# Deep Learning for Computer Vision

Fall 2022

https://cool.ntu.edu.tw/courses/189345 (NTU COOL)

http://vllab.ee.ntu.edu.tw/dlcv.html (Public website)

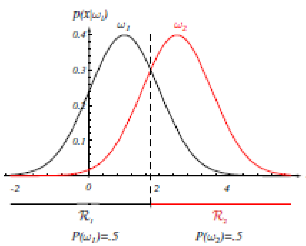
Yu-Chiang Frank Wang 王鈺強, Professor

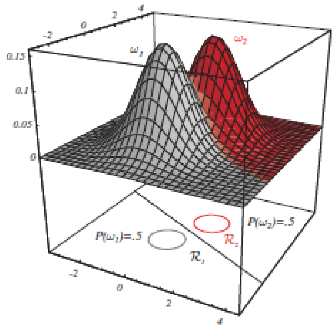
Dept. Electrical Engineering, National Taiwan University

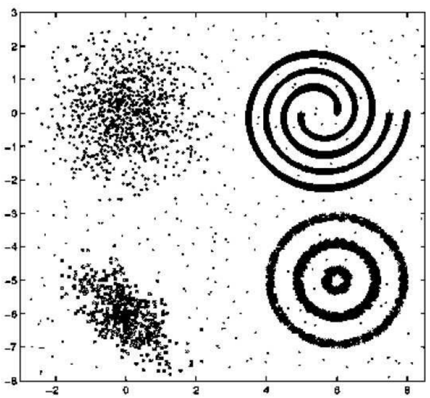
2022/09/06

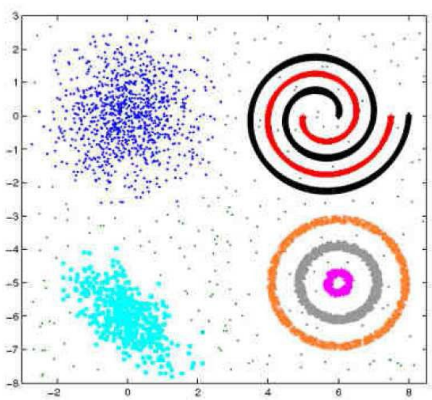
# What’s to Be Covered in This Lecture…

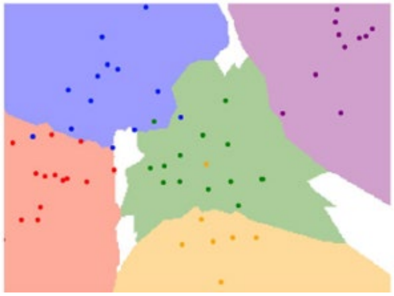
- From Probability to Bayes Decision Rule
- Unsupervised vs. Supervised Learning
  - Clustering & Dimension Reduction
  - Training, testing, & validation
  - Linear Classification





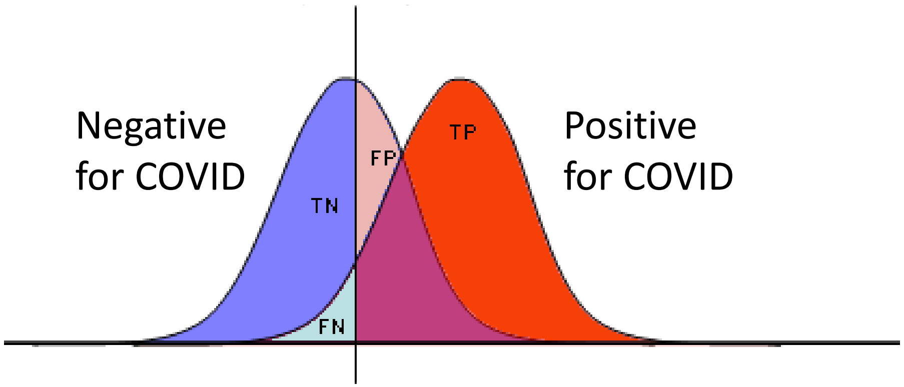




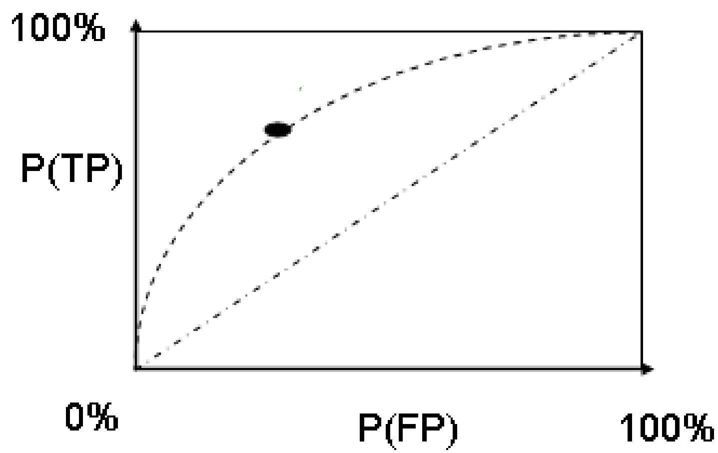
# Example: Testing/Screening of COVID-19

Distributions between positive/negative test results (e.g., PCR, antibody, etc.)

- the further away from each other, the better
- e.g., more accurate COVID diagnosis




| | |
|---|---|
| TP | FP |
| FN | TN |
| 1 | 1 |



# Bayesian Decision Theory

- Fundamental statistical approach to classification/detection tasks
- Take a 2-class classification/detection task as an example:
  - Let's see if a student would pass or fail the course of DLCV,
    with a probabilistic variable $\omega$ (i.e., $\omega = \omega_1$ 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.,
- Decision rule based on priors only
  - If the only available info is the prior, what would be a reasonable decision rule?
  - Decide $\omega_1$ if

    otherwise decide $\omega_2$ .
  - What's the incorrect classification rate (or error rate) $P_e$?

# Class-Conditional Probability Density (or Likelihood)

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

- Here's (hopefully) the hypothetical class-conditional densities reflecting the time of the students spending on DLCV who eventually pass/fail this course.

# 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.


- Bayes formula:

  $P(\omega_j, \boldsymbol{x})$

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

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

- Remark: Posterior probability $P(\omega | \boldsymbol{x})$
  - The probability of a certain state of nature ω given an observable $\boldsymbol{x}$.

# Decision Rule & Probability of Error


- For a given observable ***x*** (e.g., # of GPUs),
  the decision rule (to take DLCV or not) will be now based on:

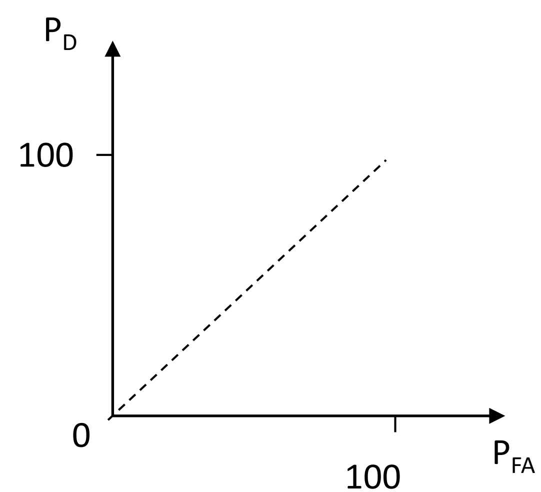
- Hit (detection, TP), false alarm (FA, FP), miss (false reject, FN), rejection (TN)

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

# From Bayes Decision Rule to Detection Theory

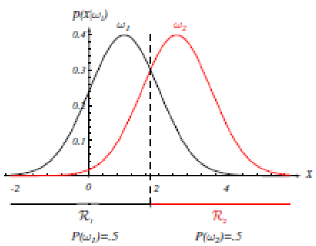
- Hit (detection, TP), false alarm (FA, FP), miss (false reject, FN), rejection (TN)

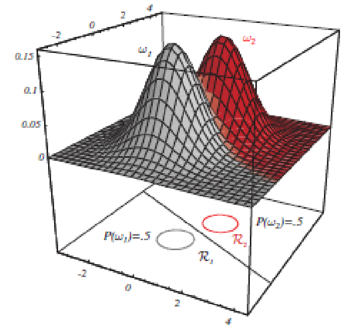
- Receiver Operating Characteristics (ROC)
  - To assess the effectiveness of the designed features/classifiers
  - False alarm ($P_{FA}$ or FP) vs. detection ($P_d$ or TP) rates
  - Which curve/line makes sense? (a), (b), or (c)?

