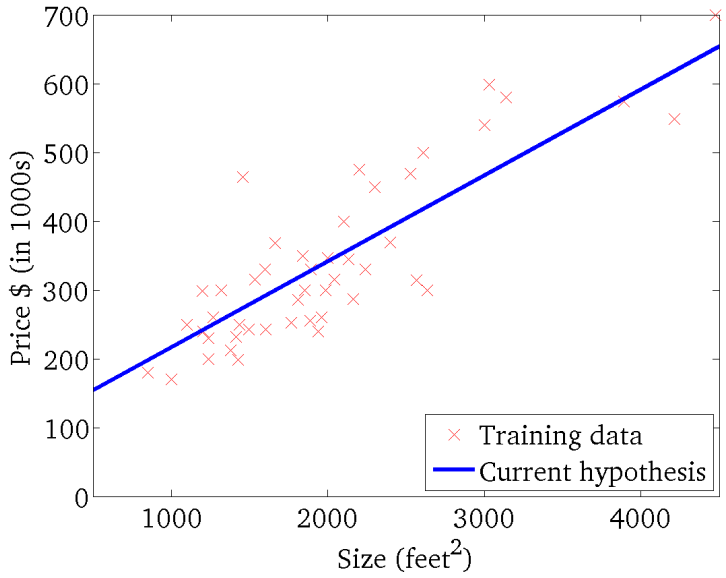
Course Review

Logistic Regression

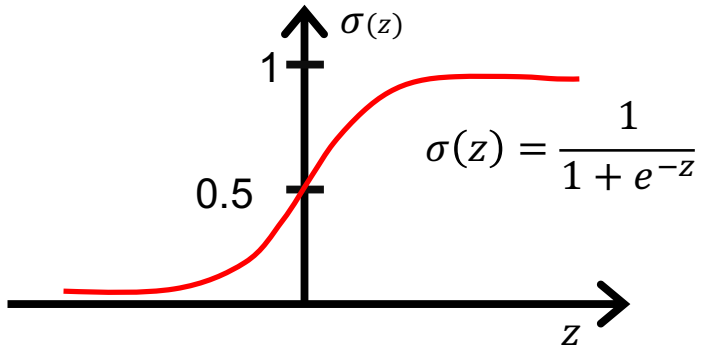
- In classification, the target variable $y \in \{0, 1\}$
- In linear regression, the output $f(\boldsymbol{x}) = \boldsymbol{xw}$ fall in the range $[-\infty, +\infty]$



- The output value of linear regression is not compatible with the target values in classification tasks

- Sigmoid/logistic function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



- Logistic regression

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{xw})$$

- Linear regression

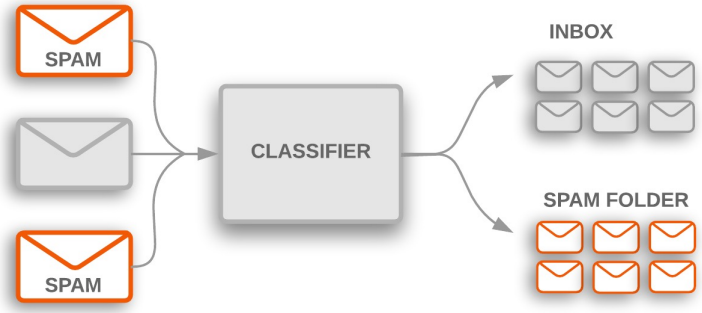
$$f(\boldsymbol{x}) = \boldsymbol{xw}$$

- The output range is transformed from $[-\infty, +\infty]$ to $[0, 1]$

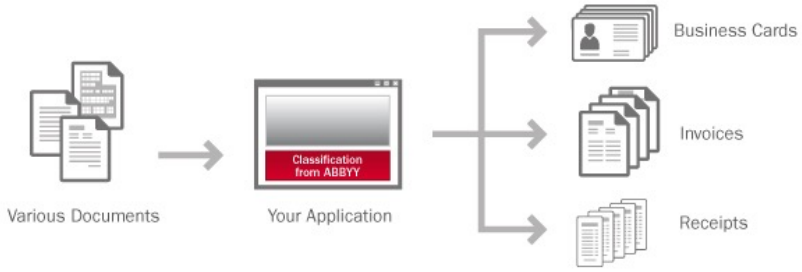
Examples for Logistic Regression

- Image category classification

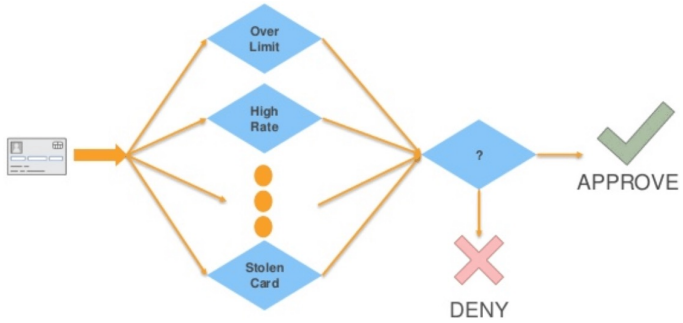
- Spam e-mails detection



- Document automatic categorization



- Transaction fraud detection



Cost Function

- Goal
  - If the true label $y = 1$, we want $f(\boldsymbol{x}) = \sigma(\boldsymbol{xw})$ to be close to 1
  - If the true label $y = 0$, we want $f(\boldsymbol{x}) = \sigma(\boldsymbol{xw})$ to be close to 0

- To achieve this goal, we can define a cost function similar to that in regression

$$L(\boldsymbol{w}) = (\sigma(\boldsymbol{xw}) - y)^2$$

- Alternatively, we can also seek to minimize

$$L(\boldsymbol{w}) = \begin{cases} -\log\big(\sigma(\boldsymbol{x}\boldsymbol{w})\big) & \text{if } y = 1 \\ -\log\big(1 - \sigma(\boldsymbol{x}\boldsymbol{w})\big) & \text{if } y = 0 \end{cases}$$

- The objective above can be equivalently written as

$$L(\boldsymbol{w}) = -y \log\big(\sigma(\boldsymbol{x}\boldsymbol{w})\big) - (1 - y) \log\big(1 - \sigma(\boldsymbol{x}\boldsymbol{w})\big)$$

If $y = 1$, $L(w)$ *reduces to* $L(w) = -log\big(\sigma(xw)\big)$;
Otherwise, if $y = 0$, $L(w)$ *reduces to* $L(w) = -log\big(1 - \sigma(xw)\big)$

- The loss above is called the *cross-entropy loss*

Bernoulli Distribution

- The Bernoulli distribution

$$p(z) = \begin{cases} \pi, & \text{if } z = 1 \\ 1 - \pi, & \text{if } z = 0 \end{cases}$$

where $\pi \in [0, 1]$ is the probability of z being 1

- The $p(z)$ can be concisely expressed as

$$p(z) = \pi^z \cdot (1 - \pi)^{1-z}$$

where $z = 0$ or 1

Binary Classification

- To achieve binary classification, the conditional probability is assumed to be a Bernoulli distribution

$$p(y|\boldsymbol{x}) = \big(\sigma(\boldsymbol{xw})\big)^{y} \cdot \big(1 - \sigma(\boldsymbol{xw})\big)^{1-y}$$

where $\pi = \sigma(\boldsymbol{xw})$; and $y = 0$ or 1

- The training objective is to maximize the log-likelihood function

$$\log p(y|\boldsymbol{x}) = y \log \sigma(\boldsymbol{xw}) + (1-y) \log\big(1 - \sigma(\boldsymbol{xw})\big)$$

