

Deep Learning for Computer Vision

( Spring 2018

<http://vllab.ee.ntu.edu.tw/dlcv.html> (primary)

<https://ceiba.ntu.edu.tw/1062DLCV> (grade, etc.)

FB: [DLCV Spring 2018](#)

Yu-Chiang Frank Wang 王鈺強, Associate Professor

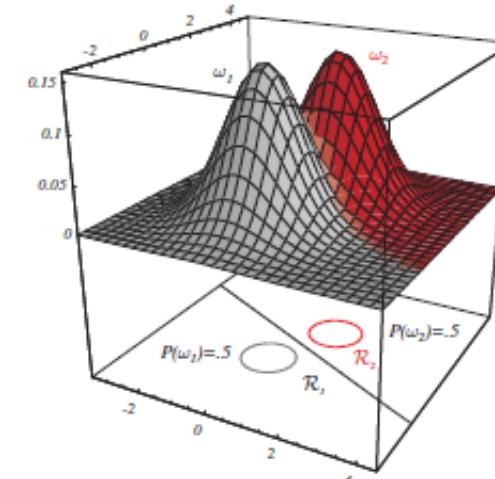
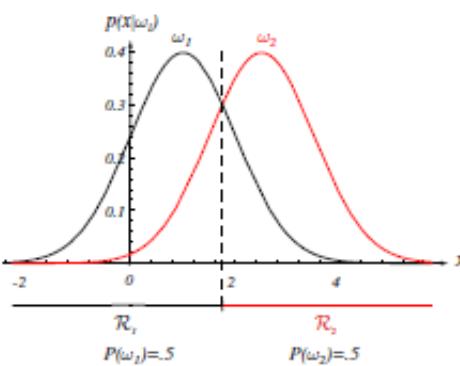
Dept. Electrical Engineering, National Taiwan University

Tight (yet tentative) Schedule

Week	Date	Topic	Remarks
0	3/7	Course Logistics + Intro to Computer Vision	
1	3/14	Machine Learning 101	HW #1 out (due in 1 week)
2	3/21	Image Representation	HW #1 due
3	3/28	Interest Point: From Recognition to Tracking	HW #2 out
4	4/4	N/A	Spring Break
5	4/11	Intro to Neural Networks + CNN	HW #2 due
6	4/18	Detection & Segmentation	HW #3 out
7	4/25	Generative Models	
8	5/2	Visualization and Understanding NNs	HW #3 due / HW #4 out
9	5/9	Recurrent NNs and Seq-to-Seq Models (I)	Project team
10	5/16	Recurrent NNs and Seq-to-Seq Models (II)	HW #4 due / HW #5 out
11	5/23	Deep Reinforcement Learning for Visual Apps	Project proposal
12	5/30	Learning Beyond Images (2D/3D, depth, etc.)	
13	6/6	Transfer Learning for Visual Analysis	HW #5 due
14	6/13	Vision & Language / Guest Lectures	Project midterm progress
15	6/20	Industrial Visit (e.g., Microsoft @ TPE)	CVPR week
16	6/27 (?)	Final Project Presentation	

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.,

$$\sum_{i=1}^C P(\omega_i) = 1$$

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 incorrect classification is equal, what is a reasonable decision rule?
 - Decide ω_1 if $P(\omega_1) > P(\omega_2)$;
otherwise decide ω_2 .
 - What's the incorrect classification rate (or error rate) P_e ?

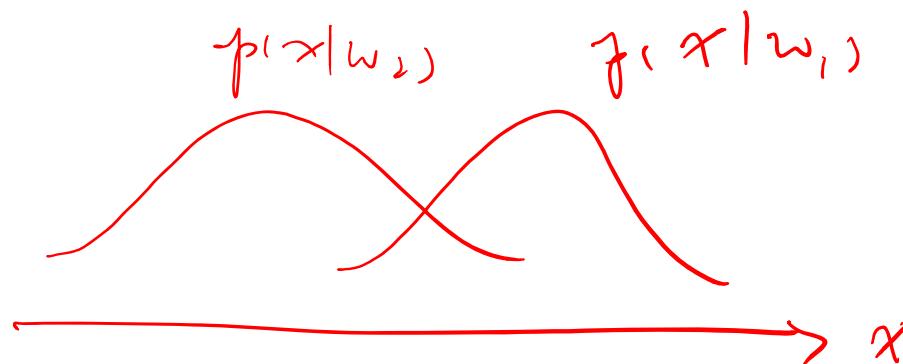
$$P_e = \min [P(\omega_1), P(\omega_2)]$$

Class-Conditional Probability Density (or Likelihood)

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

$$p(x|\omega_i), \quad i=1 \text{ or } 2$$

- 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?
- Posterior probability:
 - The probability of a certain state of nature ω given an observable x .
- Bayes formula:

$$P(\omega_j | x) = \underbrace{P(\omega_j | x)}_{\text{likelihood}} \cdot p(x) = \underbrace{p(x | \omega_j) P(\omega_j)}_{\substack{\text{evidence} = \sum_{i=1}^c p(x | \omega_i) P(\omega_i)}} \quad \begin{matrix} \xrightarrow{\text{Prior}} \\ \uparrow p(x) \end{matrix}$$

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

Decision Rule & Probability of Error

- For a given observable x , the decision rule will be now based on:

Decide ω_1 if $P(\omega_1|x) > P(\omega_2|x)$; otherwise, ω_2
 $\rightarrow \omega^* = \arg \max_i P(\omega_i|x)$

- What's the probability of error $P(\text{error})$ (or P_e)?

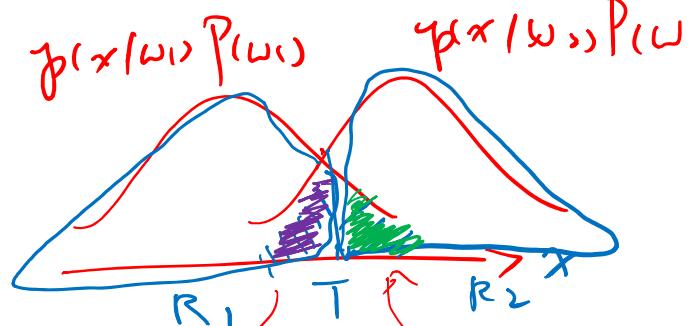
$$P_e = \min_i [P(\omega_i|x), P(\omega_{\bar{i}}|x)]$$

$$= \int_{-\infty}^{\infty} P(\text{error}, x) dx$$

$$= \int_{-\infty}^{\infty} P(\text{error}|x) \cdot P(x) dx$$

From Bayes Decision Rule to Detection Theory

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



$$P_C = \int_{-\infty}^T p(x|\omega_1) P(\omega_1) dx \rightarrow \text{correct rejection}$$

$$+ \int_T^\infty p(x|\omega_2) P(\omega_2) dx \rightarrow \text{hit}$$

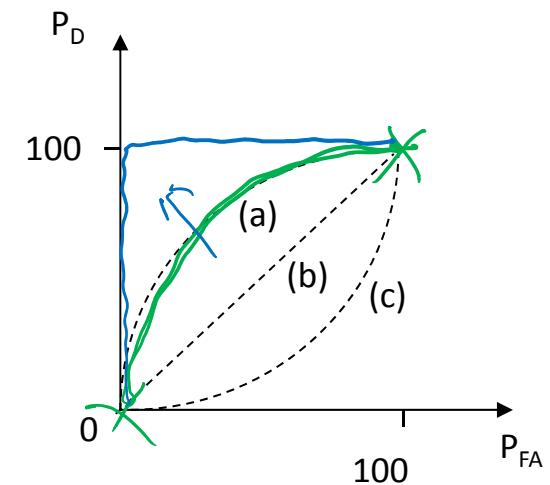
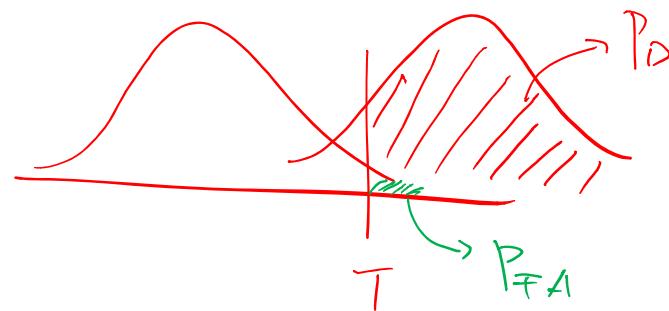
det rate

$$P_{FA} = \int_T^\infty p(x|\omega_1) P(\omega_1) dx$$

$$P_{MR} = \int_{-\infty}^T p(x|\omega_2) P(\omega_2) dx$$

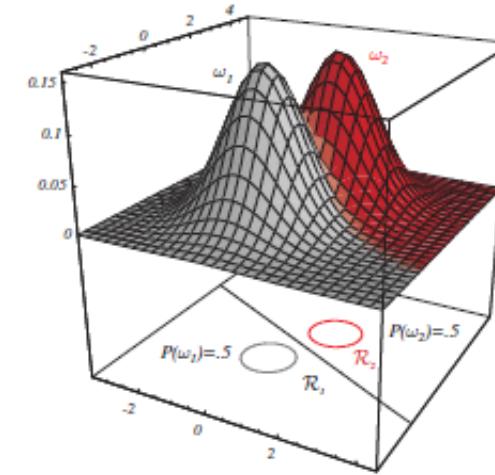
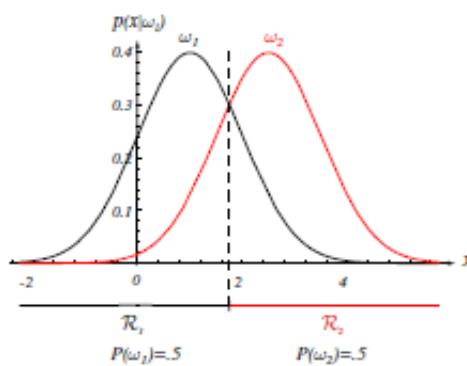
From Bayes Decision Rule to Detection Theory (cont'd)

- 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)?

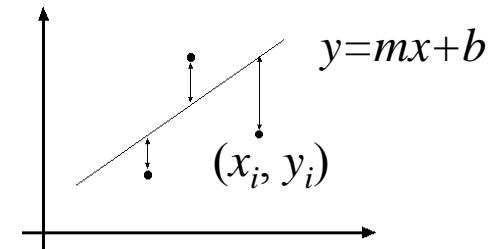


What's to Be Covered Today...

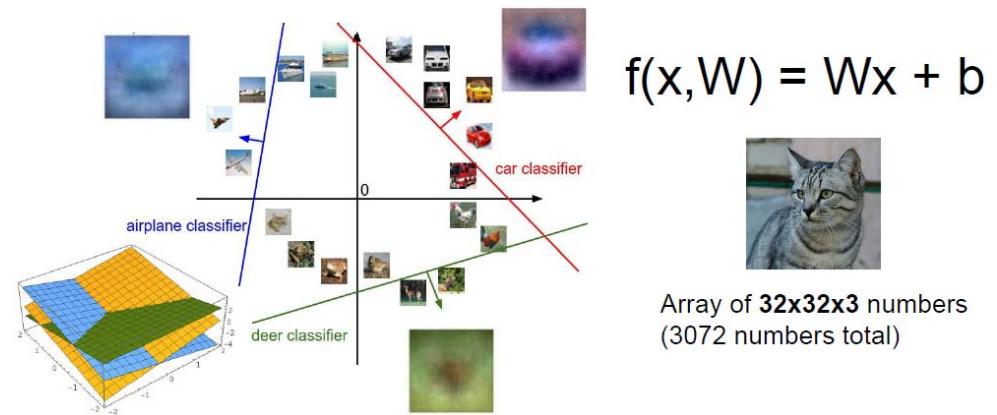
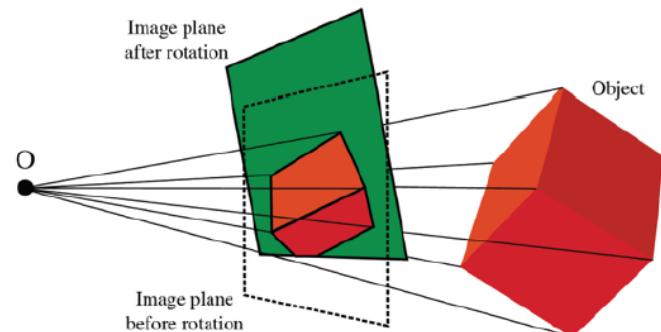
- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Dimension Reduction & Clustering
 - Linear Classification
 - Training, testing, & validation



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.



