Data Mining

PCA & Neural Network

https://data-mining.github.io/winter-2026/

CS 453/553 – Winter 2026
Yu Wang, Ph.D.
Assistant Professor
Computer Science
University of Oregon

Fun Facts about ML/AI/DM

Meta | Our approach | Research | Product experiences | Llama | Blog

TOOLS

Faiss

Faiss (Facebook AI Similarity Search) is a library that allows developers to quickly search for embeddings of multimedia documents that are similar to each other. It solves limitations of traditional query search engines that are optimized for hash-based searches, and provides more scalable similarity search functions.

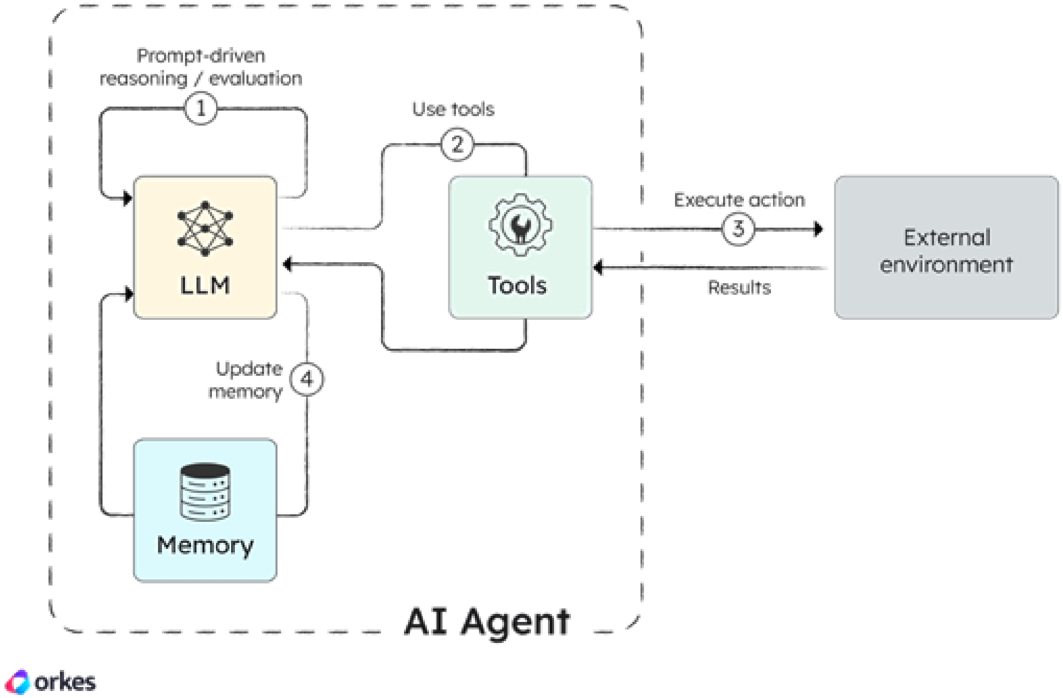
Efficient similarity search

With Faiss, developers can search multimedia documents in ways that are inefficient or impossible with standard database engines (SQL). It includes nearest-neighbor search implementations for million-to-billion-scale datasets that optimize the memory-speed-accuracy tradeoff. Faiss aims to offer state-of-the-art performance for all operating points.

Faiss contains algorithms that search in sets of vectors of any size, and also contains supporting code for evaluation and parameter tuning. Some if its most useful algorithms are implemented on the GPU. Faiss is implemented in C++, with an optional Python interface and GPU support via CUDA.

Fun Facts about ML/AI/DM

$\phi(A|B)$

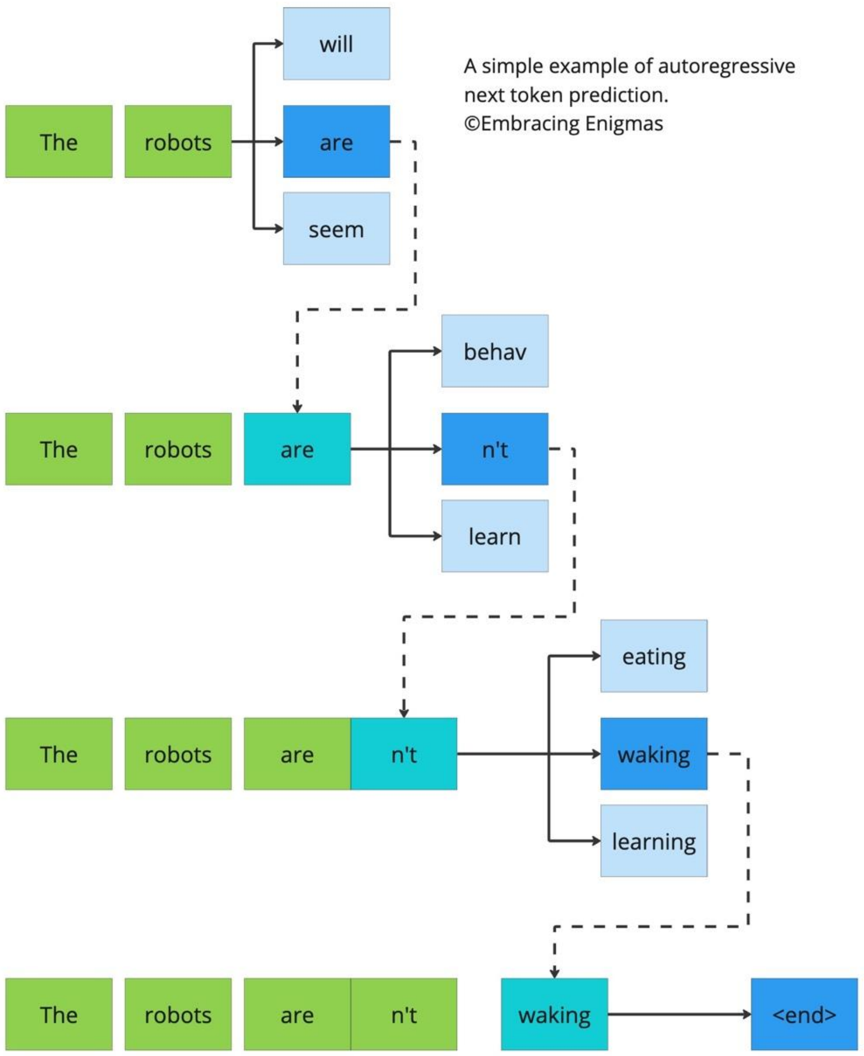


SFT

RL

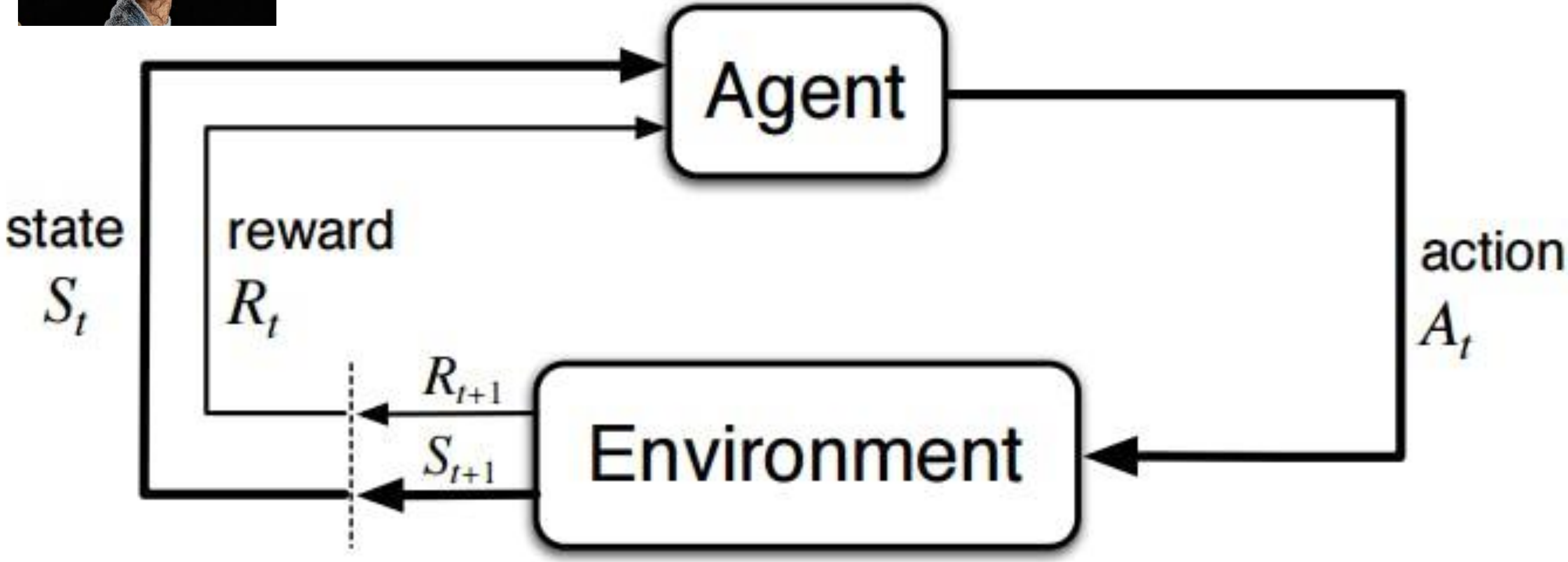
Fun Facts about ML/AI/DM

SFT

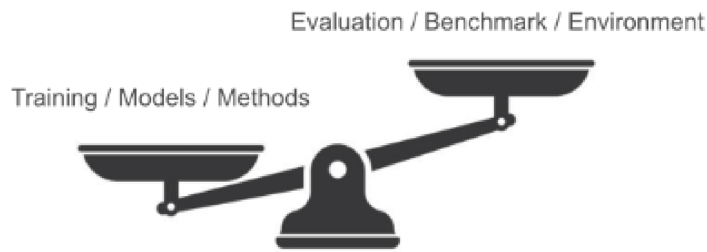


Fun Facts about ML/AI/DM

RL



The first half



Learning internal representations by error-propagation 59436 *
DE Rumelhart, GE Hinton, RJ Williams
Parallel Distributed Processing: Explorations in the Microstructure of ...

