

Computer Vision: from Recognition to Geometry

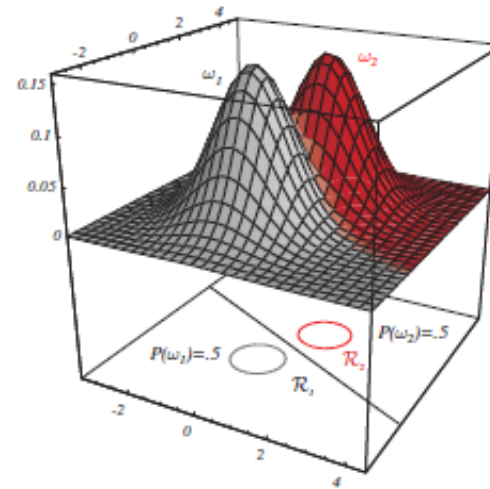
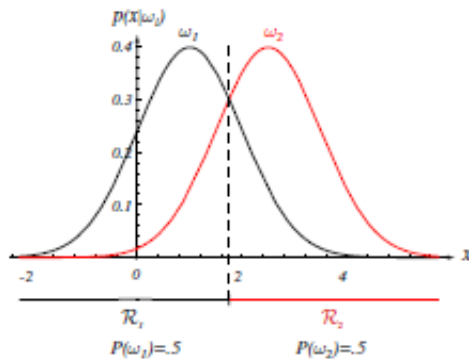
Lecture 4: Machine Learning 101

Yu-Chiang Frank Wang 王鈺強

Dept. Electrical Engineering, National Taiwan University

What's to Be Covered Today...

- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Clustering & Dimension Reduction
 - Training, testing, & validation
 - Linear Classification



Bayesian Decision Theory

- Fundamental statistical approach to classification/detection tasks
- Example (for a 2-class scenario):
 - Let's see if a student would **pass** or **fail** the course of DLCV.
 - Define a probabilistic variable ω describe the case of pass or fail.
 - That is, $\omega = \omega_1$ for pass, and $\omega = \omega_2$ for fail.
- **Prior Probability**
 - The **a priori** or **prior** probability reflects the knowledge of how likely we expect a certain state of nature before observation.
 - $P(\omega = \omega_1)$ or simply $P(\omega_1)$ as the prior that the next student would pass DLCV.
 - The priors must exhibit *exclusivity* and *exhaustivity*, i.e.,

Prior Probability (cont'd)

- Equal priors

- If we have *equal* numbers of students pass/fail DLCV, then the priors are equal; in other words, the priors are uniform.

- Decision rule based on priors only

- If the only available info is the prior, and the cost of any type of incorrect classification is equal, what would be a reasonable decision rule?
- Decide ω_1 if

otherwise decide ω_2 .

- What's the incorrect classification rate (or **error rate**) P_e ?

Class-Conditional Probability Density (or Likelihood)

- The **probability density function (PDF)** for input/observation \mathbf{x} given a state of nature ω is written as:

- Here's (hopefully) the hypothetical class-conditional densities reflecting the studying time of students who pass/fail DLCV.

Posterior Probability & Bayes Formula

- If we know the prior distribution and the class-conditional density, can we come up with a better decision rule?
 - Yes We Can! By calculating the posterior probability.
- Posterior probability $P(\omega|\mathbf{x})$:
 - The probability of a certain state of nature ω given an observable \mathbf{x} .
- Bayes formula:
 $P(\omega_j, \mathbf{x})$

$$P(\omega_j|\mathbf{x})$$

And, we have $\sum_{j=1}^C P(\omega_j|\mathbf{x}) = 1$.

Decision Rule & Probability of Error

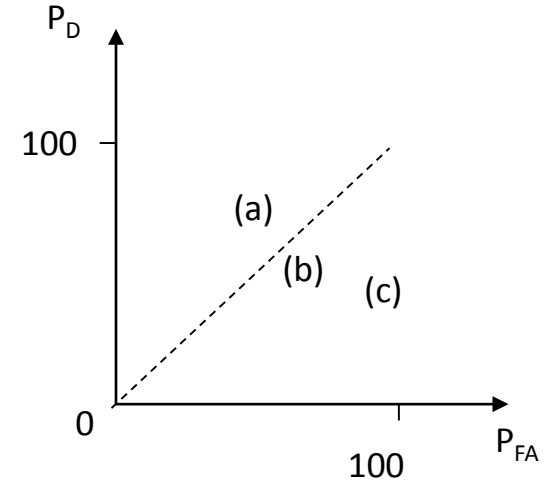
- For a given observable \mathbf{x} , the decision rule will be now based on:
- What's the probability of error $P(\text{error})$ (or P_e)?

From Bayes Decision Rule to Detection Theory

- Hit (or detection), false alarm, miss (or false reject), correct & false rejection

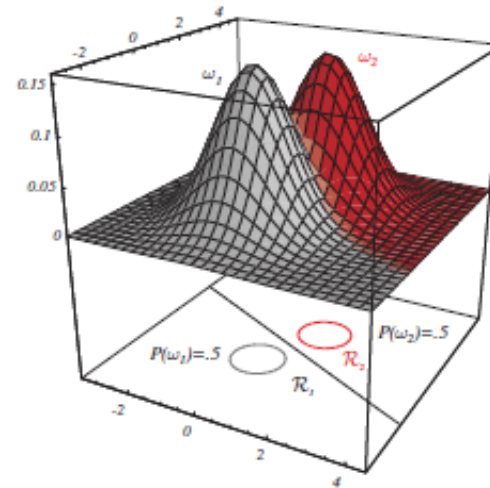
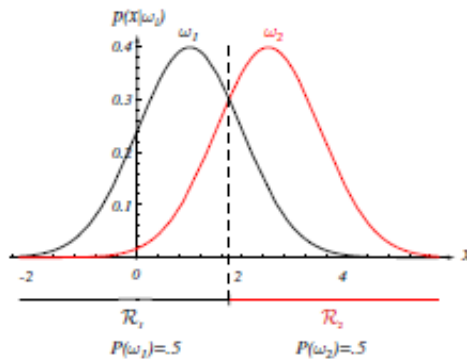
From Bayes Decision Rule to Detection Theory

- Receiver Operating Characteristics (ROC)
 - To assess the effectiveness of the designed features/classifiers/detectors
 - False alarm (P_{FA}) vs. detection (P_D) rates
 - Which curve/line makes sense? (a), (b), or (c)?

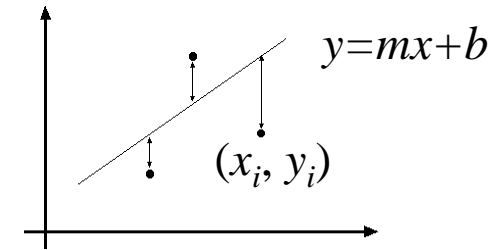


What's to Be Covered Today...

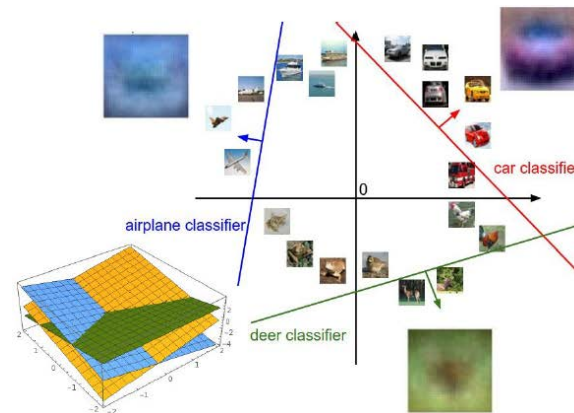
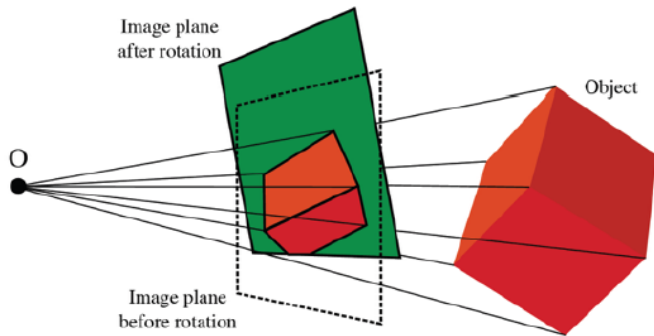
- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Dimension Reduction & Clustering
 - Linear Classification
 - Training, testing, & validation



Why Review Linear Systems?



- Aren't DL models considered to be non-linear?
- Yes, but there are lots of things (e.g., problem setting, feature representation, regularization, etc.) in the fields of learning and vision starting from linear formulations.



$$f(x, W) = Wx + b$$



Array of $32 \times 32 \times 3$ numbers
(3072 numbers total)

Rank of a Matrix

- Consider \mathbf{A} as a $m \times n$ matrix, the rank of matrix \mathbf{A} , or $\text{rank}(\mathbf{A})$, is determined as the **maximum number of linearly independent row/column vectors**.
 - Thus, we have $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^T)$ and $\text{rank}(\mathbf{A}) \leq \text{or } \geq (?) \min(m, n)$.
- If $\text{rank}(\mathbf{A}) = \min(m, n)$, \mathbf{A} is a **full-rank** matrix.
- If $m = n = \text{rank}(\mathbf{A})$ (i.e., \mathbf{A} is a squared matrix and full-rank), what can we say about \mathbf{A}^{-1} ?
- If $m = n$ but $\text{rank}(\mathbf{A}) < m$, what can we do to compute \mathbf{A}^{-1} in practice?

Solutions to Linear Systems

- Let $\mathbf{A} \mathbf{x} = \mathbf{b}$, where \mathbf{A} is a $m \times n$ matrix, \mathbf{x} is $n \times 1$ vector (to be determined), and \mathbf{b} is a $m \times 1$ vector (observation).
- We consider $\mathbf{A} \mathbf{x} = \mathbf{b}$ as a linear system with m equations & n unknowns.

- If $m = n$ & $\text{rank}(\mathbf{A}) = m$, we can solve \mathbf{x} by Gauss Elimination, or simply $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. (\mathbf{A}^{-1} is the **inverse** matrix of \mathbf{A}).
- For a matrix $\mathbf{A} = \mathbf{x}\mathbf{x}^T$, what is the rank of \mathbf{A} ?

