# Stanford CS229 Summary

Yingxin Wu

## 目录

note:

m:# of training examples.

n:# of features.

# 1 Gradient Descent

**scenario** Batch Gradient Descent: obtain the (locally) optimal solution for small datasets.

Stochastic Gradient Descent: obtain the (locally) optimal solution for large datasets.

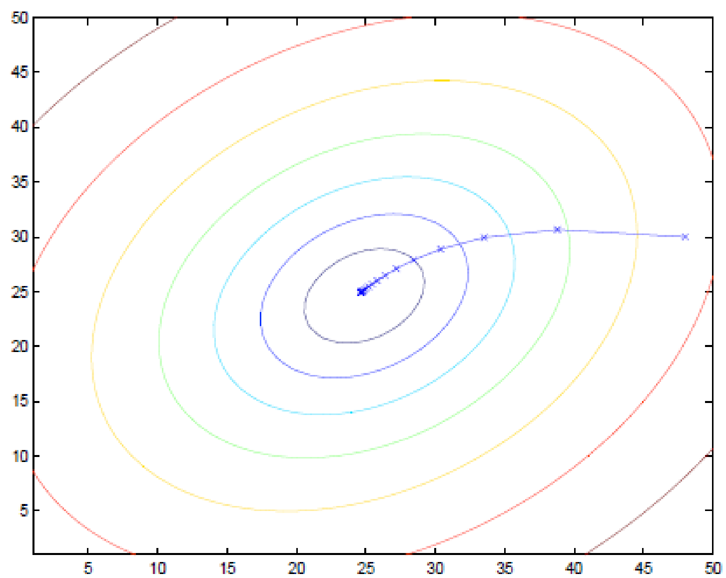


图 1: gradient descent

**input** $\theta_0 \in R^n$,$X^i$(i=1,...,m),Y($Y_i$ is the original output value of $X^i$).

**output** $\theta \in R^n$.

**B's methodology** decrease the loss function–repeatedly update each parameter in the gradient direction untill convergence.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{1}$$

$$\theta_i := \theta_i - \alpha\frac{\partial}{\partial\theta_i}J(\theta) \tag{2}$$

**S' methodology** Repeat{

for j = 1 to m{

$$\begin{aligned} \theta_i :=& \theta_i - \alpha(h_\theta(x^{(j)}) - y^{(j)})x_i^{(j)} \\ &(for\ all\ i) \end{aligned} \tag{3}$$

}

}

**Matrix form**

$$\theta = (X^TX)^{-1}X^TY \tag{4}$$

## 2 Newton's Method

**scenario** Solve the likelihood function maximization of logistic regression in classification problems and so on.

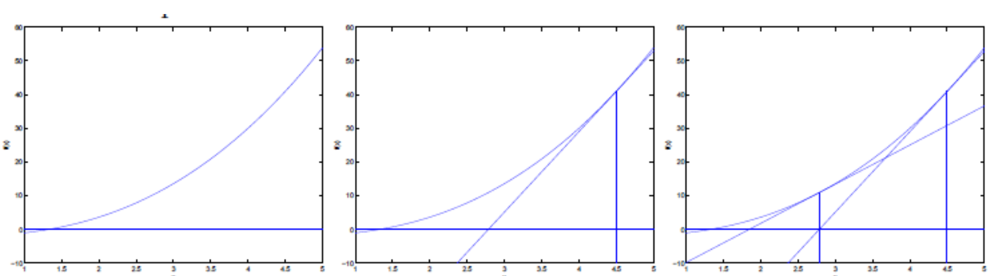

图 2: Newton's Method

**input** $\theta_0$,f(x).

**output** $\theta$

**methodology**

$$\theta := \theta - \frac{f(\theta)}{f'(\theta)} \tag{5}$$

Gradient decent only considers local optimum while Newton's method also takes global optimum into consideration.(NM:quadric surface, GD:plane).