???

Imagenet classification with deep convolutional neural networks 171956 *
A Krizhevsky, I Sutskever, GE Hinton
Advances in neural information processing systems 25

ImageNet Large Scale Visual Recognition Challenge 49275
O Russakovsky, J Deng, H Su, J Krause, S Satheesh, S Ma, Z Huang, ...
arXiv preprint arXiv:1409.0575, 2014

Playing atari with deep reinforcement learning 16853
V Mnih, K Kavukcuoglu, D Silver, A Graves, I Antonoglou, D Wierstra, ...
arXiv preprint arXiv:1312.5602, 2013

The arcade learning environment: An evaluation platform for general agents 3844
MG Bellemare, Y Naddaf, J Veness, M Bowling
Journal of Artificial Intelligence Research 47, 253-279

Attention is all you need 168293
A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, AN Gomez, ...
Advances in neural information processing systems 30, 2017

Findings of the 2014 workshop on statistical machine translation 1332
O Bojar, C Buck, C Federmann, B Haddow, P Koehn, J Leveling, C Monz, ...
Proceedings of the ninth workshop on statistical machine translation, 12-58

Language models are few-shot learners 48882 *
T Brown, B Mann, N Ryder, M Subbiah, JD Kaplan, P Dhariwal, ...
Advances in neural information processing systems 33, 1877-1901

Superglue: A stickier benchmark for general-purpose language understanding systems 2483
A Wang, Y Pruksachatkun, N Nangia, A Singh, J Michael, F Hill, O Levy, ...
Advances in neural information processing systems 32

Fun Facts about ML/AI/DM

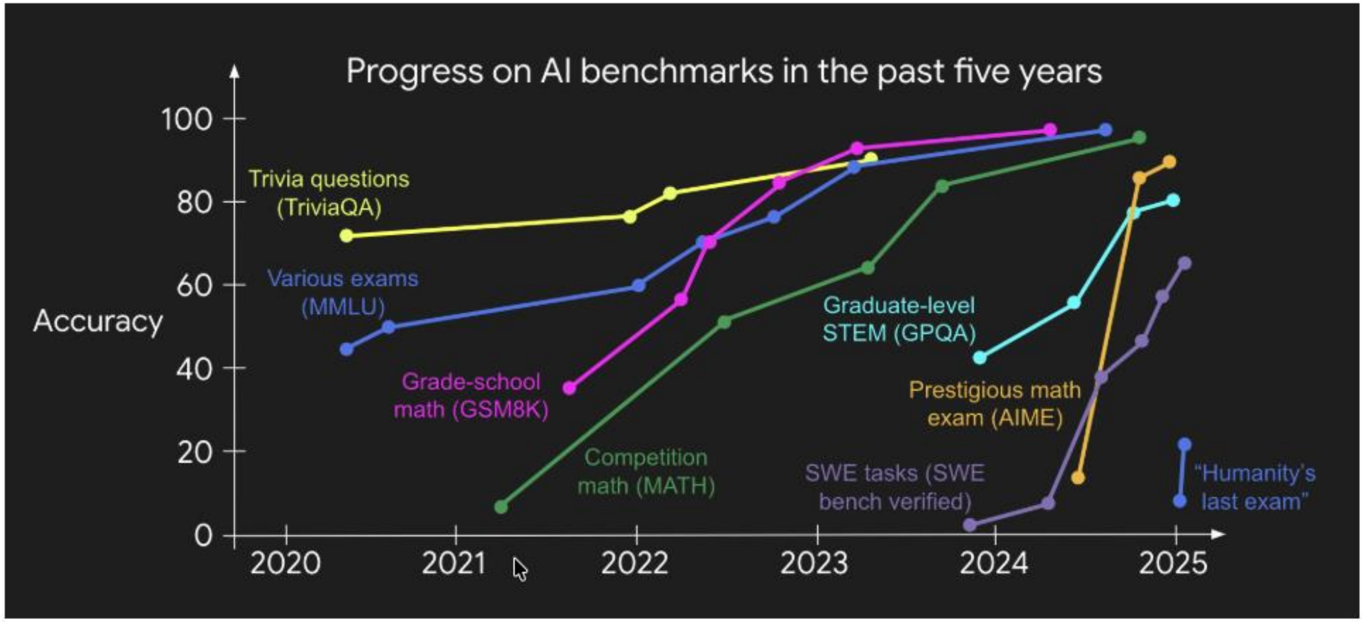
The second half

This recipe is completely changing the game. To recap the game of the first half:

- We develop novel training methods or models that hillclimb benchmarks.
- We create harder benchmarks and continue the loop.

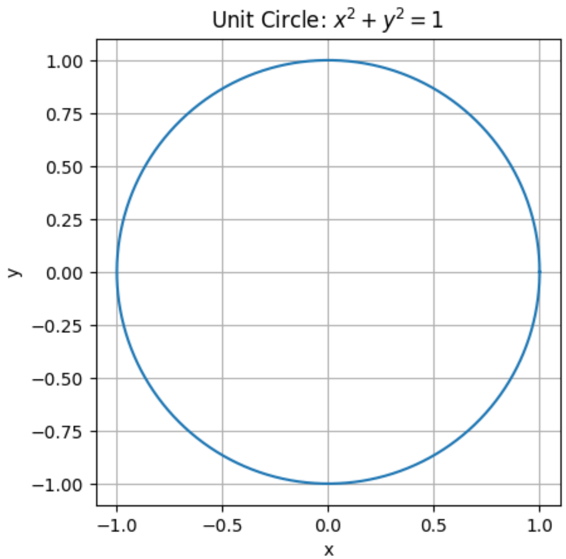
This game is being ruined because:

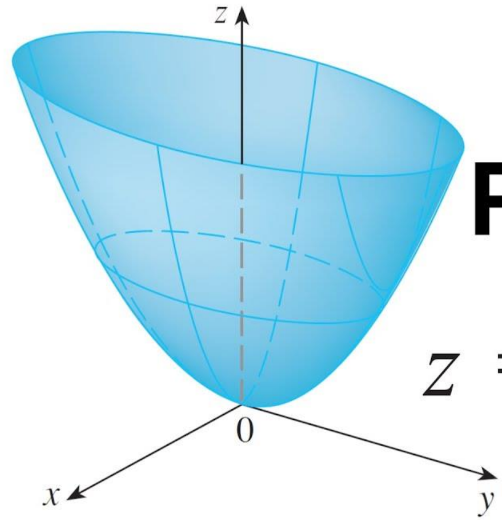
- The recipe has essentially standardized and industried benchmark hillclimbing without requiring much more new ideas. As the recipe scales and generalizes well, your novel method for a particular task might improve it by 5%, while the next o-series model improve it by 30% without explicitly targeting it.
- Even if we create harder benchmarks, pretty soon (and increasingly soon) they get solved by the recipe. My colleague Jason Wei made a beautiful figure to visualize the trend well:



Question Time!

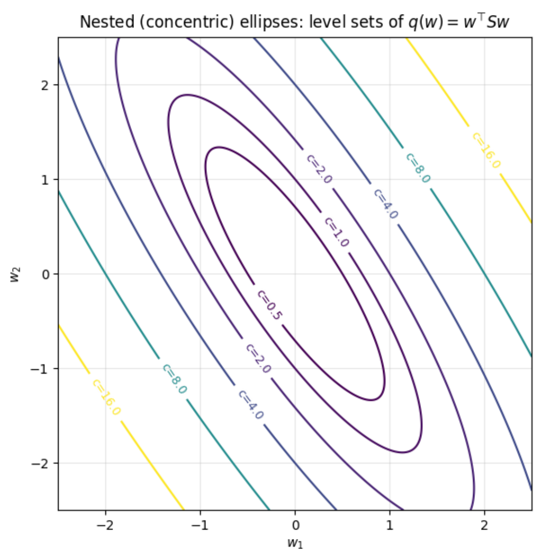
PCA – Math





Elliptic Paraboloid

$$z = 4x^2 + y^2$$



PCA – Math

Define $\mathbf{w}$ and S

Let

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \qquad S = \begin{bmatrix} s_{11} & s_{12} \\ s_{12} & s_{22} \end{bmatrix} \quad \text{(symmetric)}$$