Pseudoinverse of a Matrix (I)

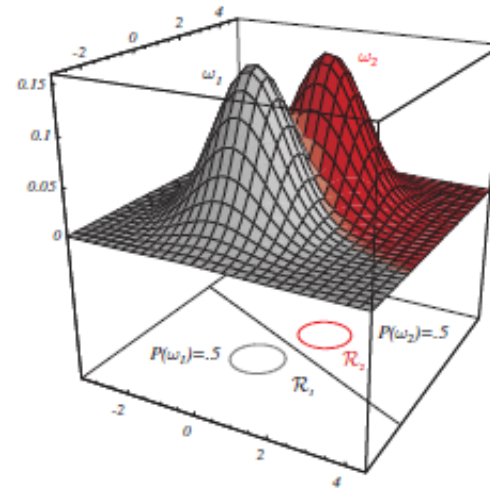
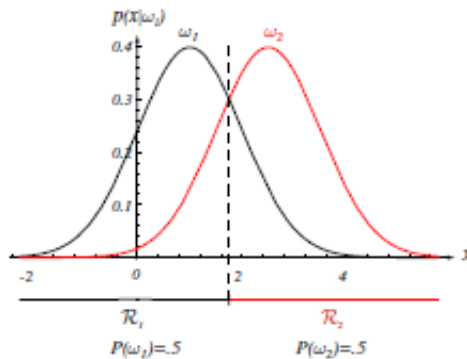
- Given $\mathbf{A} \mathbf{x} = \mathbf{b}$, if $m < n$ & $\text{rank}(\mathbf{A}) = m$...
 - We have more # of unknowns than # of equations, i.e., **underdetermined**.
 - Which \mathbf{x} is typically desirable?
 - Ever heard of the optimization technique of **Lagrange multipliers**?

Pseudoinverse of a Matrix (II)

- Given $\mathbf{A} \mathbf{x} = \mathbf{b}$, if $m > n$ & $\text{rank}(\mathbf{A}) = n$...
 - We have more # of equations than # of unknowns, i.e., **overdetermined**.
 - How to get a desirable \mathbf{x} ?

What's to Be Covered Today...

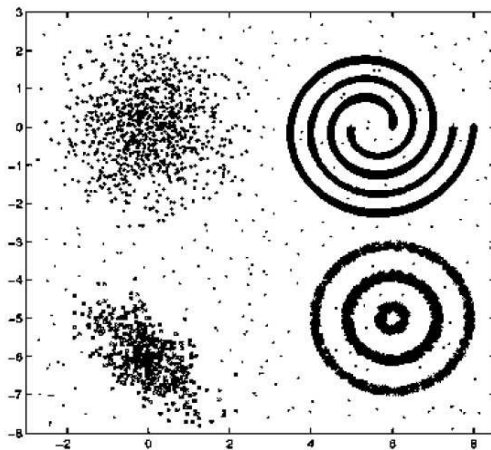
- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Clustering
 - Unsup. vs. Sup. Dimension Reduction
 - Training, testing, & validation
 - Linear Classification



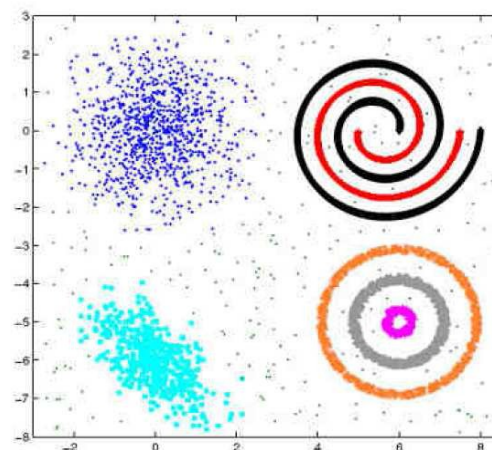
Clustering



- Clustering is an unsupervised algorithm.
 - Given:
 - a set of N unlabeled instances $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$; # of clusters K
 - Goal: group the samples into K partitions
- Remarks:
 - High within-cluster (intra-cluster) similarity
 - Low between-cluster (inter-cluster) similarity
 - But...how to determine a proper similarity measure?



(a) Input data



(b) Desired clustering

Similarity is NOT Always Objective...



Clustering (cont'd)

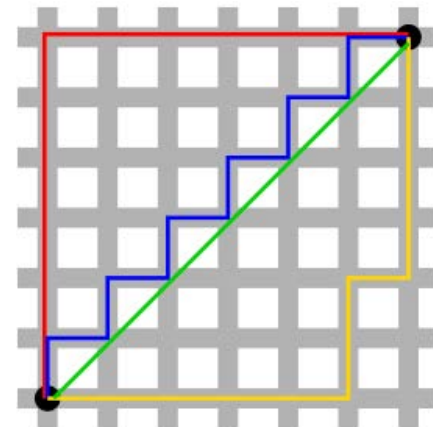
- Similarity:

- A key component/measure to perform data clustering
- **Inversely proportional** to distance
- Example distance metrics:

- Euclidean distance (L2 norm): $d(x, z) = \|x - z\|_2 = \sqrt{\sum_{i=1}^D (x_i - z_i)^2}$

- Manhattan distance (L1 norm): $d(x, z) = \|x - z\|_1 = \sum_{i=1}^D |x_i - z_i|$

- Note that p -norm of x is denoted as:



Clustering (cont'd)

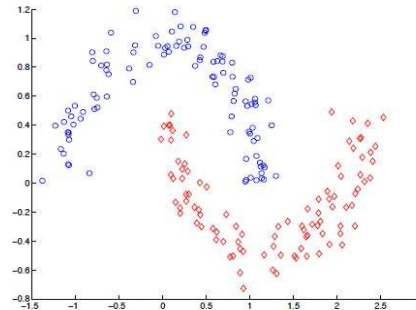
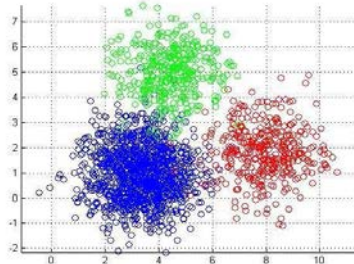
- Similarity:
 - A key component/measure to perform data clustering
 - Inversely proportional to distance
- Example distance metrics:
 - Kernelized (non-linear) distance:

$$d(x, z) = \|\Phi(x) - \Phi(z)\|_2^2 = \|\Phi(x)\|_2^2 + \|\Phi(z)\|_2^2 - 2\Phi(x)^T \Phi(z)$$

- Taking Gaussian kernel for example: $K(x, z) = \Phi(x)^T \Phi(z) = \exp\left(-\frac{\|x-z\|_2^2}{2\sigma^2}\right)$, we have

And, distance is more sensitive to **larger/smaller** σ . Why?

- For example, L2 or kernelized distance metrics for the following two cases?

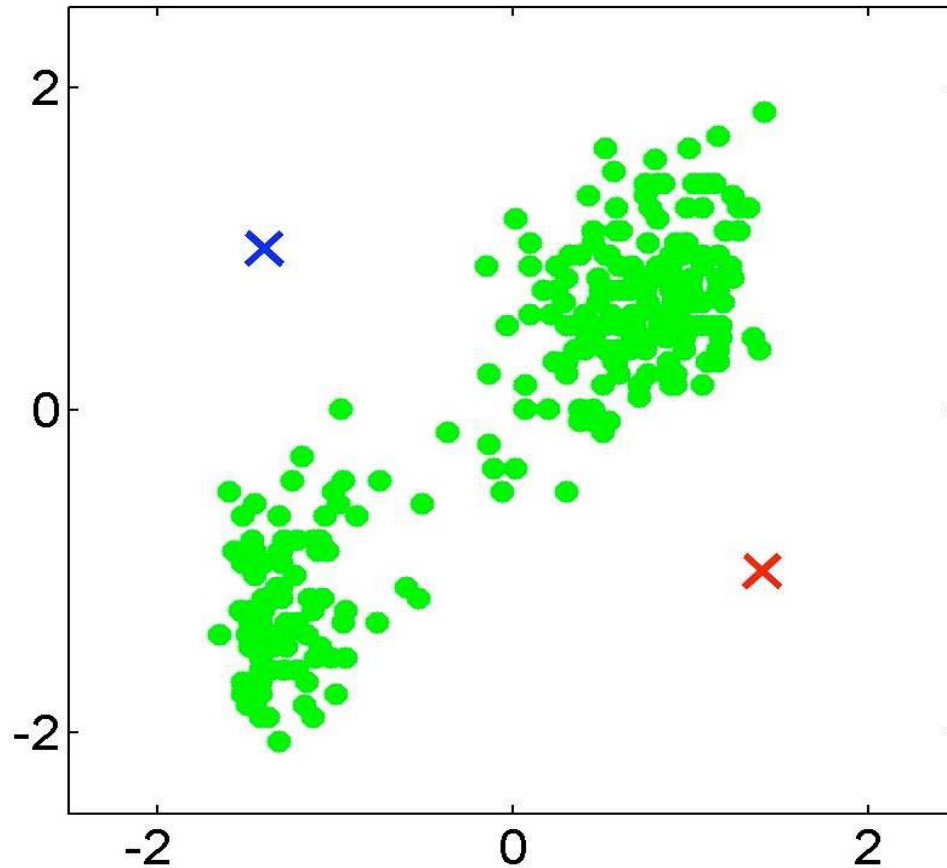


K-Means Clustering

- **Input:** N examples $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ ($\mathbf{x}_n \in \mathbb{R}^D$); number of partitions K
- **Initialize:** K cluster centers μ_1, \dots, μ_K . Several initialization options:
 - Randomly initialize μ_1, \dots, μ_K anywhere in \mathbb{R}^D
 - Or, simply choose any K examples as the cluster centers
- **Iterate:**
 - Assign each of example \mathbf{x}_n to its closest cluster center
 - Recompute the new cluster centers μ_k (mean/centroid of the set C_k)
 - Repeat while not converge
- **Possible convergence criteria:**
 - Cluster centers do not change anymore
 - Max. number of iterations reached
- **Output:**
 - K clusters (with centers/means of each cluster)