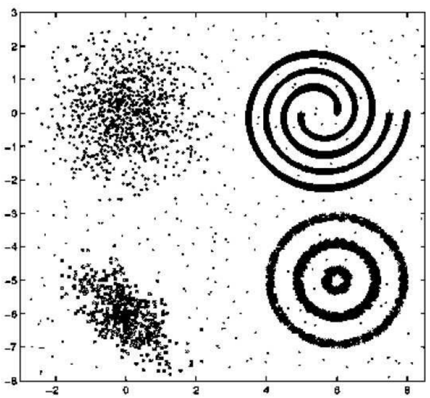


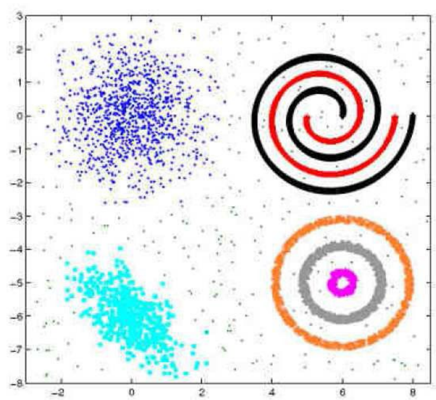
# What's to Be Covered Today…

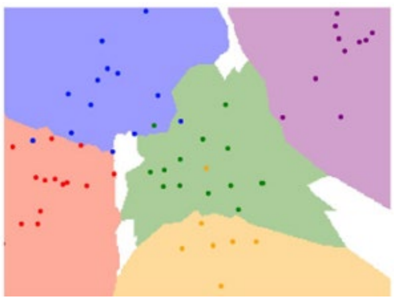
- From Probability to Bayes Decision Rule
- Unsupervised vs. Supervised Learning
    - Clustering & Dimension Reduction
    - Training, testing, & validation
    - Linear Classification





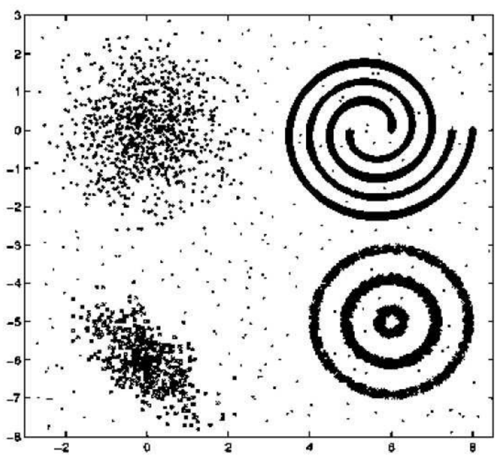




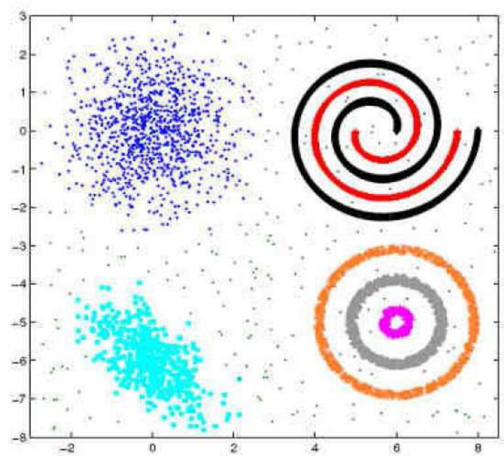


# Clustering

- Clustering is an unsupervised algorithm.
  - Given:
    a set of N unlabeled instances $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$; # 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?



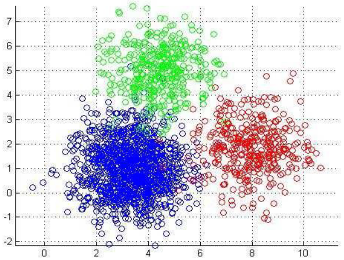
(a) Input data

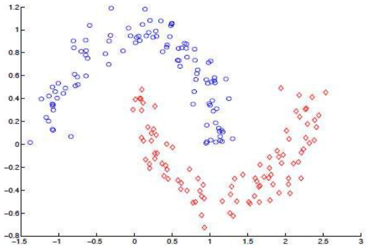


(b) Desired clustering

# Similarity is NOT Always Objective…

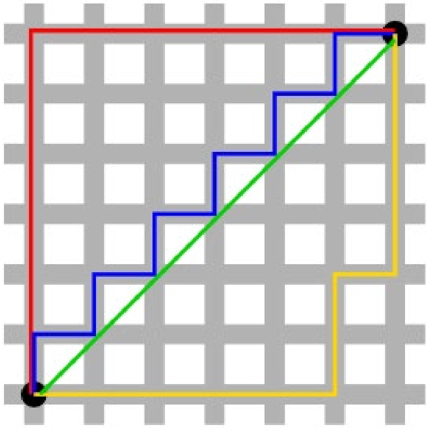
# Clustering (cont'd)





- 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:

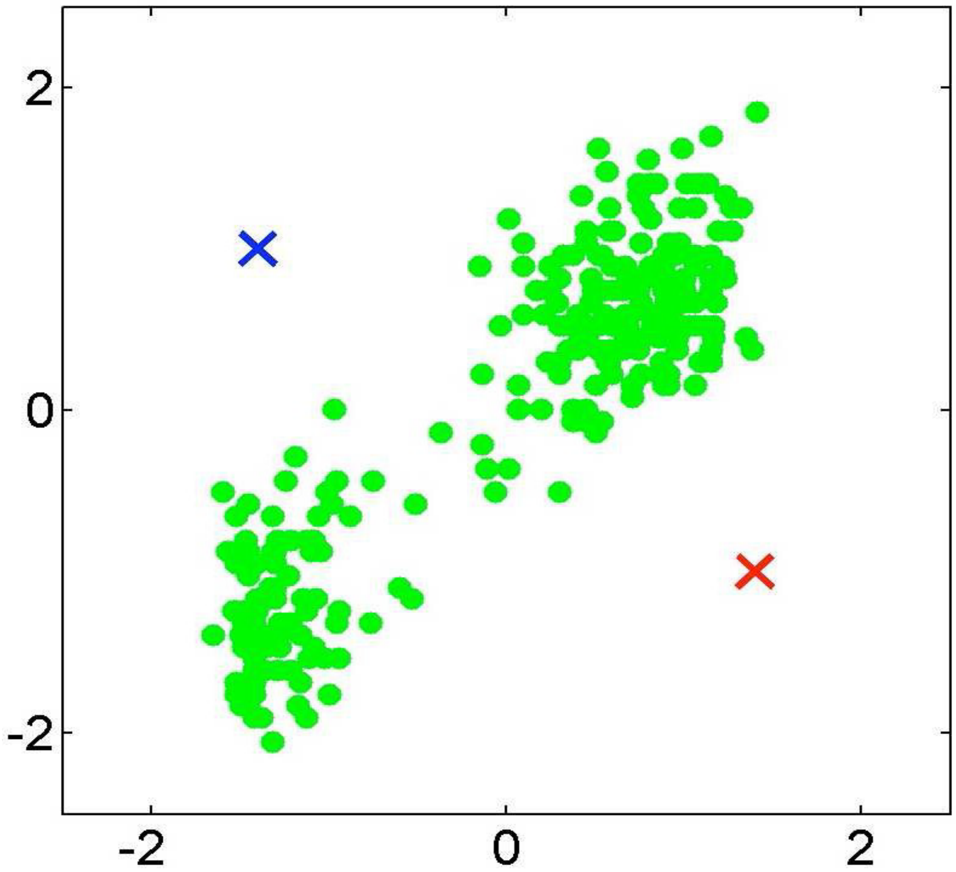


# K-Means Clustering

- Input: $N$ examples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ $(\boldsymbol{x}_n \in \mathrm{R}^D)$; number of partitions $K$
- Initialize: $K$ cluster centers $\mu_1, \ldots, \mu_K$. Several initialization options:
  - Randomly initialize $\mu_1, \ldots, \mu_K$ anywhere in $\mathrm{R}^D$
  - Or, simply choose any $K$ examples as the cluster centers
- Iterate:
  - Assign each of example $\boldsymbol{x}_n$ to its closest cluster center
  - Recompute the new cluster centers $\mu_k$ (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)