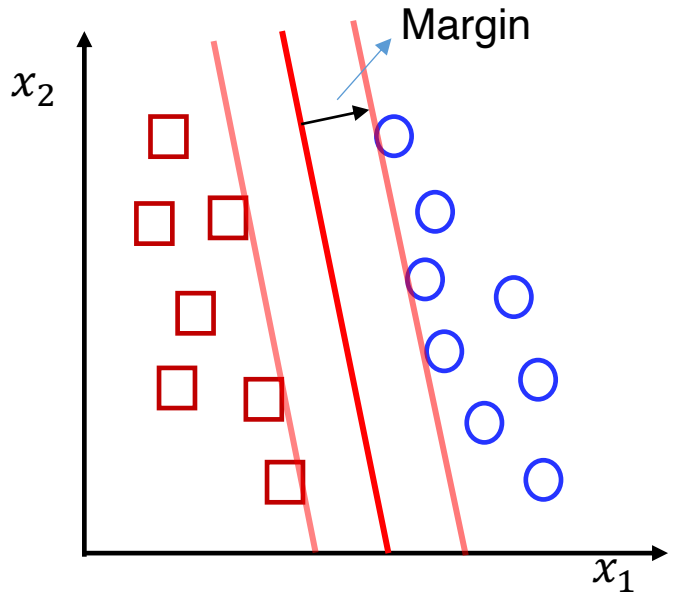
Recall that the logistic regression minimizes

$$cross\ entropy \triangleq -y \log \sigma(\boldsymbol{xw}) - (1-y) \log\big(1 - \sigma(\boldsymbol{xw})\big)$$

Maximizing $\log p(y|\boldsymbol{x})$ is equivalent to minimize the cross entropy

The Maximum-Margin Objective

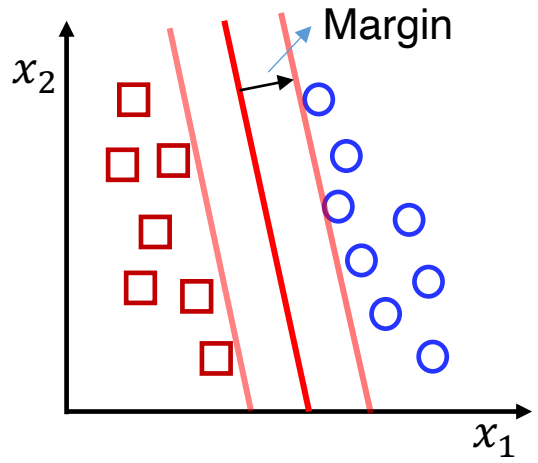
- To perform well on unseen data, the intuition is to find a hyperplane that makes the margin as large as possible



When the margin is large, *we can expect that an unseen sample has a larger chance to be categorized correctly*

- The margin of a hyperplane under a dataset is given by the minimum distance, *i.e.*,

$$Margin = \min_{\ell} \frac{y^{(\ell)} \cdot \left(\boldsymbol{w}^T \boldsymbol{x}^{(\ell)} + b\right)}{\|\boldsymbol{w}\|}$$



- Thus, the maximum-margin classifier is to find the $\boldsymbol{w}^*$ and b^* that maximize the margin, *i.e.*,

$$\boldsymbol{w}^*, b^* = \arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{\|\boldsymbol{w}\|} \min_{\ell} \left[y^{(\ell)} \cdot \left(\boldsymbol{w}^T \boldsymbol{x}^{(\ell)} + b\right) \right] \right\}$$

But how to optimize is unknown

The Transformed Objective Function

- Optimizing another objective function that shares the same optima as the original problem
  - If $\boldsymbol{w}^*$ and b^* is the optima to $\frac{1}{\|\boldsymbol{w}\|}\min_{\ell}\left[y^{(\ell)}\cdot\left(\boldsymbol{w}^T\boldsymbol{x}^{(\ell)}+b\right)\right]$, then $\kappa\boldsymbol{w}^*$ and κb^* must also be the optima for all $\kappa\in\mathbb{R}$
  - There must exist an optima $\kappa\boldsymbol{w}^*$ and κb^* that satisfy the constraints

$$y^{(\ell)}\cdot\left(\kappa\boldsymbol{w}^{*T}\boldsymbol{x}^{(\ell)}+\kappa b^*\right)\geq 1 \quad \text{for all } \ell=1,2,\cdots,n$$

Among all $\boldsymbol{w}, b$ that satisfy $y^{(\ell)}\cdot\left(\boldsymbol{w}^T\boldsymbol{x}^{(\ell)}+b\right)\geq 1$ for all ℓ, *the $\boldsymbol{w}$ and b with the smallest $\|\boldsymbol{w}\|^2$ must be the optima $\boldsymbol{w}^*$* and *b^* that maximizes*

$$\frac{1}{\|\boldsymbol{w}\|}\min_{\ell}\left[y^{(\ell)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(\ell)}+b\right)\right]$$

- Therefore, the maximum-margin hyperplane can be found by solving the optimization problem below

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$s.t.: y^{(\ell)} \cdot \left(\boldsymbol{w}^T \boldsymbol{x}^{(\ell)} + b\right) \geq 1, \quad \text{for } \ell = 1, 2, \cdots, N$$

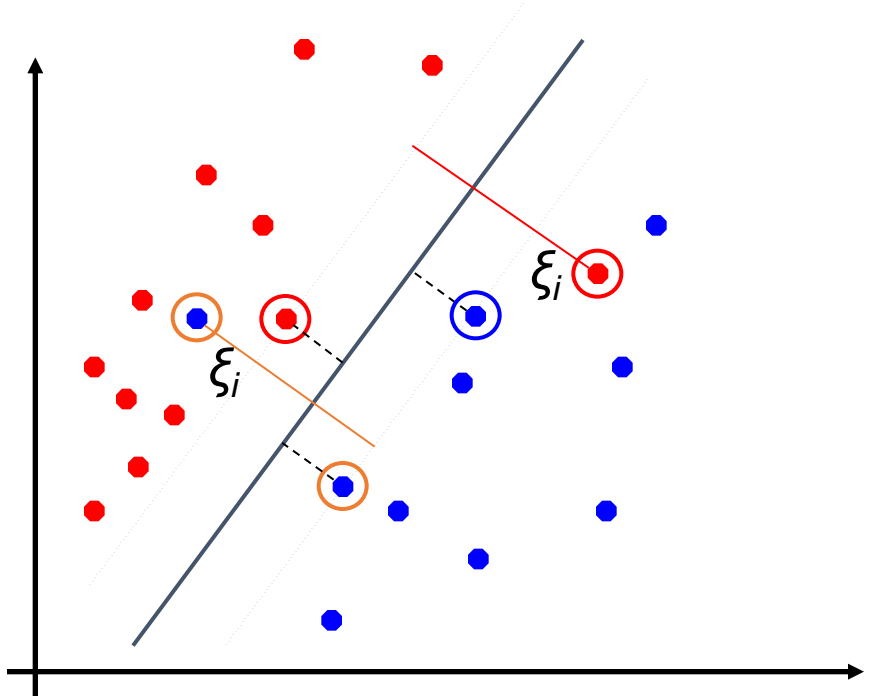
- This is a quadratic optimization problem. Its optimal solution can be found by *numerical methods* efficiently