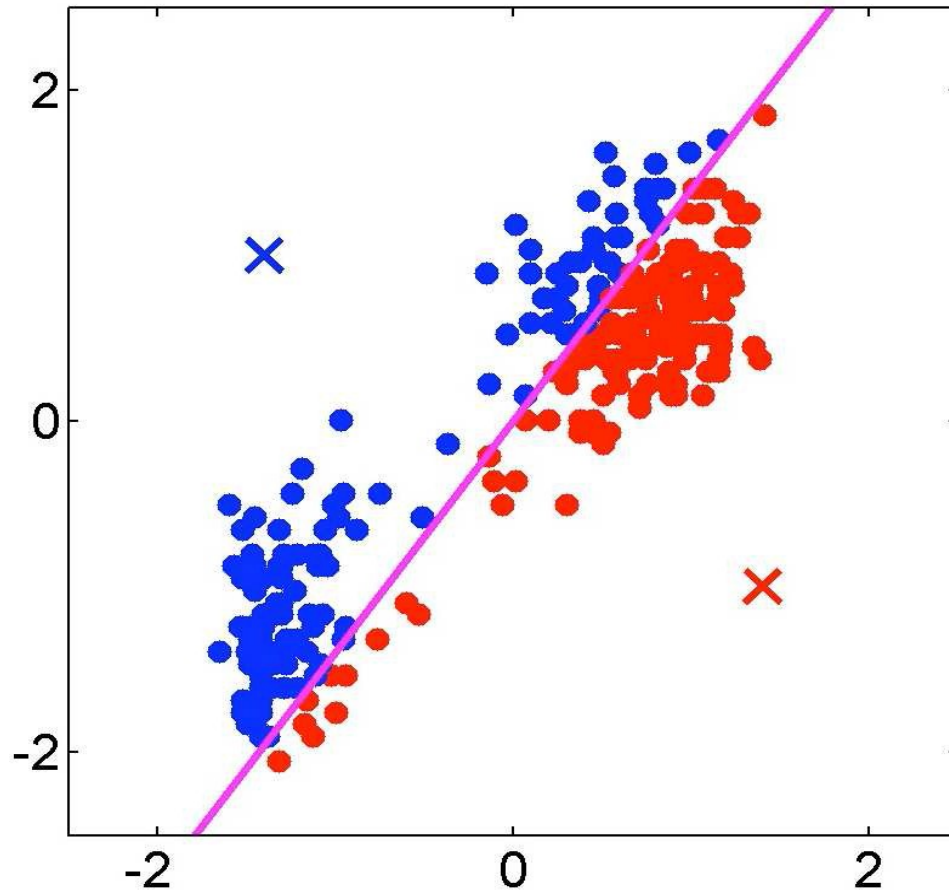
K-Means Clustering

- Example ($K = 2$): Initialization, iteration #1: pick cluster centers



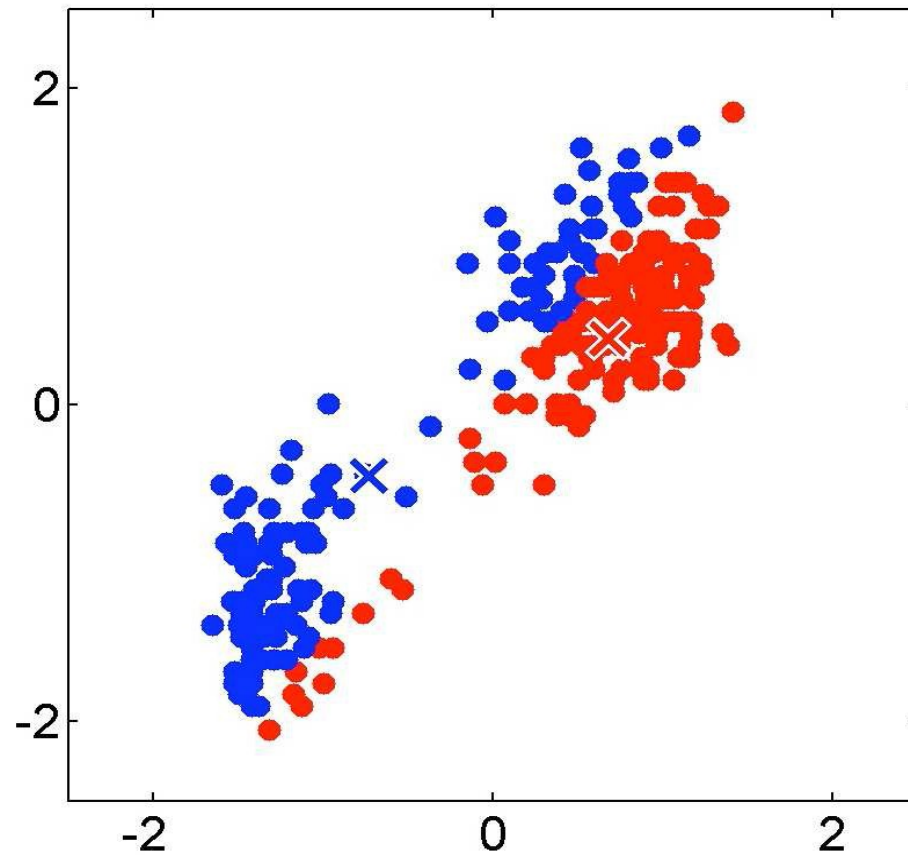
K-Means Clustering

- Example ($K = 2$): iteration #1-2, assign data to each cluster



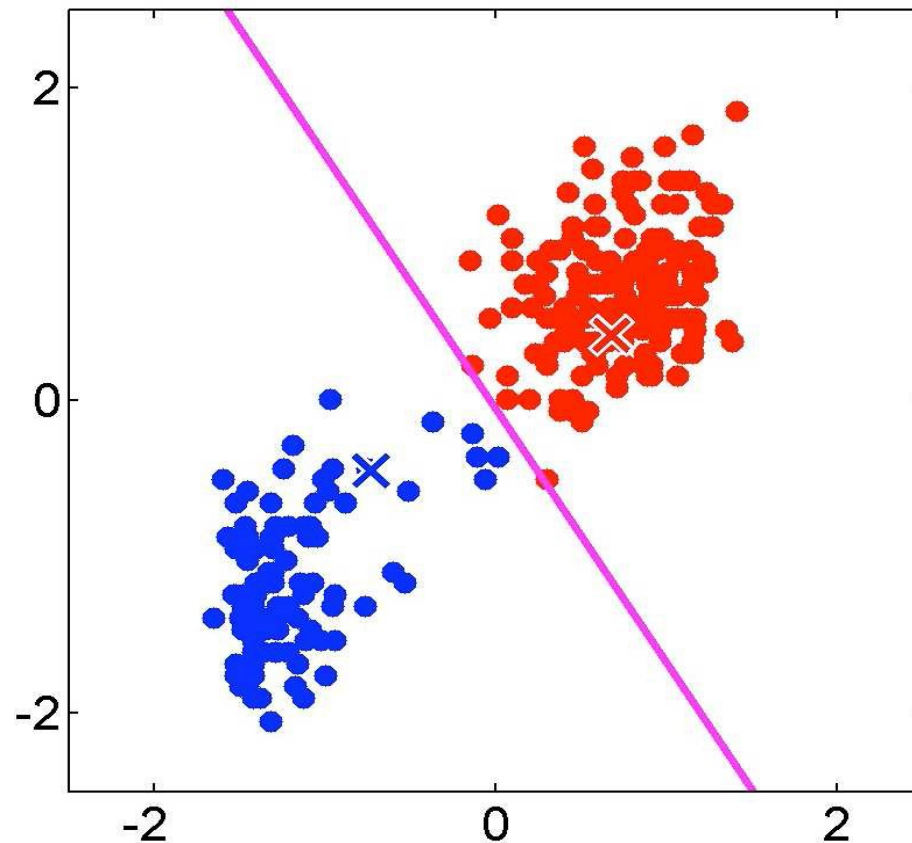
K-Means Clustering

- Example ($K = 2$): iteration #2-1, update cluster centers



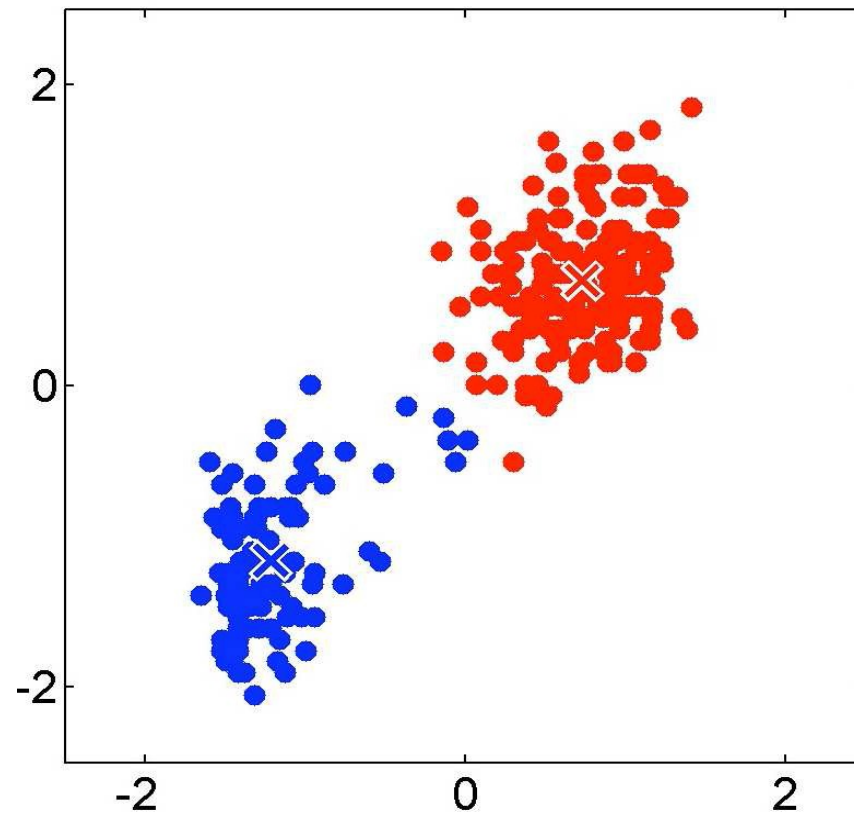
K-Means Clustering

- Example ($K = 2$): iteration #2, assign data to each cluster



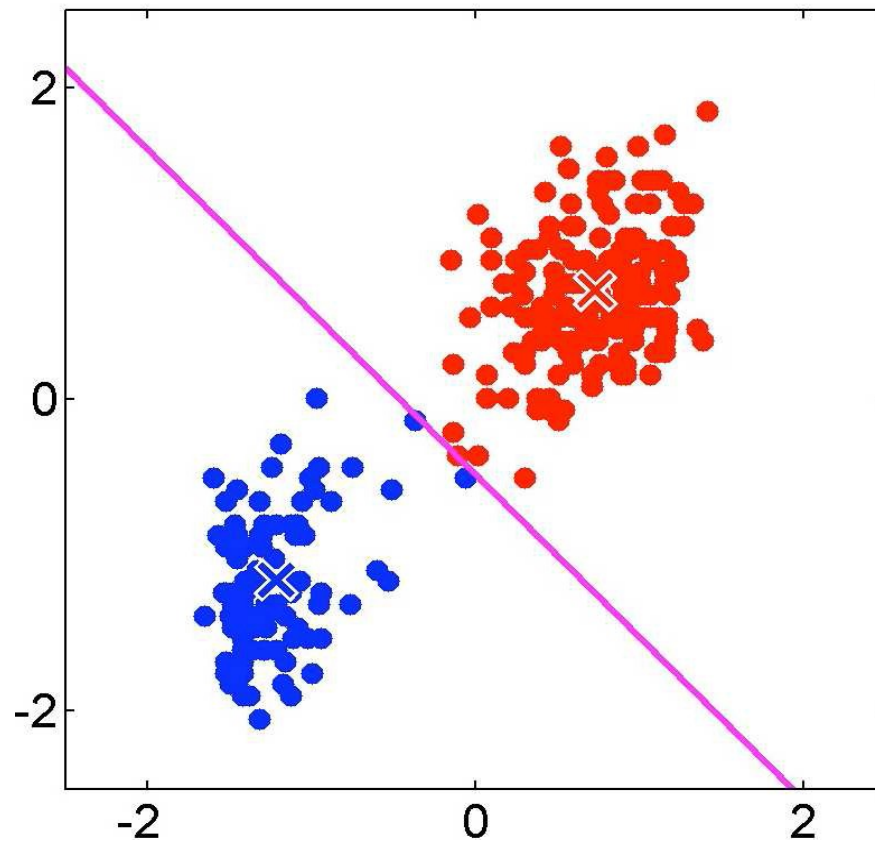