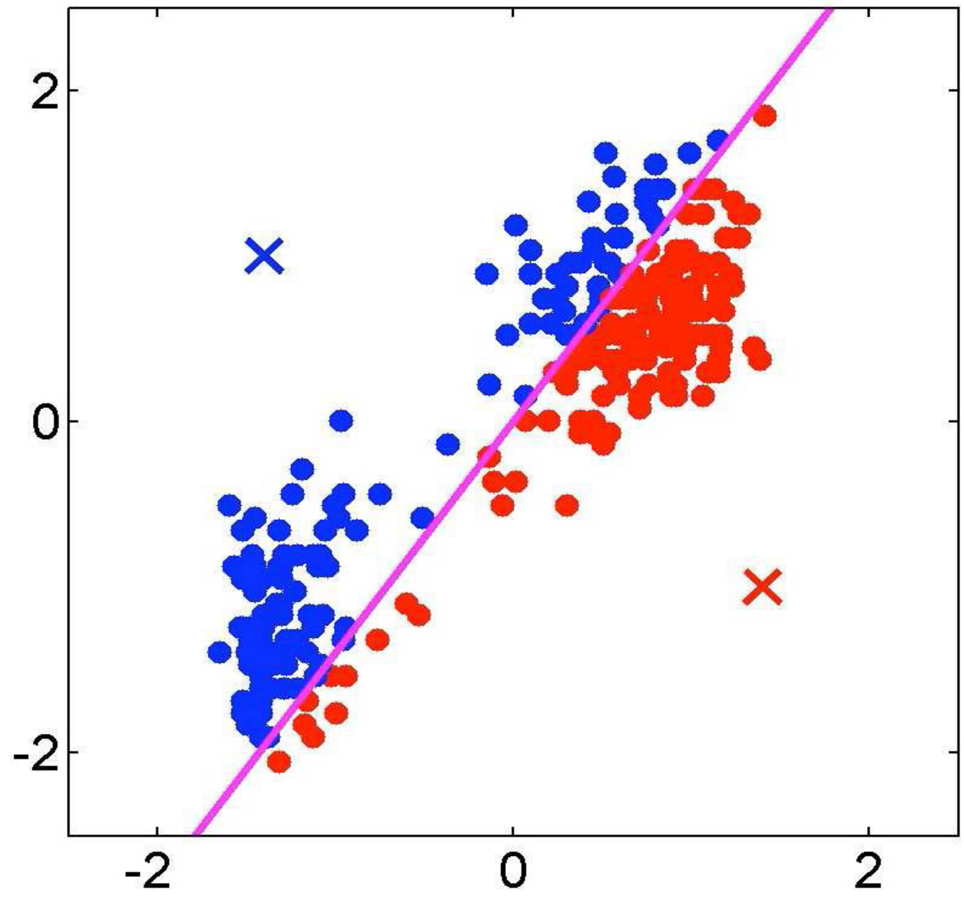
# K-Means Clustering

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



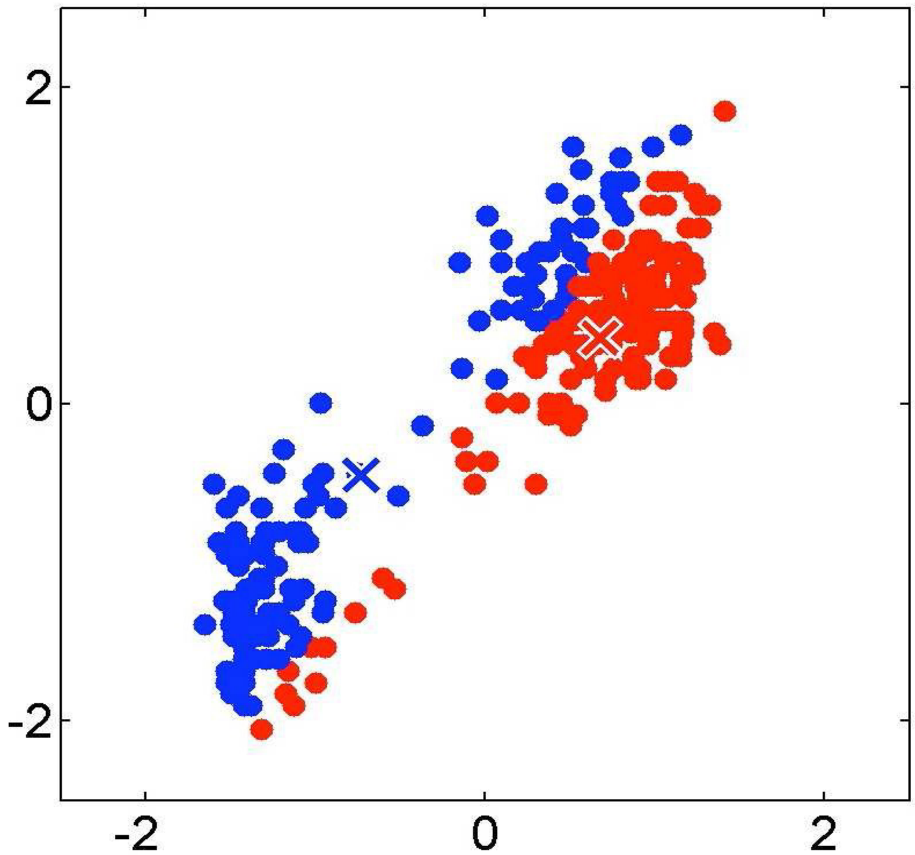
# K-Means Clustering

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



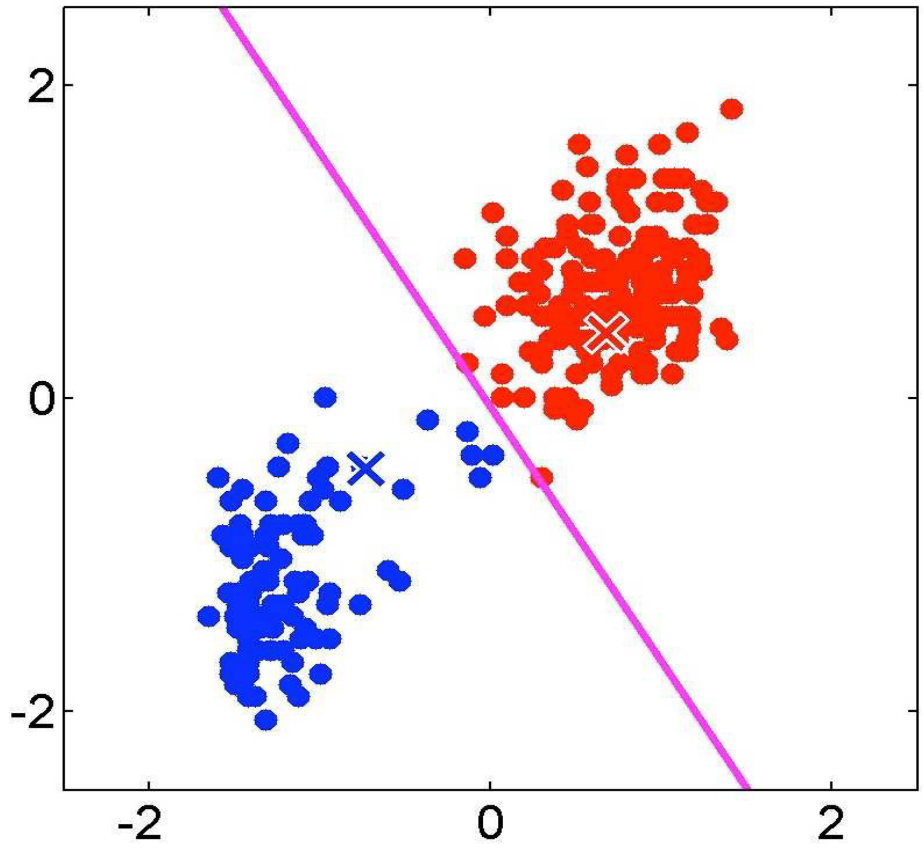
# K-Means Clustering

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



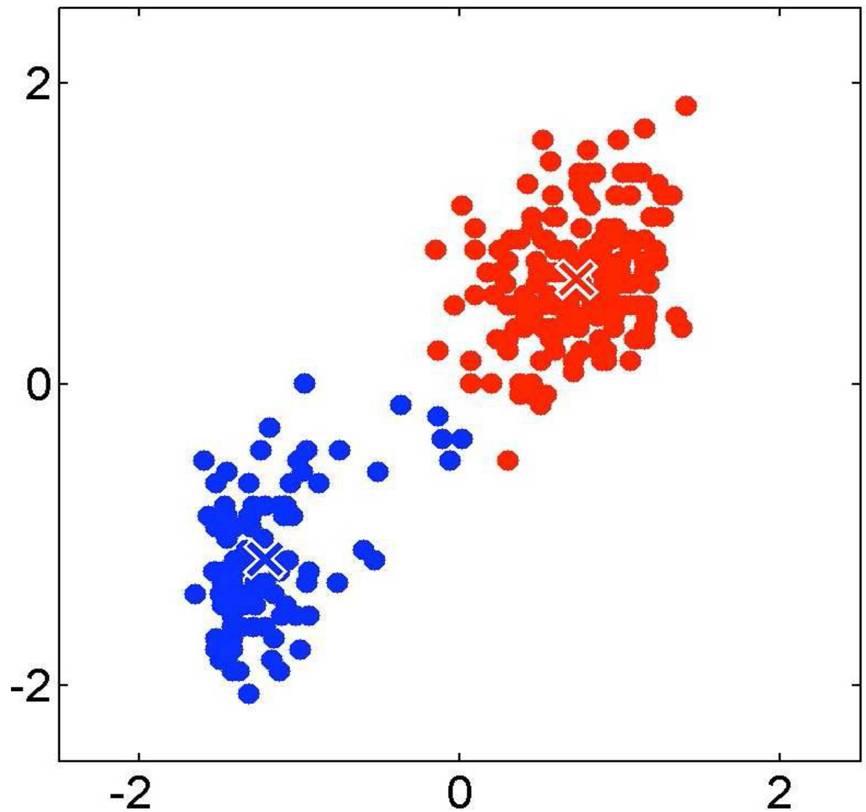
# K-Means Clustering

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



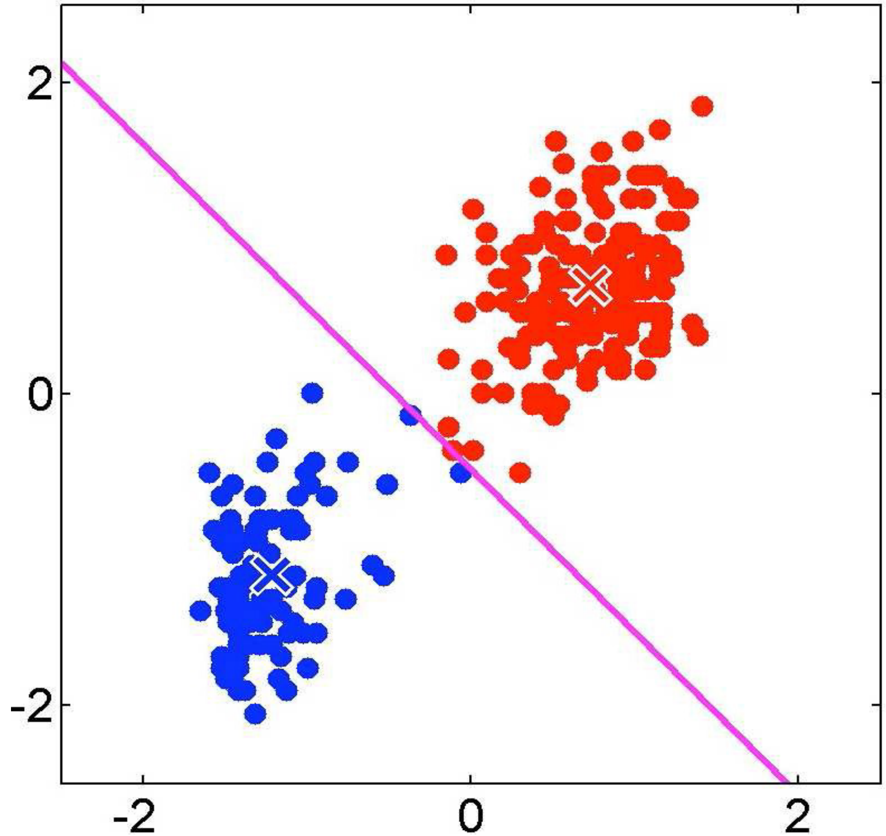
# K-Means Clustering

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



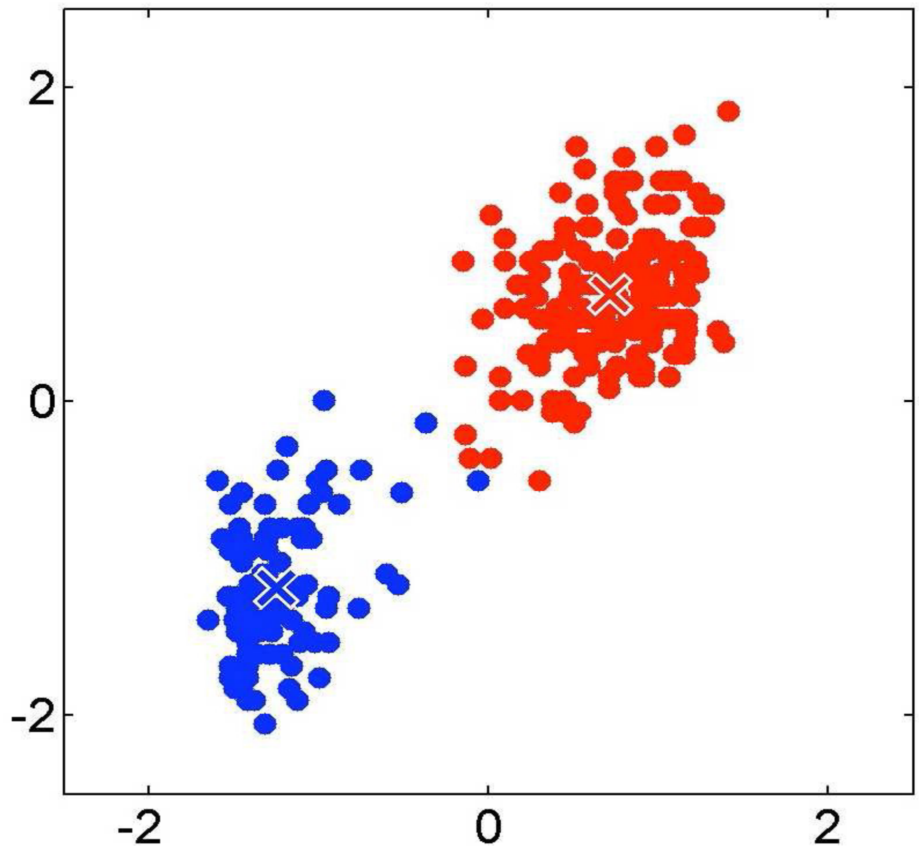
# K-Means Clustering

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



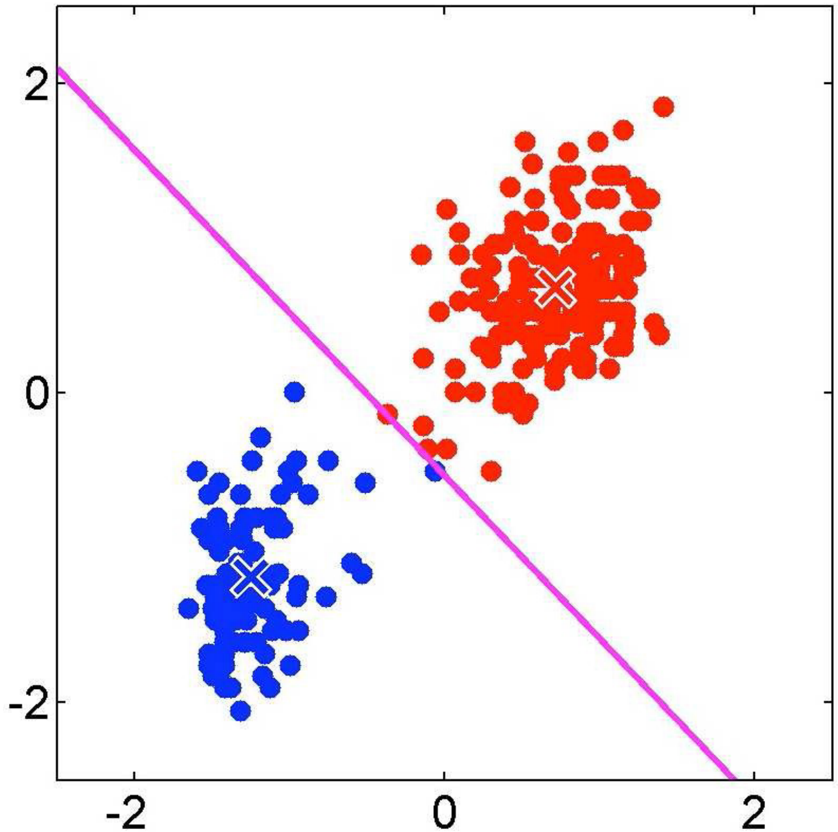
# K-Means Clustering

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