# 3 Coordinate Assent

**scenario** optimilize $W(\alpha_1, \alpha_2, ..., \alpha_n)$ with respect to each variable.

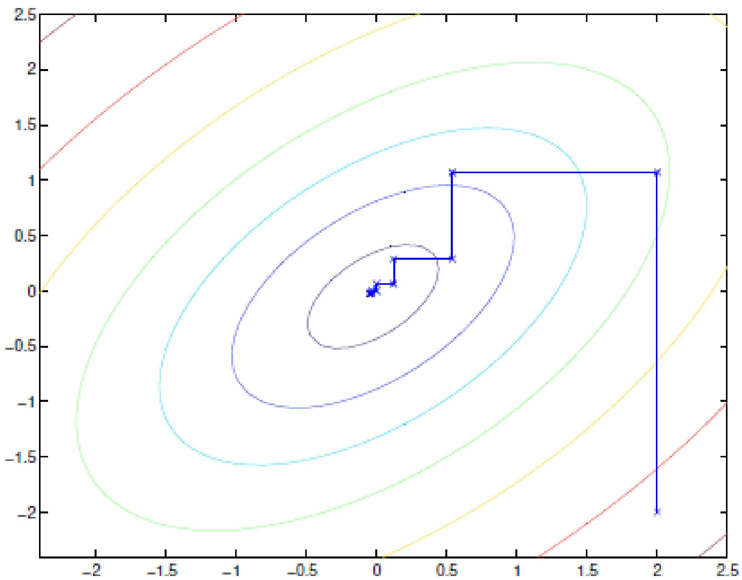


图 3: Coordinate Assent

**methodology** Loop until convergence{

For i = 1, 2, ...,n{

$\alpha_i := \underset{\hat{\alpha}_i}{argmax}\, W(\alpha_1, ..., \hat{\alpha}_i, ..., \alpha_n)$

}

}

## 4 Linear Regression

**scenario** get $\hat{y}$ when ys are continuous. e.g. factor analysis, trend forecast, planning scheme.

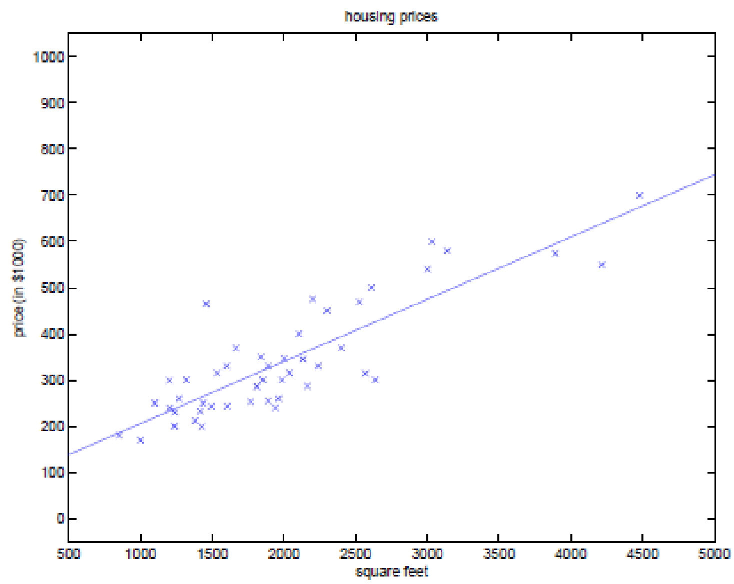


图 4: linear regression

$$h_\theta(X) = \theta^T X \tag{6}$$

$$y^{(i)} = h_\theta(x) + \epsilon^{(i)} \tag{7}$$

Regard $\epsilon$ as i.i.d which follows gaussion distribution $N(0, \sigma^2)$.Then add complexity penalty factor(L1,L2) in Elastic Net.

$$J(\theta) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda(\rho \sum_{j=1}^{n} |\theta_j| + (1-\rho) \sum_{j=1}^{n} \theta_j^2) \tag{8}$$

**local linear regression** define weight $\omega$

$$\omega^{(i)} = \exp\{-\frac{(x^{(i)} - x_0)^2}{2\tau^2}\} \tag{9}$$

Given $x_0$, fit $\theta$ to minimize $\sum \omega^{(i)}(y^{(i)} - h_\theta(x^{(i)}))^2$. And $\tau$ is called bandwidth parameter which controls how quickly the weight of a training example falls off with distance of its $x^{(i)}$ from the query point $x_0$.