K-Means Clustering

- Example (K = 2): iteration #3-1



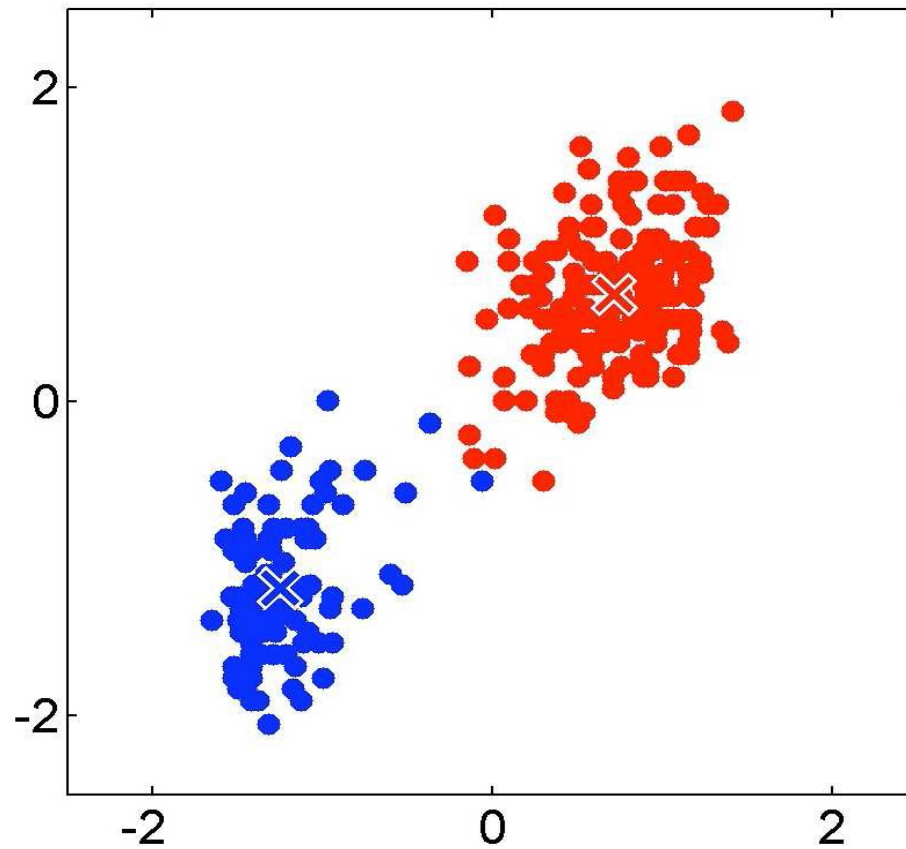
K-Means Clustering

- Example (K = 2): iteration #3-2



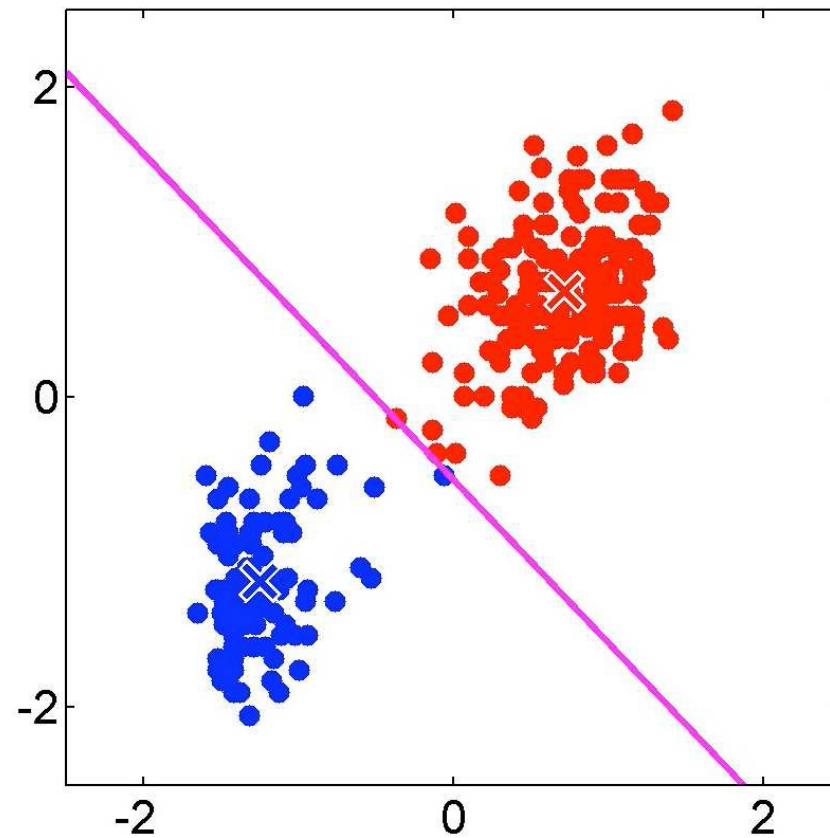
K-Means Clustering

- Example ($K = 2$): iteration #4-1



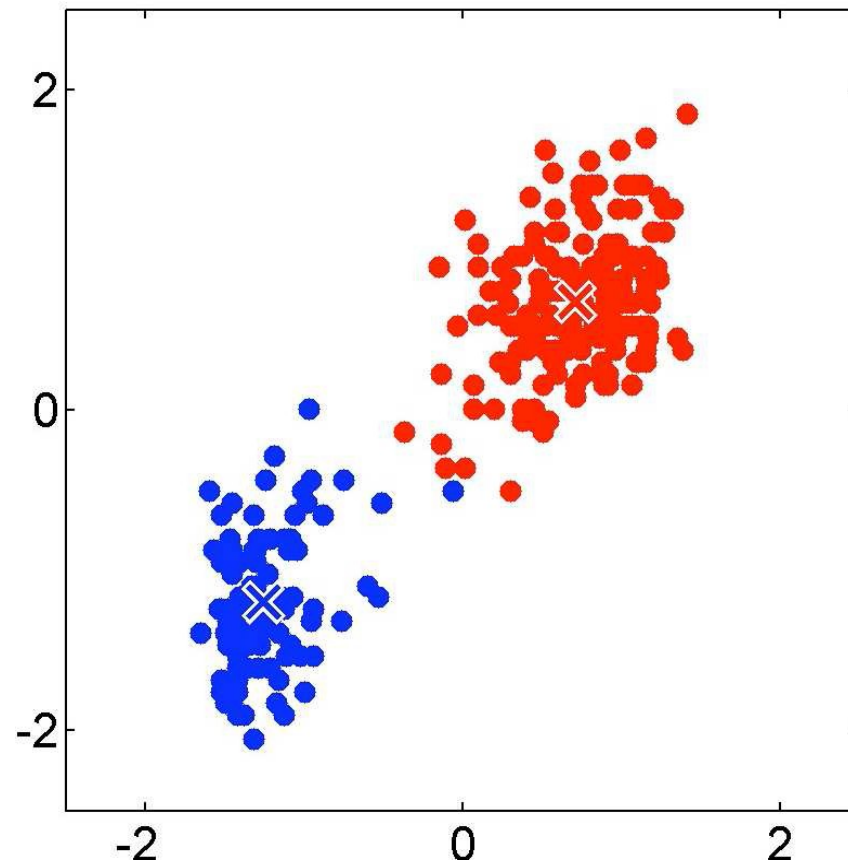
K-Means Clustering

- Example (K = 2): iteration #4-2



K-Means Clustering

- Example ($K = 2$): iteration #5, cluster means are not changed.

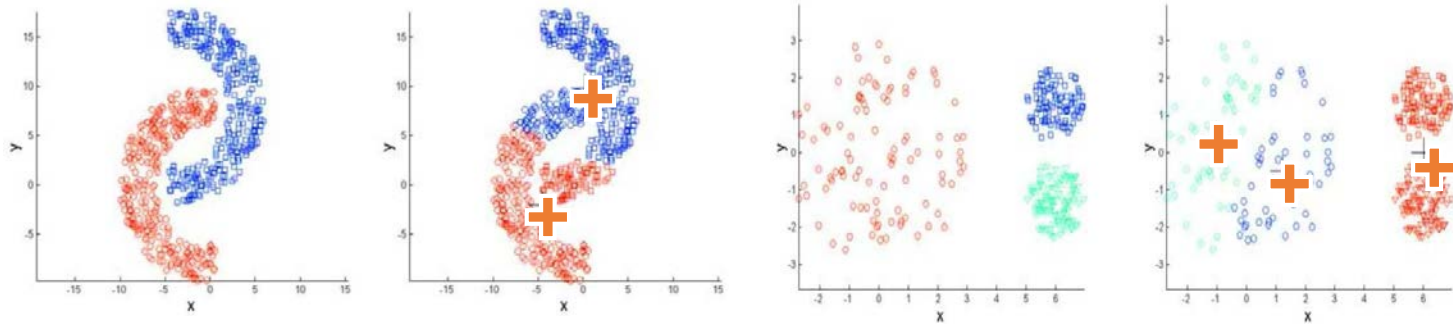


K-Means Clustering (cont'd)

- Proof in 1-D case (if time permits).

