Supervised Learning via Generative Models, Unsupervised Learning: Clustering

CS771: Introduction to Machine Learning

Plan today

- Generative models for supervised learning
- Unsupervised Learning
  - Clustering

Supervised Learning via Generative Models

- Idea: Estimate the conditional distribution $p(y|\boldsymbol{x})$ as follows

$$p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, y)}{p(\boldsymbol{x})}$$

Called a "generative model" because the distribution of the inputs $\boldsymbol{x}$ is also modeled

One model is learned, we can even generate "synthetic" $\boldsymbol{x}$ from the model and y, thus "generative"!

- For classification where output y is discrete, this boils down to

$$p(y = k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, y = k)}{p(\boldsymbol{x})} = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})}$$

Both $p(y = k)$ and $p(\boldsymbol{x}|y = k)$ can be estimated using the training data (will soon see how)

$p(\boldsymbol{x}|y = k)$ denotes the distribution of the inputs from class k

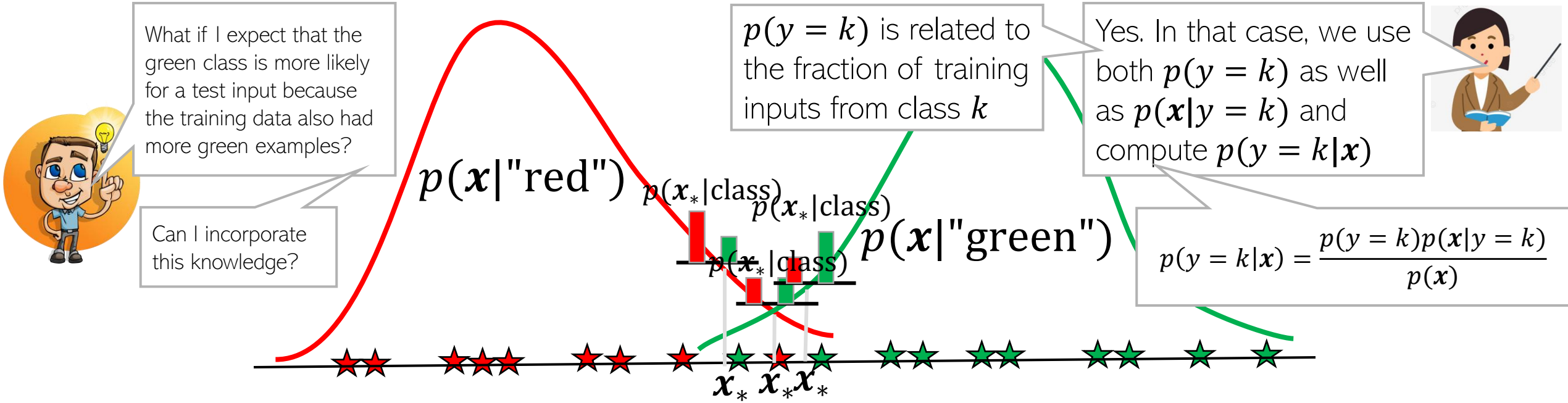
- If we assume $p(y = k)$ to be the same for all the classes then

$$p(y = k|\boldsymbol{x}) \propto p(\boldsymbol{x}|y = k)$$

Just compute the probability of the given input $\boldsymbol{x}$ for each possible class $k = 1,2, \ldots, K$

Generative Classification: An Illustration

- Learn the probability distribution $p(x|y=k)$ of inputs from each class k



- We usually assume some form for $p(x|y=k)$ (e.g., Gaussian) and estimate the parameters of that distribution (MLE/MAP/full posterior)
- We then predict label of test input $\boldsymbol{x}_*$ by comparing probabilities under each class
  - Or can report the probability of belonging to each class (soft prediction)

Generative Classification: Some Nomenclature

- Recall the general form of generative classification

$$p(y = k|\boldsymbol{x}) = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})}$$

- $p(y = k)$: "Class marginal" or "class prior" probability
  - Without looking at the input $\boldsymbol{x}$, what's the probability that $\boldsymbol{x}$ belongs to class k
- $p(\boldsymbol{x}|y = k)$: "Class-conditional distribution" of the inputs
  - The probability distribution/density of inputs from class k
- $p(\boldsymbol{x})$: "Marginal distribution" of the inputs, and by definition it is

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(y = k)p(\boldsymbol{x}|y = k)$$

- We use the training data to estimate the class-marginal and class-conditionals

Fraction of training inputs from class k can be a rough proxy for $p(y = k)$

No need to estimate $p(\boldsymbol{x})$ since it is defined in terms of class marginals and class conditionals

Estimating Class Marginals

- Estimating class marginals $p(y = k)$ is usually straightforward
- Since labels are discrete, we assume class marginal $p(y)$ to be a multinoulli

If only two classes, assume Bernoulli

$\pi_k = p(y = k)$

These probabilities sum to 1: $\sum_{k=1}^{K} \pi_k = 1$

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \pi_2, \ldots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

- Given N i.i.d. labelled examples $\{(x_n, y_n)\}_{n=1}^{N}$, $y_n \in \{1,2,\ldots,K\}$ the MLE soln

$$\boldsymbol{\pi}_{MLE} = \underset{\boldsymbol{\pi}}{\text{argmax}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{\pi})$$