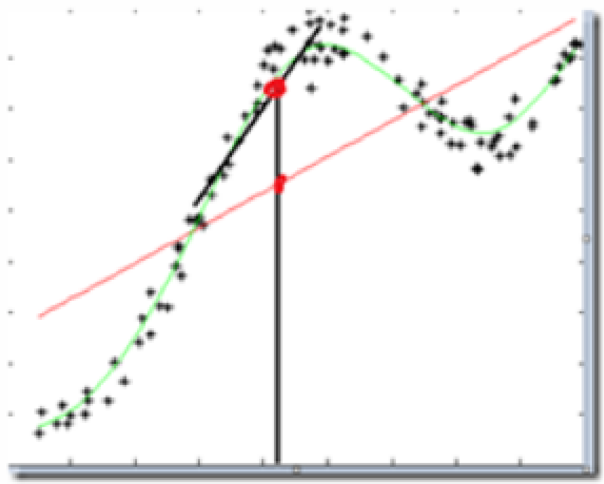

图 5: local linear regression

# 5 Logistic Regression

**scenario** Classify Xs when ys $\in$ {0,1}, namely output the probability for specific X.

**input** $X^{(i)}$(i=1,...,m),Y($Y_i$ is the classification of $X^{(i)}$).

**output** probability for $X^{(new)}$ in class 1.

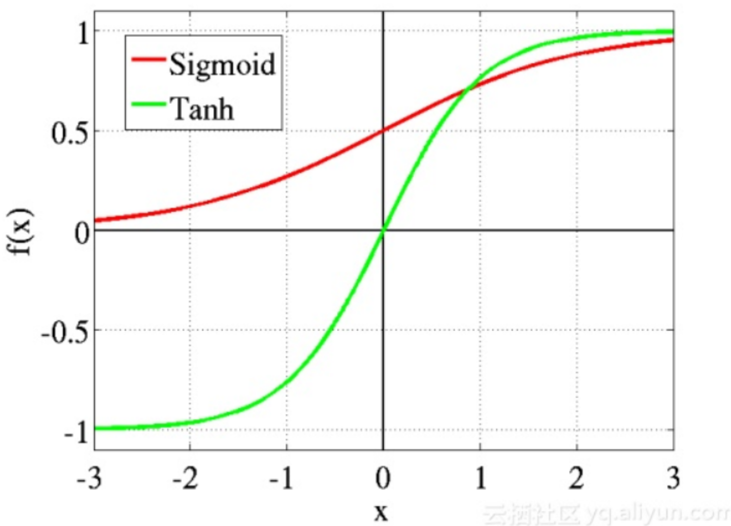


图 6: logistic regression

**methodology**

$$h_\theta(X) = \frac{1}{1+\exp(-\theta^T X)} \tag{10}$$

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y_i - h_\theta(X^{(i)}))X_j^{(i)} \tag{11}$$

# 6 Softmax regression

**scenario** Classify Xs when $Y_i \in \{1,2,...,k\}$,$Y_i$ is the classification of $X^{(i)}$, and labels are mutually exclusive.

**input** $X^{(i)}$(i=1,...,m),Y.

**output** probability for $X^{(new)}$ in class 1,2...,k.

**methodology** assume $\phi_1, \phi_2, ..., \phi_k$ are the probablities for X in class 1,2,...,k.
define

$$T(y) = \begin{bmatrix} 0 \\ .. \\ 0 \\ 1 \\ 0 \\ .. \end{bmatrix} \tag{12}$$

if y=k.

$$\eta = \begin{bmatrix} log\frac{\phi_1}{\phi_k} \\ log\frac{\phi_2}{\phi_k} \\ ... \\ log\frac{\phi_{k-1}}{\phi_k} \end{bmatrix} \tag{13}$$

Therefore

$$\phi_i = \frac{\exp(\eta_i)}{1 + \sum_{j=1}^{k-1} \exp(\eta_j)} \tag{14}$$

according to general linear model(GML), substitude $\eta = \theta^T X$ into eqn(14), and do maximum likelihood estimation for $\theta$.

# 7 Gaussian Discriminant Analysis

**scenario** assume that $p(X|y)$ is distributed according to a multivariate normal distribution, we can classify $X^{new}$ by comparing $p(y = k|X), k = 1, 2, ..., K$. In that case, generative algorithms(e.g. GDA) will be better than distrimination alogorithms(e.g. logistic regression, softmax regression) when classifying X, since more info. is included in the distribution of X given y.