- With the optimal $\boldsymbol{w}^*$ and b^*, an unseen data $\boldsymbol{x}$ can be classified as

$$\hat{y}(\boldsymbol{x}) = sign(\boldsymbol{w}^{*T}\boldsymbol{x} + b^*)$$

Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification Mathematically

The old formulation:

Find **w** and b such that
$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Modified formulation incorporates slack variables:

Find **w** and b such that
$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$

Parameter *C* can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

Ensemble Methods Overview

- It is difficult to learn a *strong classifier* that can always classify instances correctly
- But it is easy to learn a lot of *'weak' classifiers*

  A weak classifier may not perform well on the whole dataset, but may perform well on a fraction of samples, *e.g.*, some may be good at recognizing 'cat', while some others may be good at recognizing 'dog'

- If weak classifiers perform well on different fractions of samples, it is possible to *obtain a strong classifier by combining these weak classifiers in an appropriate way*
- Two questions

  1) How to produce these weak classifiers?
  2) How to combine the weak classifiers?

Two Types of Combining Methods

1) Unweighted average

   - Majority vote

2) Weighted average

   - Give better classifiers bigger weighting

For example, consider a two-class classification problem {-1, 1}

Two basic classifiers: $\hat{y}_1 = sign(f_1(\boldsymbol{x}))$ $\hat{y}_2 = sign(f_2(\boldsymbol{x}))$

Final classifiers: $\hat{y}_e = sign\left(\alpha_1 f_1(\boldsymbol{x}) + \alpha_2 f_2(\boldsymbol{x})\right)$

Remark: The weak classifiers could be of any kind, *e.g.* decision trees, SVM, neural networks, logistic regression *etc.*

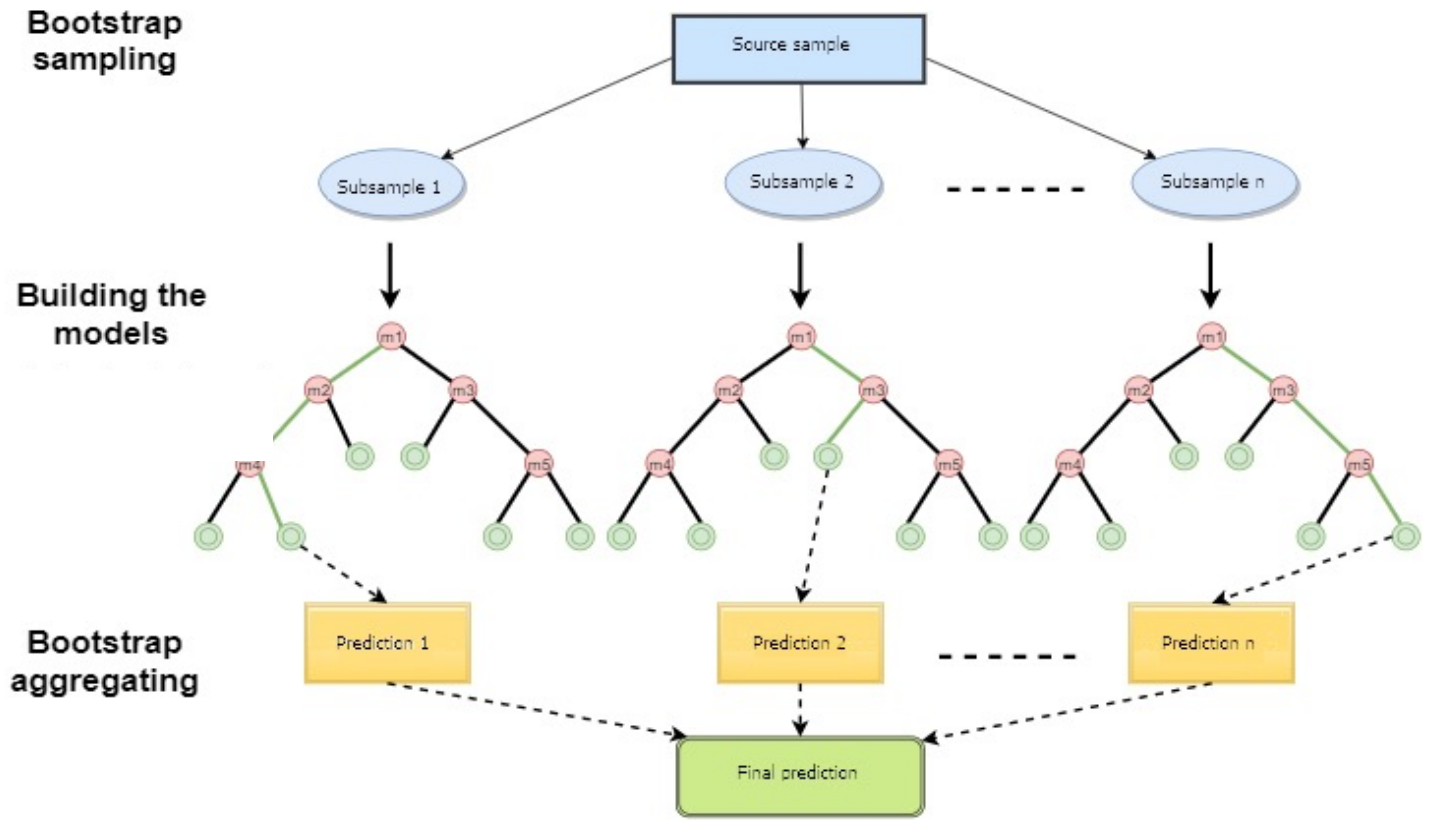
Deriving Weak Classifiers

- We don't know how to obtain classifiers that perform well on different fractions of samples
- Instead, we seek to obtain classifiers that are as diverse as possible, that is, encouraging their predictions *to be uncorrelated*. For example,

1) Creating subsets of the training dataset by bootstrapping
   - Randomly draw N' samples from the N-sample training dataset *with replacement*
   - Repeat the above procedure K times, generating subsets $S_1, S_2, \cdots, S_K$

2) Training a decision tree on each of the subset S_k

3) Combine K decision trees into one by majority voting



At testing, pass test data through all K classifiers, using the majority voting result as the final prediction

Overview of Boosting Methods

- *Weak classifiers derivation*

  Repeat the following steps several times

  1) Identifying the examples that are incorrectly classified

  2) Re-training the classifier by *giving more weighting to the misclassified examples*

- *Combining*

  Combing the prediction results of each classifier by *weighted average*

How to *weight the examples and prediction* results is the key

- Adaboost algorithm

1) Initialize the weight of examples as $\omega_0^{(n)} = \frac{1}{N}$ for $n = 1, \cdots, N$