Write out $\mathbf{w}^\top S\mathbf{w}$

First,

$$S\mathbf{w} = \begin{bmatrix} s_{11}w_1 + s_{12}w_2 \\ s_{12}w_1 + s_{22}w_2 \end{bmatrix}$$

Then,

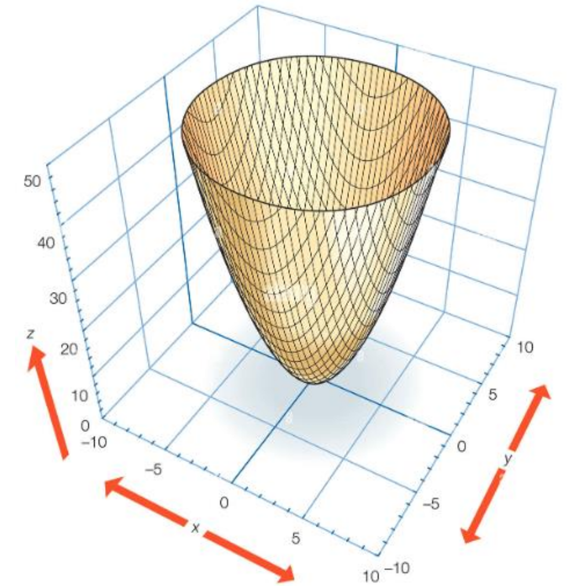
$$\mathbf{w}^\top S\mathbf{w} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} s_{11}w_1 + s_{12}w_2 \\ s_{12}w_1 + s_{22}w_2 \end{bmatrix}$$

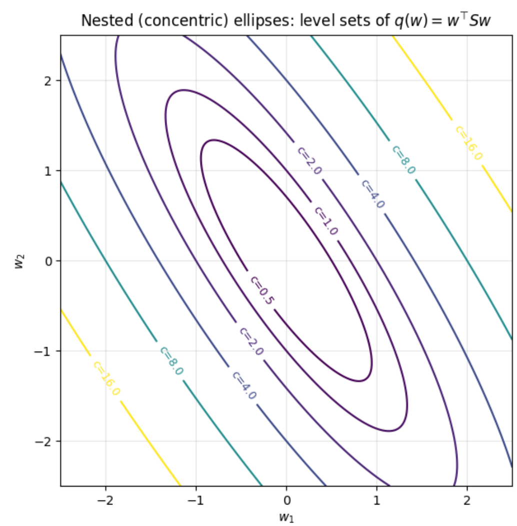
So,

$$\boxed{\mathbf{w}^\top S\mathbf{w} = s_{11}w_1^2 + 2s_{12}w_1w_2 + s_{22}w_2^2}$$

And the corresponding surface is:

$$\boxed{z = s_{11}w_1^2 + 2s_{12}w_1w_2 + s_{22}w_2^2}$$

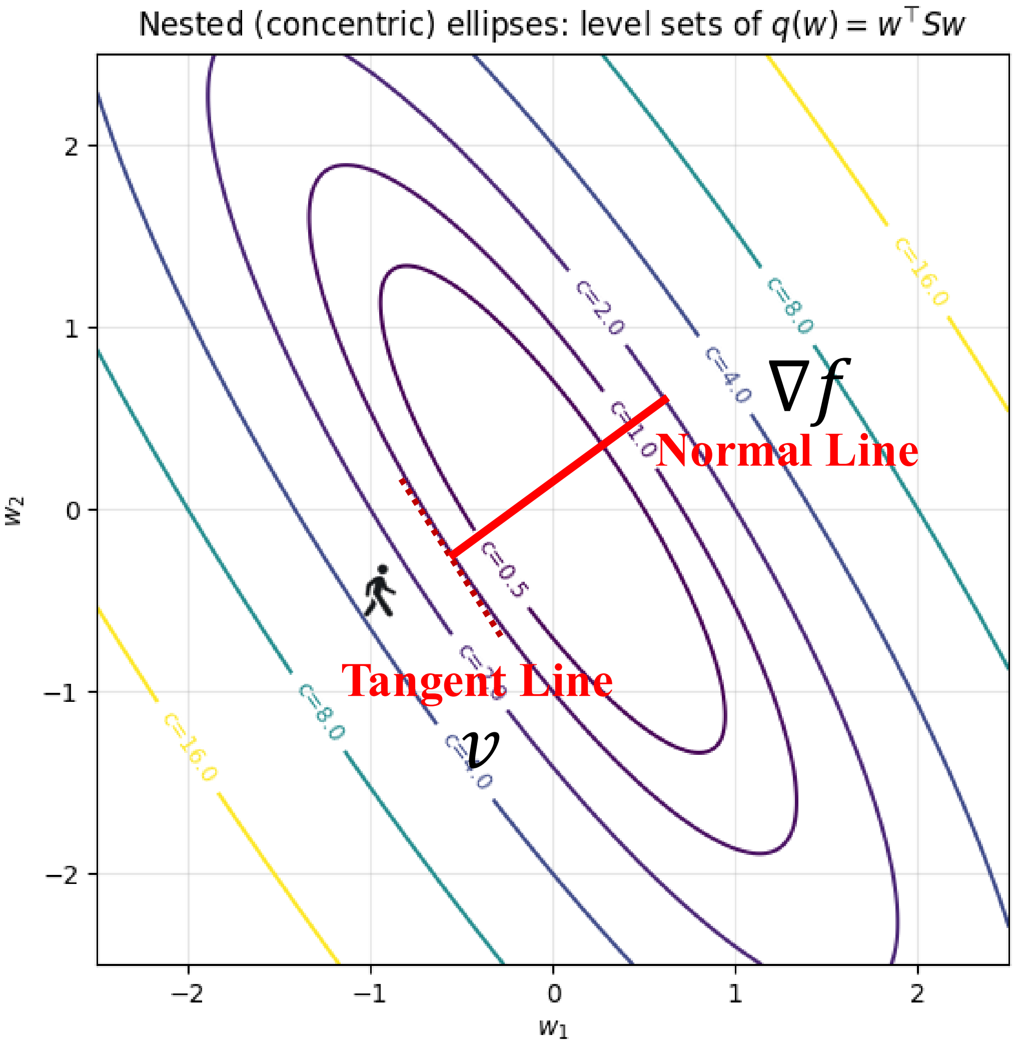




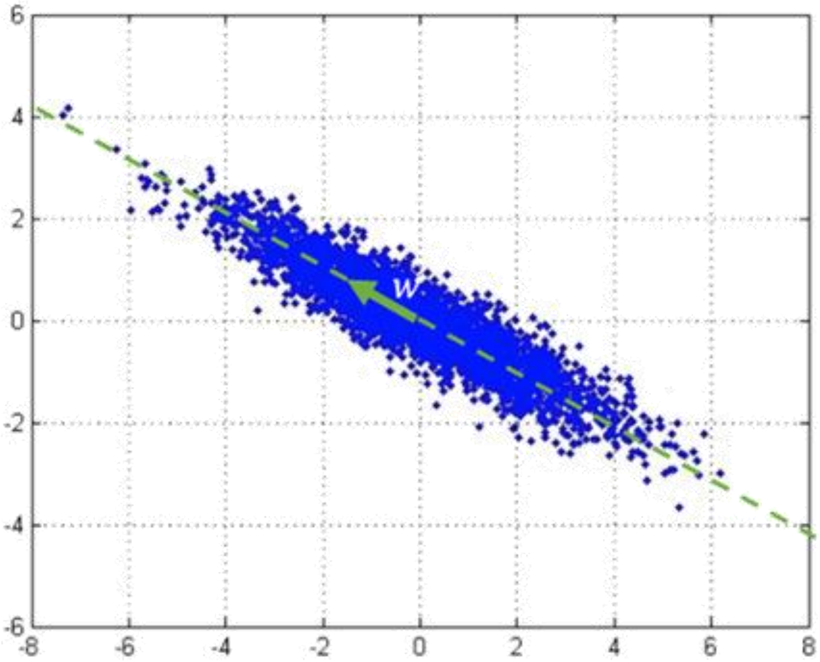
PCA – Math

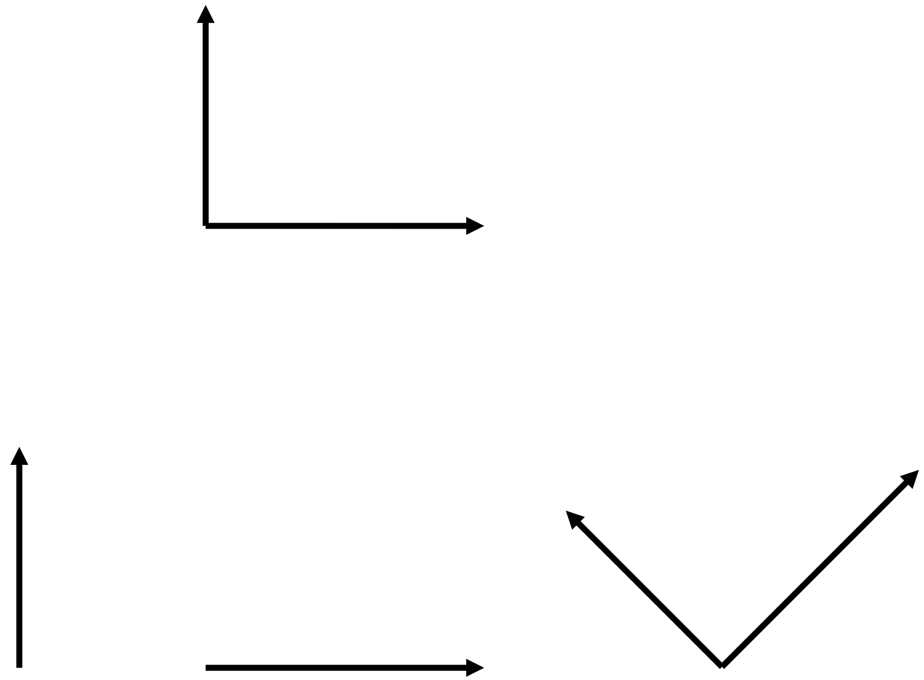
$$f(w_1, w_2) = c$$

$$\nabla f \cdot v = 0$$



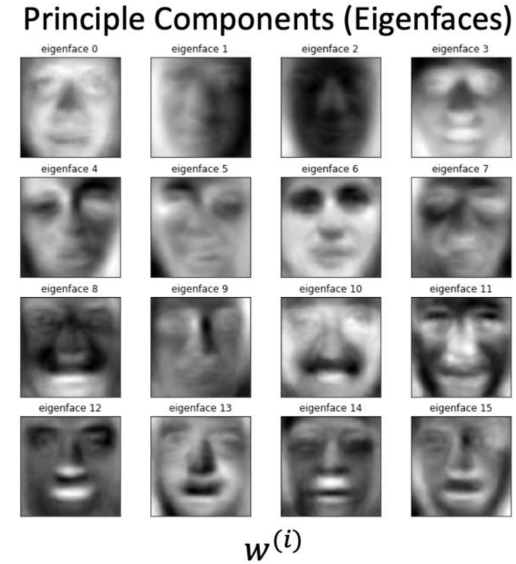
PCA – Principal Component Analysis



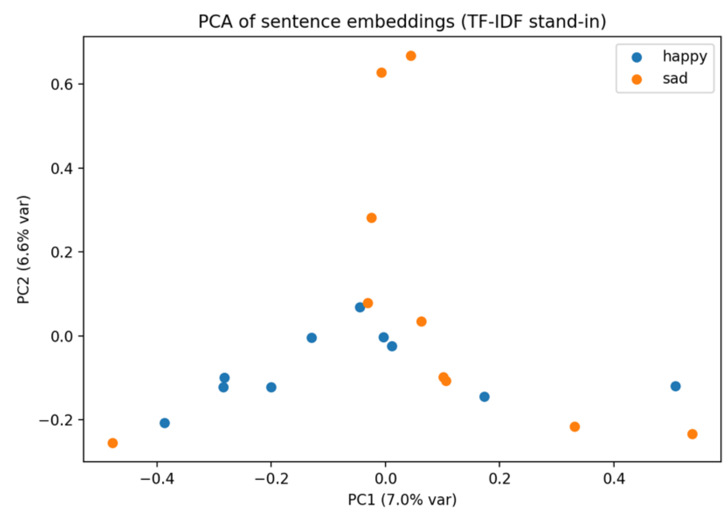


If you want to represent these cluster of points, how do you plan to do?

PCA – Application

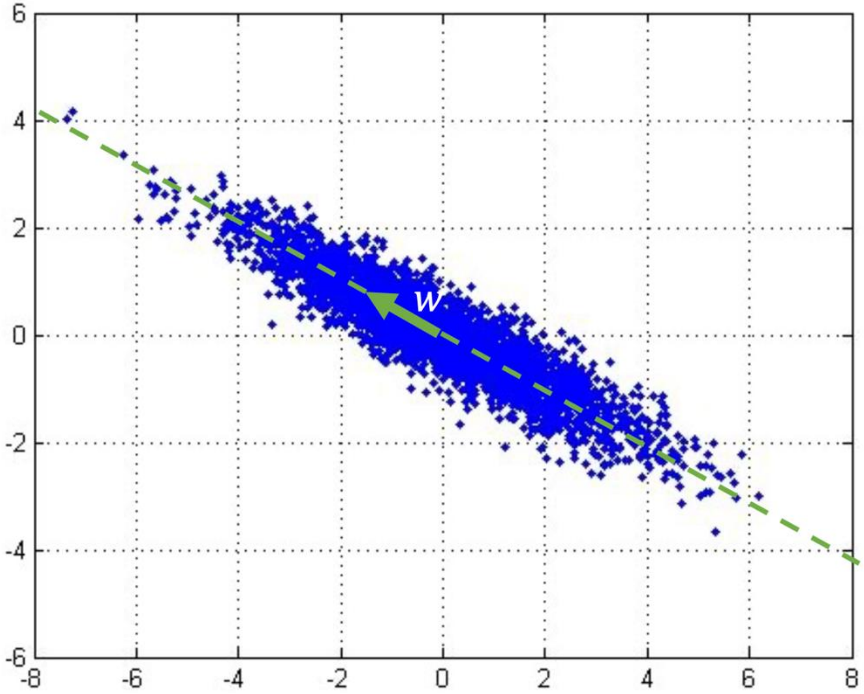



```python
# ----- 1) Toy dataset: happy vs sad -----
happy = [
    "I'm feeling great today!",
    "This made my day so much better.",
    "I love how everything turned out.",
    "What a wonderful surprise!",
    "I'm so proud of what we achieved.",
    "That was hilarious—I can't stop smiling.",
    "I'm excited for what comes next.",
    "Life feels bright and hopeful.",
    "I'm grateful for your help.",
    "This is the best news I've heard all week."
]
sad = [
    "I feel really down today.",
    "Nothing seems to be going right.",
    "I'm disappointed with how it ended.",
    "I can't stop thinking about what I lost.",
    "I feel exhausted and hopeless.",
    "This is heartbreaking.",
    "I'm worried things won't improve.",
    "I miss the way things used to be.",
    "I feel alone in this.",
    "It's been a very rough week."
]
```



PCA – Linear Dimensionality Reduction

2D Gaussian dataset

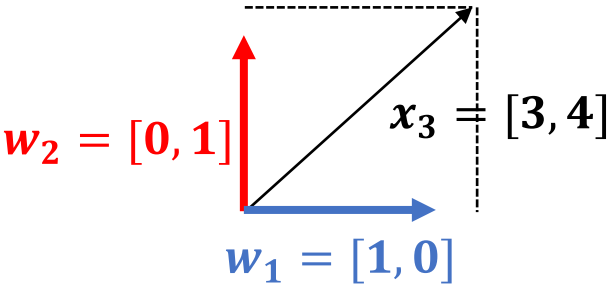


Reduction to 1D

Which linear direction you want choose?

What would be a good reduction?

- Find w such that it maximizes the variance of the projected data
- Find w such that it minimizes the reconstruction error



$$\cos\theta = \frac{w_1^T x_3}{|w_1|_2 |x_3|_2} \qquad x_3 = (w_1^T x_3) w_1 + (w_2^T x_3) w_2$$

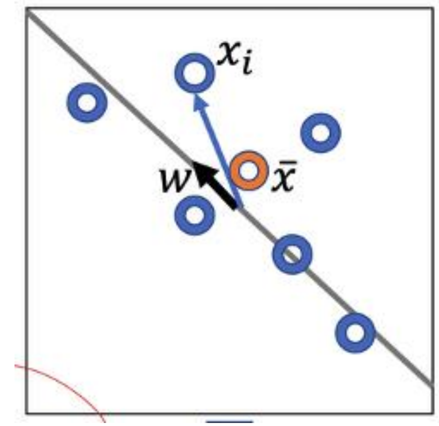
$$\cos\theta |x_3|_2 = \frac{w_1^T x_3}{|w_1|_2 |x_3|_2} = \frac{w_1^T x_3}{|w_1|_2} = w_1^T x_3$$

PCA – Linear Dimensionality Reduction

$$var = \frac{1}{N}\sum_{i=1}^{N}(w^T(x_i - \bar{x}))^2 = \frac{1}{N}\sum_{i=1}^{N} w^T(x_i - \bar{x})(x_i - \bar{x})^T w$$