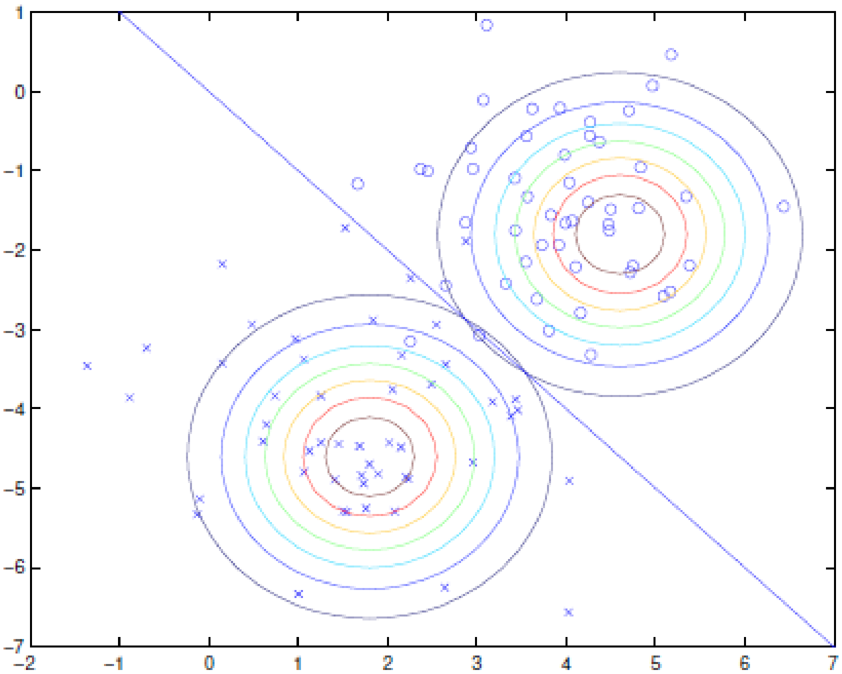

图 7: GDA

**input** $X^i$(i=1,...,m),Y.

**output** class for $X^{new}$.

**methodology** adopt Bayse.

$$
\begin{aligned}
predict\_class &= \underset{y}{argmax}\, P(y|X) \\
&= \underset{y}{argmax}\, \frac{P(X|y)P(y)}{P(X)} \\
&= \underset{y}{argmax}\, P(X|y)P(y)
\end{aligned}
\tag{15}
$$

**relation to logistic regression** if $p(X|y)$ is multivariate gaussian ,then $p(X|y)$ necessarily follows a logistic function. The converse, however, is not true.

GDA makes stronger modeling assumptions, and is more data efficient (i.e., requires less training data to learn "well") when the modeling assumptions are correct or at least approximately correct.

Logistic regression makes weaker assumptions, and is significantly more robust to deviations from modeling assumptions.

## 8 Naive Bayse

**scenario** model P($X|y$) and use it to calculate P($y|X$)(classify X).

**methodology**

$$
\begin{aligned}
P(X|y) =& P(X_1, X_2, ..., X_n|y) \\
=& P(X_1|y)P(X_2|X_1, y)...P(X_n|X_1, X_2, ..., y) \\
=& \prod_{i=1}^{n} P(X_i|y)
\end{aligned}
\tag{16}
$$

$$
P(y = k|X) = \frac{P(X|y = k)P(y = k)}{P(X)} = \frac{\prod_{i=1}^{n} P(X_i|y = k)P(y = k)}{P(X)}
\tag{17}
$$

ps: for set like{12.3, 45.5, 67.0,...,98} which contains continous value, we can discretize it into intervals[0,9],[10,19]...[90,100], namely class 1,2,...10.

$\triangle$ When the original, continuous-valued attributes are not well-modeled by a multivariate normal distribution, discretizing the features and using Naive Bayes (instead of GDA) will often result in a better classifier.

**Laplace Smoothing** when feature $X_k$ in testing examples does not exits in training examples(P($X_k|y$) = 0), correction should be done to aviod info. of other features being erased in Naive Bayse algorithm.

$$
\hat{P}(X_i|y) = \frac{\#(X_i, y) + 1}{\#y + N}
\tag{18}
$$

N : #possible types of $X_i$.

**Example 1: Multi-Variate Bernoulli Event Model** for item 1,2,...,m, create feature vectors $X^{(i)}(X_j^{(i)} \in \{0, 1\})$. $X_j^{(i)} = 0$ indictes item i doesn't have feature j, and 1 otherwise.Then adopt Naive Bayse to classify item i based on $X^{(i)}$.

- input : $X^{(1)}, X^{(2)}, ...X^{(m)}$, $Y \in R^m$,$X^{(new)}$.
- output: class for $X^{(new)}$.

**Example 2: Multinomial Event Model** if #features >> 1. Then Multi-Variate Bernoulli Event Model is no longer reasonable. So we can create feature vector Xs based on the discretization of items instead. Then adopt Naive Bayse to classify item i based on $X^i$.