Subject to constraint $\sum_{k=1}^{K} \pi_k = 1$

- MLE solution is $p(y = k) = \pi_k = N_k/N$ where $N_k = \sum_{n=1}^{N} \mathbb{I}[y = k]$
  - Thus $p(y = k) = \pi_k$ is simply the fraction of inputs from class k
- Can also compute MAP estimate or full posterior of $\boldsymbol{\pi}$ using a Dirichlet prior

Estimating Class-Conditionals

To be estimated using the N_k training inputs $\{\boldsymbol{x}_n : y_n = k\}$ from class k

- Can assume a distribution $p(\boldsymbol{x}|y = k) = p(\boldsymbol{x}|\theta_k)$ for inputs of each class k
- If $\boldsymbol{x}$ is D-dimensional, $p(\boldsymbol{x}|\theta_k)$ will be a D-dimensional distribution
- Can compute MLE/MAP estimate or full posterior of θ_k
  - This essentially is a density estimation problem for the class-cond.
  - In principle, can use any density estimation method
- Choice of the form of $p(\boldsymbol{x}|\theta_k)$ depends on various factors
  - Nature of input features, e.g.,
    - If $\boldsymbol{x} \in \mathbb{R}^D$, can use a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
    - If $\boldsymbol{x} \in \{0,1\}^D$, can use D Bernoullis (one for each feature)
    - Can also choose other more sophisticated distributions
  - Amount of training data available (important)
    - If D large and N_k small, it will be difficult to get a good estimate θ_k

E.g., if $p(\boldsymbol{x}|\theta_k)$ is multivariate Gaussian then assume it to have a diagonal covariance matrix instead of full covariance matrix

Such assumptions greatly reduce the number of parameters to be estimated

In such cases, we may need to regularize θ_k or make some simplifying assumptions on $p(\boldsymbol{x}|\theta_k)$, such as features being conditionally independent given class e.g., $p(\boldsymbol{x}|\theta_k) = \prod_{d=1}^{D} p(x_d|\theta_{kd})$ - naïve Bayes

Especially if the number of features (D) is very large because large value of D means k consists of a large number of parameters (e.g., in the Gaussian case, $\theta_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, D params for $\boldsymbol{\mu}_k$ and $O(D^2)$ params for $\boldsymbol{\Sigma}_k$. Can overfit

Generative Classification: At Test Time

- Recall the form of the conditional distribution of the label

Class-marginal accounts for the frequency of class k labels in the training data

Thus generative classification explicitly takes into account the shape and size of each class

Class-conditional distribution of inputs accounts for the shape/spread of class k

$$p(y_* = k|\boldsymbol{x}_*) = \frac{p(y_* = k) \times p(\boldsymbol{x}_*|y_* = k)}{p(\boldsymbol{x}_*)}$$

$$\propto p(y_* = k) \times p(\boldsymbol{x}_*|y_* = k)$$

Assuming we are using point estimates for the class marginal and class conditional

$$\propto \hat{\pi}_k \times p(\boldsymbol{x}_*|\hat{\theta}_k)$$

Interpretation: Probability of $\boldsymbol{x}_*$ belonging to class k is proportional to the fraction of training inputs from class k times the probability of $\boldsymbol{x}_*$ under the distribution of inputs from class k

- If we assume the class-marginal to be uniform ($p(y_* = k) = 1/K$) then

$$p(y_* = k|\boldsymbol{x}_*) \propto p(\boldsymbol{x}_*|\hat{\theta}_k)$$

- The most likely label is $y_* = \text{argmax}_{k \in \{1,2,\ldots,K\}} p(y_* = k|\boldsymbol{x}_*)$

An Example: Gaussian Class-conditionals

- The generative classification model $p(y=k|\boldsymbol{x}) = \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}$

A benefit of modeling each class by a distribution (recall that LwP had issues)

- Assume each class-conditional $p(\boldsymbol{x}|y=k)$ to be a Gaussian

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_k|}}\exp[-(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)]$$

Since the Gaussian's covariance models its shape, we can learn the shape of each class ☺

- Class marginal is multinoulli $p(y=k) = \pi_k, \pi_k \in (0,1), \sum_{k=1}^K \pi_k = 1$
- Let's denote the parameters of the model collectively by $\theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
  - Can estimate these using MLE/MAP/Bayesian inference
  - The MLE solution for $\boldsymbol{\pi}$: $\pi_k = N_k/N$ (exercise)
  - MLE solution for $\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{y_n=k}\boldsymbol{x}_n$, $\boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{y_n=k}(\boldsymbol{x}_n-\boldsymbol{\mu}_k)(\boldsymbol{x}_n-\boldsymbol{\mu}_k)^\top$

Can also do MAP estimation for $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ using a Gaussian prior on $\boldsymbol{\mu}_k$ and inverse Wishart prior on $\boldsymbol{\Sigma}_k$

Exercise: Try to derive this. I will provide a separate note containing the derivation

- If using point est (MLE/MAP) for θ, predictive distribution will be

$$p(y_*=k|\boldsymbol{x}_*,\theta) = \frac{\pi_k|\boldsymbol{\Sigma}_k|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}_*-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_*-\boldsymbol{\mu}_k)\right]}{\sum_{k=1}^K \pi_k|\boldsymbol{\Sigma}_k|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}_*-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_*-\boldsymbol{\mu}_k)\right]}$$

Can predict the most likely class for the test input $\boldsymbol{x}_*$ by comparing these probabilities for all values of k