2) For the k-th iteration, train a classifier $h_k(\boldsymbol{x})$ with the training examples weighted by $w_{k-1}^{(n)}$

3) Evaluate the weighted classification error

$$\epsilon_k = \frac{\sum_{n=1}^{N} \omega_{k-1}^{(n)} I\left(y_i \neq h_k(\boldsymbol{x}_n)\right)}{\sum_{n=1}^{N} \omega_{k-1}^{(n)}}$$

4) Determine the vote stake of the k-th classifier

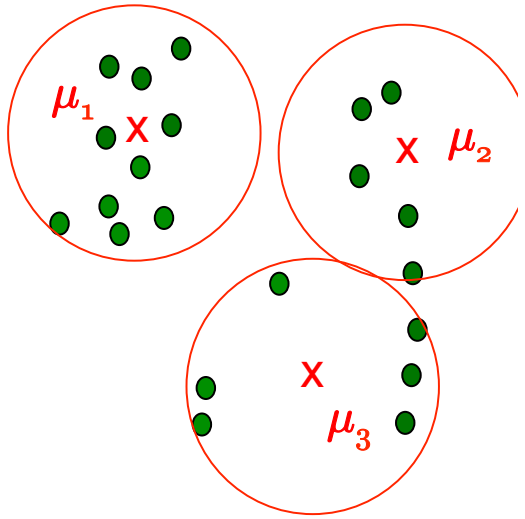
$$\alpha_k = \frac{1}{2} \ln\left(\frac{1-\epsilon_k}{\epsilon_k}\right)$$

5) Update the weights of examples as

$$\omega_k^{(n)} = \omega_{k-1}^{(n)} \exp\{-y_i h_k(\boldsymbol{x}_i) \alpha_k\}$$

K-Means Algorithm

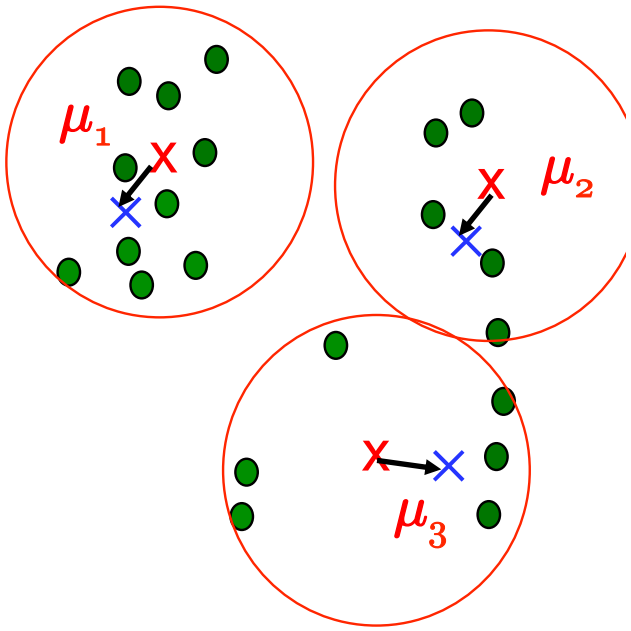
- Designate K centers $\boldsymbol{\mu}_k$ for $k = 1, \cdots, K$, and then evaluate the distance between every data $\boldsymbol{x}^{(n)}$ and all centers $\boldsymbol{\mu}_k$



- Data $\boldsymbol{x}^{(n)}$ is assigned to the cluster k that leads to smallest distance

$$r_{nk} = \begin{cases} 1, & if\ k = \arg\min_j \left\| \boldsymbol{x}^{(n)} - \boldsymbol{\mu}_j \right\|^2 \\ 0, & otherwise \end{cases}$$

- Updating the centers using the mean of samples within a cluster



Two questions

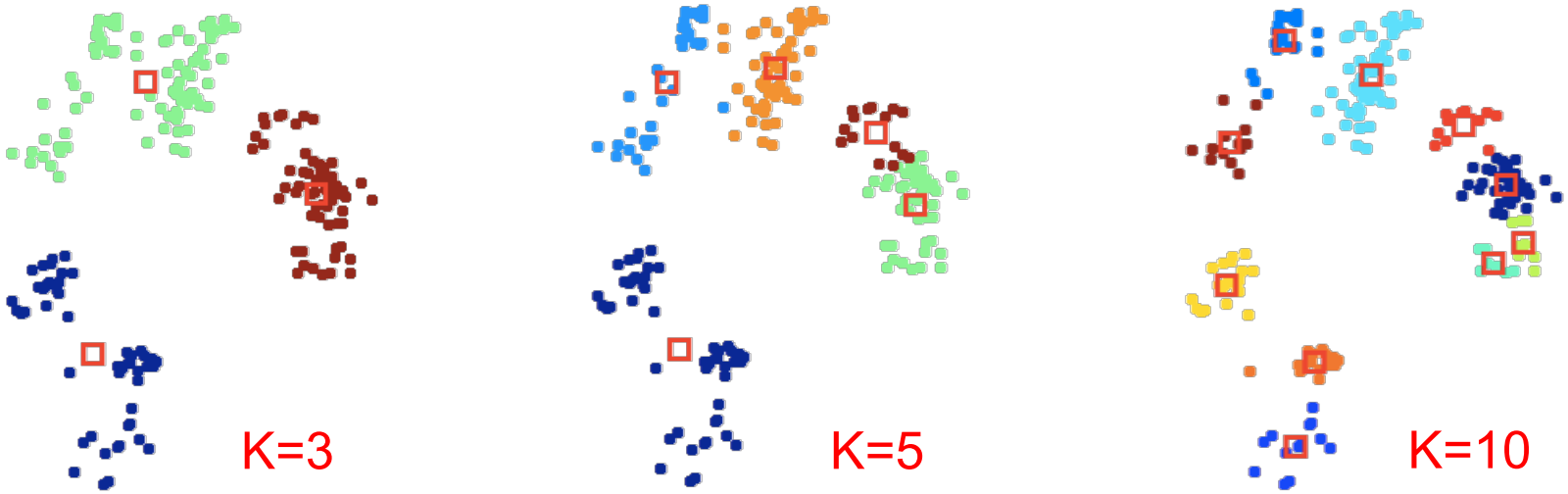
1) What does the algorithm really do?
2) Is the algorithm guaranteed to converge?