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.

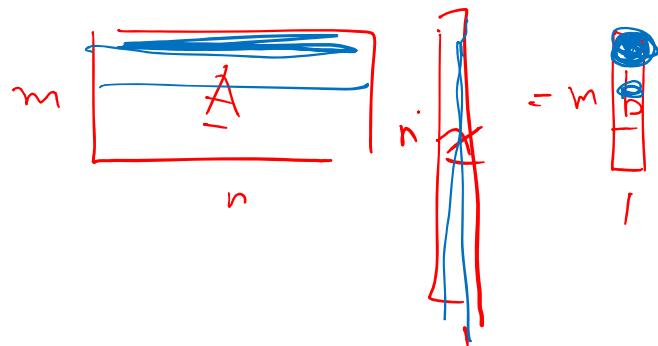
$$\underline{\mathbf{A}} = \begin{matrix} m \\ \hline \end{matrix} \left[\begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \right] \begin{matrix} n \\ \hline \end{matrix} \in \mathbb{R}^{m \times n}$$

- Thus, we have $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^T)$ and $\text{rank}(\mathbf{A}) \leq \min(m, n)$.
- 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} ?
 $\underline{\underline{=}}$
- If $m = n$ but $\text{rank}(\mathbf{A}) < m$, what can we do to compute \mathbf{A}^{-1} in practice?

$$\underline{\mathbf{A}} \leftarrow \underline{\mathbf{A}} + \sigma \underline{\mathbf{I}} , \quad \sigma : \text{small const.}$$

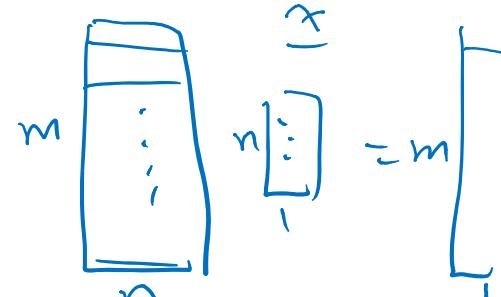
Solutions to Linear Systems

- 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} .)

Pseudoinverse of a Matrix (I)



- 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} ?

$$\underline{e} = \underline{b} - \underline{\mathbf{A}} \underline{x}$$

$$\begin{aligned} J(\underline{x}) &= \frac{1}{2} \cdot (\underline{e}^T \cdot \underline{e}) = \frac{1}{2} \cdot (\underline{b}^T - \underline{x}^T \cdot \underline{\mathbf{A}})^T (\underline{b} - \underline{\mathbf{A}} \underline{x}) \\ &= \frac{1}{2} \cdot (\underline{b}^T \cdot \underline{b} - 2\underline{b}^T \underline{\mathbf{A}} \cdot \underline{x} + \underline{x}^T \underline{\mathbf{A}}^T \underline{\mathbf{A}} \cdot \underline{x}) \end{aligned}$$

$1 \times m \quad m \times n \quad n \times 1$

$$\frac{\partial J}{\partial \underline{x}} = -\underline{\mathbf{A}}^T \cdot \underline{b} + \underline{\mathbf{A}}^T \cdot \underline{\mathbf{A}} \cdot \underline{x} = 0$$

$$(\underline{\mathbf{A}}^T \underline{\mathbf{A}}) \underline{x} = \underline{\mathbf{A}}^T \underline{b} \Rightarrow \underline{x} = \boxed{(\underline{\mathbf{A}}^T \underline{\mathbf{A}})^{-1} \cdot \underline{\mathbf{A}}^T \cdot \underline{b}}$$

p^{inv} of $\underline{\mathbf{A}}$

$\underline{\mathbf{A}}^T \underline{\mathbf{A}}$ is $n \times n$
full rank

Pseudoinverse of a Matrix (II)

A diagram illustrating the equation $\underline{A} \underline{x} = \underline{b}$. On the left, there is a green bracket above the matrix \underline{A} and a green bracket to the right of the vector \underline{x} . An equals sign is between them. To the right of the equals sign is another green bracket above the vector \underline{b} .

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

$$J(\underline{x}) = \frac{1}{2} \underline{x}^T \underline{x} + \underbrace{\underline{\lambda}^T (\underline{A} \underline{x} - \underline{b})}_{1 \times m \quad m \times 1}$$

$$\frac{\partial J}{\partial \underline{x}} = \underline{x} + \underline{A}^T \underline{\lambda} = 0$$

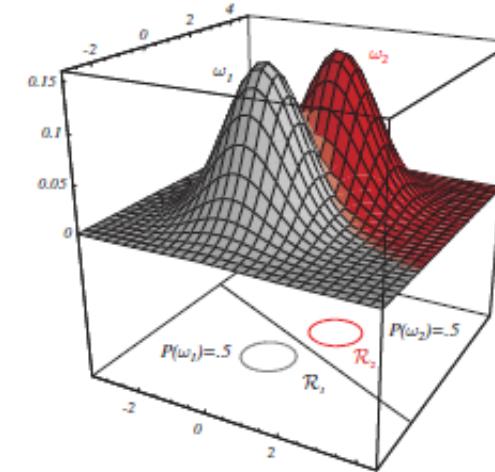
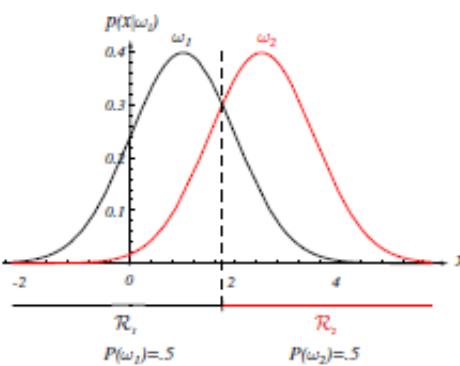
$$\underline{A} \underline{x} = - \underbrace{\underline{A} \underline{A}^T}_{m \times m} \underline{\lambda} = \underline{b} \Rightarrow \underline{\lambda} = - (\underline{A} \underline{A}^T)^{-1} \cdot \underline{b}$$

$$\Rightarrow \underline{x} = + \underbrace{\underline{A}^T (\underline{A} \underline{A}^T)^{-1}}_{\text{pinv of } \underline{A}} \cdot \underline{b}$$

pinv of \underline{A}

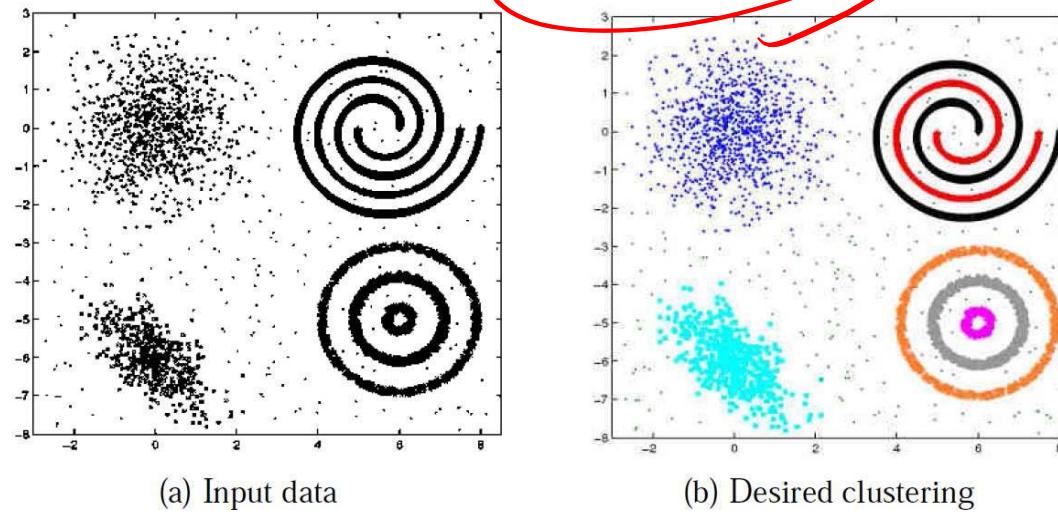
What's to Be Covered Today...

- 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



Clustering

- Clustering is an unsupervised algorithm.
 - Given:
a set of N unlabeled instances $\{x_1, \dots, x_N\}$; # 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?



Similarity is NOT Always Objective...



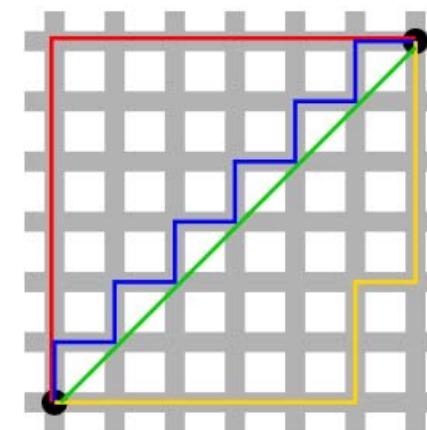
Clustering (cont'd)

- 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:

$$\|x\|_p = \left(\sum_{i=1}^D (x_i)^p \right)^{1/p}$$



Clustering (cont'd)

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

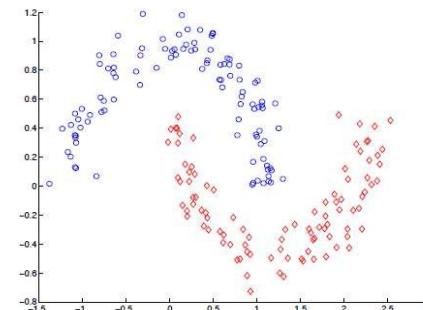
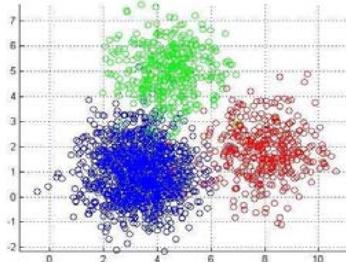
$$d(x, z) = \|\Phi(x) - \Phi(z)\|_2^2 = \|\Phi(x)\|_2^2 + \|\Phi(z)\|_2^2 - 2\Phi(x)^T \Phi(z)$$