$$= w^T\left(\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T\right)w$$

$$= w^T S w$$

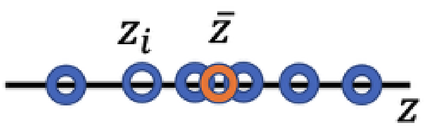


Constrained Optimization

$$\max_{w} \quad w^T S w$$
$$s.t. \; w^T w = 1$$

KKT Condition

$$Sw = \lambda w$$



PCA – Linear Dimensionality Reduction

Constrained Optimization

$$\max_{w} \; w^T S w$$
$$s.t. \; w^T w = 1$$

KKT Condition

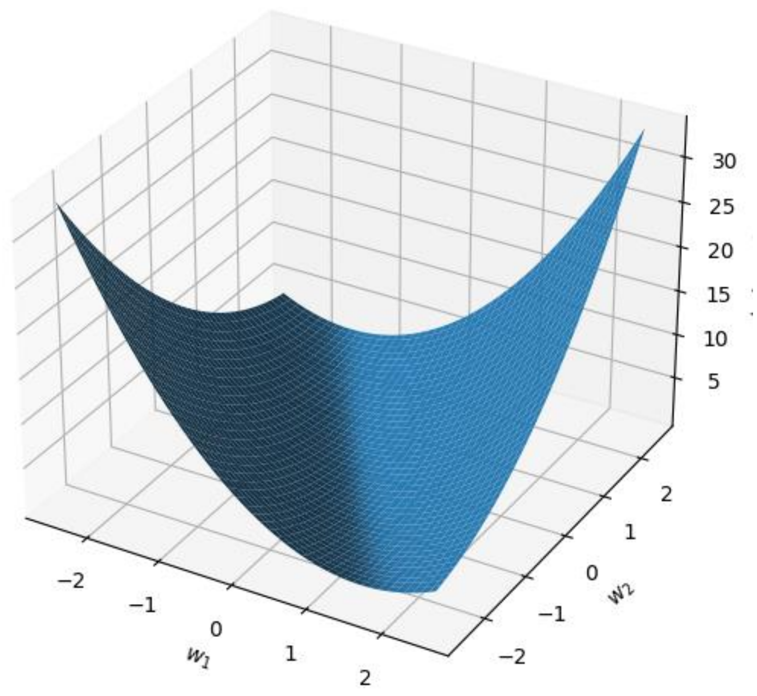
$$Sw = \lambda w$$

$$\nabla(w^\top S w) = 2Sw$$

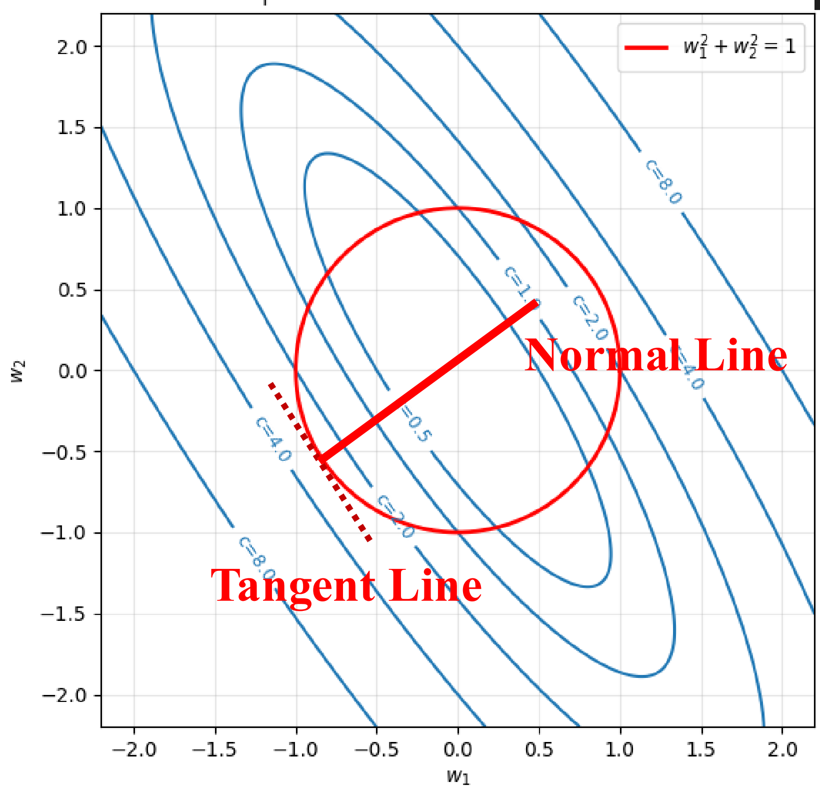
$$\nabla(w^\top w) = 2w$$

$$2Sw = 2\lambda w$$

The "bowl": elliptic paraboloid of $q(w) = w^\top S w$



Nested ellipses of $w^T S w$ with the unit circle constraint

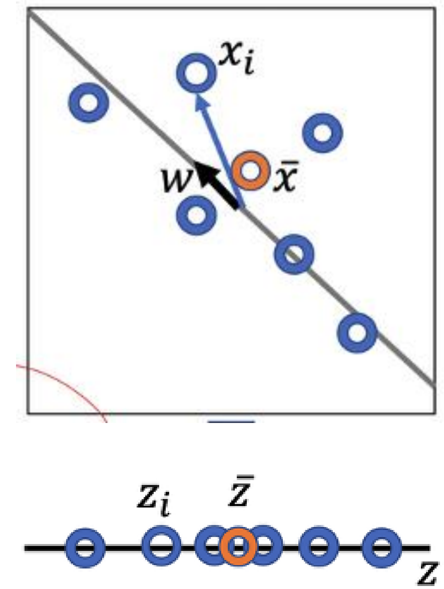


PCA – Linear Dimensionality Reduction

$$var = \frac{1}{N}\sum_{i=1}^{N}(w^T(x_i - \bar{x}))^2 = \frac{1}{N}\sum_{i=1}^{N} w^T(x_i - \bar{x})(x_i - \bar{x})^T w$$

$$= w^T\left(\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T\right)w$$

$$= w^T S w$$



Constrained Optimization

$$\max_{w} \quad w^T S w$$
$$s.t. \; w^T w = 1$$

KKT Condition

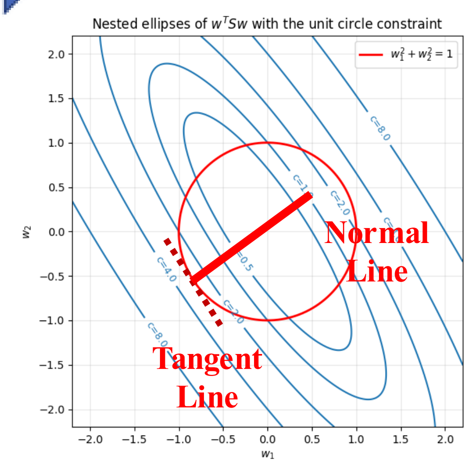
$$Sw = \lambda w$$

$$L(w, \alpha) = w^T S w + \alpha(w^T w - 1)$$

$$\nabla_w L(w, \alpha) = 2Sw + 2\alpha w = 0$$

$$\nabla_w L(w, \alpha) = Sw - \lambda w = 0$$

$$Sw = \lambda w$$



Solve Eigenvector

$$Sw = \lambda w$$

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

$$w_1, w_2, \ldots, w_n$$

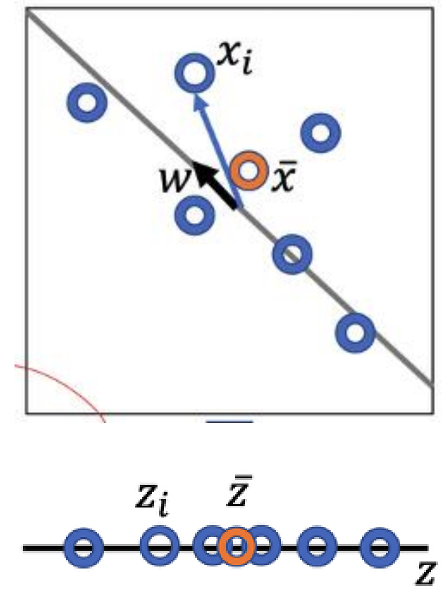
PCA – Linear Dimensionality Reduction

$$var = \frac{1}{N}\sum_{i=1}^{N}(w^T(x_i - \bar{x}))^2 = \frac{1}{N}\sum_{i=1}^{N} w^T(x_i - \bar{x})(x_i - \bar{x})^T w$$