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_{n=1}^{N} r_{nk}\, \boldsymbol{x}_n}{\sum_{n=1}^{N} r_{nk}}$$

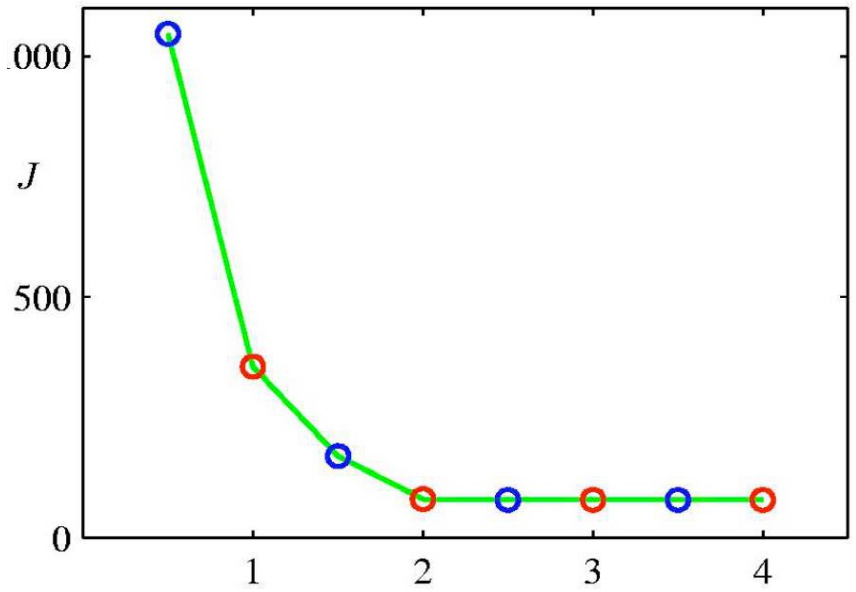
- Repeating the assignment and center updating steps above

Issue: Number of Clusters

- How to set the value for K is extremely important to the final clustering result



- The distance J decreases as the number of clusters K increases. Thus, we cannot determine K by seeking the minimum of J

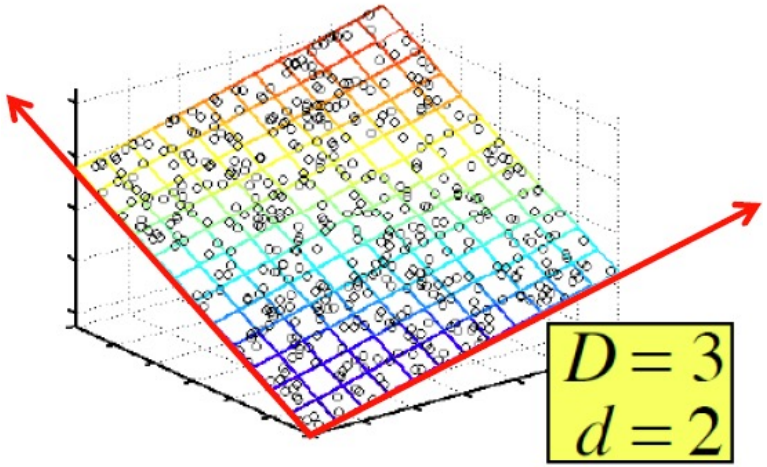


1) One possible method is to choose the elbow point (here $K = 2$)

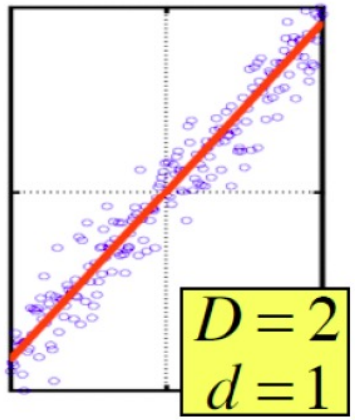
2) Another possible method is to determine the best K value according to the performance of downstream applications

Why the dimensionality could be reduced?

- The high-dimensional data often resides on a low-dimensional intrinsic space approximately



3-dimensional data lies on a 2-dimensional plane approximately

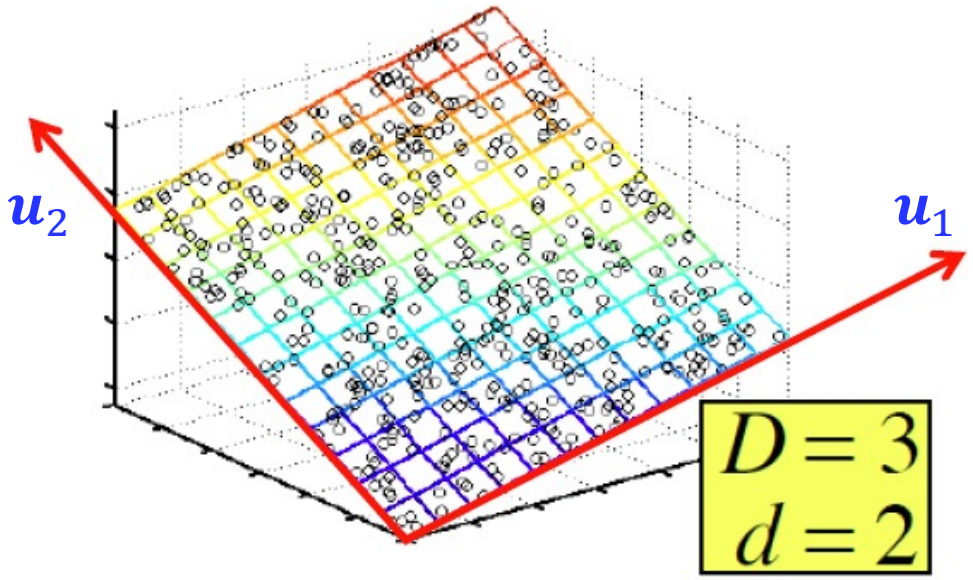


2-dimensional data lies on a 1-dimensional line approximately

The key is to find *the principal directions* under which the dimensions of data can be significantly reduced

Finding the Best Directions

- **Objective:** Given data $\{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$ from $\mathbb{R}^D$, finding the orthogonal directions $\boldsymbol{u}_i$ under which the original data can be best represented



$$\boldsymbol{x}^{(n)} \approx \sum_{i=1}^{M} \alpha_i^{(n)} \boldsymbol{u}_i$$

- Suppose the best directions $\{\boldsymbol{u}_i\}_{i=1}^{M}$ are given, what are the best coefficients $\alpha_i^{(n)}$?

$$\alpha_i^{(n)} = \boldsymbol{u}_i^T \boldsymbol{x}^{(n)}$$

Instead of representing the data $\boldsymbol{x}^{(n)}$ directly, we first center the data to the origin, *i.e.*, subtracting each data point $\boldsymbol{x}^{(n)}$ by its mean

$$\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}, \quad \text{where } \bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}^{(n)}$$

- The objective can now be described as minimizing the error between $\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}$ and its best approximant $\tilde{\boldsymbol{x}}^{(n)} = \sum_{i=1}^{M} \alpha_i^{(n)} \boldsymbol{u}_i$ in the $span(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_M)$

$$E = \frac{1}{N}\sum_{n=1}^{N} \left\| \left(\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}\right) - \tilde{\boldsymbol{x}}^{(n)} \right\|^2$$

where the best coefficient α_i is known equal to

$$\alpha_i = \boldsymbol{u}_i^T \left(\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}\right)$$

Content-based Recommendations

Main idea: Recommend items to customer ***x*** similar to previous items rated highly by ***x***

Example:

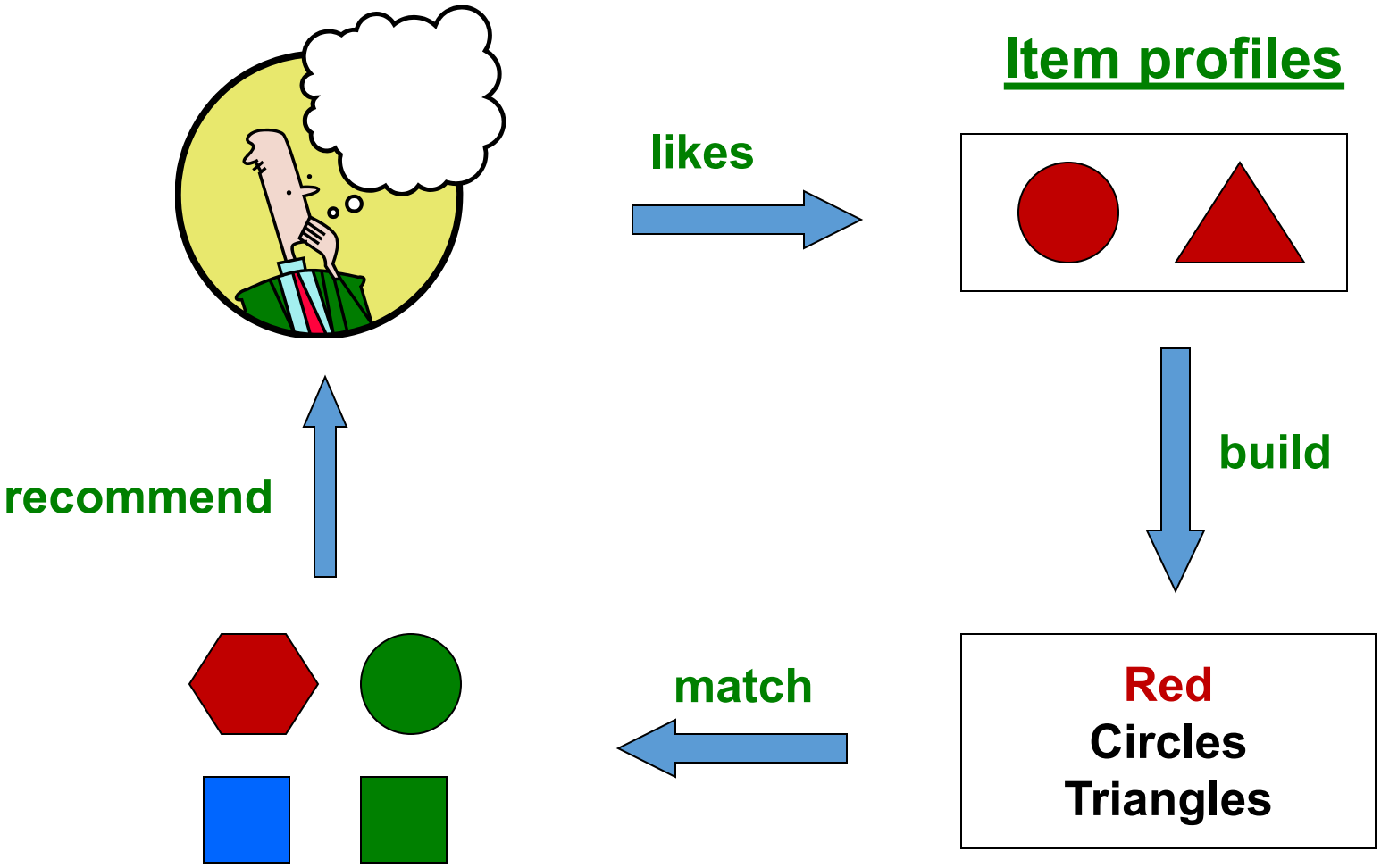
Movie recommendations

- Recommend movies with same actor(s), director, genre, …

Websites, blogs, news

- Recommend other sites with “similar” content

Plan of Action



Item Profiles

For each item, create an **item profile**

Profile is a set (vector) of features

- **Movies:** author, title, actor, director,…
- **Text:** Set of "important" words in document

How to pick important features?

- Usual heuristic from text mining is **TF-IDF**

  (Term frequency * Inverse Doc Frequency)

  - **Term … Feature**
  - **Document … Item**

User Profiles and Prediction

User profile possibilities:

- Weighted average of rated item profiles
- **Variation:** weight by difference from average rating for item
- …

Prediction heuristic:

- Given user profile ***x*** and item profile ***i***, estimate $u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$

Pros: Content-based Approach

+: No need for data on other users

- No cold-start or sparsity problems

+: Able to recommend to users with unique tastes

+: Able to recommend new & unpopular items

- No first-rater problem

+: Able to provide explanations

- Can provide explanations of recommended items by listing content-features that caused an item to be recommended

Cons: Content-based Approach

–: Finding the appropriate features is hard

- E.g., images, movies, music

–: Recommendations for new users

- **How to build a user profile?**

–: Overspecialization

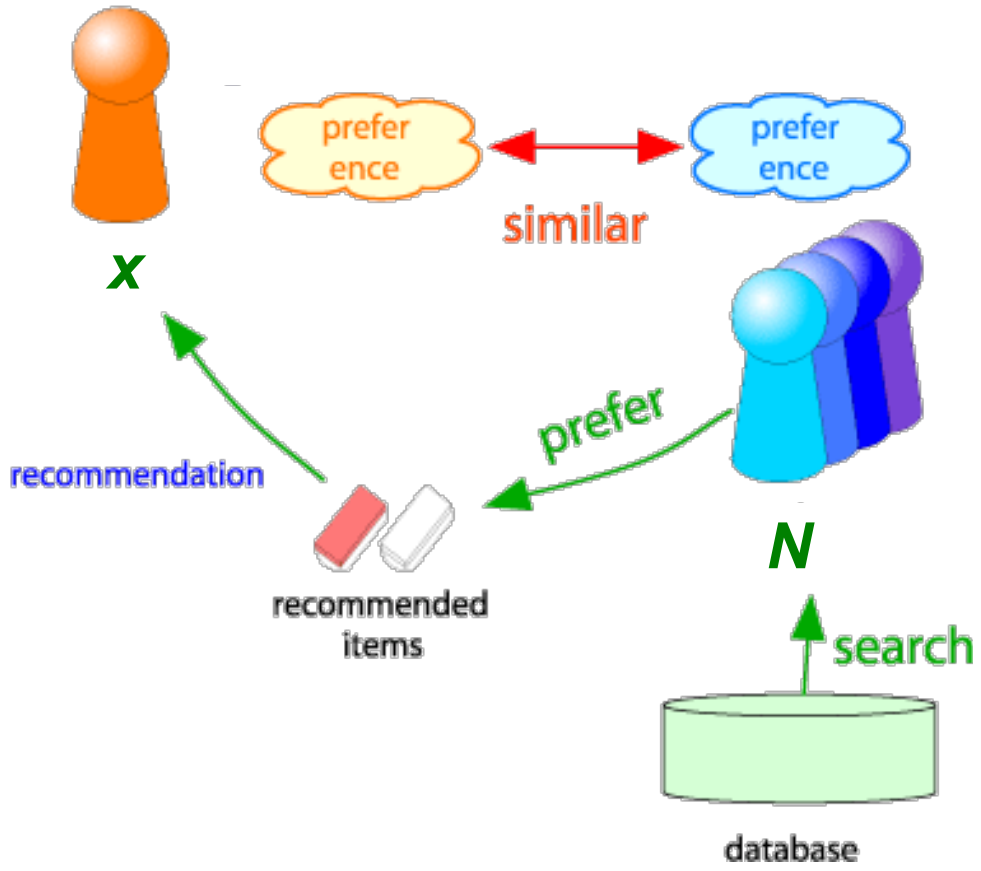
- Never recommends items outside user's content profile
- People might have multiple interests
- **Unable to exploit quality judgments of other users**