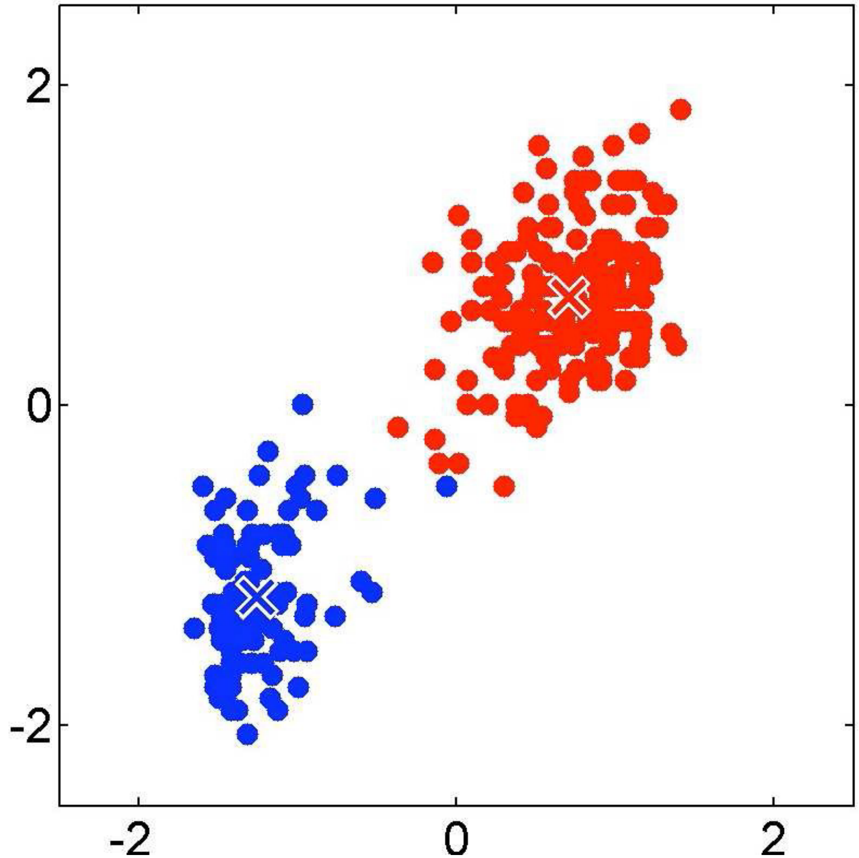
# K-Means Clustering

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



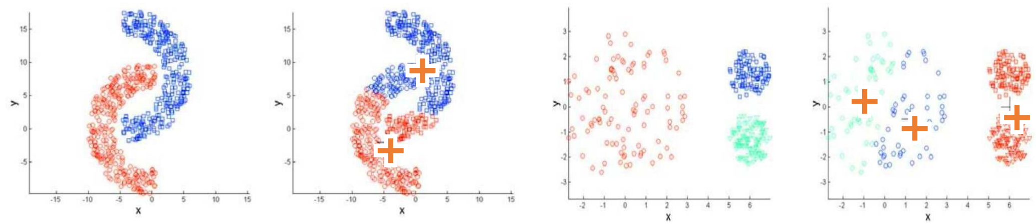
# K-Means Clustering

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



# K-Means Clustering (cont'd)

- Easy to implement, but…
  - Preferable for round shaped clusters with similar sizes

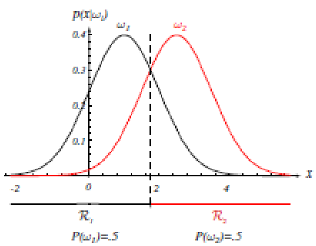


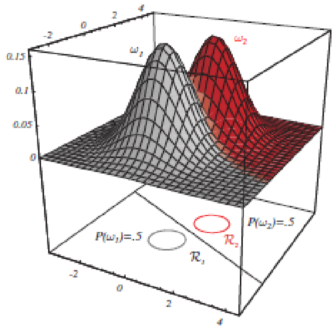
  - Limitations
    - Sensitive to initialization →
    - Sensitive to outliers →
    - Hard assignment only →

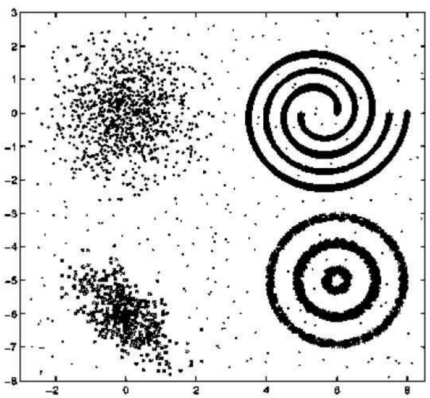
- Remarks
  - Expectation-maximization (EM) algorithm
  - Speed-up possible by hierarchical clustering (e.g., 100 = $10^2$ clusters)

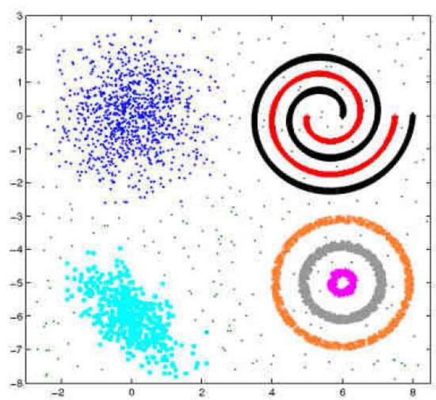