- 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

$$\|\Phi(x)\|_2^2 = 1 \quad \sigma < \sqrt{\Phi(x)^T \Phi(z)} < 1$$

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

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

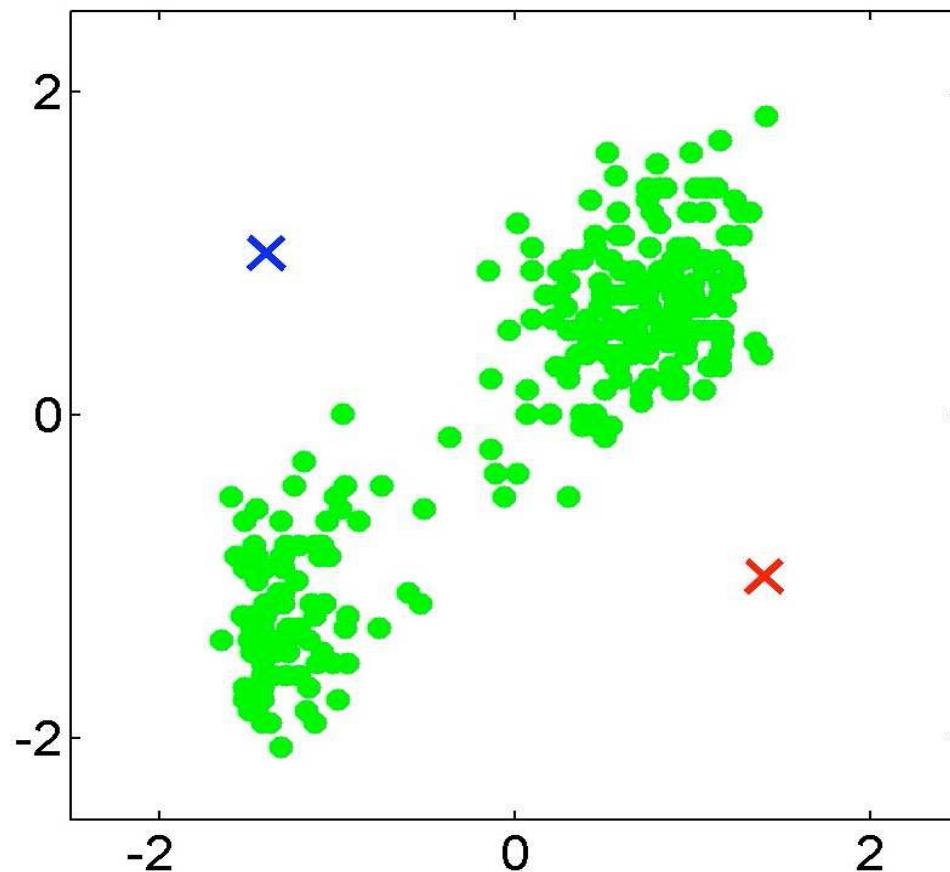


K-Means Clustering

- **Input:** N examples $\{x_1, \dots, x_N\}$ ($x_n \in \mathbb{R}^D$); 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 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)