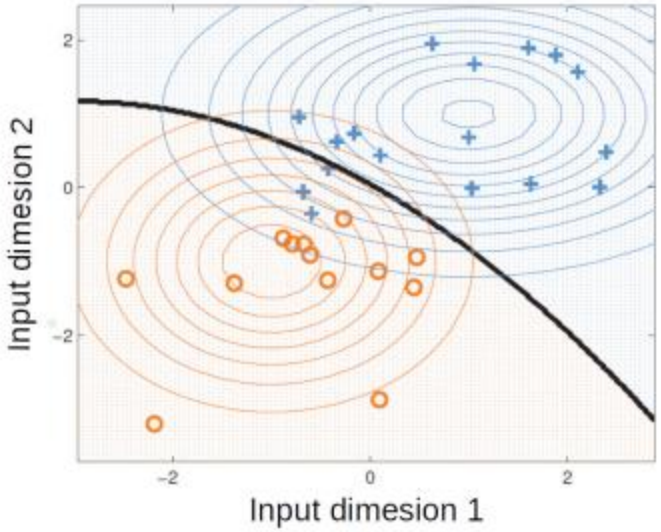
Note that the exponent has a Mahalanobis distance like term. Also, accounts for the fraction of training examples in class k

Decision Boundary with Gaussian Class-Conditional

- As we saw, the prediction rule when using Gaussian class-conditional

$$p(y=k|\boldsymbol{x},\theta)=\frac{\pi_k|\boldsymbol{\Sigma_k}|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K}\pi_k|\boldsymbol{\Sigma_k}|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}$$

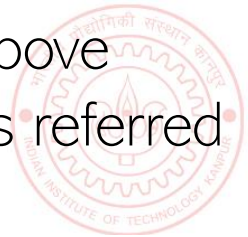
- The decision boundary between any pair of classes will be a quadratic curve



Reason: For any two classes k and k' at the decision boundary, we will have $p(y=k|x,\theta)=p(y=k'|x,\theta)$. Comparing their logs and ignoring terms that don't contain $\boldsymbol{x}$, can easily see that

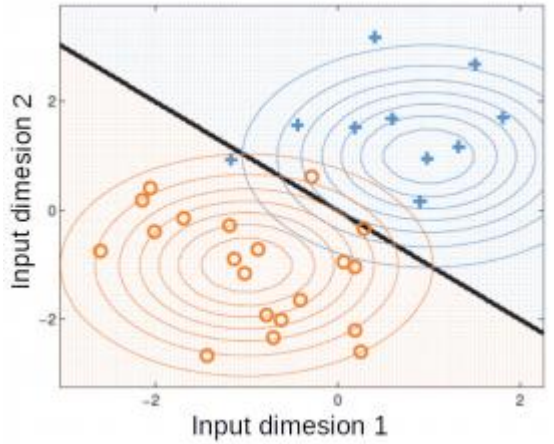
$$(\mathbf{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)-(\mathbf{x}-\boldsymbol{\mu}_{k'})^\top\boldsymbol{\Sigma}_{k'}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{k'})=0$$

Decision boundary contains all inputs $\boldsymbol{x}$ that satisfy the above. This is a quadratic function of $\boldsymbol{x}$ (this model is sometimes referred to Quadratic Discriminant Analysis)



Decision Boundary with Gaussian Class-Conditional

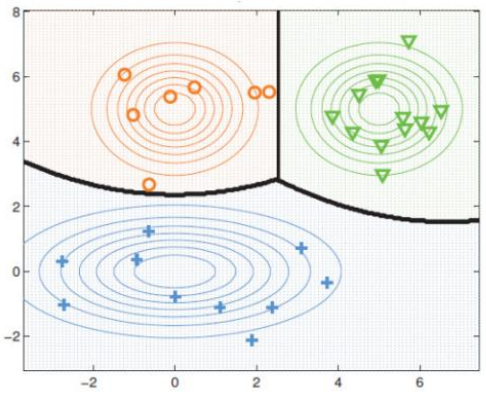
- Assume all classes are modeled using the same covariance matrix $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$
- In this case, the decision boundary b/w any pair of classes will be linear

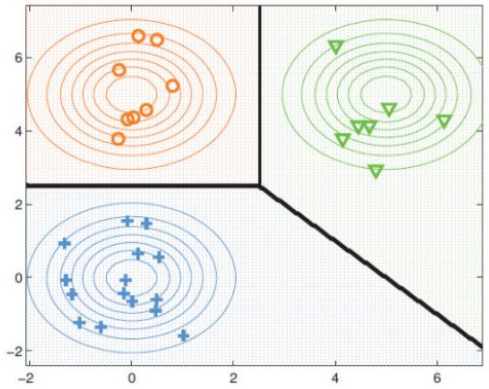


Reason: Again using $p(y = k|x, \theta) = p(y = k'|x, \theta)$, comparing their logs and ignoring terms that don't contain $\boldsymbol{x}$, we have

$$(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - (\mathbf{x} - \boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{k'}) = 0$$

Quadratic terms of $\boldsymbol{x}$ will cancel out; only linear terms will remain; hence decision boundary will be a linear function of $\boldsymbol{x}$ (Exercise: Verify that we can indeed write the decision boundary between this pair of classes as $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ where $\boldsymbol{w}$ and b depend on $\boldsymbol{\mu}_k$, $\boldsymbol{\mu}_{k'}$ and $\boldsymbol{\Sigma}$)





If we assume the covariance matrices of the Gaussian class-conditionals for any pair of classes to be equal, then the learned separation boundary b/w this pair of classes will be linear; otherwise, quadratic as shown in the figure on left