# What’s to Be Covered Today…

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



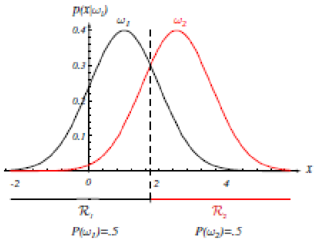


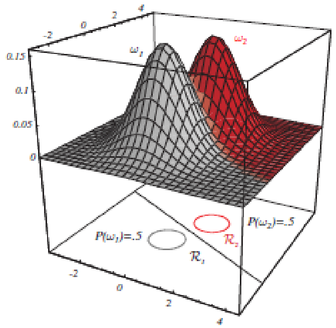


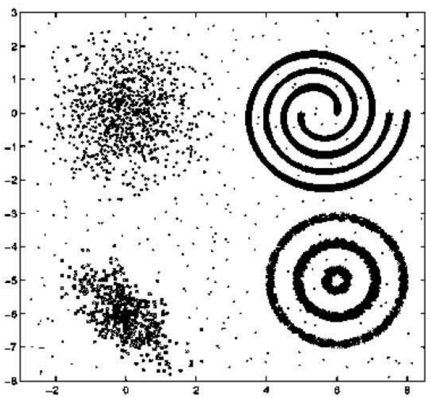


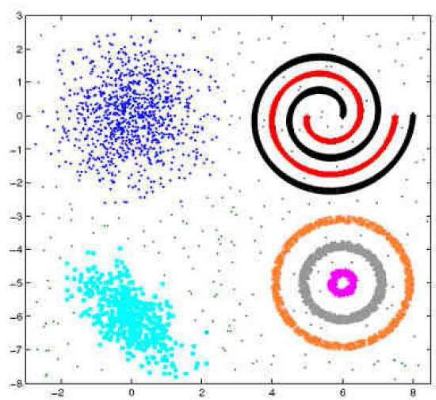
# What’s to Be Covered Today…

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



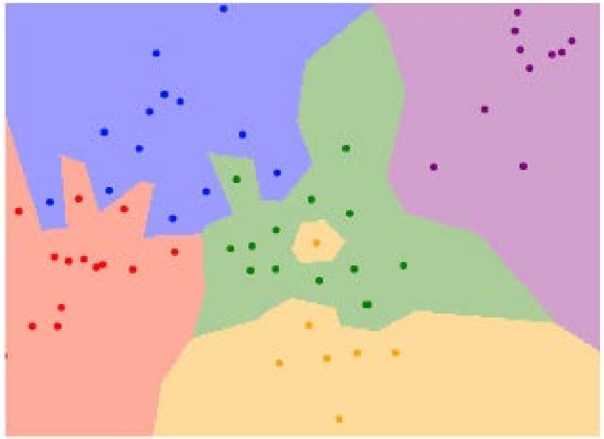




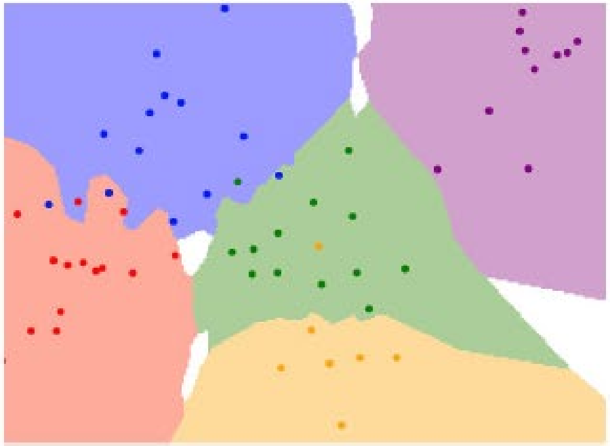


# Hyperparameters in ML

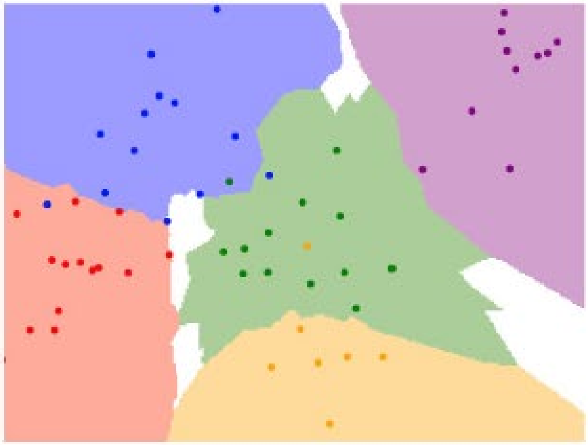
- In many cases, we need to determine the model (hyper)parameters in advance.
  - E.g., for k-NN (k-nearest neighbor classifier), what is the best k value?
  - We need to determine such hyperparameters in an educated way instead of guessing.
  - Let’s see what we should do for hyperparameter selection.