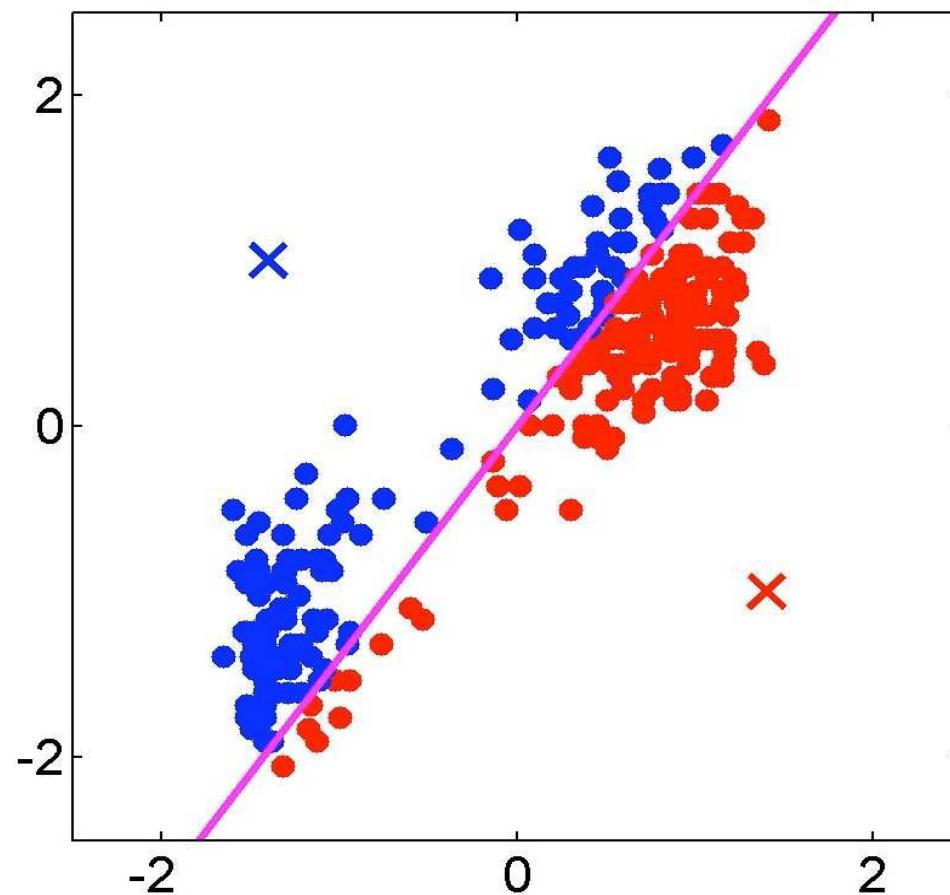
K-Means Clustering

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



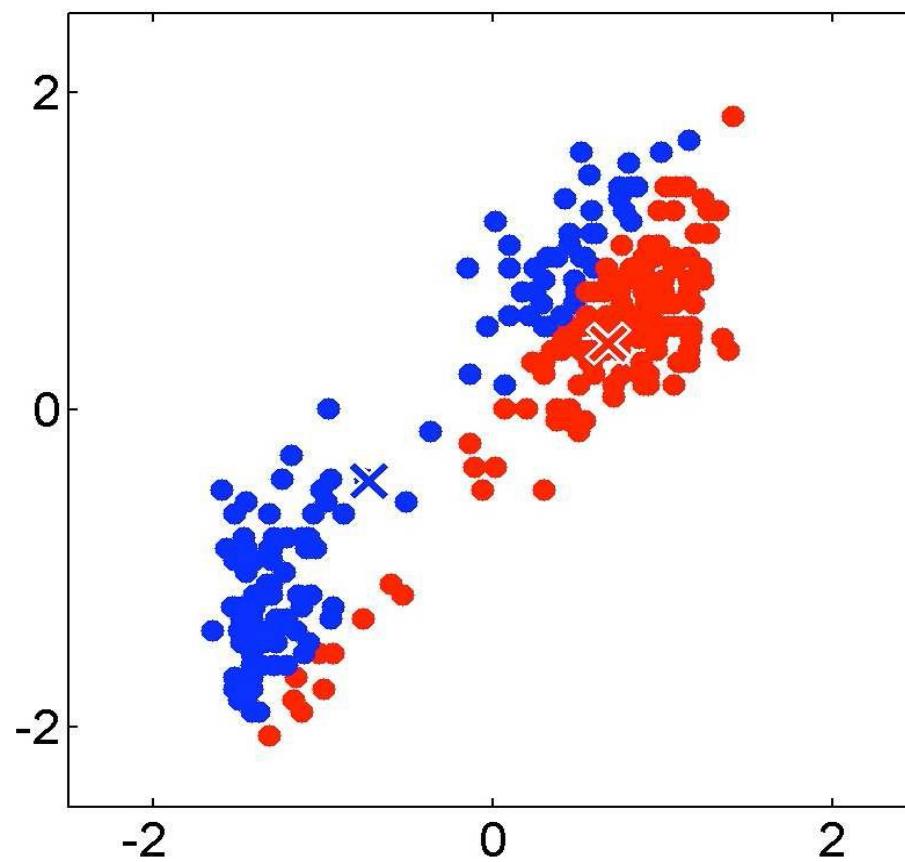
K-Means Clustering

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



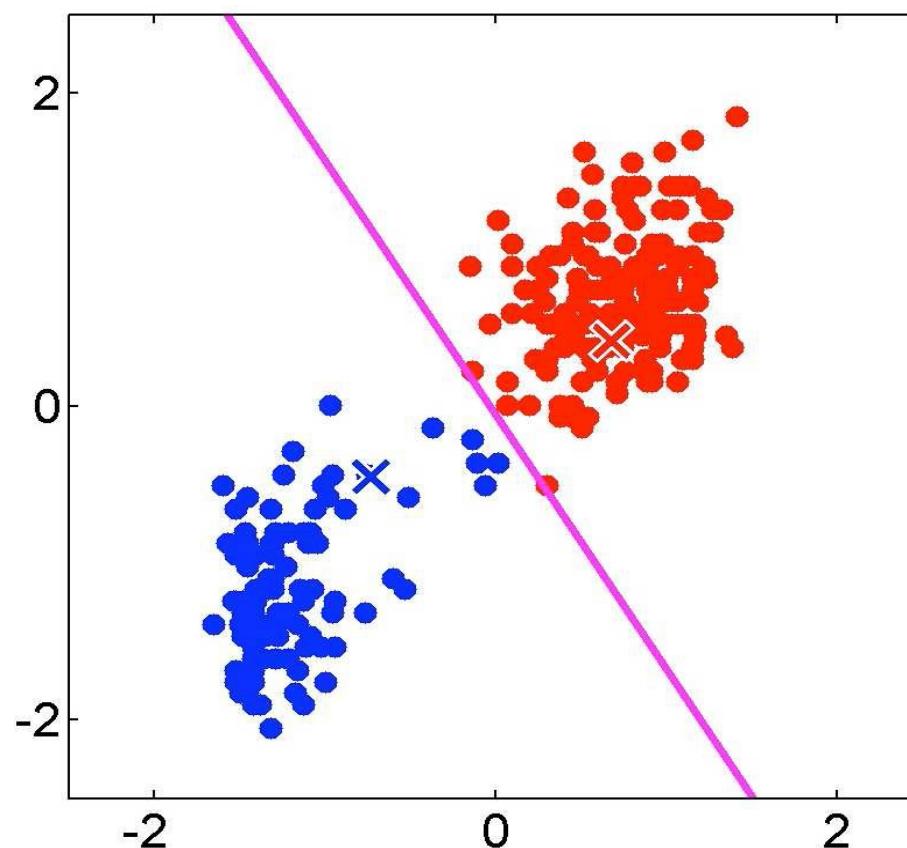
K-Means Clustering

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



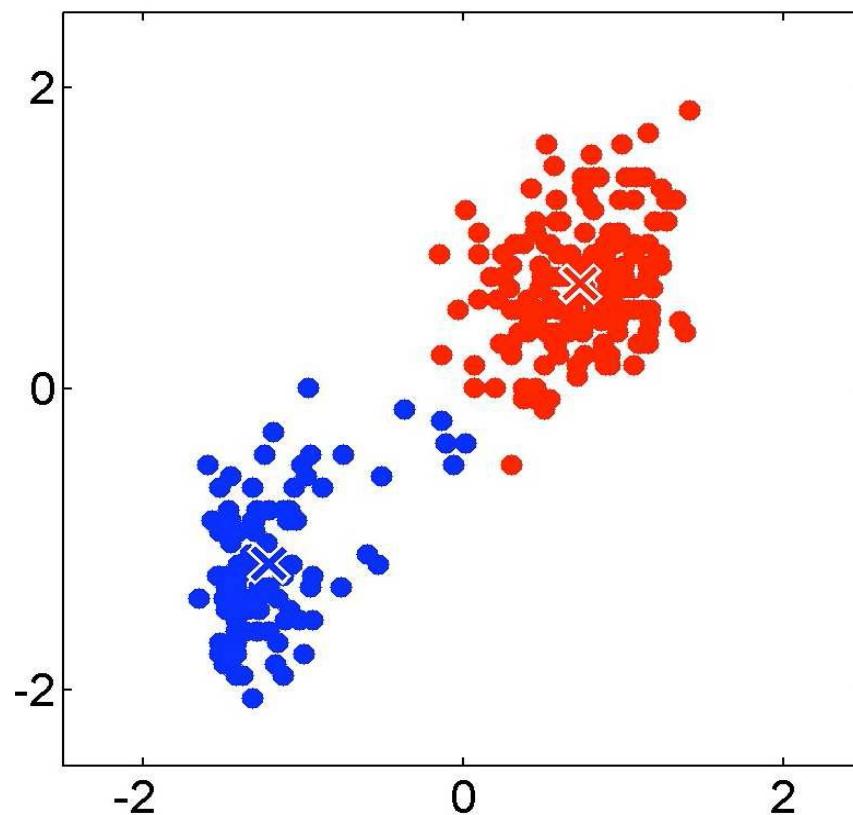
K-Means Clustering

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



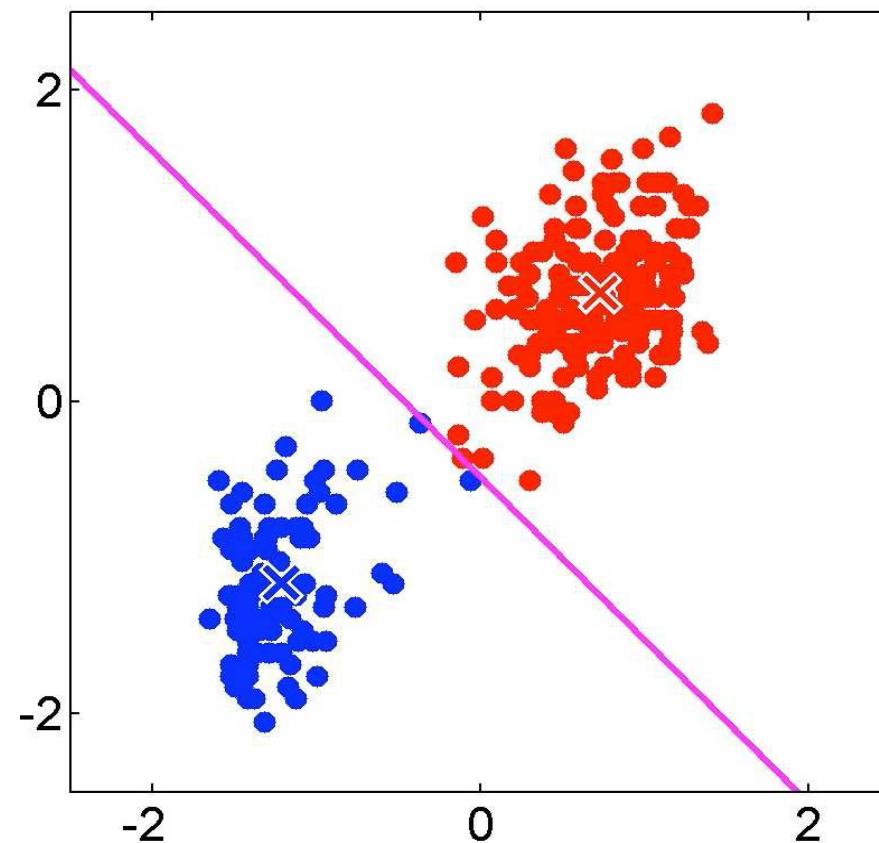
K-Means Clustering

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



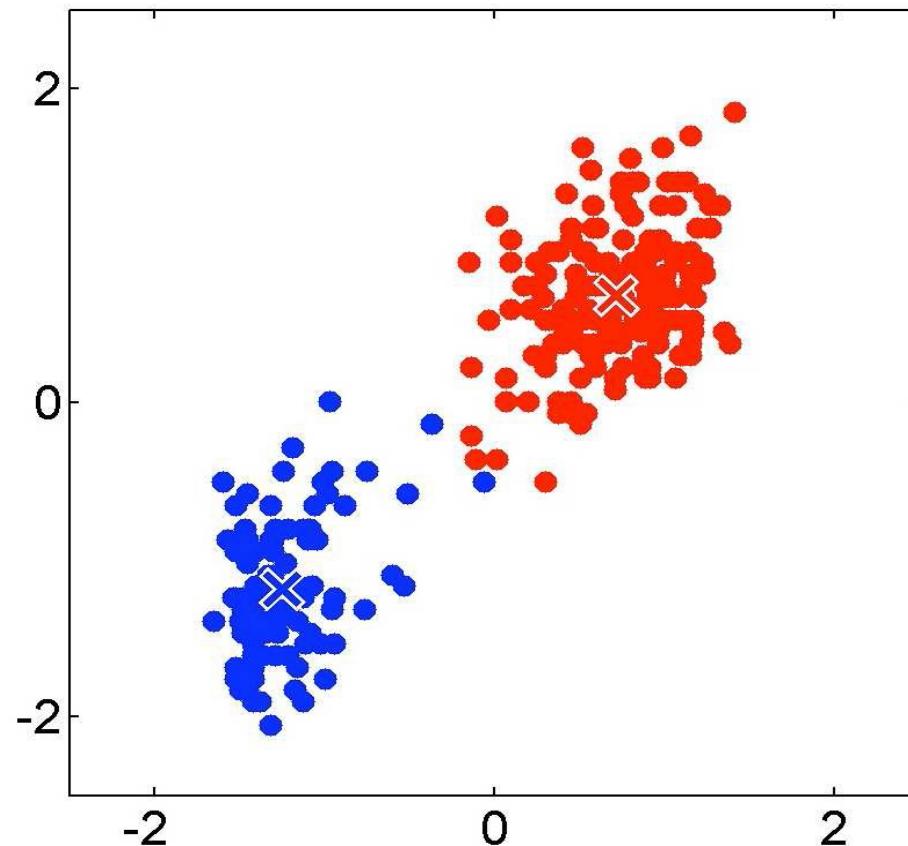
K-Means Clustering

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



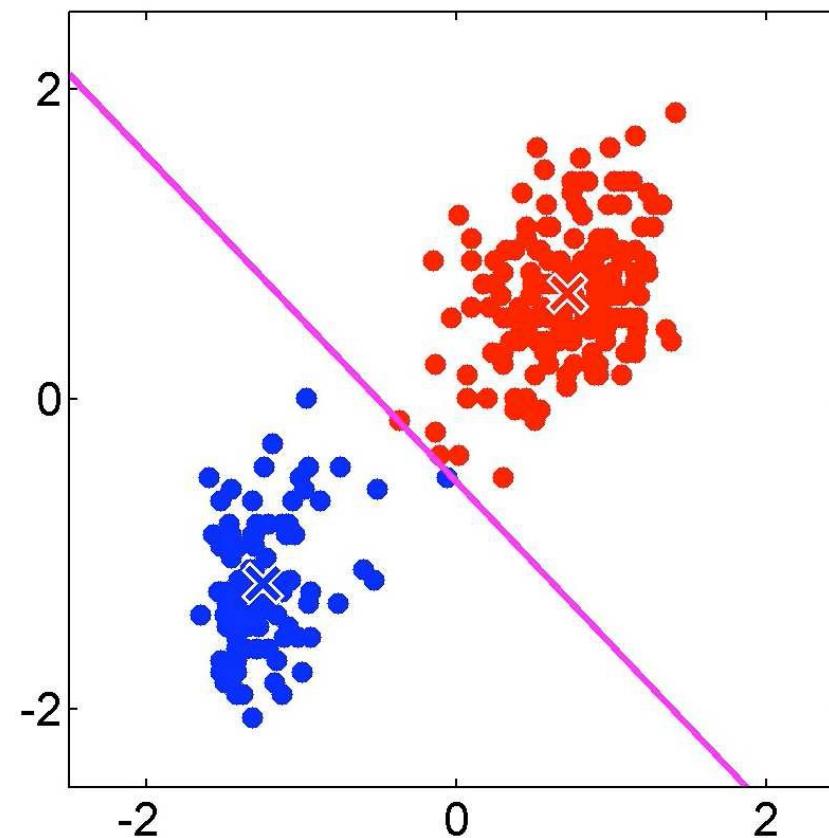
K-Means Clustering

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



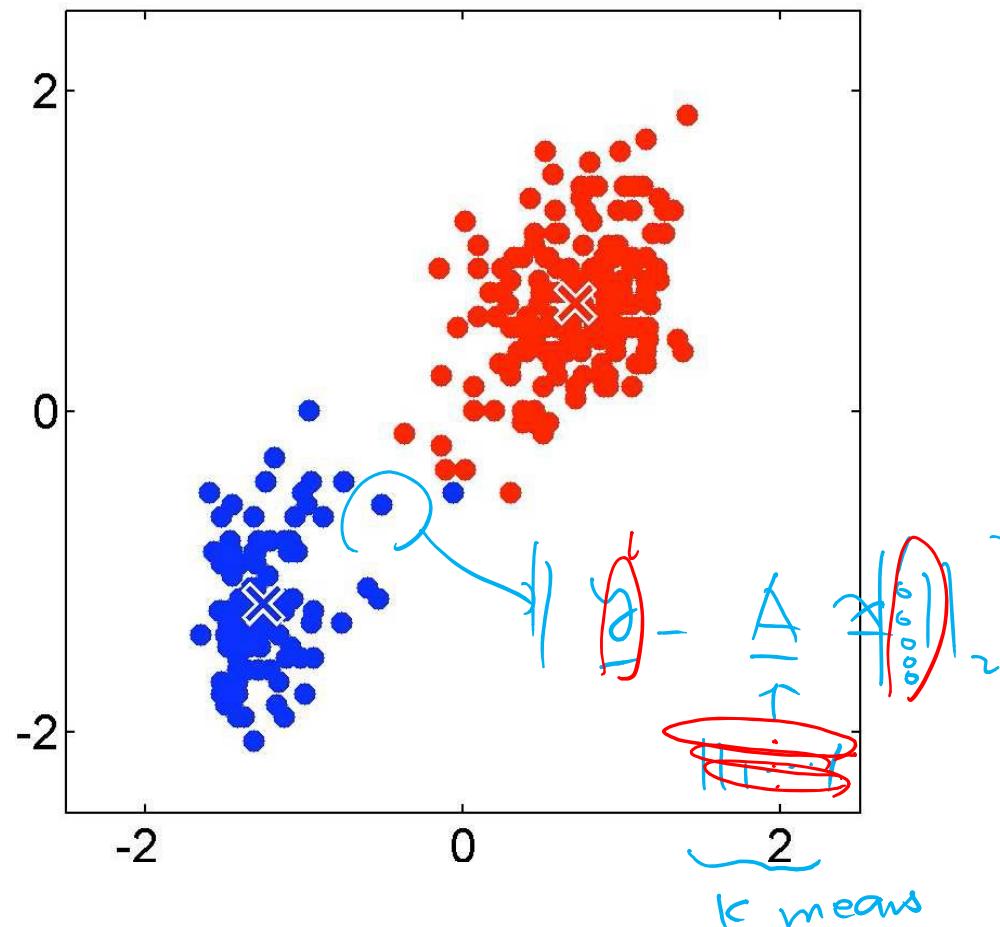
K-Means Clustering

- 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)

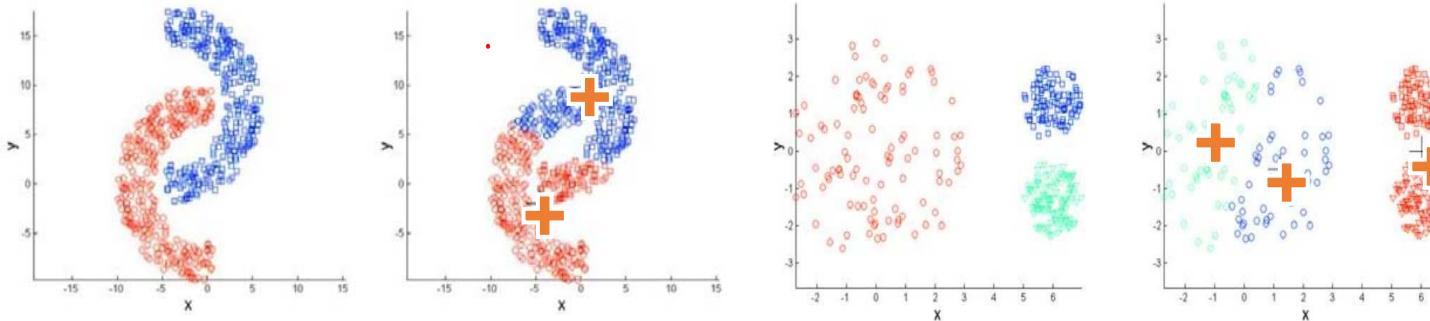
- Limitation

$$\text{1a } \underline{A} \in \mathbb{R}^{d \times K}$$

- Sensitive to initialization; how to alleviate this problem?
- Sensitive to outliers; possible change from K-means to...
- Hard assignment only. Mathematically, we have...