A Closer Look at the Linear Case

- For the linear case (when $\boldsymbol{\Sigma_k = \Sigma, \forall k}$), the class conditional probability

$$p(y = k|\boldsymbol{x}, \theta) \propto \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

- Expanding further, we can write the above as

$$p(y = k|\boldsymbol{x}, \theta) \propto \exp\left[\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k\right] \exp\left[\boldsymbol{x}^\top \Sigma^{-1}\boldsymbol{x}\right]$$

- Therefore, the above class posterior probability can be written as

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}{\sum_{k=1}^{K} \exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}$$

$$\boldsymbol{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k \qquad b_k = -\tfrac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k$$

If all Gaussians class-cond have the same covariance matrix (basically, of all classes are assumed to have the same shape)

- The above has *exactly* the same form as softmax classification

A Very Special Case: LwP Revisited

- Note the prediction rule when $\boldsymbol{\Sigma_k = \Sigma, \forall k}$

$$\hat{y} = \arg\max_k p(y = k|\boldsymbol{x}) = \arg\max_k \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

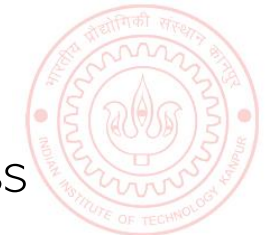
$$= \arg\max_k \log \pi_k - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

- Also assume all classes to have equal no. of training examples, i.e., $\boldsymbol{\pi_k = 1/K}$. Then

$$\hat{y} = \arg\min_k (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

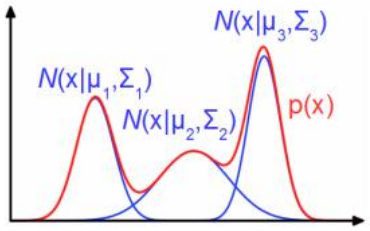
The Mahalanobis distance matrix = $\boldsymbol{\Sigma^{-1}}$

- Equivalent to assigning $\boldsymbol{x}$ to the "closest" class in terms of a Mahalanobis distance
- If we further assume $\boldsymbol{\Sigma = I_D}$ then the above is exactly the LwP rule
  - Thus LwP assumes spherical classes with roughly equal number of inputs from each class



Unsupervised Generative Classification

- In generative classification, we estimate
  - Class marginal distribution $p(y) = \text{multinoulli}(y|\boldsymbol{\pi})$
  - K class-conditional distributions $p(x|\theta_k), k = 1,2,\ldots,K$



- Can we estimate $\{\pi, \theta_1, \theta_2, \ldots, \theta_K\}$ if the training labels are not known?

- It then becomes an unsupervised learning problem
  - Mixture modeling or clustering

- We will look at it later but the general idea is based on ALT-OPT
  1. Guess the label of each input given current estimate of $\{\pi, \theta_1, \theta_2, \ldots, \theta_K\}$
  2. Re-estimate $\{\pi, \theta_1, \theta_2, \ldots, \theta_K\}$ given the label guesses
  3. Alternate between steps 1 and 2 till convergence

Prediction step of generative classification

Parameter estimation step of generative classification

Discriminative vs Generative

Proponents of discriminative models: Why bother modeling $\boldsymbol{x}$ if $\boldsymbol{y}$ is what you care about? Just model $\boldsymbol{y}$ directly instead of working hard to model $\boldsymbol{x}$ by learning the class-conditional

- Recall that discriminative approaches model $p(\boldsymbol{y}|\boldsymbol{x})$ directly
- Generative approaches model $p(\boldsymbol{y}|\boldsymbol{x})$ via $p(\boldsymbol{x}, \boldsymbol{y})$
- Number of parameters: Discriminative models have fewer parameters to be learned
  - Just the weight vector/matrix $\boldsymbol{w}/\boldsymbol{W}$ in case of logistic/softmax classification
- Ease of parameter estimation: Debatable as to which one is easier
  - For "simple" class-conditionals, easier for gen. classifn model (often closed-form solution)
  - Parameter estimation for discriminative models (logistic/softmax) usually requires iterative methods (although objective functions usually have global optima)
- Dealing with missing features: Generative models can handle this easily
  - E.g., by integrating out the missing features while estimating the parameters (will see later)
- Inputs with features having mixed types: Generative model can handle this
  - Appropriate $p(\boldsymbol{x}_d|\boldsymbol{y})$ for each type of feature in the input. Difficult for discriminative models

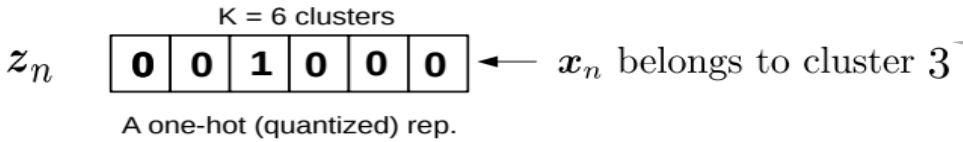
Discriminative vs Generative (Contd)

- Leveraging unlabeled data: Generative models can handle this easily by treating the missing labels are latent variables and are ideal for Semi-supervised Learning. Discriminative models can't do it easily
- Adding data from new classes: Discriminative model will need to be re-trained on all classes all over again. Generative model will just require estimating the class-cond of newly added classes
- Have lots of labeled training data? Discriminative models usually work very well
- Final Verdict? Despite generative classification having some clear advantages, both methods can be quite powerful (the actual choice may be dictated by the problem)
  - Important to be aware of their strengths/weaknesses, and also the connections between these
- Possibility of a Hybrid Design? Yes, Generative and Disc. models can be combined, e.g.,
  - "Principled Hybrids of Generative and Discriminative Models" (Lassere et al, 2006)
  - "Deep Hybrid Models: Bridging Discriminative & Generative Approaches" (Kuleshov & Ermon, 2017)