User-User Collaborative Filtering

Consider user ***x***

Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings

Estimate ***x***'s ratings based on ratings of users in ***N***



Finding "Similar" Users

Let r_x be the vector of user x's ratings

Cosine similarity measure

- $\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$

r_x, r_y as points:
$r_x = \{1, 0, 0, 1, 3\}$
$r_y = \{1, 0, 2, 2, 0\}$

- **Problem:** Treat missing ratings as "negative", even smaller than than low ratings, e.g. 1

Pearson correlation coefficient

- S_{xy} = items rated by both users x and y

$$sim(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

r_x, r_y … avg. rating of **x**, **y**

Item-Item Collaborative Filtering

So far: User-user collaborative filtering

Another view: Item-item

- For item ***i***, find other similar items
- Estimate rating for item ***i*** based on ratings for similar items
- Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

s_{ij}… similarity of items ***i*** and ***j***
r_{xj}…rating of user ***x*** on item ***j***
N(i;x)… set items rated by ***x*** similar to ***i***

Pros of Collaborative Filtering

+ Works for any kind of item

- No feature selection needed

- Question
  - The item-item CF methods look similar to the content-based recommendation methods, both evaluating the similarities between items and then recommending the similar ones
  - But why the CF methods don't have the issue of feature selection for items?

Cons of Collaborative Filtering

- **- Cold Start:**
  - Need enough users in the system to find a match
- **- Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- **- First rater:**
  - Cannot recommend an item that has not been previously rated, e.g., new items
- **- Popularity bias:**
  - Tend to recommend popular items
  - Cannot recommend items to someone with unique taste

Hybrid Methods

Implement two or more different recommenders and combine predictions

- Perhaps using a linear model

Add content-based methods to collaborative filtering

- Item profiles for new item problem
- Demographics to deal with new user problem

EM Algorithm

- Algorithm

  E-step: Evaluating the expectation

$$\mathcal{Q}\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)}\right)=\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})}[\log p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})]$$

  M-step: Updating the parameter

$$\boldsymbol{\theta}^{(t+1)}=\arg\max_{\boldsymbol{\theta}}\mathcal{Q}\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)}\right)$$

- Key integrant in EM

  1) The posteriori distribution $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})$

  2) The expectation of joint distribution $\log p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})$ *w.r.t.* the posteriori

  3) Maximization

Re-representing the Log-likelihood

- The log-likelihood can be reformulated as

$\forall$ distribution $q(\boldsymbol{z})$

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log p(\boldsymbol{x})$$

$$= \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}) q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}) q(\boldsymbol{z})}$$

$$= \underbrace{\sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})}}_{\mathcal{L}(q,\boldsymbol{\theta})} + \underbrace{\sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})}}_{KL(q||p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}))}$$

$$= \mathcal{L}(q,\boldsymbol{\theta}) + KL(q||p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})), \qquad \text{for } \forall\, \boldsymbol{\theta}, q(\boldsymbol{z})$$

Remark: The KL-divergence is used to *measure the distance* between two distributions q and p, which is defined as

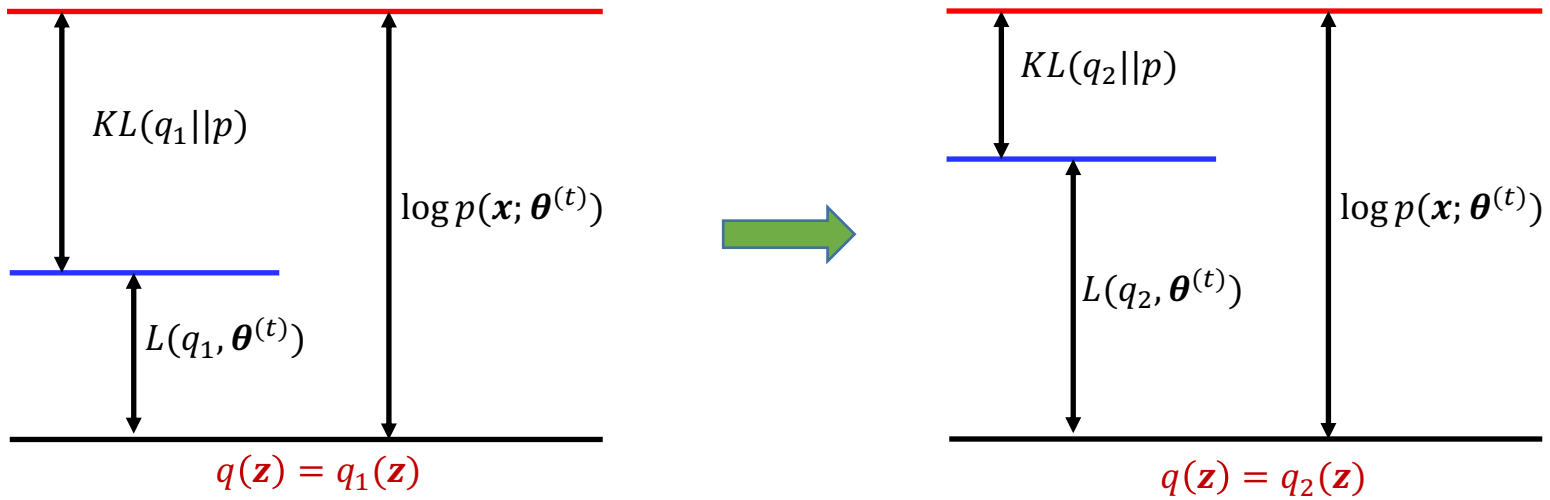
$$KL(q||p) \triangleq \int q(z) \log \frac{q(z)}{p(z)} dz \geq 0$$

- Thus, with the parameter at the t-th iteration denoted as $\boldsymbol{\theta}^{(t)}$, we have

$$\log p\left(\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right) = \mathcal{L}\left(q,\boldsymbol{\theta}^{(t)}\right) + KL(q||p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}))$$

This equality holds for any distribution $q(\boldsymbol{z})$

- Different $q(\boldsymbol{z})$ will lead to different decomposition of $\log p\left(\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right)$



Theoretical Justification for EM

$$\log p(\boldsymbol{x};\boldsymbol{\theta}^{(t)}) = \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}^{(t)})}{q(\boldsymbol{z})} + \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})}$$