- input : $X^{(1)}, X^{(2)}, ...X^{(m)}(X_j^{(i)} \in \{1, 2, ..., k\}$, say, a key to certain word), $Y \in R^m$,$X^{(new)}$.
- output: class for $X^{(new)}$.

# 9 Perceptron

**scenario** when $X_1, ...X_m$ are linearly separable, and $y_i \in$ {-1, 1}.

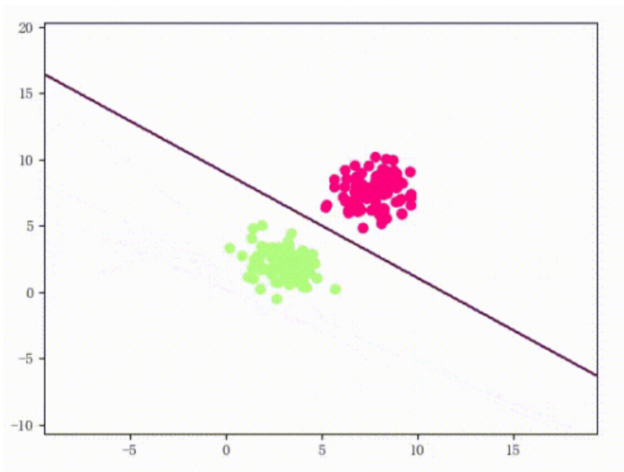


图 8: perceptron

**methodology** try to find a hyperplane that separates all binary categories.
minimaize loss function

$$L(\omega, b) = -\sum_{i=1}^{m} y_i(\omega^T X_i + b) \tag{19}$$

Concretely, update the gradient with a misclassified point$(X_i, y^i)$

$$\omega := \omega + \alpha y_i X_i \tag{20}$$

$$b := b + \alpha y_i \tag{21}$$

## 10 Support Vector Machine

**scenario** suitable for small-scale non-parametric* linear classification problems. Especially, after transforming SVM into the dual problem, the classification only needs to calculate the distance between SVM and a few support vectors, which has obvious advantages in the calculation of complex kernel functions and can greatly simplify the model and calculation.

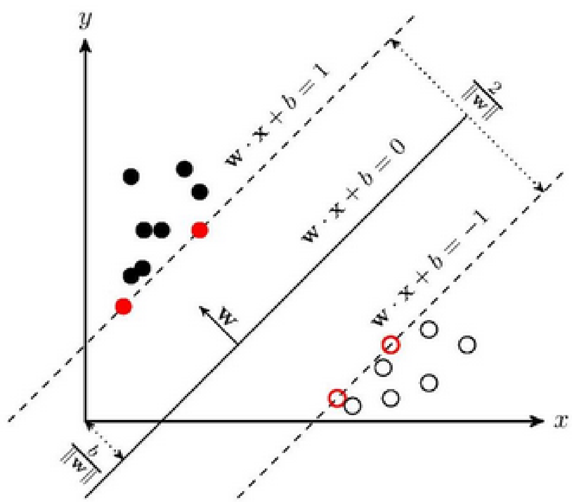


图 9: svm

**methodology** $X \xrightarrow{\phi} \phi(X)$ so that X is linearly separable.

$$
\begin{aligned}
&\underset{\omega,b}{min}\, \frac{1}{2}\left\|\omega\right\|^2 \\
&s.t.\ y_i(\omega^T\phi(X_i)+b) \geq 1,\ \ i=1,2,...,m
\end{aligned}
\tag{22}
$$

Dual problem:

$$
\begin{aligned}
&\underset{\alpha}{max}\, W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \phi(X_i)^T\phi(X_j) \\
&s.t.\ \sum_{i=1}^{m}\alpha_i y_i = 0, \\
&\alpha_i \geq 0,\ \ i=1,2,...,m
\end{aligned}
\tag{23}
$$

**optimization** SMO(Sequential minimal optimization):

Repeat{

Select $\alpha_i, \alpha_j$
Hold all $\alpha_k$ fixed, except $\alpha_i, \alpha_j$
Optimize W($\alpha$) w.r.t. $\alpha_i, \alpha_j$(subject to all the constraints).
}