$$= w^T\left(\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T\right)w$$

$$= w^T S w$$

$$var = w^T \lambda w = \lambda$$



Constrained Optimization

$$\max_{w} \quad w^T S w$$
$$s.t. \; w^T w = 1$$

KKT Condition

$$Sw = \lambda w$$

$$L(w, \alpha) = w^T S w + \alpha(w^T w - 1)$$

$$\nabla_w L(w, \alpha) = 2Sw + 2\alpha w = 0$$

$$\nabla_w L(w, \alpha) = Sw - \lambda w = 0$$

$$Sw = \lambda w$$

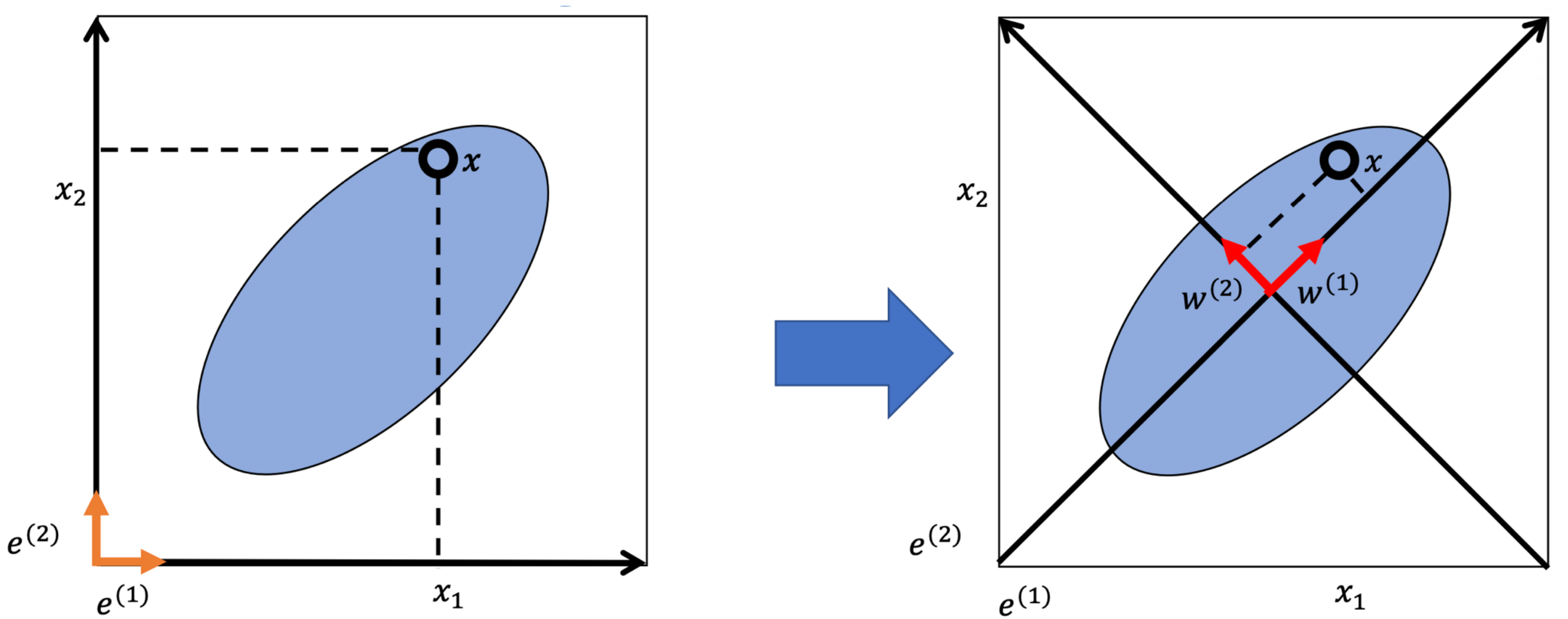
Solve Eigenvector

$$Sw = \lambda w$$

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

$$w_1, w_2, \ldots, w_n$$

PCA – Maximizing Variance



$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow x = x_1 \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{e^{(1)}} + x_2 \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{e^{(2)}}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow x = \bar{x} + \left(x^T w^{(1)}\right) w^{(1)} + \left(x^T w^{(2)}\right) w^{(2)}$$

PCA – Maximizing Variance

```
[11]: import torch
      import numpy as np
      import matplotlib.pyplot as plt
      from torchvision import datasets, transforms

      # Load MNIST dataset
      transform = transforms.ToTensor()
      mnist_data = datasets.MNIST(root='./data', train=True, download=True, transform=transform)
      images = mnist_data.data.float()
      labels = mnist_data.targets
```

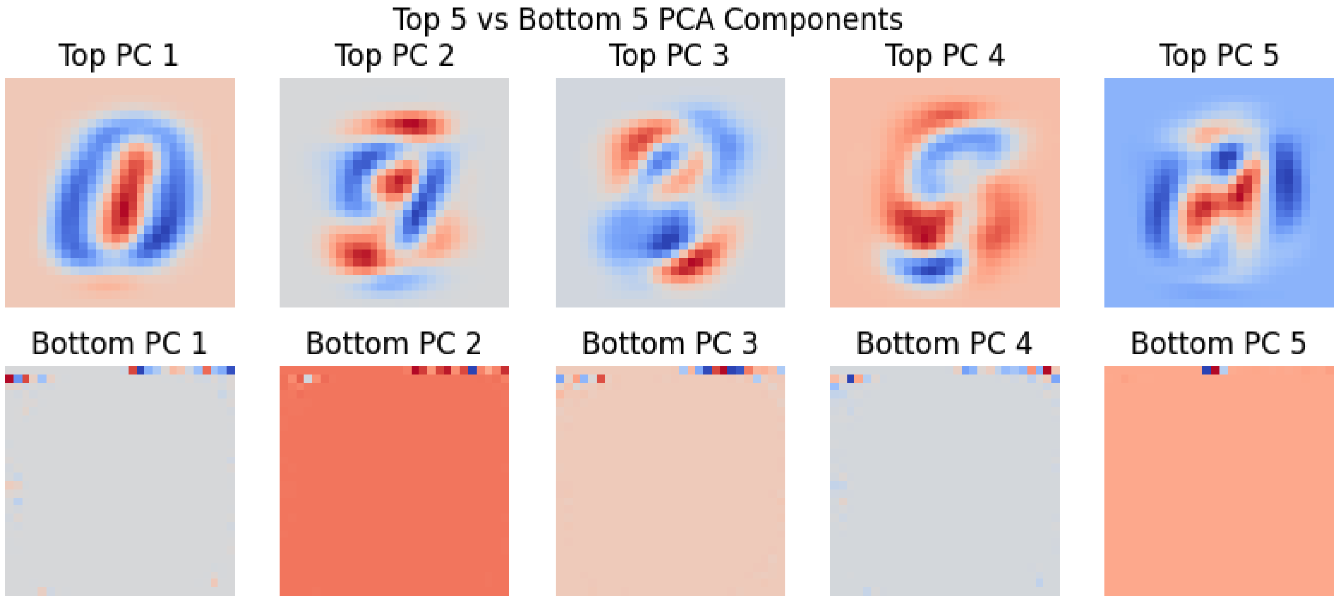
```
[13]: # Flatten images to vectors of size 784
      N, H, W = images.shape  # N=60000, H=28, W=28
      X = images.view(N, H*W)  # shape: (60000, 784)

      # Normalize data
      mean_image = X.mean(dim=0)
      X_centered = X - mean_image

      # Compute covariance matrix
      cov = (X_centered.T @ X_centered) / (N-1)

      # Eigen-decomposition
      eigenvalues, eigenvectors = torch.linalg.eigh(cov)
      # Sort eigenvalues and eigenvectors in descending order
      eigenvalues, indices = torch.sort(eigenvalues, descending=True)
      eigenvectors = eigenvectors[:, indices]
```

```
[14]: ### Visualization 1: Original MNIST Dataset
      fig, axes = plt.subplots(1, 10, figsize=(10, 2))
      for i in range(10):
          axes[i].imshow(X[i].reshape(28, 28), cmap='gray')
          axes[i].axis('off')
      plt.suptitle('Original MNIST Samples')
      plt.show()
```

Original MNIST Samples

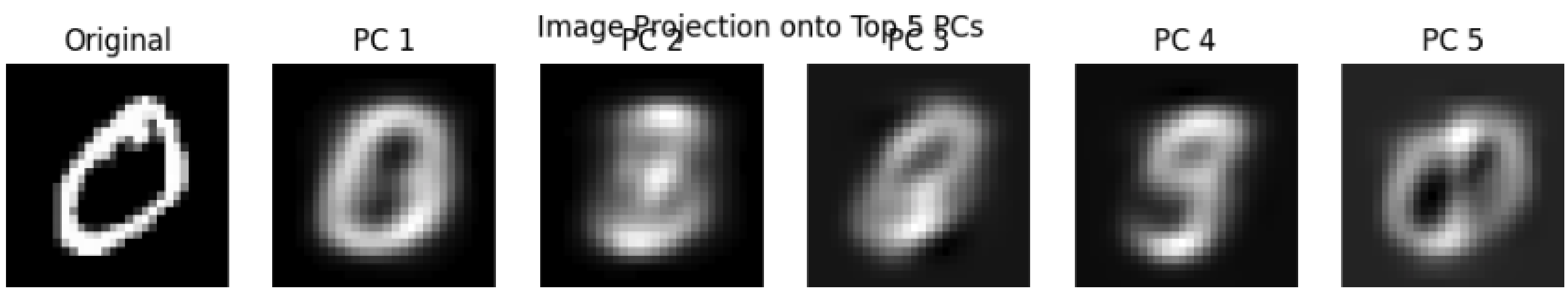
PCA – Maximizing Variance

```python
### Visualization 2: Top 5 and Bottom 5 PCA Components
fig, axes = plt.subplots(2, 5, figsize=(10, 4))
for i in range(5):
    top_comp = eigenvectors[:, i].reshape(28, 28)
    bottom_comp = eigenvectors[:, -i-1].reshape(28, 28)

    axes[0, i].imshow(top_comp, cmap='coolwarm')
    axes[0, i].set_title(f'Top PC {i+1}')
    axes[0, i].axis('off')

    axes[1, i].imshow(bottom_comp, cmap='coolwarm')
    axes[1, i].set_title(f'Bottom PC {i+1}')
    axes[1, i].axis('off')
plt.suptitle('Top 5 vs Bottom 5 PCA Components')
plt.show()
```