Unsupervised Learning

- It's about learning interesting/useful structures in the data (unsupervisedly!)
- There is no supervision (no labels/responses), only inputs $\boldsymbol{x_1, x_2, \ldots, x_N}$
- Some examples of unsupervised learning
  - Clustering: Grouping similar inputs together (and dissimilar ones far apart)
  - Dimensionality Reduction: Reducing the dimensionality of inputs
  - Estimating the probability density of inputs (which distribution $\boldsymbol{p(x|\theta)}$ "generated" the inputs)
  - Unsupervised feature/representation learning
- Most unsup. learning algos also learn a new feature representation of inputs, e.g.,
  - Clustering gives a one-hot "quantized" representation $\boldsymbol{z_n}$ for each input $\boldsymbol{x_n}$

Some clustering algos learn a soft/probabilistic clustering in which $\boldsymbol{z_n}$ will be be probability vector that sums to 1 (will see later)



Assuming each input belongs deterministically to a single cluster

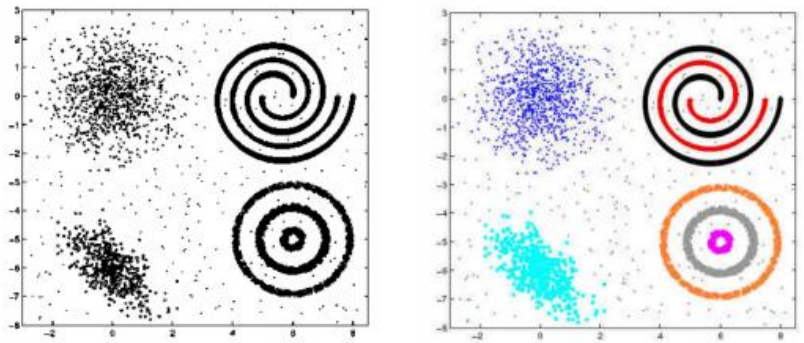
  - Dim-red gives $\boldsymbol{z_n}$, a lower-dim representation of $\boldsymbol{x_n}$
  - Unsup. Rep. learning can learn lower/higher dimensional representation $\boldsymbol{z_n}$ of $\boldsymbol{x_n}$

Clustering

In some cases, we may not know the right number of clusters in the data and may want to learn that (technique exists for doing this but beyond the scope)

- Given: N unlabeled inputs $\boldsymbol{x_1, x_2, \ldots, x_N}$; desired no. of partitions K
- Goal: Group the examples into K "homogeneous" partitions

In addition to partitioning these N inputs, we may also want to predict which partition a new test point belongs to



(a) Input data (b) Desired clustering

Picture courtesy: "Data Clustering: 50 Years Beyond K-Means", A.K. Jain (2008)

- Loosely speaking, it is classification without ground truth labels of training data
- A good clustering is one that achieves
  - High within-cluster similarity
  - Low inter-cluster similarity

Similarity can be Subjective

- Clustering only looks at similarities b/w inputs, since no labels are given
- Without labels, similarity can be hard to define

- Thus using the right distance/similarity is very important in clustering
- In some sense, related to asking: "Clustering based on what"?

Picture courtesy: http://www.guy-sports.com/humor/videos/powerpoint presentation dogs.htm

Clustering: Some Examples

- Document/Image/Webpage Clustering
- Image Segmentation (clustering pixels)

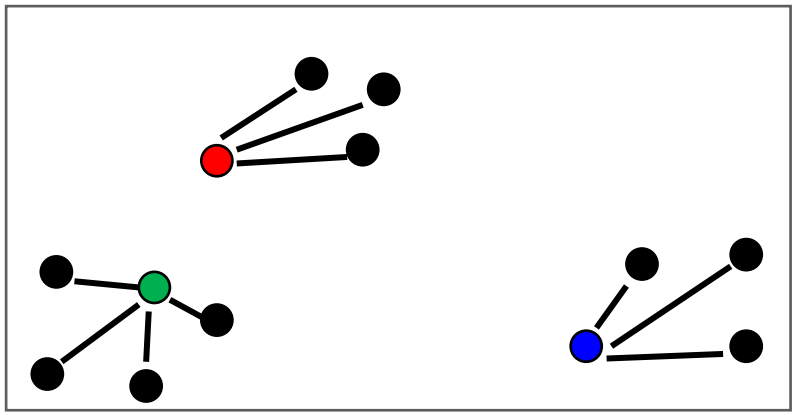
- Clustering web-search results
- Clustering (people) nodes in (social) networks/graphs
- .. and many more..