$$\min_{\underline{A}, \underline{Z}} \| \underline{y} - \underline{A}\underline{x} \|_2, \text{ s.t. } \|\underline{x}\|_0 = J, \|\underline{z}\|_1 = 1$$

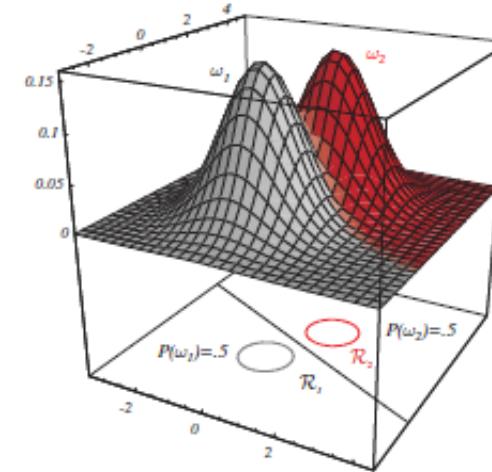
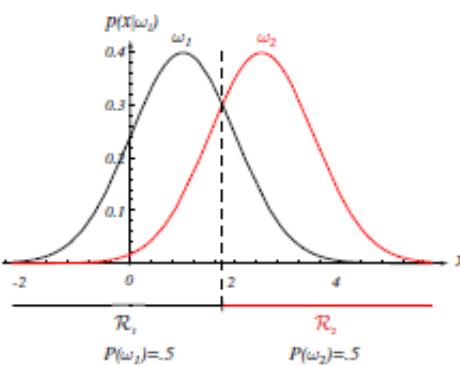
- Preferable for round shaped clusters with similar sizes



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

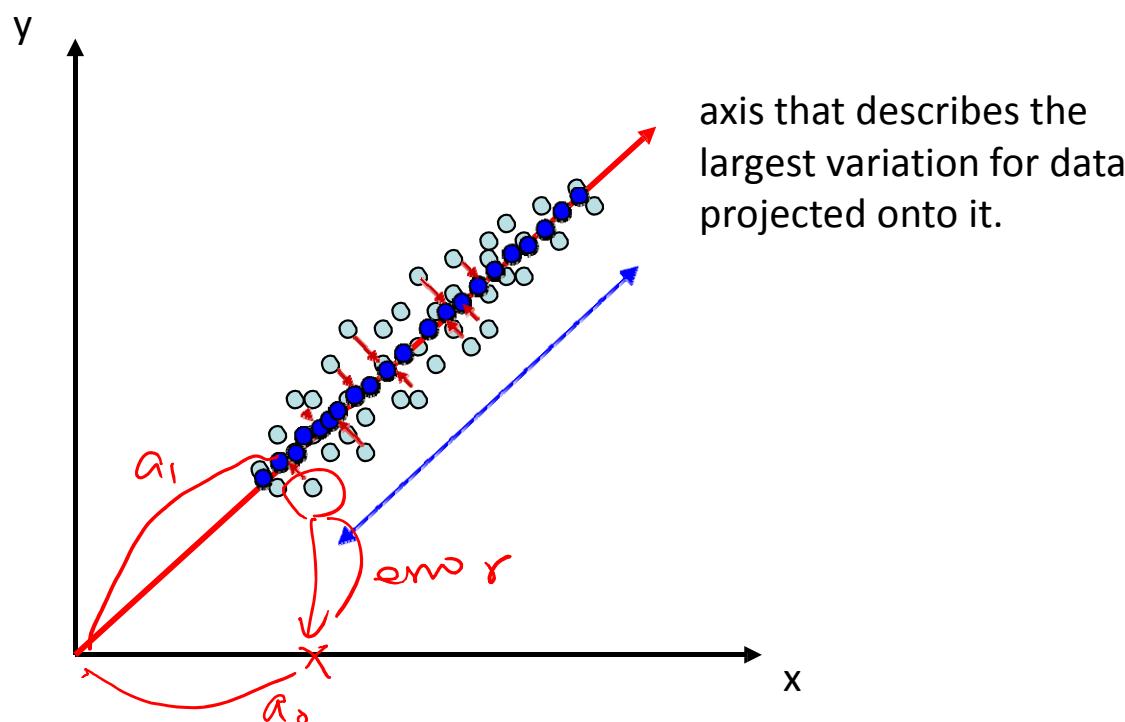
What's to Be Covered Today...

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



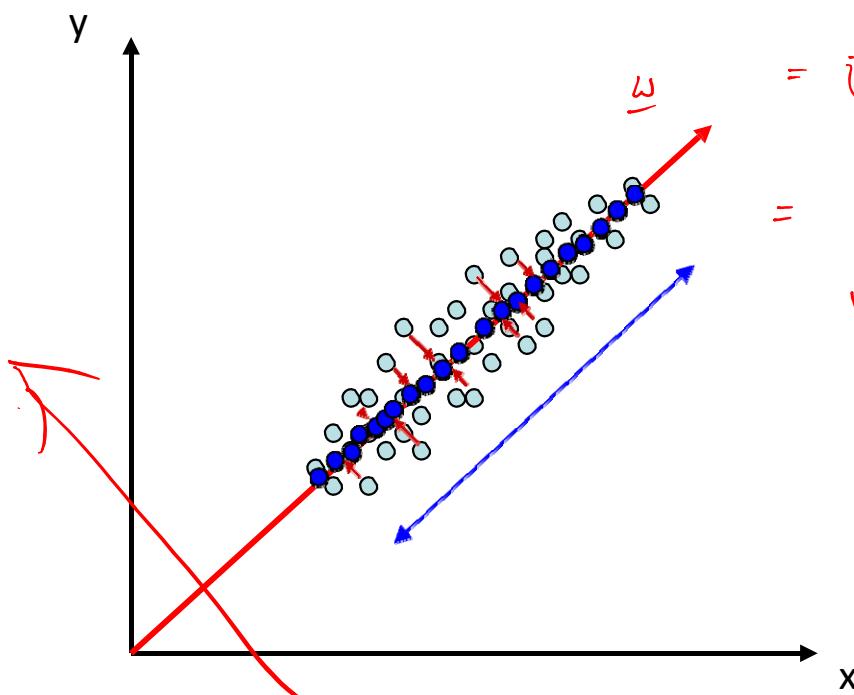
Dimension Reduction

- 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 ω maximizing the variance of the projected data
- In other words, we need to maximize $\text{var}(\omega^T \mathbf{x})$ with $\|\omega\| = 1$.



$$\begin{aligned}
 J(\underline{\omega}) &= E[(\underline{\omega}^T \mathbf{x} - \underline{\omega}^T \underline{\mu})^2] \\
 &= E[(\underline{\omega}^T \mathbf{x} - \underline{\omega}^T \underline{\mu})(\mathbf{x}^T \underline{\omega} - \underline{\mu}^T \underline{\omega})] \\
 &= \underline{\omega}^T \underbrace{E[(\mathbf{x} - \underline{\mu})(\mathbf{x} - \underline{\mu})^T]}_{d \times d} \cdot \underline{\omega} \\
 &\quad \uparrow d \qquad \uparrow d \times d \qquad \uparrow d \times 1
 \end{aligned}$$

$\underline{\Sigma}$ = covariance matrix.

$$\underline{\omega}^T \cdot \underline{\Sigma} \cdot \underline{\omega}$$

Formulation & Derivation for PCA (cont'd)

- Lagrangian optimization for PCA

$$J(\underline{w}) = \underline{w}^T \underline{\Sigma} \cdot \underline{w} - \lambda (\underline{w}^T \underline{w} - 1)$$

$$\frac{\partial J}{\partial \underline{w}} = 2 \underline{\Sigma} \cdot \underline{w} - 2 \lambda \underline{w} = 0$$

$$\underbrace{\underline{w}^T \underline{\Sigma} \underline{w}}_{d \times d \text{ } d \times 1} = \lambda \underline{w}^T \underline{w} = \lambda$$

$\lambda \times 1$

= $\lambda_1, \lambda_2, \dots, \lambda_d$

Eigenanalysis & PCA

- Eigenanalysis for PCA...find the eigenvectors e_i and the corresponding eigenvalues λ_i
 - In other words, the direction e_i captures the variance of λ_i . (e_1, \dots, e_d)
 - But, which eigenvectors to use though? All of them? $(\lambda_1, \dots, \lambda_d)$
- A $d \times d$ covariance matrix contains a maximum of d eigenvector/eigenvalue pairs.
 - Do we need to compute all of them? Which 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 ? $\cancel{N-1=99}$
 - Thus, at most 99 non-zero eigenvalues can be obtained.
 - How dimension reduction is realized? how to reconstruct the input data?

$$\sum_i e_i = \lambda e$$

↑

$$\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x})^T$$
$$= \frac{1}{N} \{ \boxed{} + \dots + \boxed{} \}$$

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 $N-1$, we have at most $N-1$ non-zero eigenvalues.
 - How dimension reduction is realized? how to reconstruct the input data?

$$\begin{aligned}
 \underline{\Sigma}^T (\underline{x} - \underline{\mu}) &= \sum_{i=1}^{N-1} \underline{Q}_i \cdot \underline{e}_i \\
 &= \underline{Q}_1 \underline{e}_1 + \dots + \underline{Q}_{N-1} \underline{e}_{N-1}
 \end{aligned}$$

$\underline{x} \in \mathbb{R}^{(M \times M)}$ \rightarrow $\underline{g} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} \in \mathbb{R}^3$