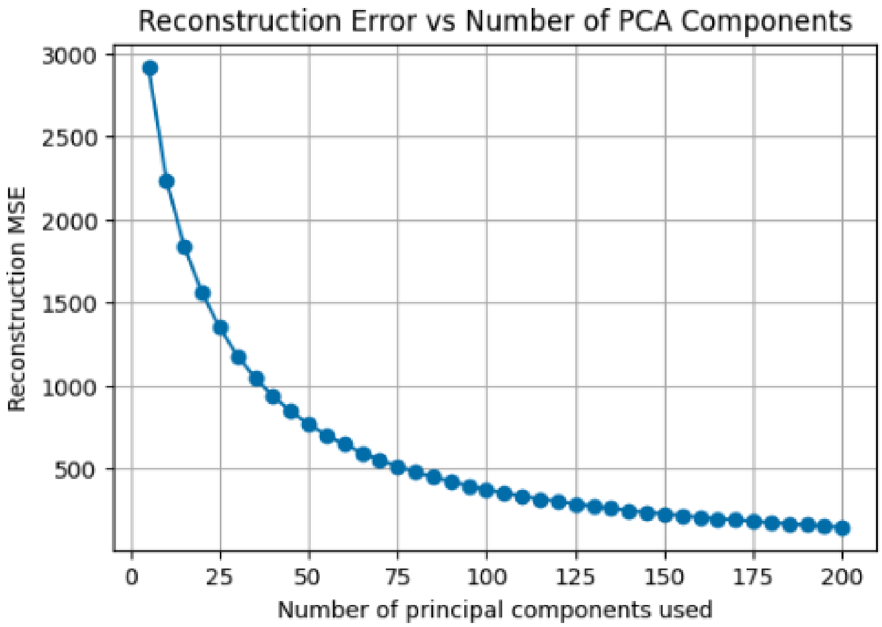
PCA – Maximizing Variance

```
### Visualization 4: Projecting a Specific Image onto PC1-5
sample_idx = 1  # Choose an image
test_image = X[sample_idx]
fig, axes = plt.subplots(1, 6, figsize=(12, 2))
axes[0].imshow(test_image.reshape(28, 28), cmap='gray')
axes[0].set_title('Original')
axes[0].axis('off')

for i in range(5):
    weight = torch.dot(test_image - mean_image, eigenvectors[:, i])
    recon = weight * eigenvectors[:, i] + mean_image
    axes[i+1].imshow(recon.reshape(28, 28), cmap='gray')
    axes[i+1].set_title(f'PC {i+1}')
    axes[i+1].axis('off')
plt.suptitle('Image Projection onto Top 5 PCs')
plt.show()
```



PCA – Maximizing Variance

```
### Visualization 5: Reconstruction Loss Curve
components_list = list(range(5, 201, 5))
errors = []
for k in components_list:
    X_recon_k = (X_centered @ eigenvectors[:, :k] @ eigenvectors[:, :k].T) + mean_image
    mse_k = torch.mean((X - X_recon_k) ** 2).item()
    errors.append(mse_k)

plt.figure(figsize=(6, 4))
plt.plot(components_list, errors, marker='o')
plt.title('Reconstruction Error vs Number of PCA Components')
plt.xlabel('Number of principal components used')
plt.ylabel('Reconstruction MSE')
plt.grid(True)
plt.show()
```



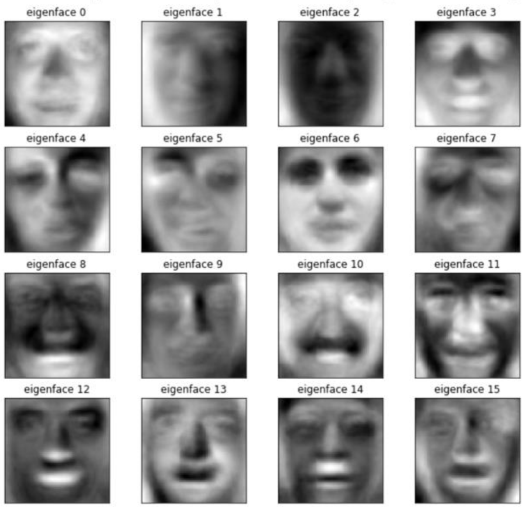
PCA – Maximizing Variance

- 64 by 64 images of celebrities
- Vectorized -> 4096-dimensional space
- Calculate PCA and visualize top PCs

Mean face

$\bar{x}$

Principle Components (Eigenfaces)



$w^{(i)}$

PCA – Maximizing Variance

Input x

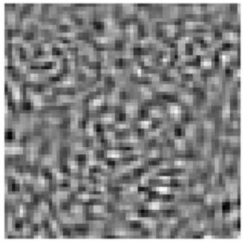
=

Mean $\bar{x}$

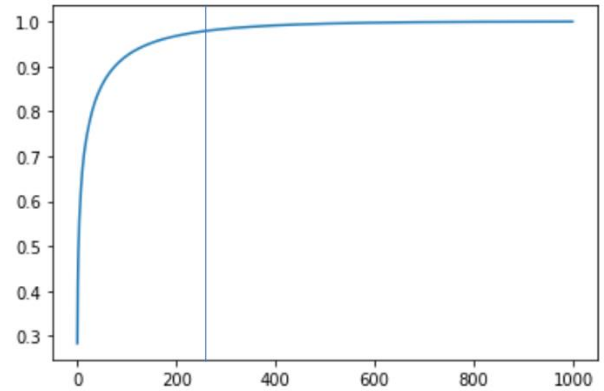
+ 341.6* $w^{(1)}$

- 12.7* $w^{(2)}$

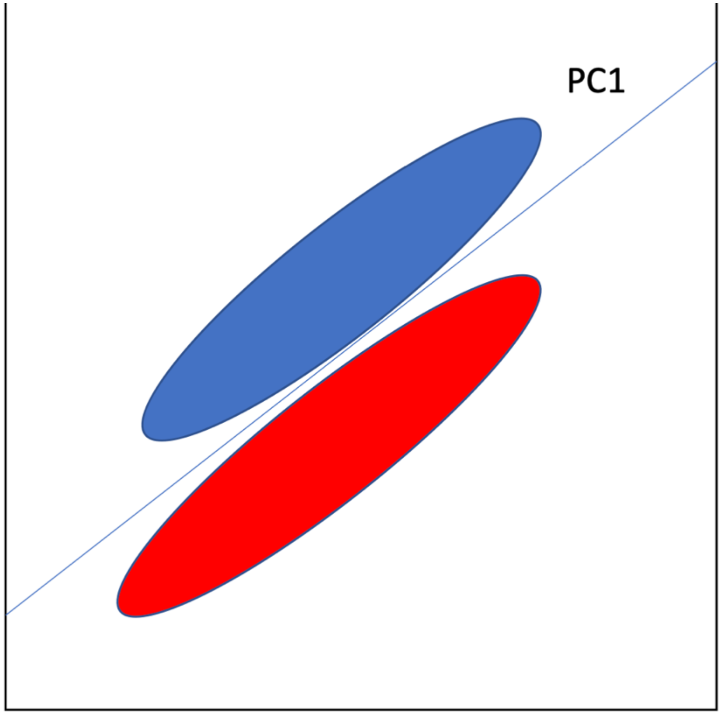
+... + 12.2* $w^{(1000)}$



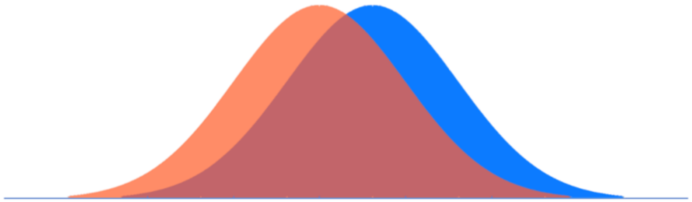
Reconstruction as a function of number of PC components



Problem with PCA



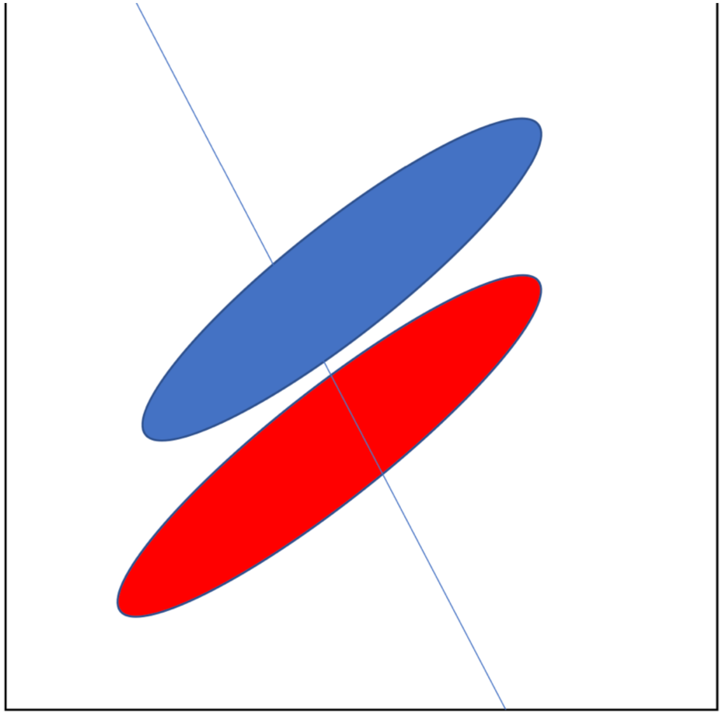
Reduction to 1D



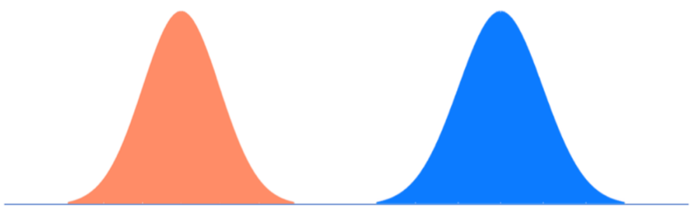
We lose the discriminatory feature!

