2 = \frac{1}{N} \sum_i (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2$$
Alternate View: Minimizing Reconstruction Error with $K$-dim subspace.

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

$$\arg\min_{\text{orthonormal basis: } u_1, u_2, \ldots u_K} \frac{1}{N} \sum_i (x_i - \text{Proj}_{u_1, \ldots u_K}(x_i))^2$$

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

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

The SVD "simultaneously" finds all $u_1, u_2, \ldots u_K$
Choosing $K$ (Hyperparameter Tuning)

How do you select $K$ for PCA?

Read CIML (similar methods for $K$-means)
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: \( f(x) = \text{sign}(w \cdot x + b) \), known as **linear**, visualized by a (hopefully) separating hyperplane in feature-space.

Algorithm: **PerceptronTrain**, an error-driven, iterative updating algorithm.
A Different View of \textsc{PerceptronTrain}: Optimization

“Minimize training-set error rate”:

$$\min_{w,b} \frac{1}{N} \sum_{n=1}^{N} [y_n \cdot (w \cdot x + b) \leq 0]$$

\(\epsilon_{\text{train}} \equiv \) zero-one loss

loss

margin = \(y \cdot (w \cdot x + b)\)
A Different View of PERCEPTRONTRAIN: Optimization

“Minimize training-set error rate”:

$$\min_{w,b} \frac{1}{N} \sum_{n=1}^{N} \left[ y_n \cdot (w \cdot x + b) \leq 0 \right]$$

$$\epsilon_{\text{train}} \equiv \text{zero-one loss}$$

This problem is NP-hard; even solving trying to get a (multiplicative) approximation is NP-hard.
A Different View of \textsc{PerceptronTrain}: Optimization

“Minimize training-set error rate”:

\[
\min_{w,b} \frac{1}{N} \sum_{n=1}^{N} [y_n \cdot (w \cdot x + b) \leq 0]
\]

\(\epsilon^{\text{train}} \equiv \) zero-one loss

What the perceptron does:

\[
\min_{w,b} \frac{1}{N} \sum_{n=1}^{N} \max(-y_n \cdot (w \cdot x + b), 0)
\]

\(\text{perceptron loss}\)

\[
\text{margin} = y \cdot (w \cdot x + b)
\]
A Different View of **PerceptronTrain**: Optimization

“Minimize training-set error rate”:

\[
\min_{\mathbf{w}, b} \frac{1}{N} \sum_{n=1}^{N} [y_n \cdot (\mathbf{w} \cdot \mathbf{x} + b) \leq 0]
\]

\[\epsilon_{\text{train}} \equiv \text{zero-one loss}\]

What the perceptron does:

\[
\min_{\mathbf{w}, b} \frac{1}{N} \sum_{n=1}^{N} \max(-y_n \cdot (\mathbf{w} \cdot \mathbf{x} + b), 0)
\]

\[\text{perceptron loss}\]
A Different View of PERCEPTRON\textsc{Train}: Optimization

“Minimize training-set error rate”:

\[
\min_{w,b} \frac{1}{N} \sum_{n=1}^{N} [y_n \cdot (w \cdot x + b) \leq 0]
\]

\[\epsilon_{\text{train}} \equiv \text{zero-one loss}\]

What the perceptron does:

\[
\min_{w,b} \frac{1}{N} \sum_{n=1}^{N} \max(-y_n \cdot (w \cdot x + b), 0)
\]

perceptron loss
Smooth out the Loss?