- 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

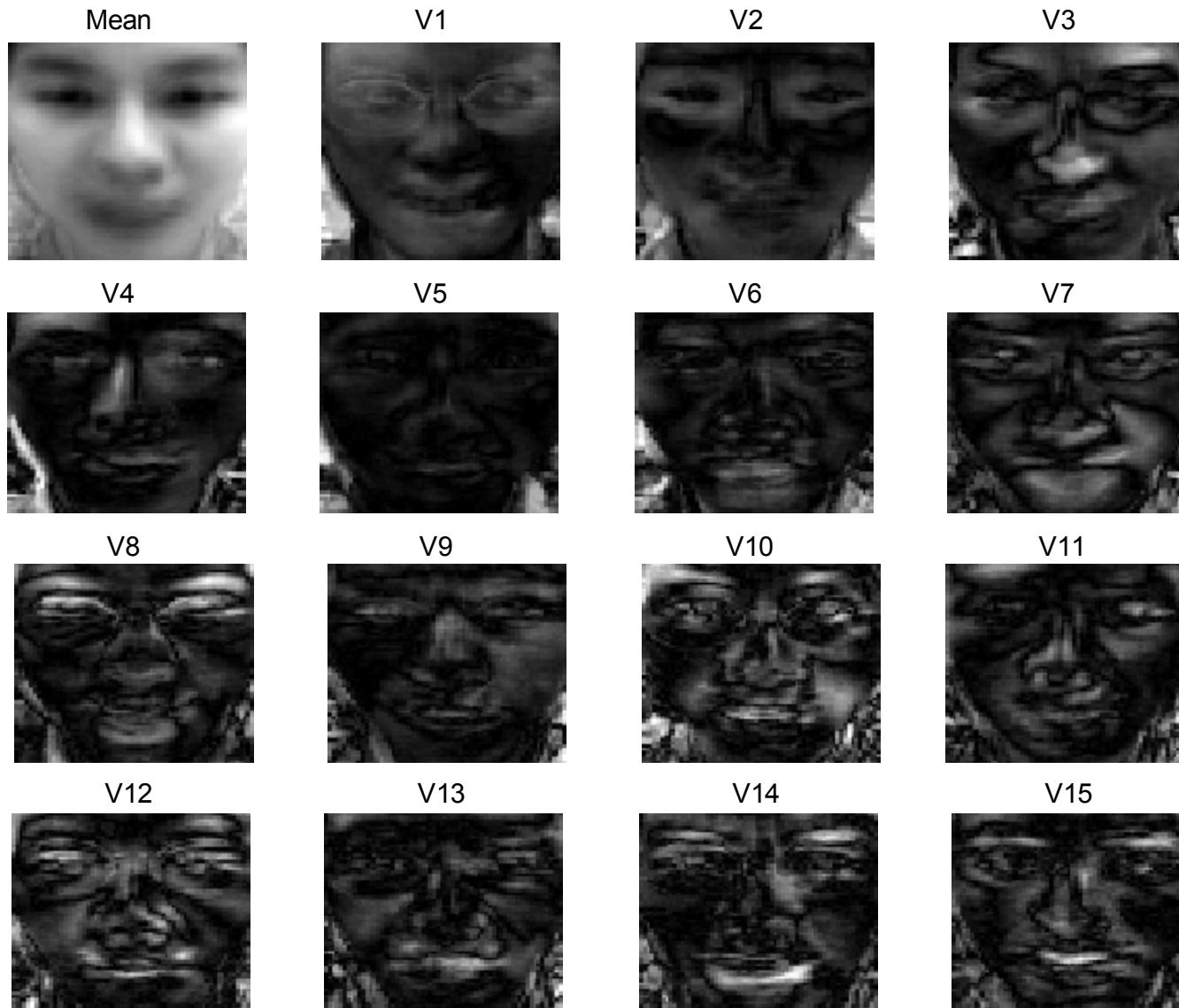
- Assume we have $N = 100$ images of size 200×200 pixels (i.e., $d = 40000$).
- What is the size of the covariance matrix? What's its rank?
- What can we do? **Gram Matrix Trick!**

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



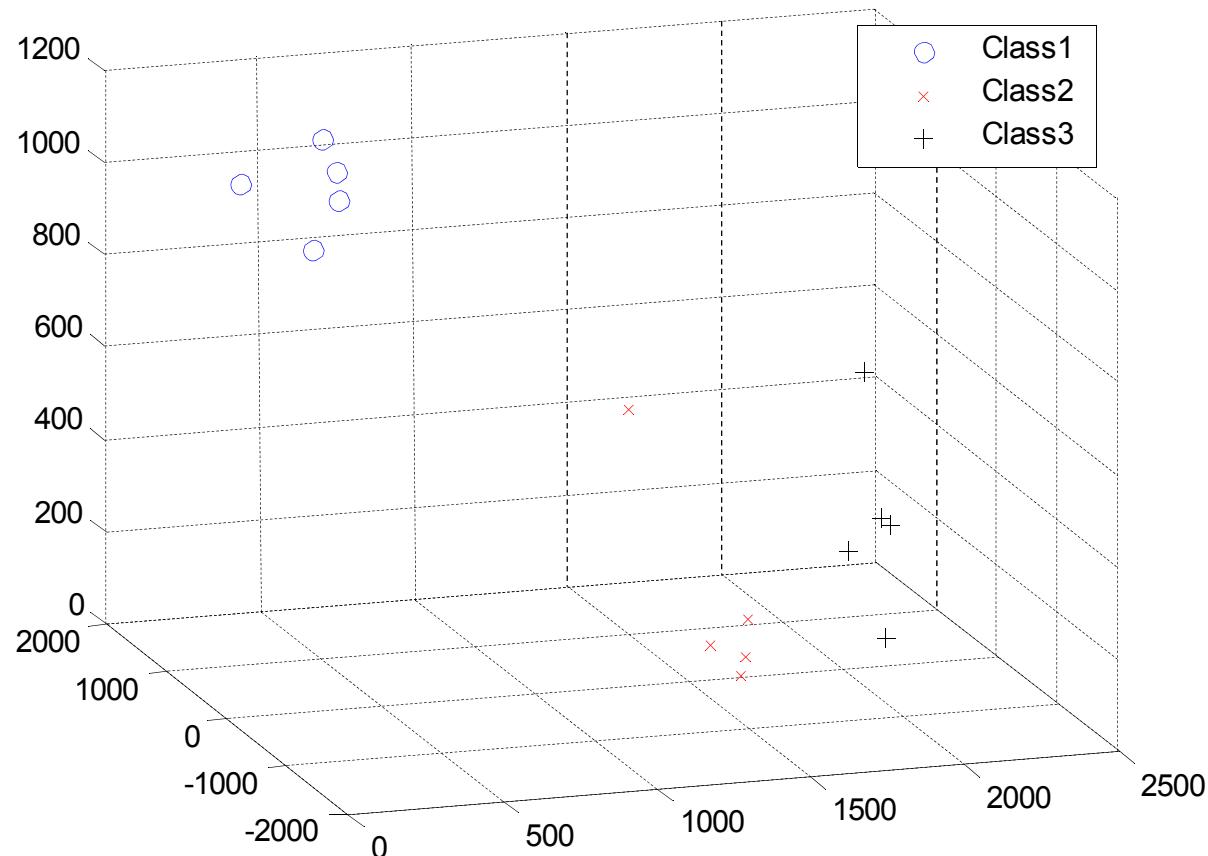
All 64 eigenvectors, MSE = 0

MSE=0.00



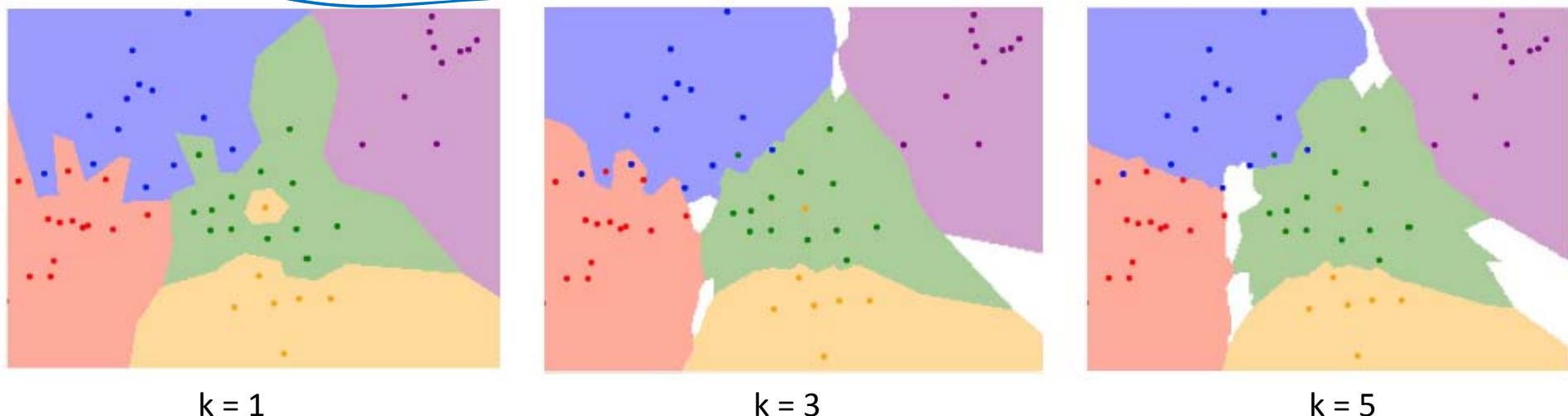
Final Remarks

- 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)

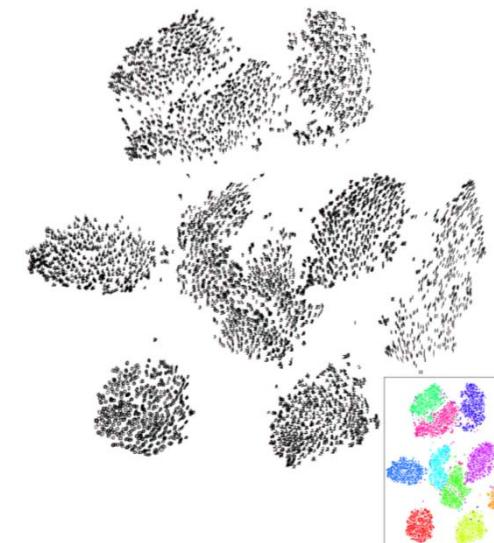
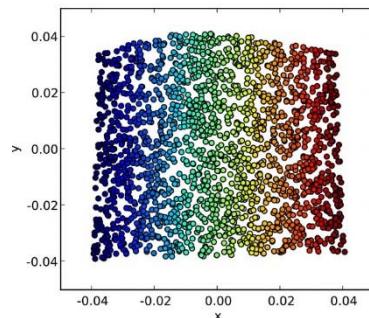
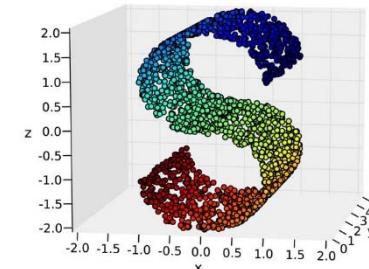
- How do we classify? For example...
 - Given a test face input, project into the same 3D space (by the same 3 eigenvectors).
 - The resulting vector in the 3D space is the **feature** for this test input.
 - We can do a simple **Nearest Neighbor (NN)** classification with Euclidean distance, which calculates 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.



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

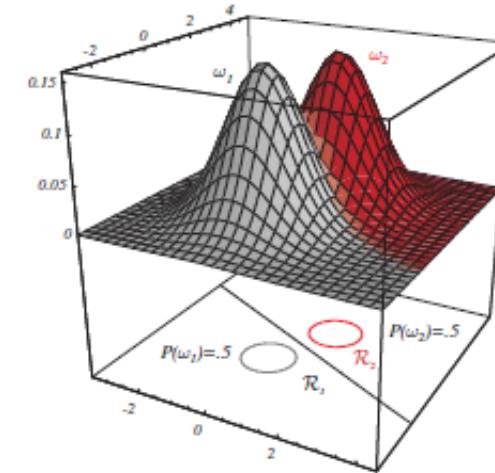
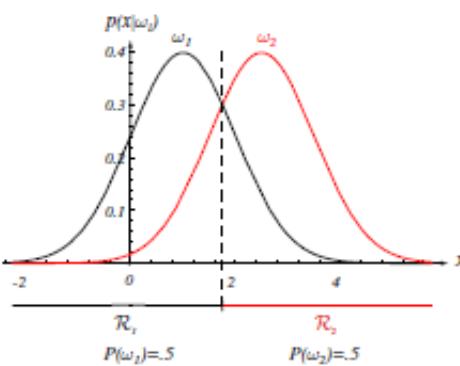
Final Remarks (cont'd)

- 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 & Dimension Reduction
 - Training, testing, & validation
 - Linear Classification