k = 1



k = 3

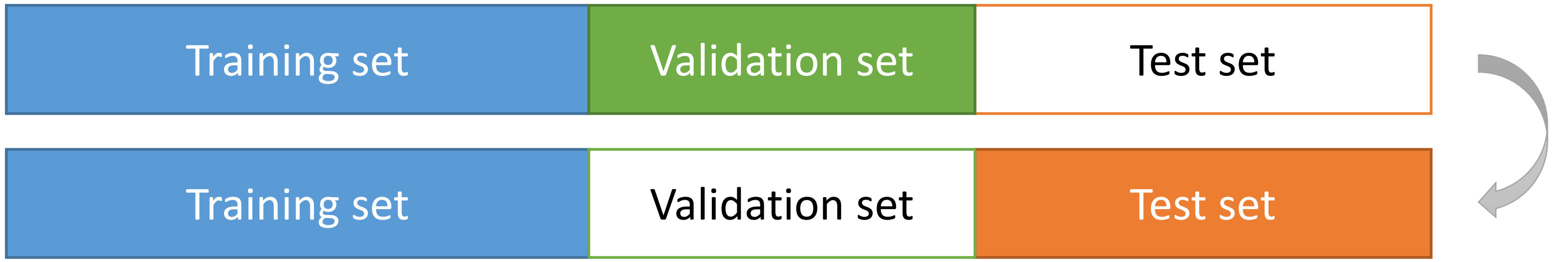


k = 5

Image credit: Stanford CS231n

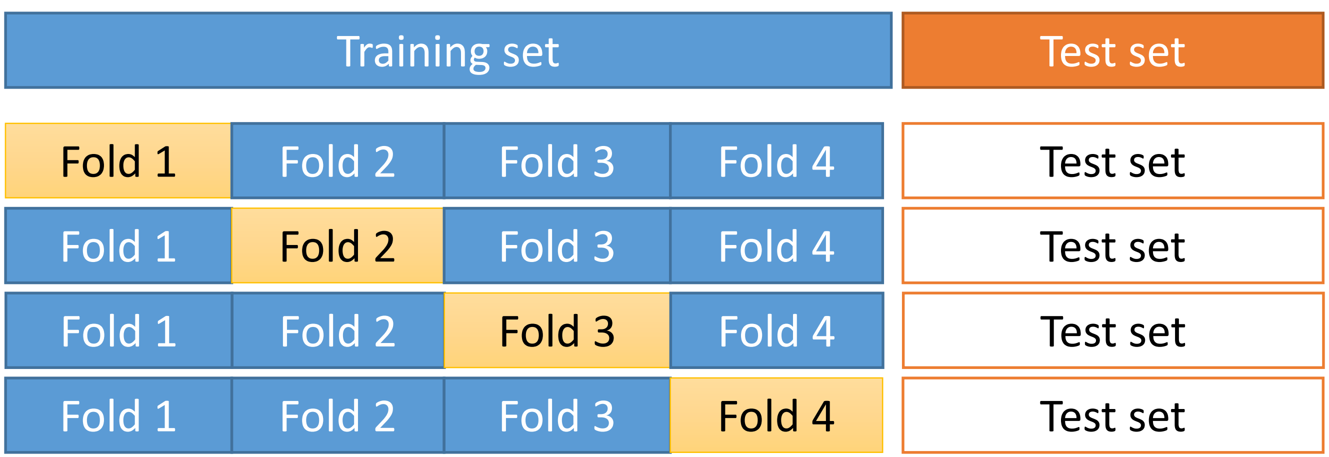
# How to Determine Hyperparameters?

- Use of validation data!
  - For the dataset of interest, it is split it into training, validation, and test sets.
  - You train your model with possible hyperparameter choices (k in k-NN), and select those work best on the validation set.
  - OK, but…



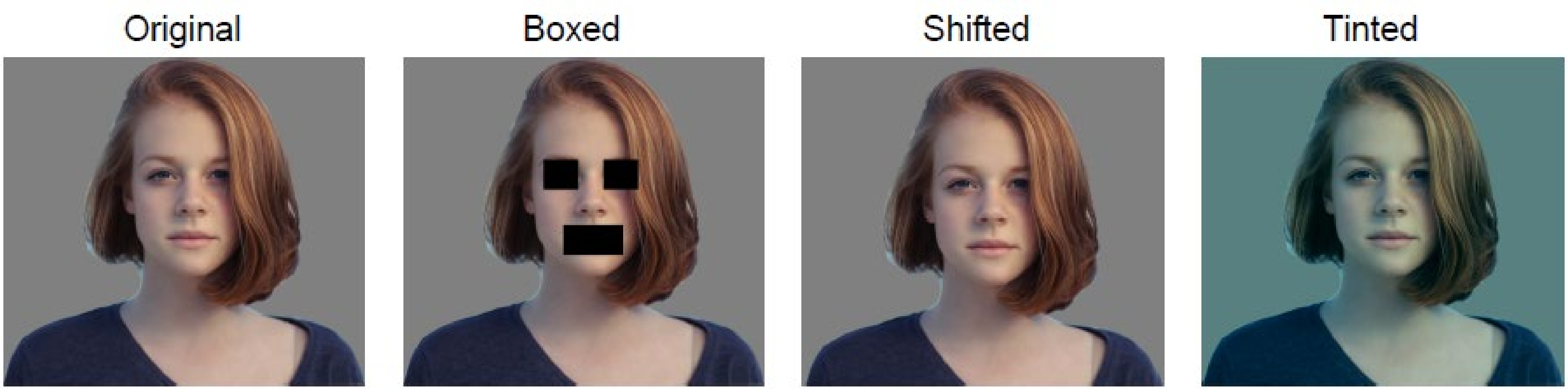
# How to Determine Hyperparameters? (cont'd)

- What if validation data not available?
  - 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) w/ the best average validation performance, followed by training the model using the entire training set.
    - Never access the test set during training!
  - E.g., a 4-fold cross-validation



# Minor Remarks on NN-based Methods

- k-NN is easy to implement but 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 !
    (NN-based methods are viewed as *data-driven* approaches.)

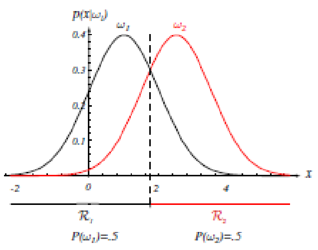


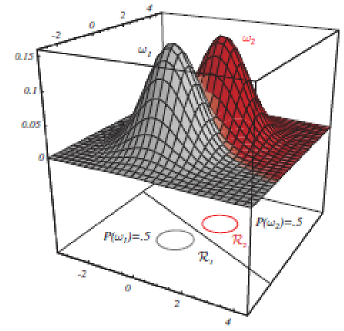

All three images have the same Euclidean distance to the original one.