PCA is unsupervised method

Problem with PCA



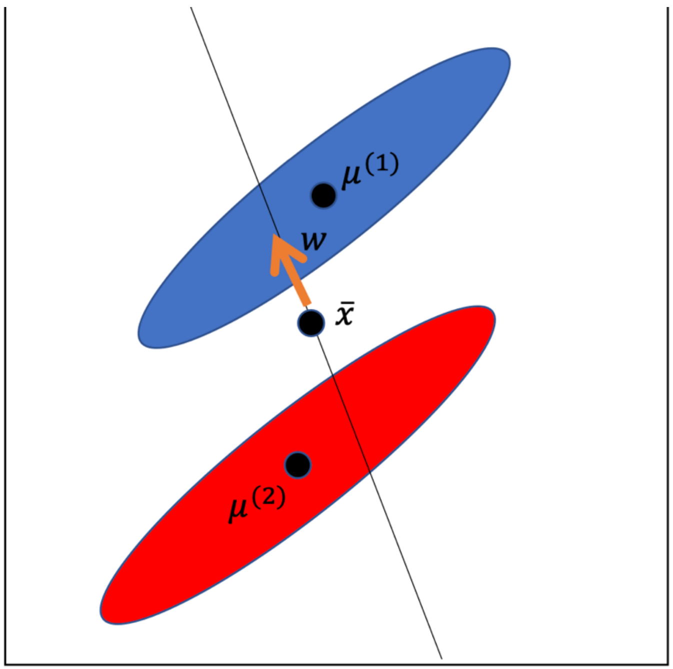
Reduction to 1D



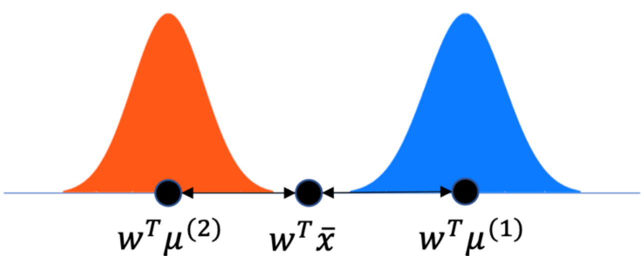
What would be a good reduction?

- Keeps the class means separate
- Keep he withing class variations small

Problem with PCA



Reduction to 1D



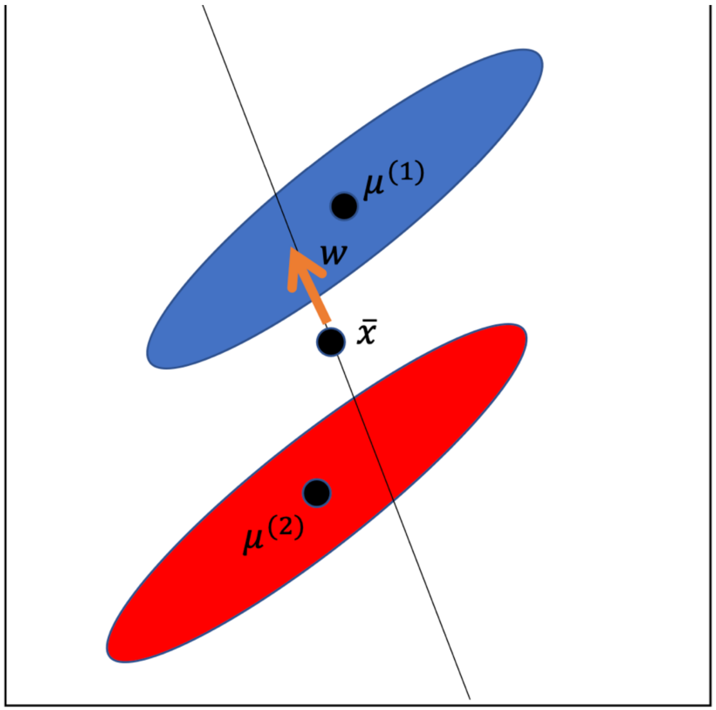
Distance between class means

$$\frac{1}{K}\sum_k \left(w^T\mu^{(k)} - w^T\bar{x}\right)^2 = \frac{1}{K}\sum_k \left(w^T\left(\mu^{(k)} - \bar{x}\right)\right)^2$$

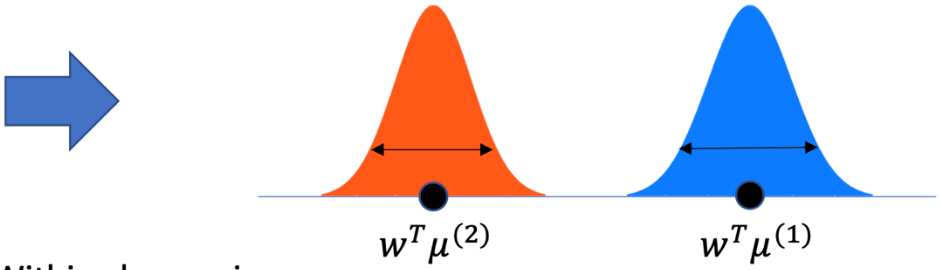
$$= w^T\left(\frac{1}{K}\sum_k \left(\mu^{(k)} - \bar{x}\right)\left(\mu^{(k)} - \bar{x}\right)^T\right)w$$

$$= w^T S_b w$$

Problem with PCA



Reduction to 1D



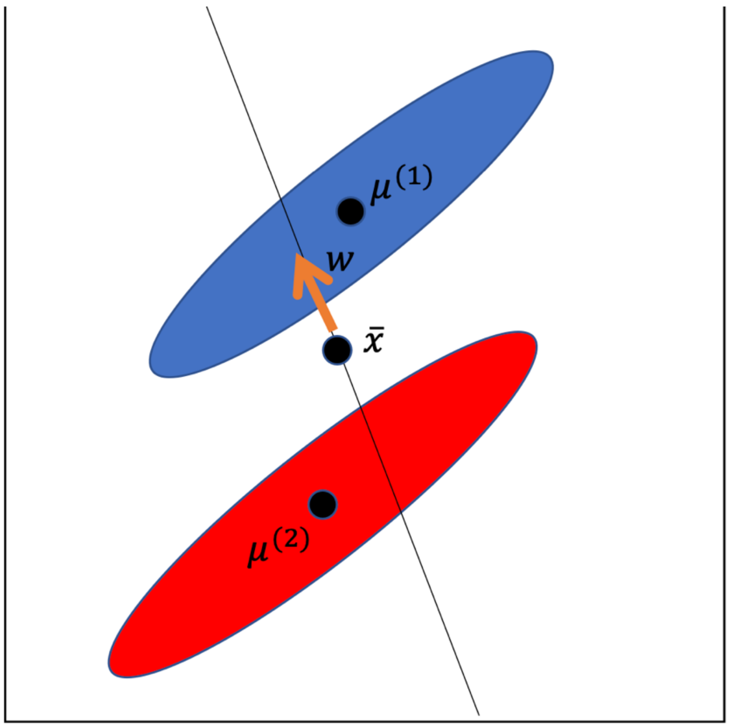
Within class variance

$$\frac{1}{N}\sum_{k}\sum_{i\in C_k}\left(w^T x^{(i)} - w^T\mu^{(k)}\right)^2$$

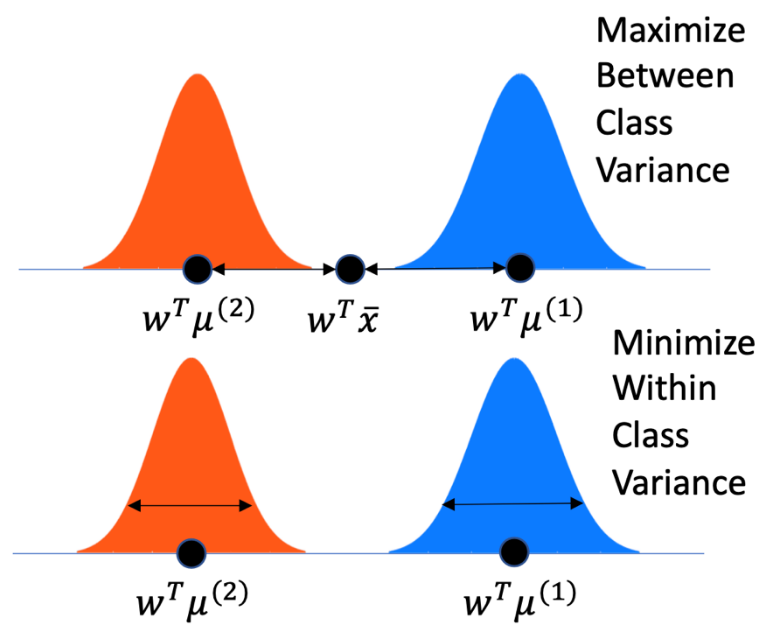
$$= w^T\left(\frac{1}{N}\sum_{k}\sum_{i\in C_k}\left(x^{(i)} - \mu^{(k)}\right)\left(x^{(i)} - \mu^{(k)}\right)^T\right)w$$

$$= w^T S_w w$$

Problem with PCA



Reduction to 1D



Question Time!