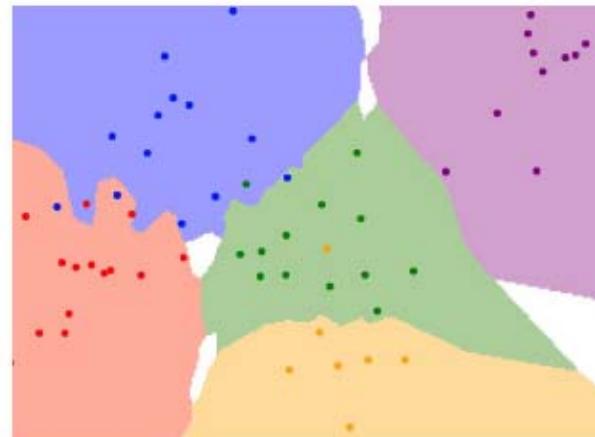


Hyperparameters in ML

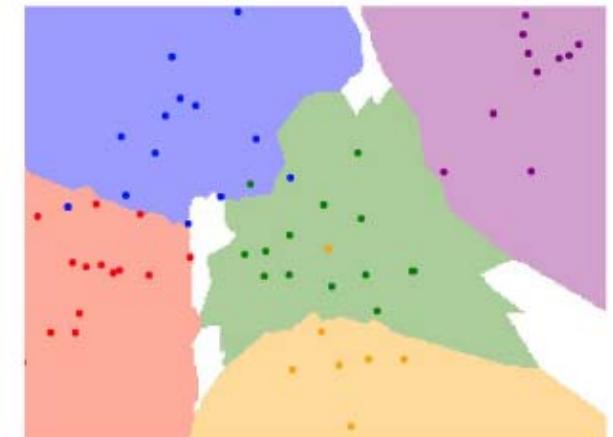
- Recall that for k-NN, we need to determine the k value in advance.
 - What is the best k value?
 - And, what is the best distance/similarity metric?
 - Similarly, take PCA for example, what is the best reduced dimension number?
- Hyperparameters: choices about the learning model/algorithim of interest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



$k = 1$



$k = 3$



$k = 5$

Image credit: Stanford CS231n

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

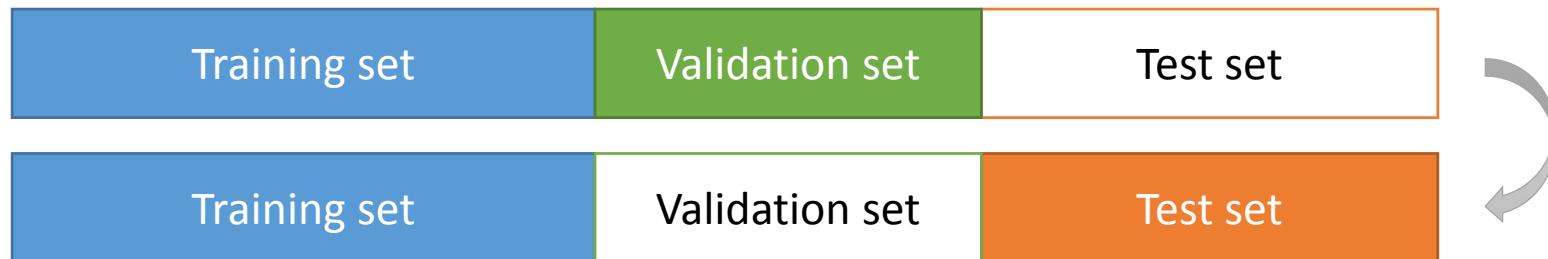
How to Determine Hyperparameters? (cont'd)

- 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.
 - 



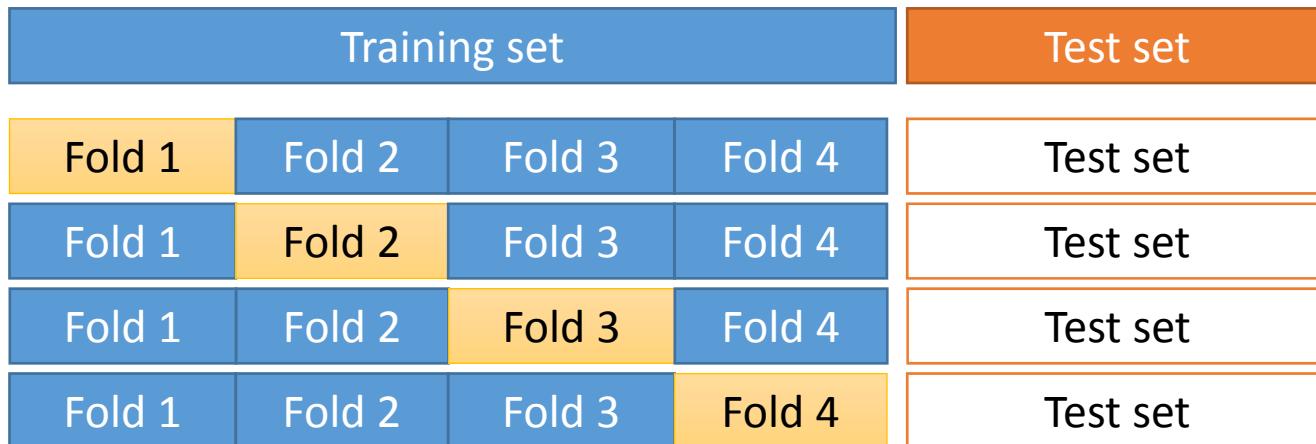
How to Determine Hyperparameters? (cont'd)

- 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...



Minor Remarks on NN-based Methods

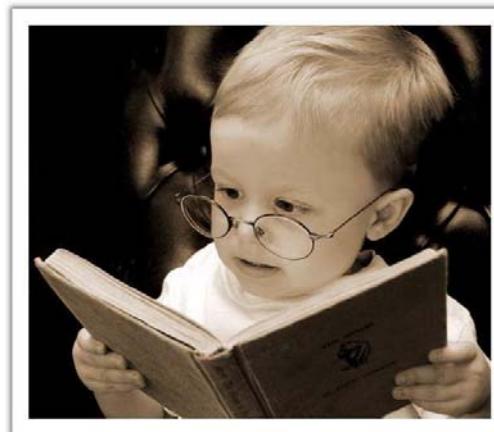
- In fact, k-NN (or even NN) is not of much interest in practice. Why?
 - Choice of *distance metrics* might be an issue. See example below.
 - Measuring distances in *high-dimensional spaces* might not be a good idea.
 - Moreover, NN-based methods require lots of and !
(That is why NN-based methods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



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

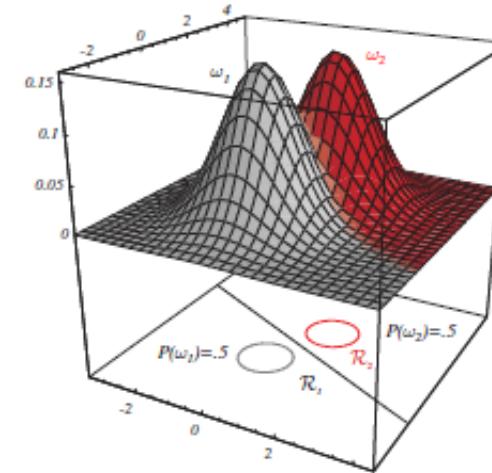
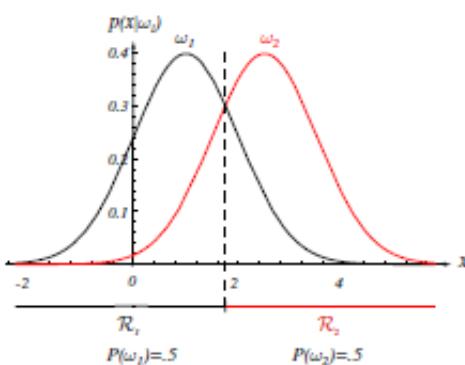
What We Learned Today...

- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Dimension Reduction, Clustering
 - Training, testing, & validation
 - Linear Classification
- Plus, HW 1 is out (see **Ceiba**) and due next week! (**3/20 Wed 01:00am**)



What's to Be Covered...

- 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



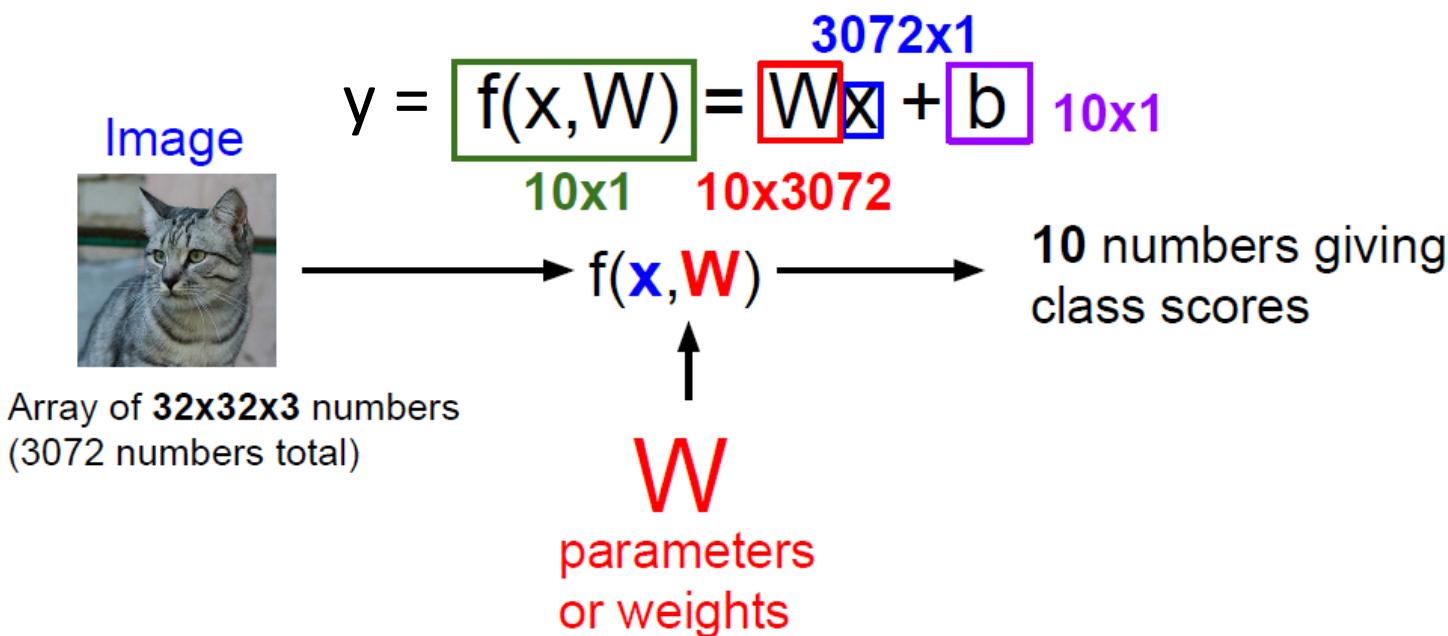
Linear Classification

- Linear Classifier
 - Can be viewed as a **parametric approach**. Why?
 - Assuming that we need to recognize 10 object categories of interest
 - E.g., CIFAR10 with 50K training & 10K test images of 10 categories.
And, each image is of size 32 x 32 x 3 pixels.