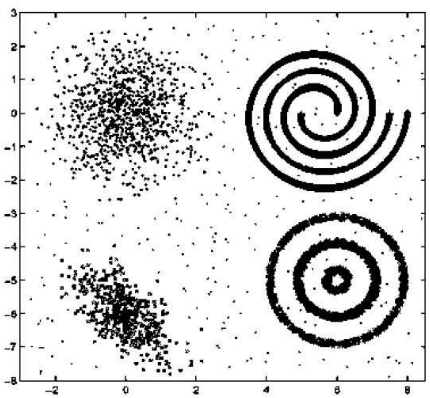
Image credit: Stanford CS231n

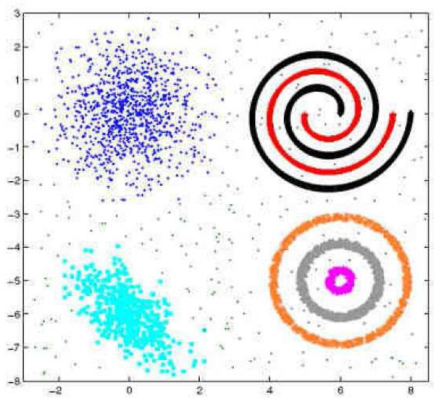
# What’s to Be Covered in This Lecture…

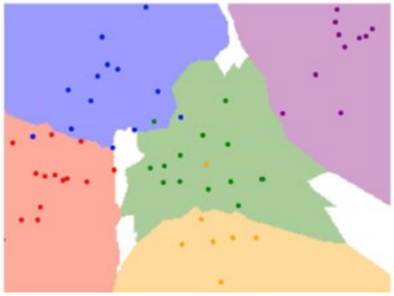
- From Probability to Bayes Decision Rule
- Unsupervised vs. Supervised Learning
  - Clustering & Dimension Reduction
  - Training, testing, & validation
  - Linear Classification





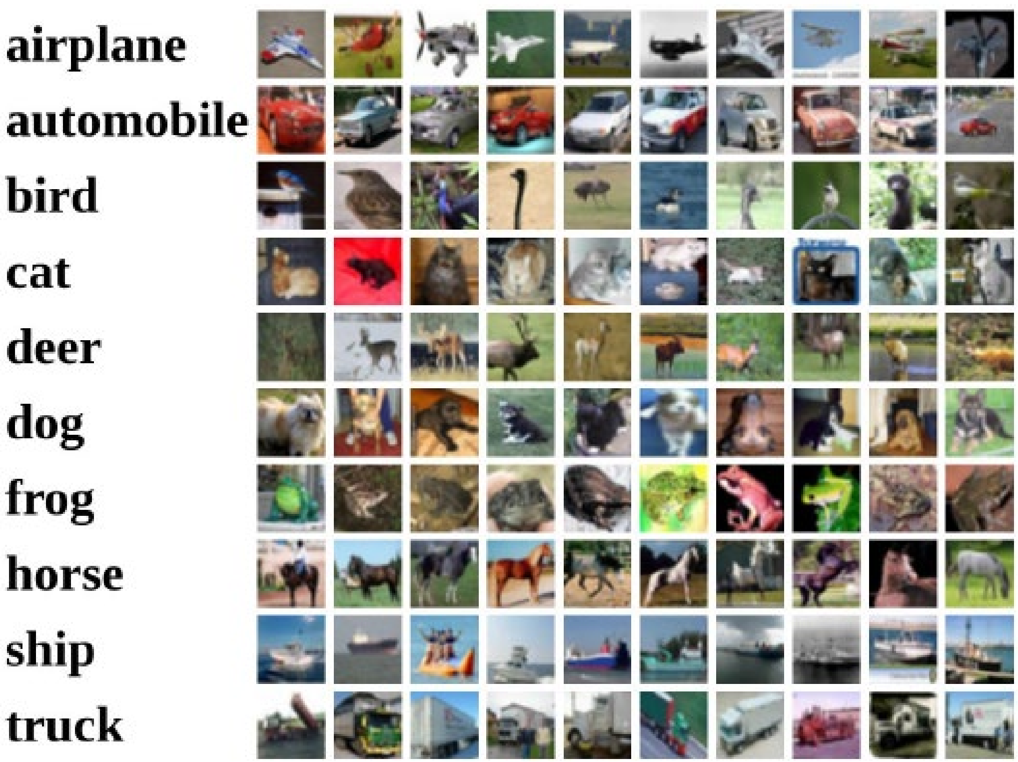






# Linear Classification

- Linear Classifier
  - Can be viewed as a parametric or algebraic approach. Why?
  - Consider that we have 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 or algebraic approach. Why?
  - Consider that we have 10 object categories of interest
  - Let's take the input image as **x**, and the linear classifier as **W**.
    We need **y** = **Wx** + **b** as a 10-dimensional output vector, indicating the score for each class.

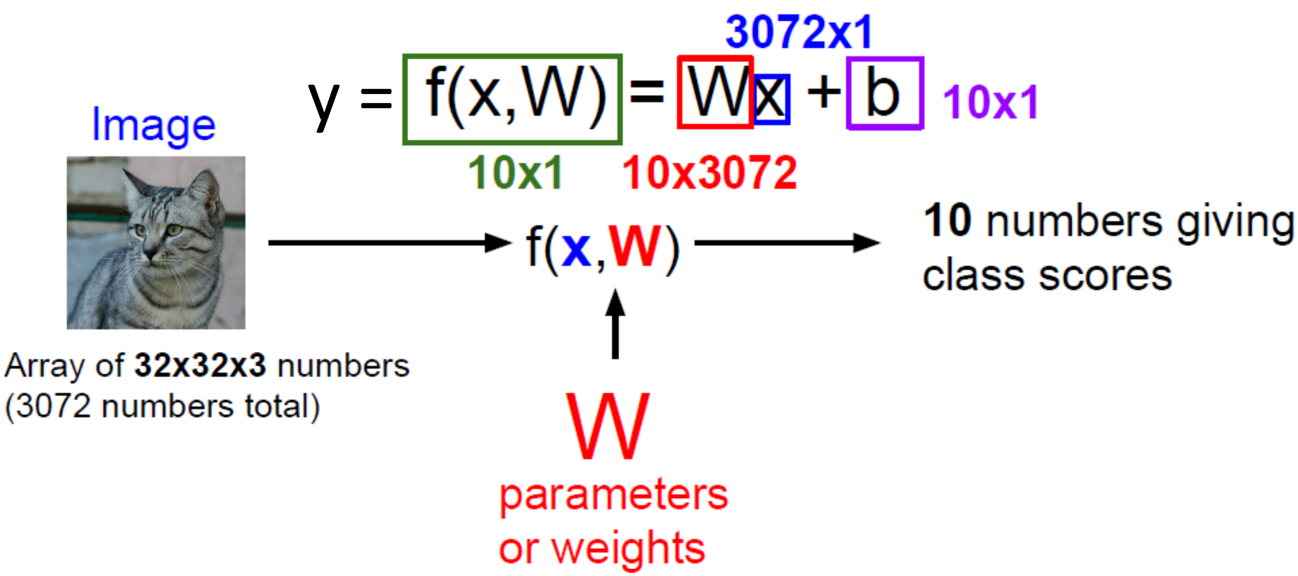



Image credit: Stanford CS231n

# Linear Classification (cont'd)

- Linear Classifier
  - Can be viewed as a parametric or algebraic approach. Why?
  - Consider that we have 10 object categories of interest
  - Let's take the input image as **x**, and the linear classifier as **W**.
    We need **y** = **Wx** + **b** as a 10-dimensional output vector, indicating the score for each class.
  - For example, an image with 2 x 2 pixels & 3 classes of interest
    we need to learn a linear classifier **W** (plus a bias **b**),
    so that desirable outputs **y** = **Wx** + **b** can be expected.

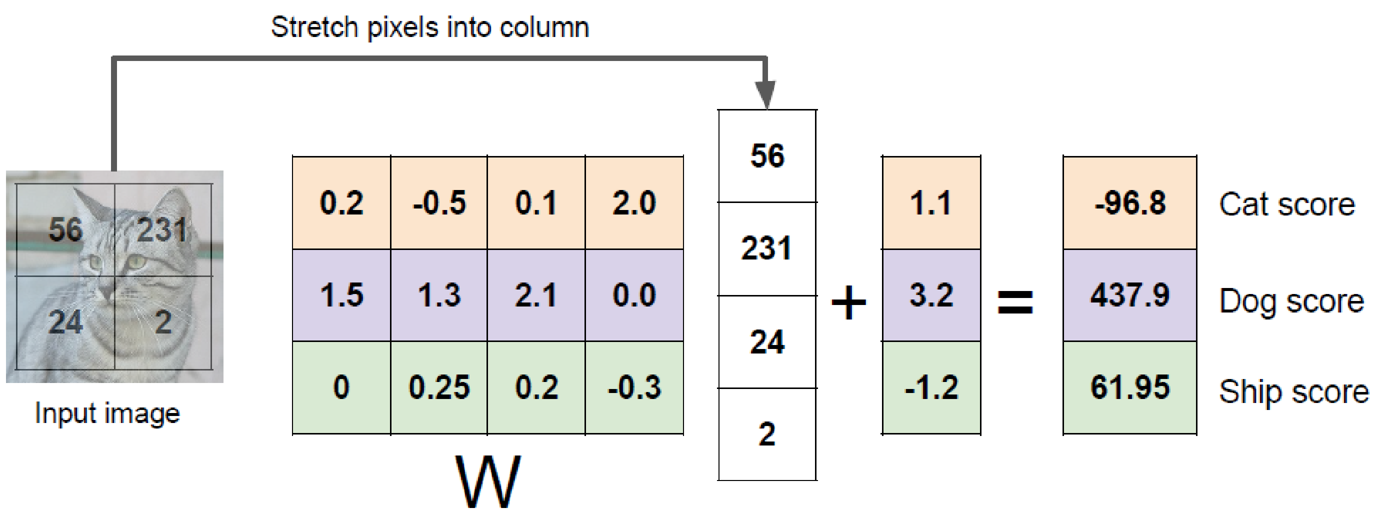
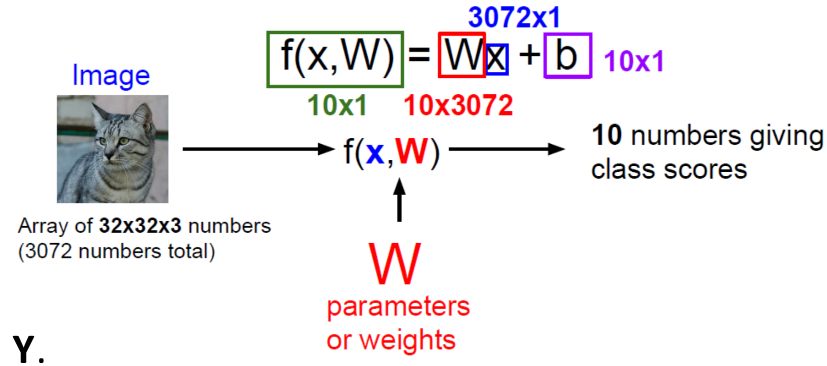