K-Means Clustering (cont'd)

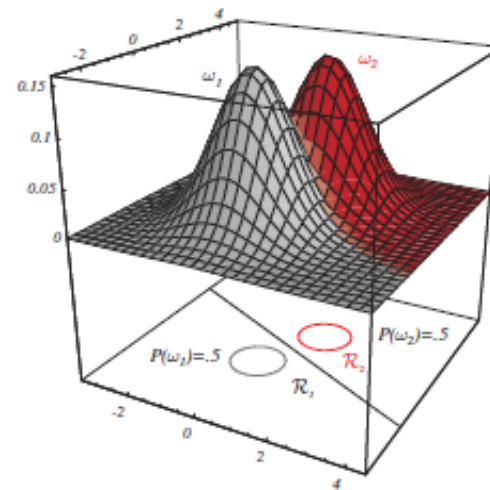
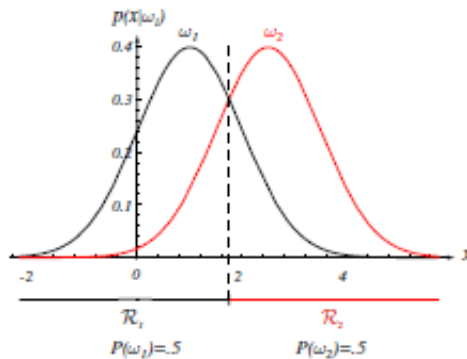
- Limitation
 - Sensitive to initialization; how to alleviate this problem?
 - Sensitive to outliers; possible change from K-means to...
 - Hard assignment only. Mathematically, we have...
- Preferable for round shaped clusters with similar sizes



- Remarks
 - Speed-up possible by hierarchical clustering
 - Expectation–maximization (EM) algorithm

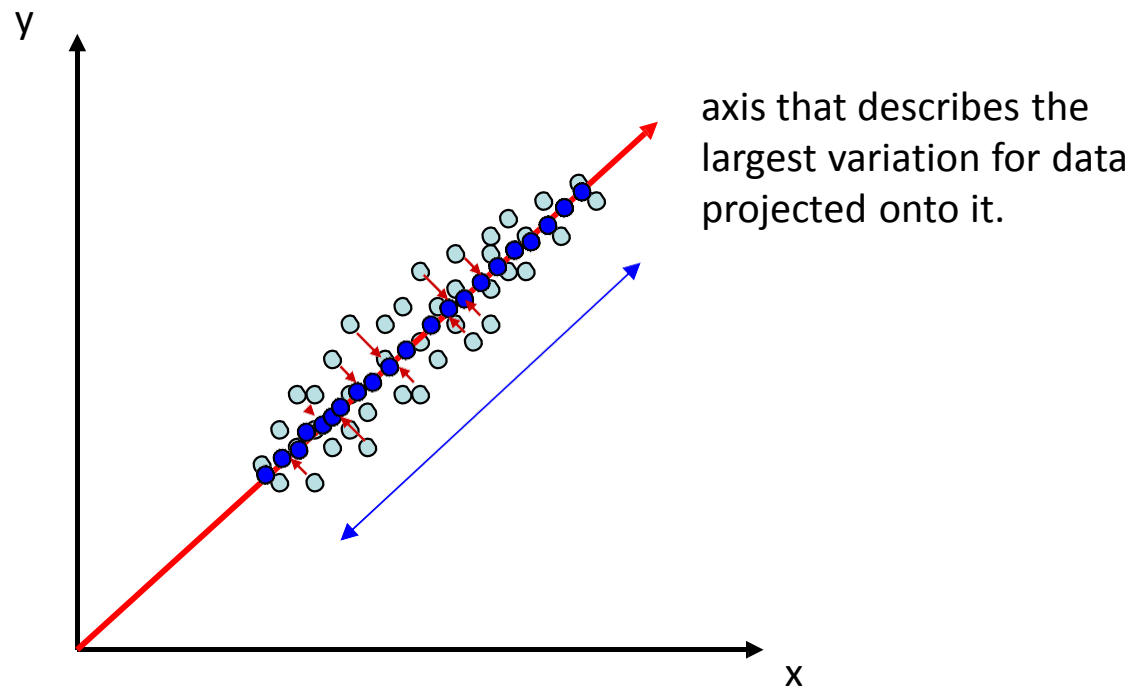
What's to Be Covered Today...

- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Clustering
 - Unsup. vs. Sup. Dimension Reduction
 - Training, testing, & validation
 - Linear Classification



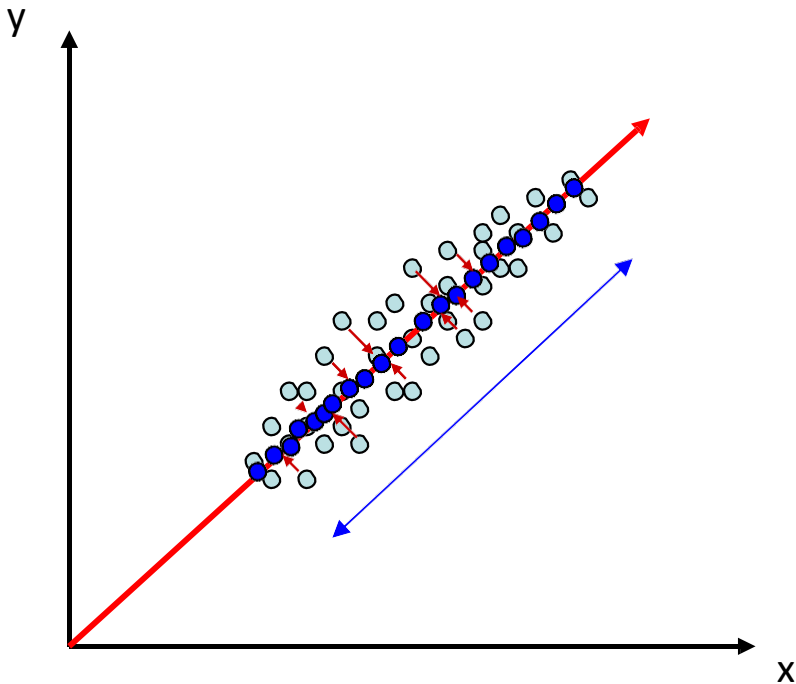
Unsupervised Dimension Reduction

- Principal Component Analysis (PCA)
 - Unsupervised & linear dimension reduction
 - Related to Eigenfaces, etc. feature extraction and classification techniques
 - Still very popular despite of its simplicity and effectiveness.
 - Goal:
 - Determine the projection, so that the variation of projected data is maximized.



Formulation & Derivation for PCA

- Input: a set of instances \mathbf{x} without label info
- Output: a projection vector $\boldsymbol{\omega}$ maximizing the variance of the projected data
- In other words, we need to maximize $\text{var}(\boldsymbol{\omega}^T \mathbf{x})$ with $\|\boldsymbol{\omega}\| = 1$.



Formulation & Derivation for PCA (cont'd)

- Lagrangian optimization for PCA

Eigenanalysis & PCA

- Eigenanalysis for PCA...find the eigenvectors \mathbf{e}_i and the corresponding eigenvalues λ_i
 - In other words, the direction \mathbf{e}_i captures the variance of λ_i .
 - But, which eigenvectors to use though? All of them?
- A $d \times d$ covariance matrix contains a maximum of d eigenvector/eigenvalue pairs.
 - Do we need to compute all of them? Which \mathbf{e}_i and λ_i pairs to use?
 - Assuming you have N images of size $M \times M$ pixels, we have dimension $d = M^2$.
 - What is the rank of Σ ?
 - Thus, at most $\min(N, d)$ non-zero eigenvalues can be obtained.
 - How dimension reduction is realized? how to reconstruct the input data?

Eigenanalysis & PCA (cont'd)

- A $d \times d$ covariance matrix contains a maximum of d eigenvector/eigenvalue pairs.
 - Assuming you have N images of size $M \times M$ pixels, we have dimension $d = M^2$.
 - With the rank of Σ as r , we have at most r non-zero eigenvalues.
 - How dimension reduction is realized? how to reconstruct the input data?

- Expanding a signal via eigenvectors as bases
 - With symmetric matrices (e.g., covariance matrix), eigenvectors are orthogonal.
 - They can be regarded as unit basis vectors to span any instance in the d -dim space.

Practical Issues in PCA

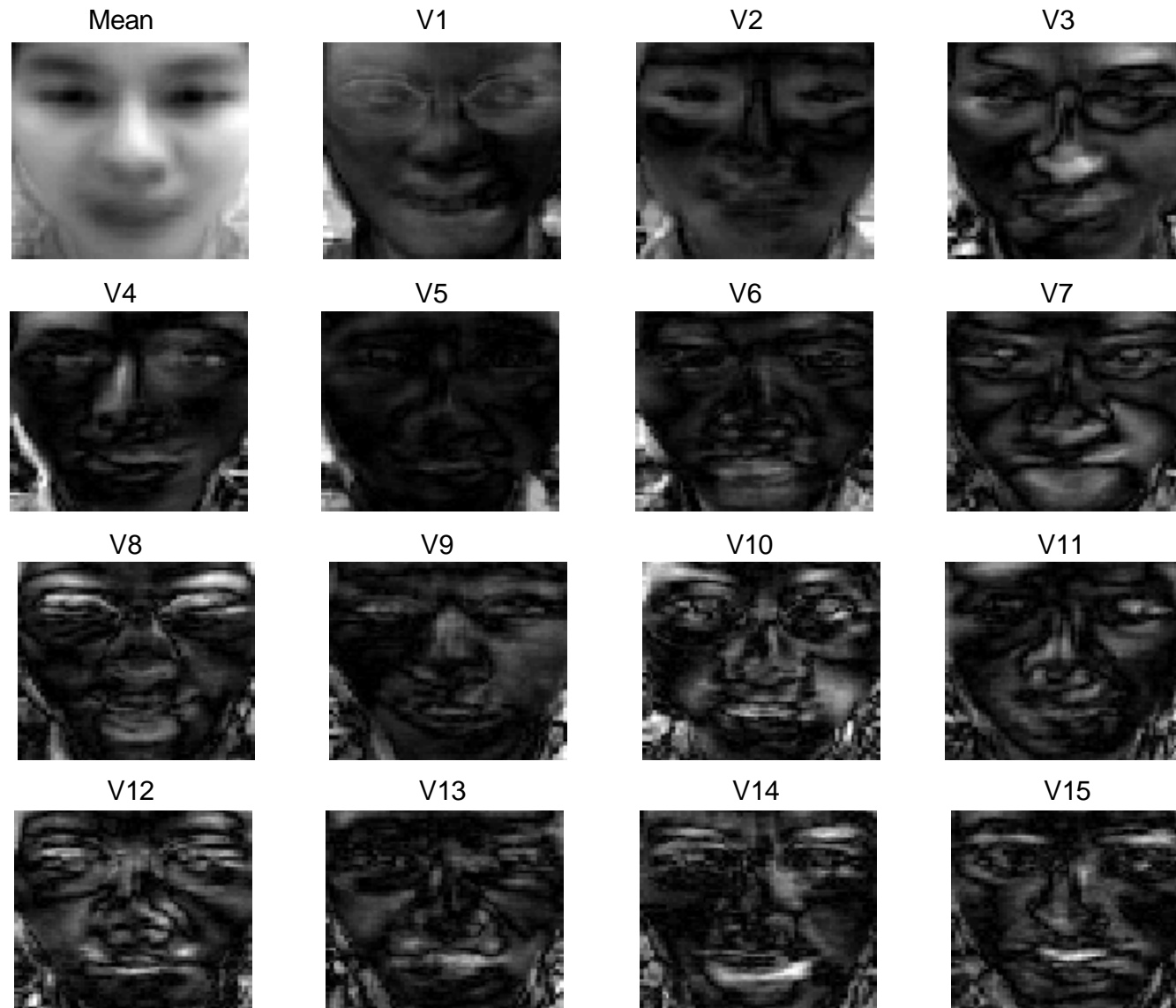
- Assume we have $N = 100$ images of size 200×200 pixels (i.e., $d = 40000$).
- What is the size of the covariance matrix? What's its rank?
- What can we do? **Gram Matrix Trick!**

Let's See an Example (CMU AMP Face Database)

- Let's take 5 face images x 13 people = 65 images, each is of size $64 \times 64 = 4096$ pixels.
- # of eigenvectors are expected to use for perfectly reconstructing the input = **64**.
- Let's check it out!