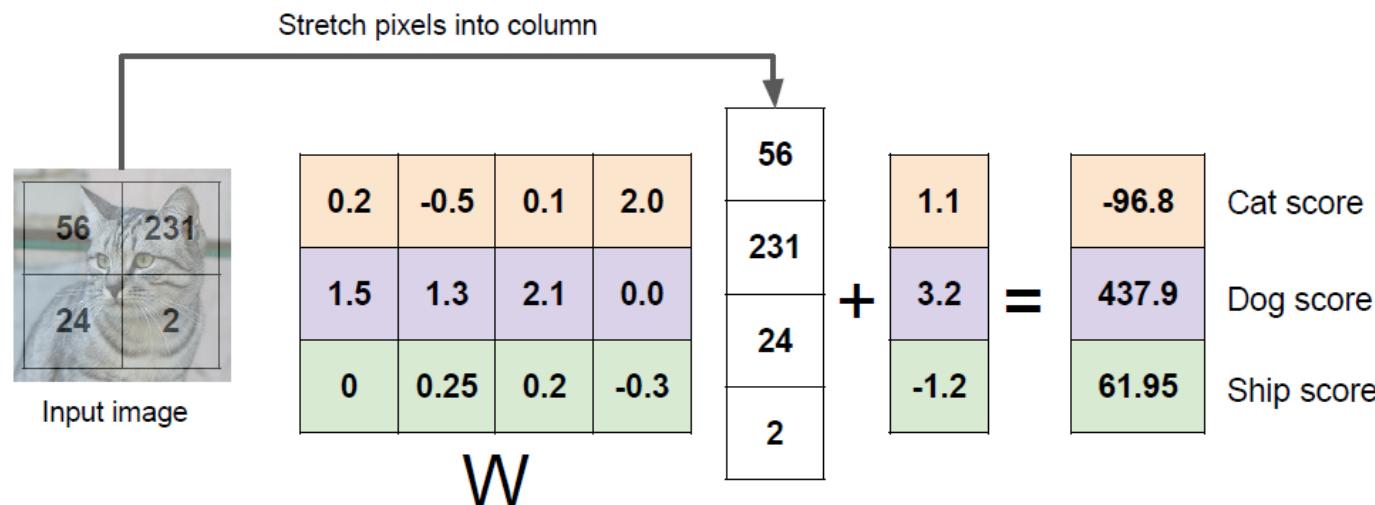
Linear Classification (cont'd)

- Linear Classifier
 - Can be viewed as a **parametric approach**. Why?
 - Assuming that we need to recognize 10 object categories of interest (e.g., CIFAR10).
 - Let's take the input image as \mathbf{x} , and the linear classifier as \mathbf{W} . We hope to see that $\mathbf{y} = \mathbf{Wx} + \mathbf{b}$ as a 10-dimensional output indicating the score for each class.



Linear Classification (cont'd)

- Linear Classifier
 - Can be viewed as a **parametric approach**. Why?
 - Assuming that we need to recognize 10 object categories of interest (e.g., CIFAR10).
 - Let's take the input image as \mathbf{x} , and the linear classifier as \mathbf{W} . We hope to see that $\mathbf{y} = \mathbf{Wx} + \mathbf{b}$ as a 10-dimensional output indicating the score for each class.
 - Take an image with 2×2 pixels & 3 classes of interest as example: we need to learn linear transformation/classifier \mathbf{W} and bias \mathbf{b} , so that desirable outputs $\mathbf{y} = \mathbf{Wx} + \mathbf{b}$ can be expected.



Some Remarks

- Interpreting $y = Wx + b$
 - What can we say about the learned W ?
 - The weights in W are trained by observing training data X and their ground truth Y .
 - Each column in W can be viewed as an exemplar of the corresponding class.
 - Thus, Wx basically performs **inner product** (or **correlation**) between the input x and the exemplar of each class. (Signal & Systems!)

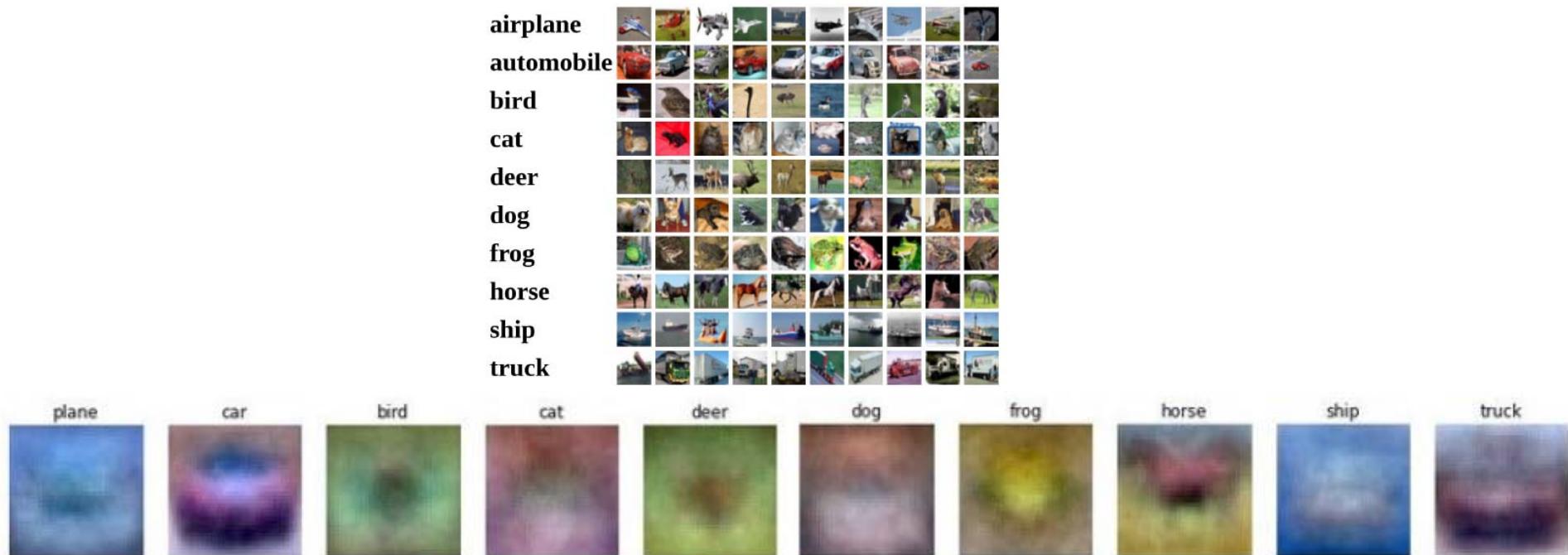
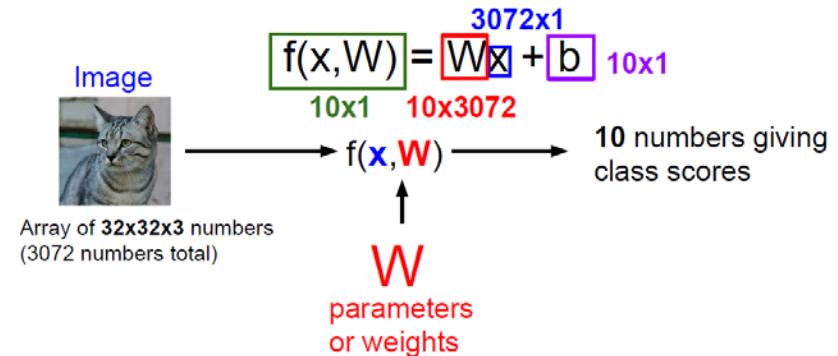
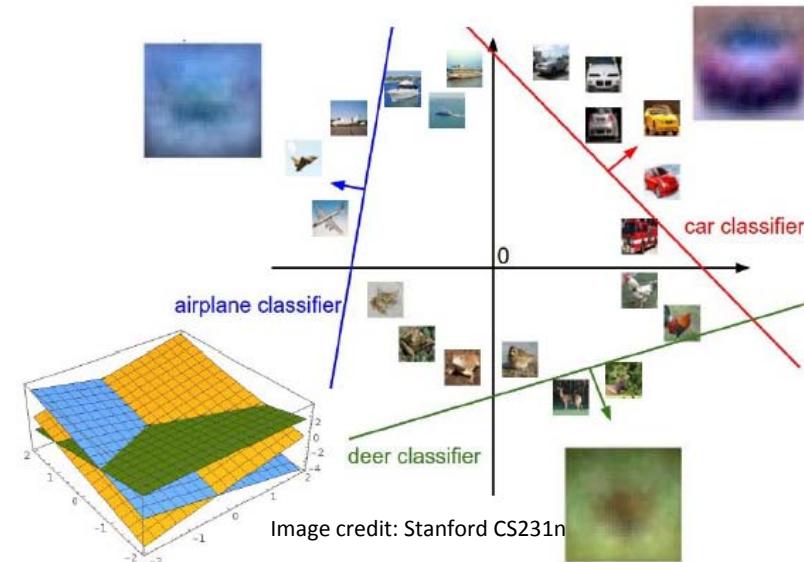
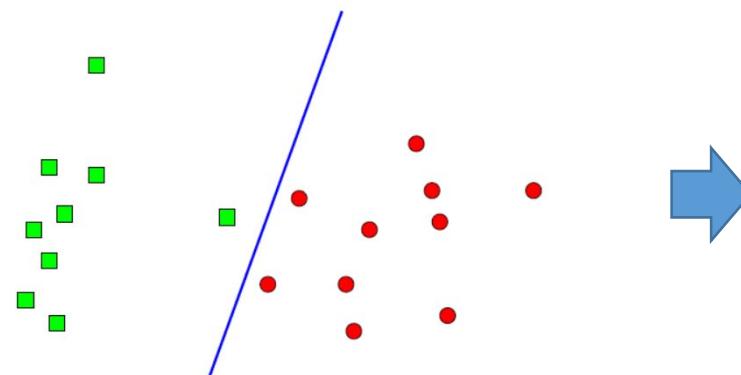


Image credit: Stanford CS231n

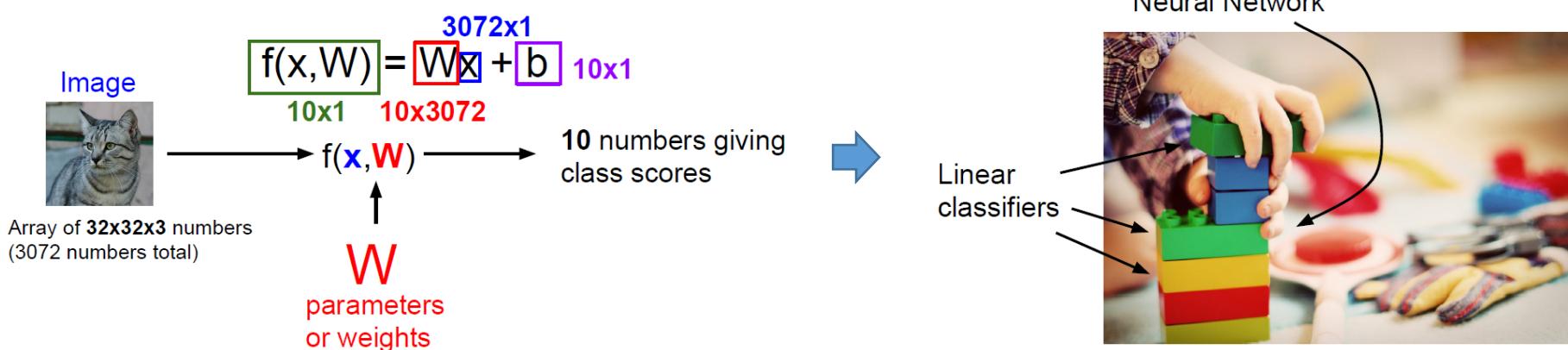
From Binary to Multi-Class Classification



- How to extend binary classifiers for solving multi-class classification?
 - 1-vs.-1 (1-against-1) vs. 1-vs.-all (1-against-rest)

Linear Classification

- Remarks
 - Starting points for many multi-class or complex/nonlinear classifier
 - How to determine a proper loss function for matching \mathbf{y} and $\mathbf{Wx} + \mathbf{b}$, and thus how to learn the model \mathbf{W} (including the bias \mathbf{b}), are the keys to the learning of an effective classification model.



What We Learned Today...

- From Probability to Bayes Decision Rule
- Brief Review of Linear Algebra & Linear System
- Unsupervised vs. Supervised Learning
 - Dimension Reduction, Clustering
 - Training, testing, & validation
 - Linear Classification
- Plus, HW 1 is out (see **Ceiba**) and due next week! (**3/20 Wed 01:00am**)