Picture courtesy: http://people.cs.uchicago.edu/~pff/segment/

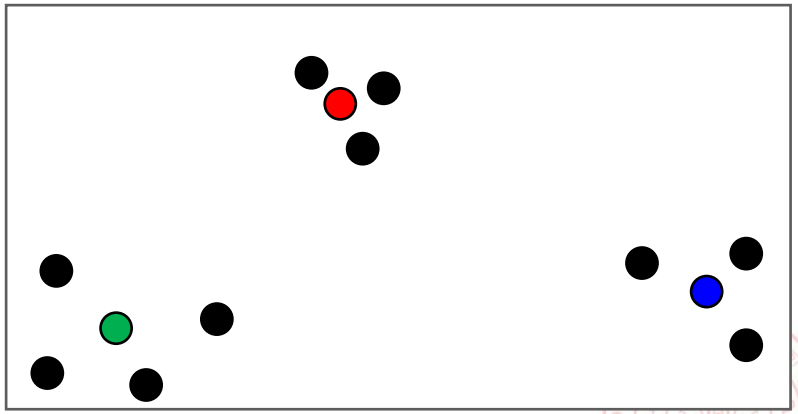
K-means Clustering

- Based on the following two steps (applied in alternating fashion until not converged)

1. Assign each input to the current closest mean



1. Recompute (update) the means



- At the very beginning, the means needs to be initialized (at random locations or using other schemes that we will see later)

K-means Clustering

Similar to LwP but in LwP the labels are known so the means can be computed in a single step without iterating multiple times

Good initialization matters (some methods exist to pick good initialization, e.g., K-means++)

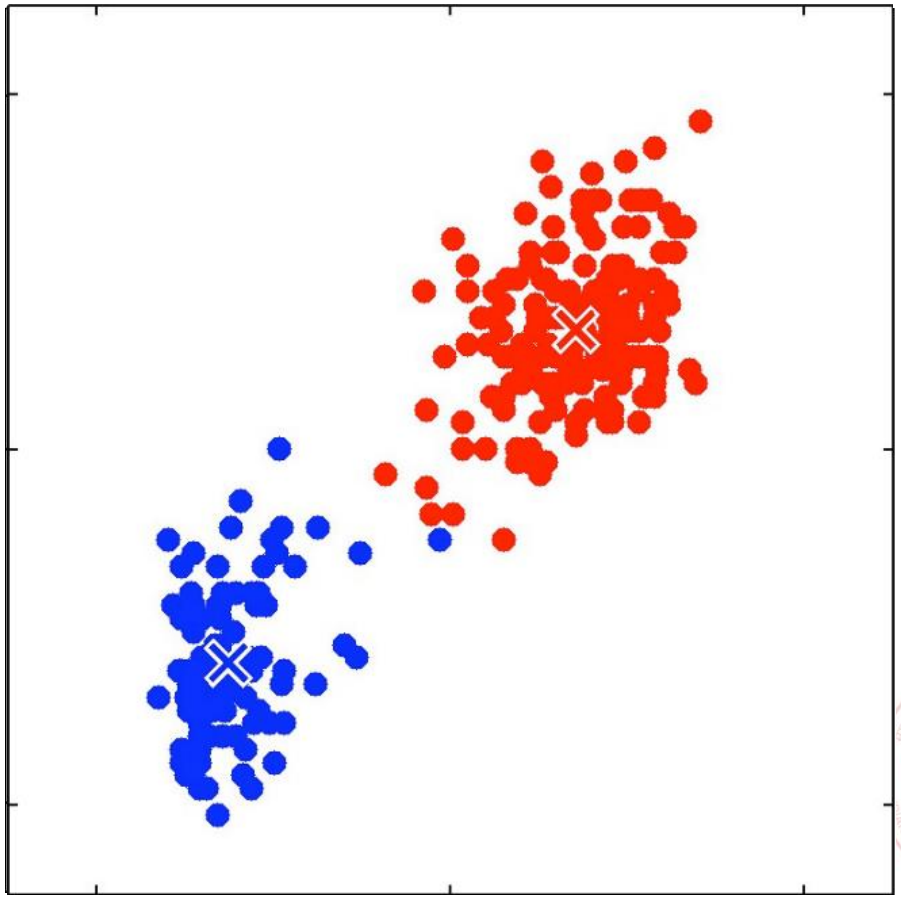
1. Randomly initialize K locations (the means)

2. Using the current means, predict the cluster id for each input (closest mean)

3. Recompute the means using the predicted cluster id of each input

4. Go to step 2 if not converged

K-means clustering with $K = 2$



K-means (a.k.a. Lloyd's) Algorithm: Formally

- **Input:** N inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$; $\boldsymbol{x}_n \in \mathbb{R}^D$; desired no. of partitions K
- **Desired Output:** Cluster ids of these N inputs and K cluster means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$

K-means Algorithm

1. Initialize K cluster means $\mu_1, \ldots, \mu_K$
2. For $n = 1, \ldots, N$, assign each point $\boldsymbol{x}_n$ to the closest cluster

   $z_n = k$ means $z_{nk} = 1$ in one-hot representation of z_n

   $$z_n = \arg\min_{k \in \{1,\ldots,K\}} ||\boldsymbol{x}_n - \mu_k||^2$$

3. Suppose $\mathcal{C}_k = \{\boldsymbol{x}_n : z_n = k\}$. Re-compute the means

   $$\mu_k = \text{mean}(\mathcal{C}_k), \quad k = 1, \ldots, K$$

4. Go to step 2 if not yet converged

K-means algo can also be seen as doing a compression by "quantization": Representing each of the N inputs by one of the $K < N$ means

Can be fixed by modeling each cluster by a probability distribution, such as Gaussian (e.g., Gaussian Mixture Model; will see later)

This basic K-means models each cluster by a single mean μ_k. Ignores size/shape of clusters

What Loss Function is *K*-means Optimizing?