What Do the Eigenvectors/Eigenfaces Look Like?



All 64 Eigenvectors, do we need them all?



Use only 1 eigenvector, MSE = 1233

MSE=1233.16



Use 2 eigenvectors, MSE = 1027

MSE=1027.63



Use 3 eigenvectors, MSE = 758

MSE=758.13



Use 4 eigenvectors, MSE = 634

MSE=634.54



Use 8 eigenvectors, MSE = 285

MSE=285.08



With 20 eigenvectors, MSE = 87

MSE=87.93



With 30 eigenvectors, MSE = 20

MSE=20.55



With 50 eigenvectors, MSE = 2.14

MSE=2.14



With 60 eigenvectors, MSE = 0.06

MSE=0.06



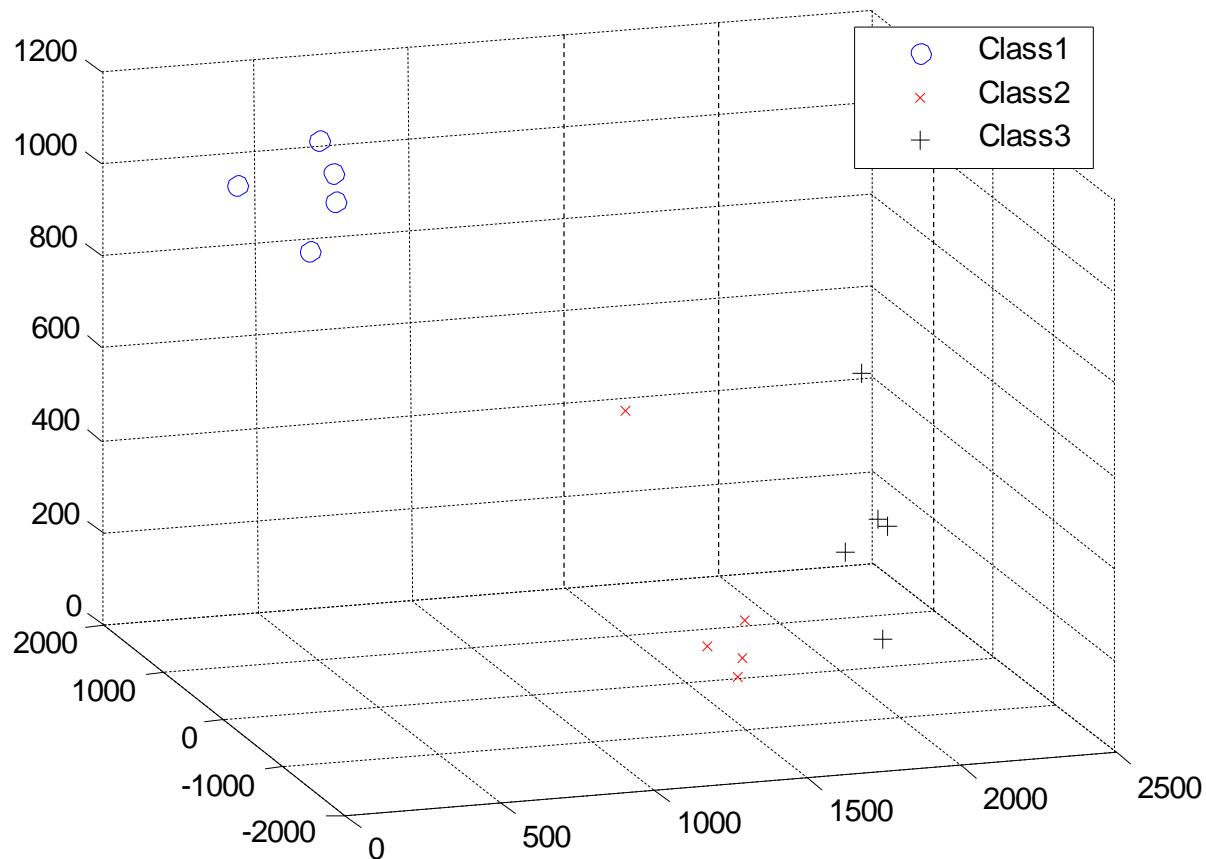
All 64 eigenvectors, MSE = 0

MSE=0.00



Final Remarks

- Linear & unsupervised dimension reduction
- PCA can be applied as a feature extraction/preprocessing technique.
 - E.g., Use the top 3 eigenvectors to project data into a 3D space for classification.

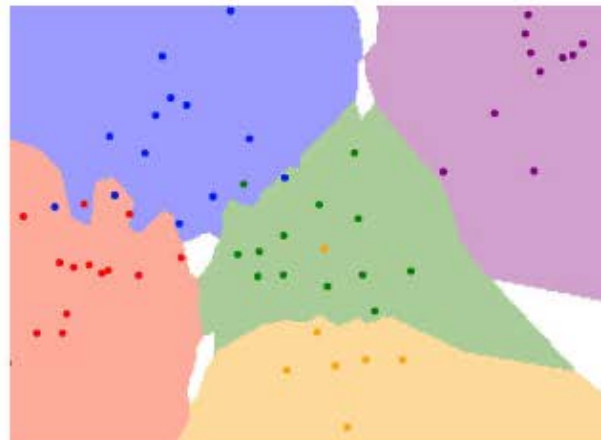


Final Remarks (cont'd)

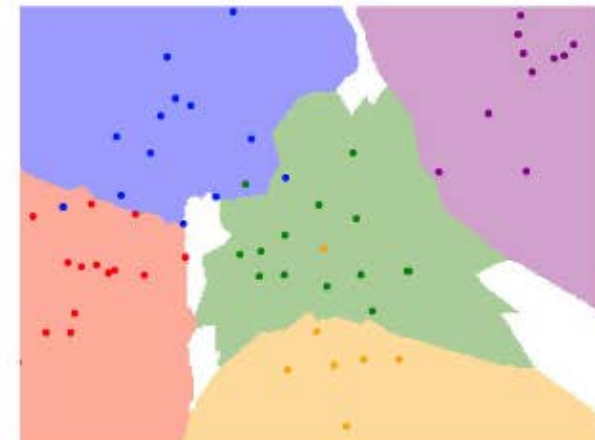
- How do we classify? For example...
 - Given a test face input, project into the same 3D space (by the same 3 eigenvectors).
 - The resulting vector in the 3D space is the **feature** for this test input.
 - We can do a simple **Nearest Neighbor (NN)** classification with Euclidean distance, which calculates the distance to all the projected training data in this space.
 - If NN, then the **label of the closest training instance** determines the classification output.
 - If **k-nearest neighbors (k-NN)**, then k-nearest neighbors need to **vote** for the decision.



k = 1



k = 3

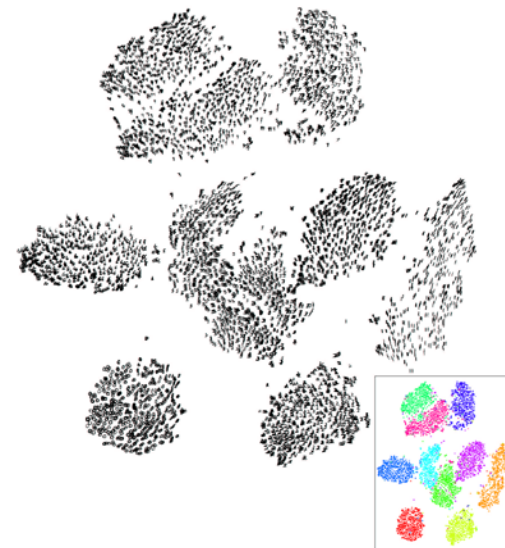
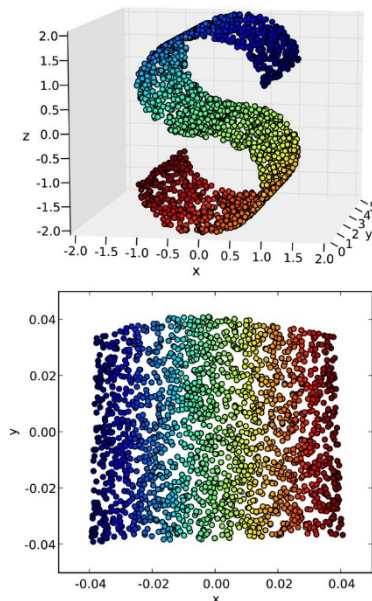


k = 5

Demo available at <http://vision.stanford.edu/teaching/cs231n-demos/knn/>

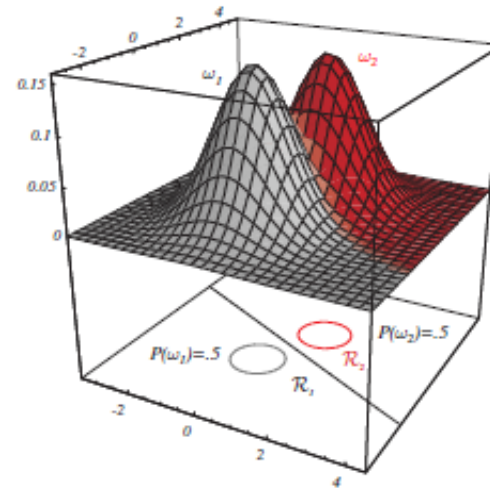
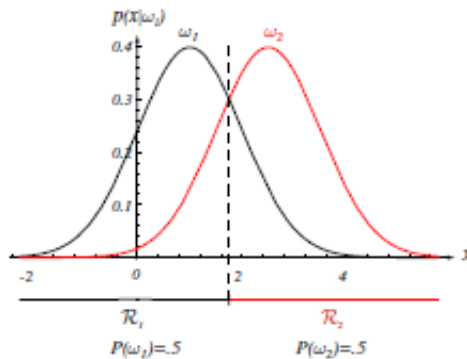
Final Remarks (cont'd)

- If labels for each data is provided → [Linear Discriminant Analysis \(LDA\)](#)
 - LDA is also known as Fisher's discriminant analysis.
 - Eigenface vs. Fisherface (IEEE Trans. PAMI 1997)
- If linear DR is not sufficient, and **non-linear DR** is of interest...
 - Isomap, locally linear embedding (LLE), etc.
 - **t-distributed stochastic neighbor embedding (t-SNE)** (by G. Hinton & L. van der Maaten)



What's to Be Covered Today...