Image credit: Stanford CS231n

# Remarks




- Interpreting **W** in **y** = **Wx** + **b**
  - The weights in **W** are learned by observing training data **X** and their ground truth **Y**.
  - Each row 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, reflecting the corresponding *similarity*.
  - How to determine a proper loss function for matching **y** and **Wx**+**b**, so that **W** can be learned by observing training data?



Image credit: Stanford CS231n

# Loss Function

- **Loss is a function of model parameter W**
  - AKA *objective function*, *cost function*, etc.
  - Tells us how good/bad our learned model **W** in **y** = **Wx** + **b** is. (The lower, the better!)
  - Given a labeled dataset $\{(x_i, y_i)\}_{i=1}^{N}$
    where **x** and **y** indicate the input instance and its label, respectively,

    Loss of a single input instance is denoted as $L_i(f(x_i, W), y_i)$,

    and that for the entire dataset is the sum or average of per-instance losses:

    $$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i) \ .$$

  - In practice, calculating full sum for L is expensive.
    - Approximate sum using a **minibatch** of instances (e.g., 32, 64, 128 samples, etc.)

# Loss Function (cont'd)

- **Cross-Entropy Loss** (Multinomial Logistic Regression)
  - Interpret classifier scores as **probabilities**
  - **Softmax function:**

$$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$ with $s = f(x_i; W)$ as the classifier output for input $\mathbf{x}_i$

  - See example below

| | | exp | Probabilities must be >= 0 | normalize | Probabilities must sum to 1 |
|---|---|---|---|---|---|
| cat | **3.2** | | **24.5** | | **0.13** |
| car | 5.1 | | 164.0 | | 0.87 |
| frog | -1.7 | | 0.18 | | 0.00 |
| | Unnormalized log-probabilities / logits | | unnormalized probabilities | | probabilities |

$$\boxed{L_i = -\log P(Y = y_i \mid X = x_i)}$$

**$L_{cat}$ = -log(0.13) = 2.04**

What about this L?
What are its possibly min/max value??

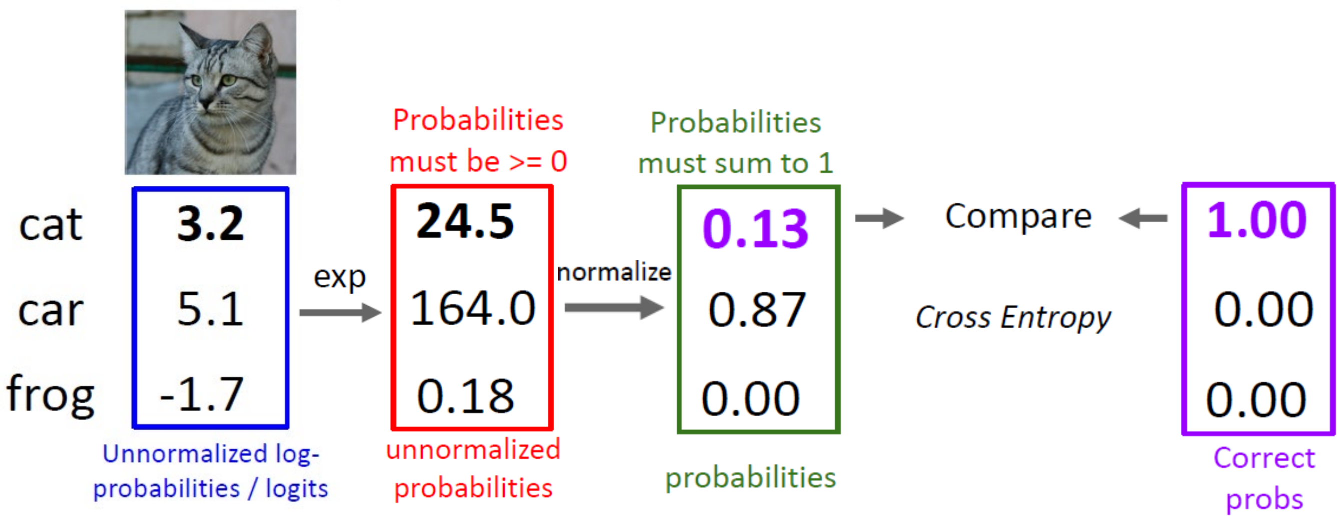
Slide credit: UMich EECS 498-007

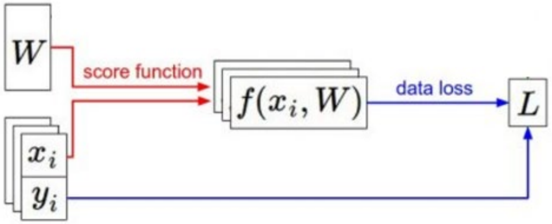
- **Cross-Entropy Loss** (cont'd)
  - **Softmax function:**

$$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$$ with $s = f(x_i; W)$ as the classifier output for input $\mathbf{x}_i$

$$L_i = -\log P(Y = y_i \mid X = x_i)$$ or $$\boxed{L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right)}$$

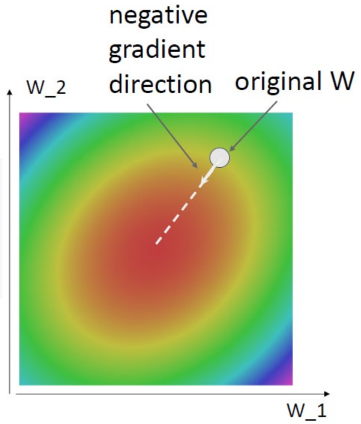
  - **(Binary) Cross Entropy Loss** (or $\mathbf{L_{BCE}}$; see example below):





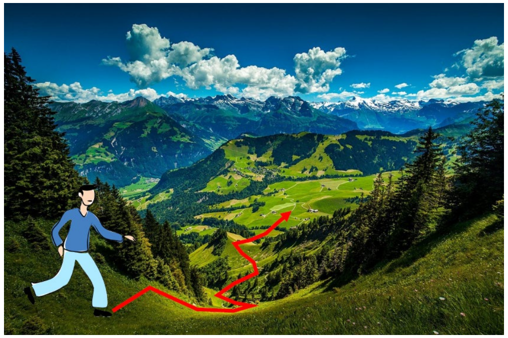

- **Searching for W from $L_{BCE}$**
  - Computing **gradients**:
    Following the slope to reach the (hopefully global) minimum for **W**.
  - **Gradient Descent** via numeric or analytic gradients:
    - Iteratively step in the direction of the negative gradient & search for **W**
    - Hyperparameters: weight initialization, # of steps, learning rate, etc.

```
# Vanilla gradient descent
w = initialize_weights()
for t in range(num_steps):
  dw = compute_gradient(loss_fn, data, w)
  w -= learning_rate * dw
```




  - **Stochastic Gradient Descent**
    - Full sum in L is expensive when large N
    - *Approximate* sum using a minibatch of instances (e.g., 32, 64, 128, etc.)
    - Additional hyperparameters of batch size and data sampling

```
# Stochastic gradient descent
w = initialize_weights()
for t in range(num_steps):
  minibatch = sample_data(data, batch_size)
  dw = compute_gradient(loss_fn, minibatch, w)
  w -= learning_rate * dw
```



Slide credit: UMich EECS 498-007

# What's to Be Covered in This Lecture…

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