- Define the distortion or "loss" for the cluster assignment of a single input $\boldsymbol{x}_n$

No labels here. We are measuring how much error we will incur if we assign $\boldsymbol{x}_n$ to its nearest cluster mean

$$\ell(\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{z}_n) = \sum_{k=1}^{K} z_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$$

$\boldsymbol{z}_n = [z_{n1}, z_{n2}, \ldots, z_{nK}]$ denotes a length K one-hot encoding of $\boldsymbol{x}_n$ and $z_{nk} = 1$ if $\boldsymbol{x}_n$ is assigned to cluster k

- Notation:
  - $\boldsymbol{X}$ is $N \times D$ matrix of inputs, $\boldsymbol{Z}$ is $N \times K$ matrix denoting cluster ids (each row is a one-hot $\boldsymbol{z}_n$)
  - $\boldsymbol{\mu}$ is $K \times D$ (each row contains the mean μ_k of cluster k)
- Total distortion over all inputs defines the K-means "loss function"

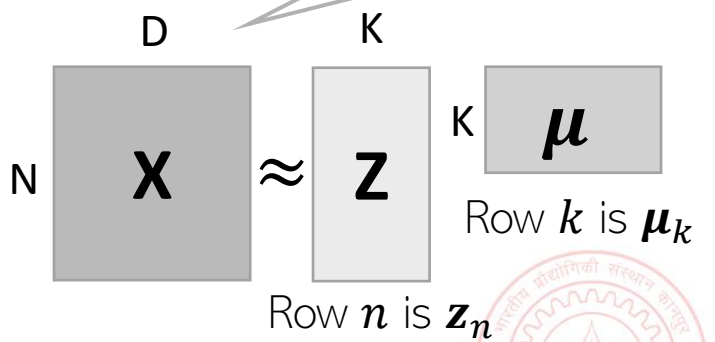
$$\ell(\boldsymbol{X}, \boldsymbol{\mu}, \boldsymbol{Z}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$$

$$= \|\boldsymbol{X} - \boldsymbol{Z}\boldsymbol{\mu}\|_F^2$$

K-means problem viewed as a matrix factorization problem

$\|.\|_F^2$ denotes the matrix Frobenius norm (squared norm for matrices)

$\boldsymbol{X}$ approximated by the product of $\boldsymbol{Z}$ and $\boldsymbol{\mu}$



- The K-means problem is to minimize this objective w.r.t. $\boldsymbol{\mu}$ and $\boldsymbol{Z}$
  - Alternating optimization on this loss would give the K-means (Lloyd's) algorithm we saw earlier!

K-means Loss: Several Forms, Same Meaning!

- Can write the K-means loss function in several ways

Mean of the cluster which input $\boldsymbol{x_n}$ got assigned to

$$\mathcal{L}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}) = \sum_{n=1}^{N} ||\boldsymbol{x}_n - \mu_{z_n}||^2$$

$$\mathcal{L}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}) = \sum_{k=1}^{K} \underbrace{\sum_{n:z_n=k} ||\boldsymbol{x}_n - \mu_k||^2}_{\text{within cluster variance}}$$

$$\mathcal{L}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} ||\boldsymbol{x}_n - \mu_k||^2$$

$$\mathcal{L}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}) = \underbrace{||\mathbf{X} - \mathbf{Z}\boldsymbol{\mu}||_F^2}_{\text{as matrix factorization}}$$

Replacing the ℓ_2 (Euclidean) squared by ℓ_1 distances gives the K-medoids algorithm (more robust to outliers)

- Note: Not just K-means but many unsup. learning algos try to minimize a distortion or reconstruction error for $\boldsymbol{X}$ by solving a problem of the form

$\boldsymbol{Z}$ denotes a new representation of the inputs and $\boldsymbol{\mu}$ denotes the other parameters of the model

$$\{\hat{\mathbf{Z}}, \hat{\boldsymbol{\mu}}\} = \arg\min_{\mathbf{Z}, \boldsymbol{\mu}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu})$$

Solving for Z

- Solving for $\boldsymbol{Z}$ with $\boldsymbol{\mu}$ fixed at $\hat{\boldsymbol{\mu}}$

$$\hat{\mathbf{Z}} = \arg\min_{\mathbf{Z}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \hat{\mu}) = \arg\min_{\mathbf{Z}} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} ||\boldsymbol{x}_n - \hat{\mu}_k||^2$$

- Still not easy. $\boldsymbol{Z}$ is discrete and above is an NP-hard problem
  - Combinatorial optimization: K^N possibilities for $\boldsymbol{Z}$ ($N \times K$ matrix with one-hot rows)

- Greedy approach: Optimize $\boldsymbol{Z}$ one row ($\boldsymbol{z_n}$) at a time keeping all others $\boldsymbol{z_n}$'s (and the cluster means $\mu_1, \mu_2, \ldots, \mu_K$) fixed

$$\hat{\boldsymbol{z}}_n = \arg\min_{\boldsymbol{z}_n} \sum_{k=1}^{K} z_{nk} ||\boldsymbol{x}_n - \hat{\mu}_k||^2 = \arg\min_{\boldsymbol{z}_n} ||\boldsymbol{x}_n - \hat{\mu}_{z_n}||^2$$

- Easy to see that this is minimized by assigning $\boldsymbol{x_n}$ to the closest mean
  - This is exactly what the K-means (Lloyd's) algo does!