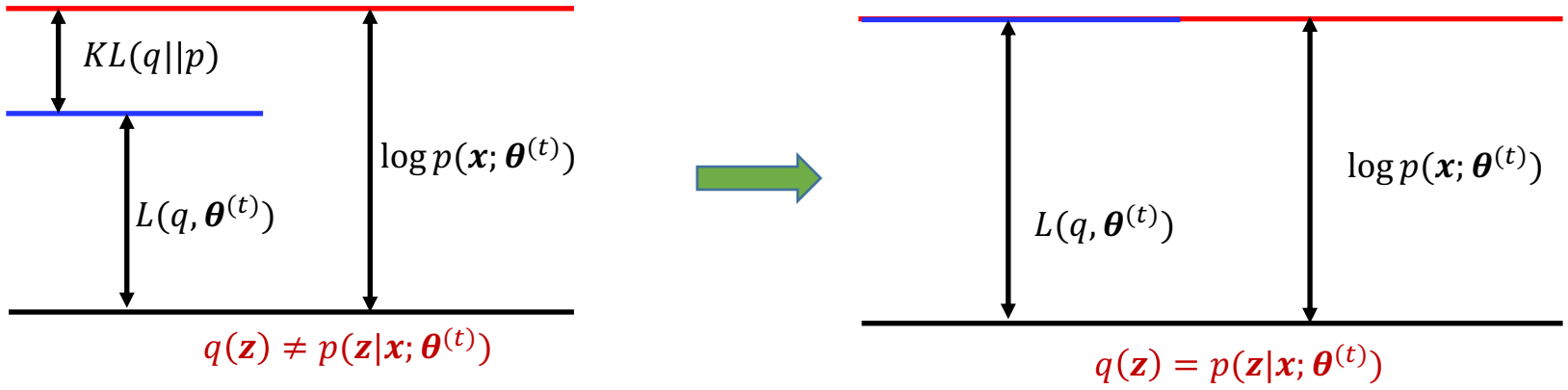
- If we set $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})$, then we have

$$KL\left(q||p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})\right) = 0$$

Thus, we have

$$\log p(\boldsymbol{x};\boldsymbol{\theta}^{(t)}) = \mathcal{L}\left(p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}),\boldsymbol{\theta}^{(t)}\right)$$

$$= \sum_{\boldsymbol{z}} p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right) \log \frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}^{(t)})}{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})}$$



$$\log p(\boldsymbol{x};\boldsymbol{\theta}^{(t)}) = \mathcal{L}\left(p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right),\boldsymbol{\theta}^{(t)}\right)$$

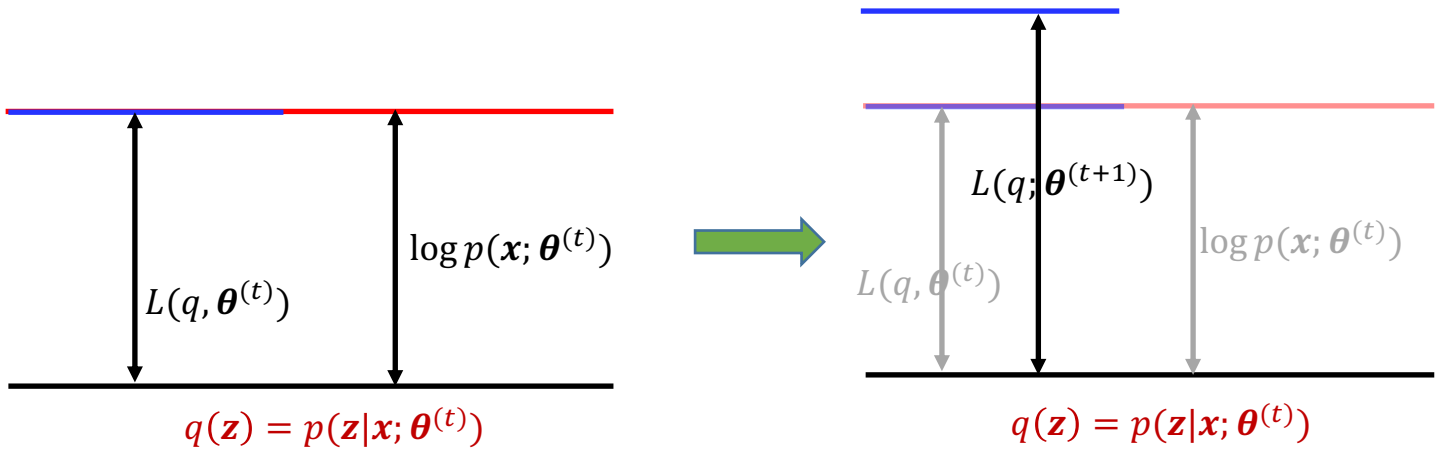
$$= \sum_{\boldsymbol{z}} p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right) \log \frac{p\left(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}^{(t)}\right)}{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})}$$

- If we update $\boldsymbol{\theta}$ as

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}\left(p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right),\boldsymbol{\theta}\right),$$

then we must have the relation

$$\mathcal{L}\left(p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right),\boldsymbol{\theta}^{(t+1)}\right) \geq \underbrace{\mathcal{L}\left(p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right),\boldsymbol{\theta}^{(t)}\right)}_{=\log p(\boldsymbol{x};\boldsymbol{\theta}^{(t)})}$$



$$\log p(\boldsymbol{x};\boldsymbol{\theta}^{(t+1)}) = \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p\left(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}^{(t+1)}\right)}{q(\boldsymbol{z})} + \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t+1)})}$$

- By setting $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})$, we obtain

$$\log p(\boldsymbol{x};\boldsymbol{\theta}^{(t+1)}) = \underbrace{\mathcal{L}\left(p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}),\boldsymbol{\theta}^{(t+1)}\right)}_{\geq \log p(\boldsymbol{x};\boldsymbol{\theta}^{(t)})} + \underbrace{KL(p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})||p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t+1)}))}_{\geq 0}$$

The KL-divergence is always non-negative

- Thus, we can see that

$$\log p(\boldsymbol{x};\boldsymbol{\theta}^{(t+1)}) \geq \log p(\boldsymbol{x};\boldsymbol{\theta}^{(t)})$$

$\max_{\boldsymbol{\theta}} \mathcal{L}\left(p\left(\boldsymbol{z}\middle|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right),\boldsymbol{\theta}\right)$ *can guarantee the increase of likelihood at each step*

- Equivalence between EM updating

$$\mathcal{Q}\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)}\right)=\mathbb{E}_{p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right)}[\log p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})] \qquad \boldsymbol{\theta}^{(t+1)}=\arg\max_{\boldsymbol{\theta}}\mathcal{Q}\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)}\right)$$

and the updating rule $\arg\max_{\boldsymbol{\theta}}\mathcal{L}\left(p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right),\boldsymbol{\theta}\right)$

$$\mathcal{L}\left(p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right),\boldsymbol{\theta}\right)=\underbrace{\sum_{\boldsymbol{z}}p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right)\log p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}_{\mathbb{E}_{p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right)}[\log p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})]}-\underbrace{\sum_{\boldsymbol{z}}p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right)\log p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right)}_{constant}$$

Therefore, $\arg\max_{\boldsymbol{\theta}}\mathcal{L}\left(p\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right),\boldsymbol{\theta}\right)\Leftrightarrow\arg\max_{\boldsymbol{\theta}}\mathcal{Q}\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)}\right)$

EM algorithm can guarantee the increase of likelihood at each step