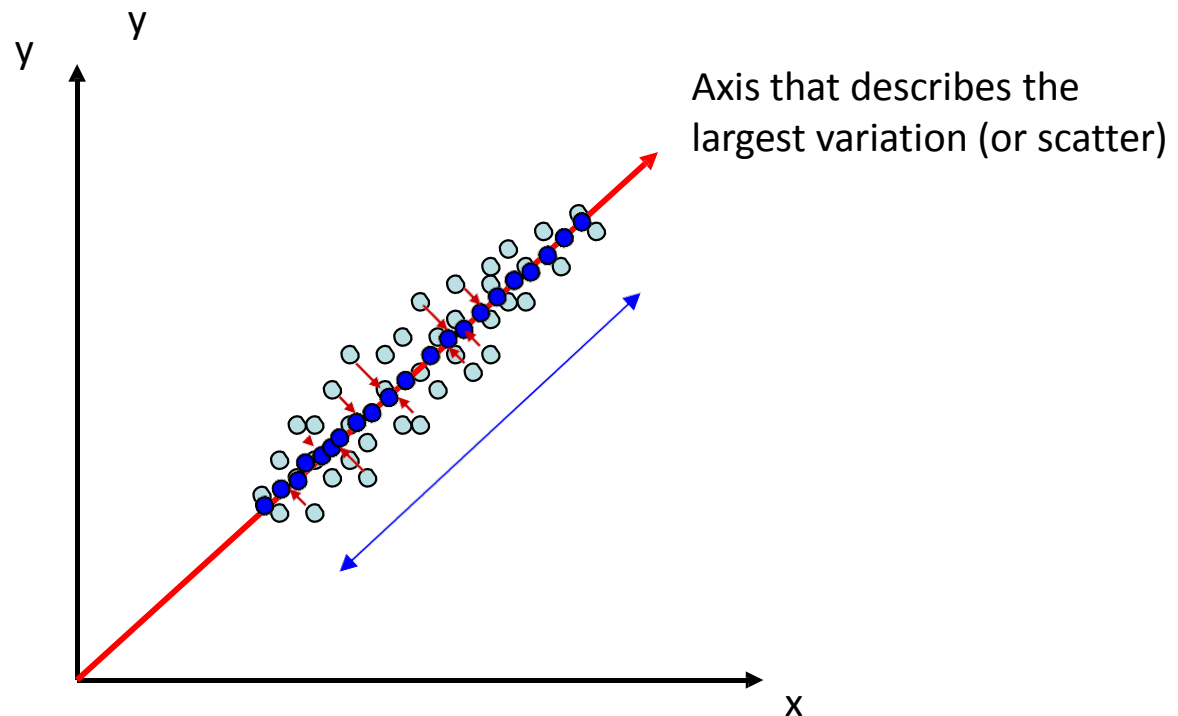
- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Clustering
 - Unsup. vs. Sup. Dimension Reduction
 - Training, testing, & validation
 - Linear Classification



What is PCA?

What are we trying to do?

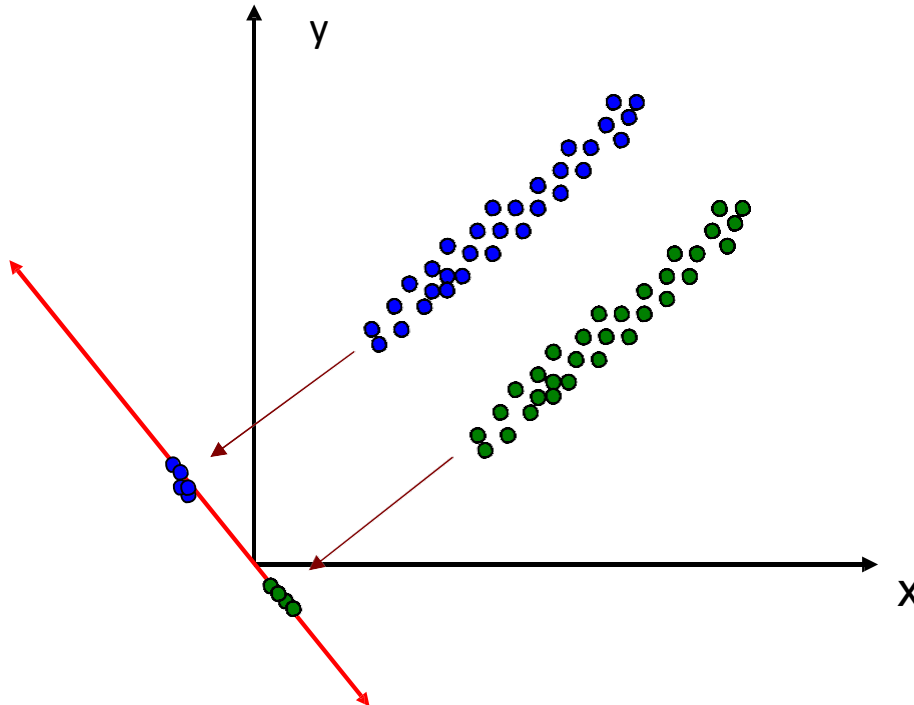
- We want to find projections of data (i.e., direction vectors that we can project the data on to) that describe the maximum variation.



What is LDA?

What are we trying to do?

- We want to find projections that separate the classes with the assumption of unimodal Gaussian modes.
- That is, to max. distance between two means while min. the variances
- =>will lead to minimize overall probability of error



Case 1: A simple 2-class problem

- We want to maximize the distance between the projected means:
e.g., maximize $|\tilde{\mu}_1 - \tilde{\mu}_2|^2$

Between Class Scatter Matrix S_B

$$\begin{aligned}(\tilde{\mu}_1 - \tilde{\mu}_2)^2 &= (\mathbf{w}^T \mu_1 - \mathbf{w}^T \mu_2)^2 \\ &= \mathbf{w}^T (\mu_1 - \mu_2) (\mu_1 - \mu_2)^T \mathbf{w} \\ &= \mathbf{w}^T S_B \mathbf{w}\end{aligned}$$

We want to maximize $\mathbf{w}^T S_B \mathbf{w}$ where S_B is the between class scatter matrix defined as:

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

NOTE: S_B is rank 1. This will be useful later on to find closed form solution for 2-class LDA

We also want to minimize....

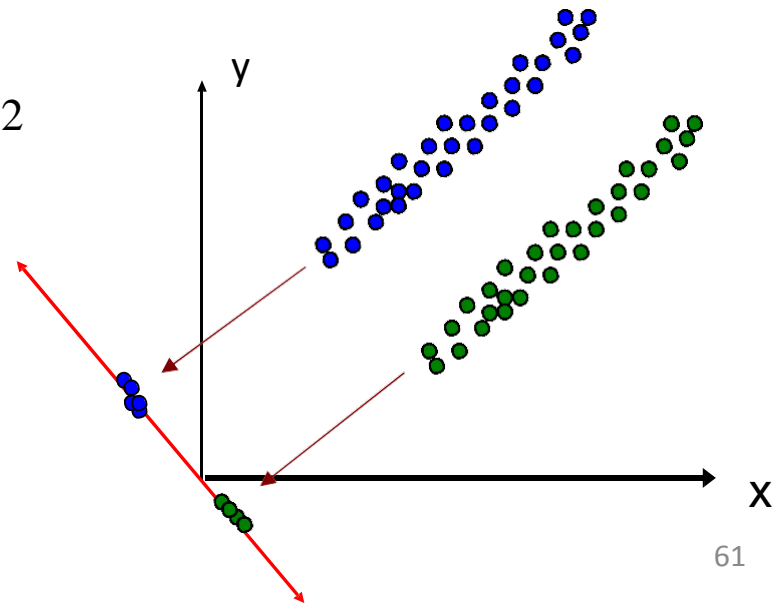
- The variance or scatter of the projected samples from each class (i.e. we want to make each class more compact or closer to its mean). The scatter from class 1 defined as s_1 is given as

$$\tilde{s}_1^2 = \sum_{i=1}^{N_1} (\tilde{x}_i - \tilde{\mu}_1)^2$$

- Thus we want to minimize the scatter of class 1 and class 2 in projected space, i.e.

minimize the total scatter

$$\tilde{s}_1^2 + \tilde{s}_2^2$$



Fisher Linear Discriminant Criterion Function

- Objective #1: We want to **maximize** the between class scatter:

$$|(\tilde{\mu}_1 - \tilde{\mu}_2)|^2$$

- Objective #2: We want to **minimize** the within-class scatter.

$$\tilde{s}_1^2 + \tilde{s}_2^2$$

- Thus we define our objective function $J(w)$ as the following ratio that we want to **maximize** in order to achieve the above objectives:

LDA

- Thus we want to find the vector \mathbf{w} that maximizes $J(\mathbf{w})$.
- Let's expand on scatter s_1 & s_2 .

$$\begin{aligned}\tilde{s}_1^2 &= \sum_{i=1}^{N_1} (\tilde{x}_i - \tilde{\mu}_1)^2 \\ &= \sum_{i=1}^{N_1} (w^T x_i - w^T \mu_1)^2 \\ &= \sum_{i=1}^{N_1} w^T (x_i - \mu_1)(x_i - \mu_1)^T w \\ &= w^T S_1 w\end{aligned}$$

$$\begin{aligned}\tilde{s}_2^2 &= \sum_{i=1}^{N_2} (\tilde{x}_i - \tilde{\mu}_2)^2 \\ &= \sum_{i=1}^{N_2} (w^T x_i - w^T \mu_2)^2 \\ &= \sum_{i=1}^{N_2} w^T (x_i - \mu_2)(x_i - \mu_2)^T w \\ &= w^T S_2 w\end{aligned}$$

Total Within-Class Scatter Matrix

- We want to minimize total within-class scatter. i.e.

$$\tilde{S}_1^2 + \tilde{S}_2^2$$

- This is equivalent to minimize $\mathbf{w}^T \mathbf{S}_w \mathbf{w}$

Solving LDA

- Maximize $J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$
- We need to find the optimal \mathbf{w} which will maximize the above ratio.
- What do we do now?