Solving for $\boldsymbol{\mu}$

- Solving for $\boldsymbol{\mu}$ with $\boldsymbol{Z}$ fixed at $\widehat{\boldsymbol{Z}}$

$$\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \mathcal{L}(\mathbf{X}, \hat{\mathbf{Z}}, \boldsymbol{\mu}) = \arg\min_{\boldsymbol{\mu}} \sum_{k=1}^{K} \sum_{n:\hat{z}_n=k} ||\boldsymbol{x}_n - \mu_k||^2$$

- Not difficult to solve (each $\boldsymbol{\mu_k}$ is a real-valued vector, can optimize easily)

$$\hat{\mu}_k = \arg\min_{\mu_k} \sum_{n:\hat{z}_n=k} ||\boldsymbol{x}_n - \mu_k||^2$$

- Note that each $\boldsymbol{\mu_k}$ can be optimized for independently

- (Verify) This is minimized by setting $\hat{\boldsymbol{\mu}}_{\boldsymbol{k}}$ to be mean of points currently in cluster $\boldsymbol{k}$
  - This is exactly what the K-means (Lloyd's) algo does!

Convergence of *K*-means algorithm

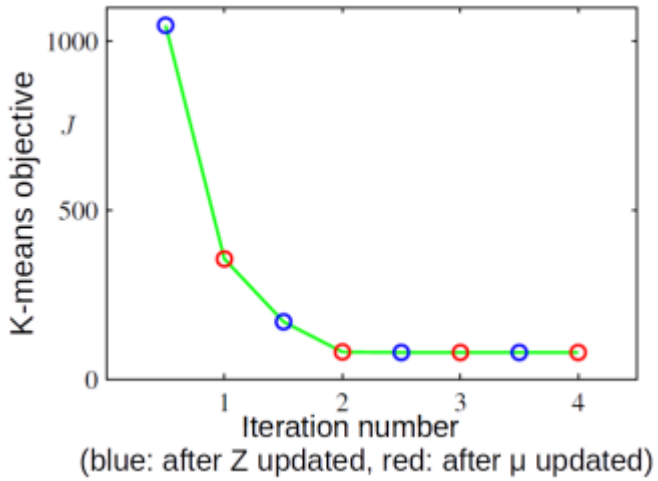
- Each step (updating $\boldsymbol{Z}$ or $\boldsymbol{\mu}$) can never increase the K-means loss
- When we update $\boldsymbol{Z}$ from $\boldsymbol{Z}^{(t-1)}$ to $\boldsymbol{Z}^{(t)}$

$$\mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\mu}^{(t-1)}) \leq \mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t-1)}, \boldsymbol{\mu}^{(t-1)}) \quad \text{because} \quad \mathbf{Z}^{(t)} = \arg\min_{\mathbf{Z}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}^{(t-1)})$$

- When we update $\boldsymbol{\mu}$ from $\boldsymbol{\mu}^{(t-1)}$ to $\boldsymbol{\mu}^{(t)}$

$$\mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\mu}^{(t)}) \leq \mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\mu}^{(t-1)}) \quad \text{because} \quad \boldsymbol{\mu}^{(t)} = \arg\min_{\boldsymbol{\mu}} \mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\mu})$$

- Thus the K-means algorithm monotonically decreases the objective

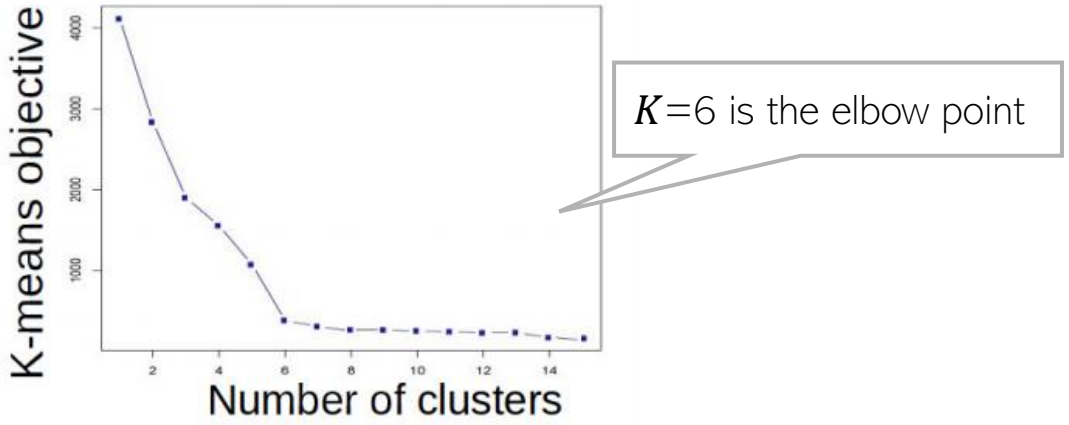


K-means: Choosing K

- One way to select K for the K-means algorithm is to try different values of K, plot the K-means objective versus K, and look at the "elbow-point"

If we use $K = N$ then K-means algo can achieve smallest possible loss (zero!). Think why.

However, if we use $K = N$ it isn't a useful clustering



K=6 is the elbow point

- Or criterion such as AIC (Akaike Information Criterion). Assuming D-dim inputs

$$AIC = 2\mathcal{L}(\hat{\boldsymbol{\mu}}, \mathbf{X}, \hat{\mathbf{Z}}) + KD$$

and choose K which gives the smallest AIC (small loss + large K values penalized)

- More advanced approaches, such as nonparametric Bayesian methods (Dirichlet Process mixture models also used, not within K-means but with other clustering algos)