Some calculus....

LDA derivation

$$S_B \mathbf{w} - J(\mathbf{w}) S_W \mathbf{w} = \mathbf{0}$$

$$S_B \mathbf{w} - \lambda S_W \mathbf{w} = \mathbf{0}$$

$$S_B \mathbf{w} = \lambda S_W \mathbf{w}$$

Generalized Eigenvalue problem

$$S_W^{-1} S_B \mathbf{w} = \lambda \mathbf{w}$$

If S_W is non-singular and invertible.

We want to maximize $J(\mathbf{w})$. This is equivalent to the derivation of the eigenvector \mathbf{w} with the largest eigenvalue. Why?

Special Case LDA Solution for 2-Class Problems

- Lets replace what S_B is for two classes and see how we can simplify to get a closed form solution.
(i.e., we would like to get a solution of the vector \mathbf{w} for the 2-class case.)
- We know that in two class case, there is only 1 \mathbf{w} vector.
Lets use this knowledge cleverly...

S_B is rank 1

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T = \mathbf{m}\mathbf{m}^T$$

$$S_B = \mathbf{m}\mathbf{m}^T = \begin{bmatrix} | & | & | \\ m(1)m & m(2)m & m(N)m \\ | & | & | \end{bmatrix}$$

S_B has only 1 linearly independent column vector \Rightarrow Rank 1 matrix

2-class LDA

$$S_W^{-1} S_B w = \lambda w$$

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$


$$S_W = \sum_{i=1}^C \sum_{j=1}^{N_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

- Basically in this generalized eigenvalue/eigenvector problem, the number of valid eigenvectors with non-zero eigenvalue is determined by the **minimum** rank of matrices S_B and S_W .
- *In this case*, there is only 1 valid eigenvector with a non-zero eigenvalue! (i.e., there is only one valid w vector solution.)

2-class LDA (cont'd)

- Lets see and simplify the 2 class case:

$$(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \mathbf{w} = \lambda \mathbf{S}_w \mathbf{w}$$


$$(\mu_1 - \mu_2)^T \mathbf{w} = \text{scalar} = \beta$$

which gives $(\mu_1 - \mu_2)\beta = \lambda \mathbf{S}_w \mathbf{w}$

2-Class LDA Closed Form Solution

$$(\mu_1 - \mu_2)\beta = \lambda S_w \mathbf{w}$$

Multi-Class LDA

- What if we have more than 2 classes...what then?
- We need more than one \mathbf{w} projection vector to provide separability.
- Let's look at our math derivations to see what changes.

Multi-Class LDA (cont'd)

- Maximize $J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$

- Lets start with the Between-Class Scatter matrix for 2 class.

$$\mathbf{S}_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

- However, \mathbf{S}_B now is the between class scatter matrix for *many* classes. We need to make all the class means furthest from each other. One way is to push them as far away from their global mean

$$\mathbf{S}_B = \sum_{i=1}^C (\mu_i - \mu)(\mu_i - \mu)^T$$

Multi-Class LDA (cont'd)

- LDA solution:
$$S_B \mathbf{w} = \lambda S_w \mathbf{w}$$

Generalized Eigenvalue problem, the number of valid eigenvectors are bound by the MINIMUM rank of matrix (S_B, S_w) . In this case S_B is typically lowest rank which is sum of C outer-product matrices. (Since they subtract the global mean, the rank is **C-1**.)

$$S_w^{-1} S_B \mathbf{w} = \lambda \mathbf{w}$$

If S_w is non-singular and invertible.

For C classes we have at most $C-1$ \mathbf{w} vectors where we can project on to. Why?

When Would LDA Fail?

- What happens when we deal with **high-dimensional** data.
- If more dimensions d than sample # N , then we run into more problems.
- S_w is singular. It will still have at most **$N-C$ non-zero eigenvalues**.

N is the total number of samples from all classes, C is the number of classes.

$$S_w^{-1} S_B w = \lambda w$$

$$S_w = \sum_{i=1}^C \sum_{j=1}^{N_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

$$S_B = \sum_{i=1}^C (\mu_i - \mu)(\mu_i - \mu)^T$$

Fisherfaces

- Solution? Fisherfaces.....
- First do PCA and keep N-C eigenvectors. Project your data on to these N-C eigenvectors. (S_w will now be full rank = N-C not d.)
- Do LDA and compute the c-1 projections in this N-C dimensional subspace.
- PCA + LDA = Fisherfaces!

(read the famous PAMI paper of 'Fisherfaces vs Eigenfaces')

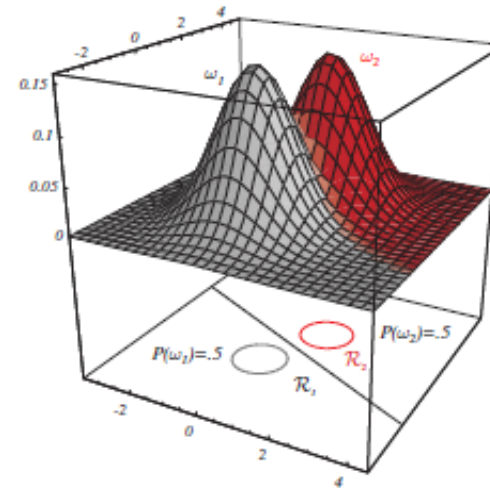
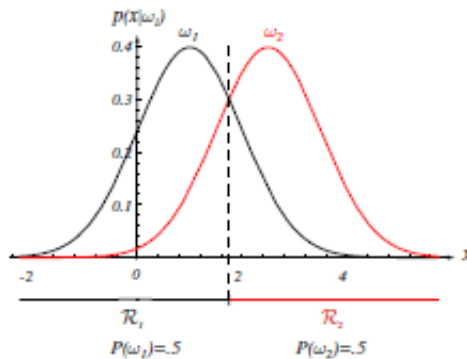
$$S_w^{-1} S_B w = \lambda w$$

$$S_w = \sum_{i=1}^C \sum_{j=1}^{N_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

$$S_B = \sum_{i=1}^C (\mu_i - \mu)(\mu_i - \mu)^T$$

What's to Be Covered Today...

- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Clustering & Dimension Reduction
 - Training, testing, & validation
 - Linear Classification

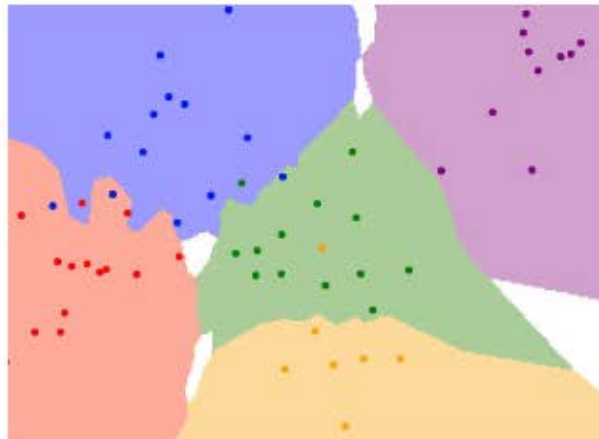


Hyperparameters in ML

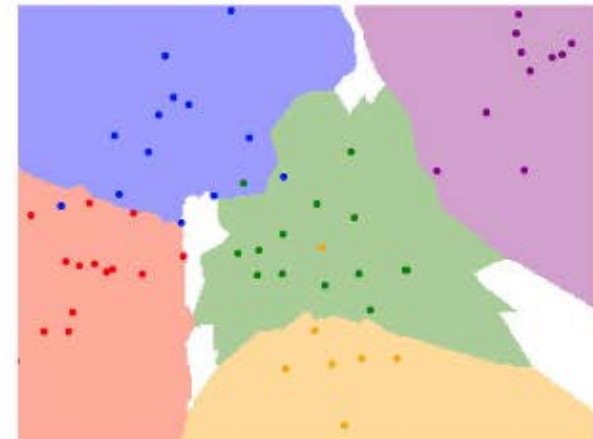
- Recall that for k-NN, we need to determine the k value in advance.
 - What is the best k value?
 - And, what is the best distance/similarity metric?
 - Similarly, take PCA for example, what is the best reduced dimension number?
- **Hyperparameters:** choices about the learning model/algorithm of interest
 - We need to determine such hyperparameters instead of learn them.
 - Let's see what we can do and cannot do...



k = 1



k = 3



k = 5

How to Determine Hyperparameters?

- Idea #1
 - Let's say you are working on face recognition.
 - You come up with your very own feature extraction/learning algorithm.
 - You take a dataset to train your model, and select your hyperparameters based on the resulting performance.



Dataset

How to Determine Hyperparameters? (cont'd)

- Idea #2
 - Let's say you are working on face recognition.
 - You come up with your very own feature extraction/learning algorithm.
 - For a dataset of interest, you split it into training and test sets.
 - You train your model with possible hyperparameter choices, and select those work best on test set data.



How to Determine Hyperparameters? (cont'd)

- Idea #3
 - Let's say you are working on face recognition.
 - You come up with your very own feature extraction/learning algorithm.
 - For the dataset of interest, it is split it into training, validation, and test sets.
 - You train your model with possible hyperparameter choices, and select those work best on the validation set.



How to Determine Hyperparameters? (cont'd)

- Idea #3.5
 - What if only training and test sets are given, not the validation set?
 - **Cross-validation** (or *k-fold* cross validation)
 - Split the training set into k folds with a hyperparameter choice
 - Keep 1 fold as validation set and the remaining $k-1$ folds for training
 - After each of k folds is evaluated, report the average validation performance.
 - Choose the hyperparameter(s) which result in the highest average validation performance.
 - Take a 4-fold cross-validation as an example...

Training set				Test set
Fold 1	Fold 2	Fold 3	Fold 4	Test set
Fold 1	Fold 2	Fold 3	Fold 4	Test set
Fold 1	Fold 2	Fold 3	Fold 4	Test set
Fold 1	Fold 2	Fold 3	Fold 4	Test set

Minor Remarks on NN-based Methods

- In fact, k-NN (or even NN) is not of much interest in practice. Why?
 - Choice of **distance metrics** might be an issue. See example below.
 - Measuring distances in **high-dimensional spaces** might not be a good idea.
 - Moreover, NN-based methods require lots of **data** and **computational power** !
(That is why NN-based methods are viewed as *data-driven* approaches.)



All three images have the same Euclidean distance to the original one.

What We Learned Today...

- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Clustering
 - Dimension Reduction,
 - Training, testing, & validation
 - Linear Classification