**L1 norm soft margin SVM** adopt this sdue to two cases:

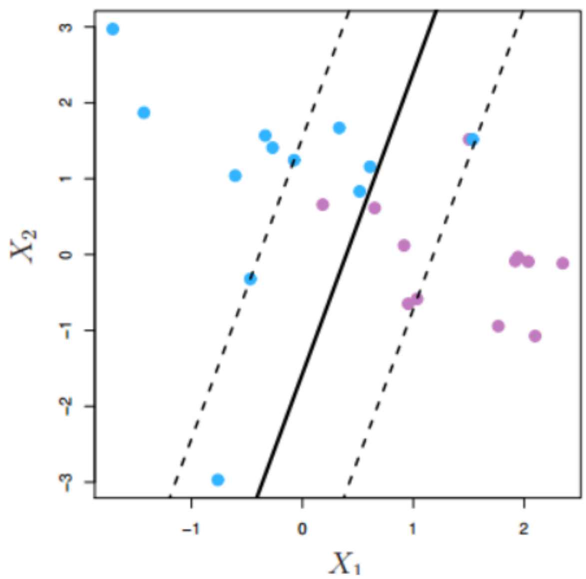


图 10: soft-svm

- data set is not linear separable
- want to overlook exception

In other words, allow some data to "cross" hyperplane.

$$\begin{aligned} &\min_{\omega,b} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}\epsilon_i \\ &s.t.\ y_i(\omega^T\phi(X_i)+b) \geq 1-\epsilon,\ \epsilon_i \geq 0, i=1,2,...,m \end{aligned} \tag{24}$$

Dual problem:
make only one correction: "$\alpha_i \geq 0,\ i=1,2,...,m$ " $\rightarrow$ "$C \geq \alpha_i \geq 0$"
*ps: $< non-parametric >$ algorithms that do not make too many assumptions about the form of

the objective function are called nonparametric machine learning algorithms.Without assumptions, the algorithm is free to learn any form of the function from the training data.

# 11 K-means

**scenario** divide points into k groups. Points in the same group have large similarity while points otherwise have large distinction.

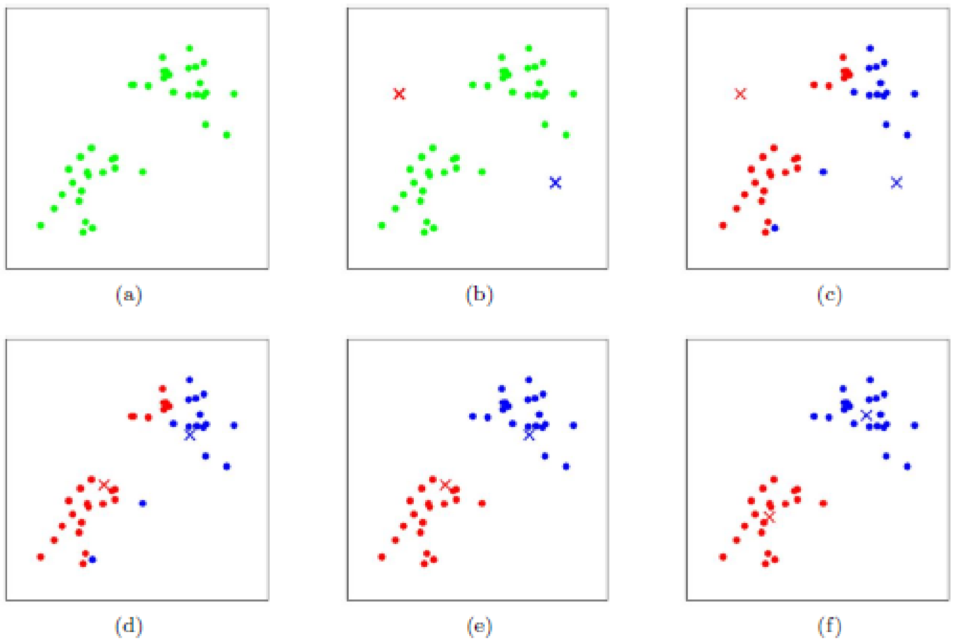


图 11: KMeans

**methodology**

- Initialize cluster centroids $\mu_1, \mu_2, ..., \mu_k$ randomly.
- Repeat until convergence:

  { For every i, set

$$c^{(i)} = \underset{j}{argmin} \left\| x^i - \mu_j \right\|^2 \tag{25}$$

  For each j, set

$$\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} c^{(i)} = j} \tag{26}$$

}

**limitation** KMeans can not be adpoted when each dimention of data is not uniform and has different varience.

## 12 Gaussian Mixture Model

**scenario** From the central limit theorem, as long as the model is sufficiently complex and the sample size is sufficient, each small region can be described by the gaussian distribution. Moreover, GMM is widely used since the gaussian function has good computational performance.

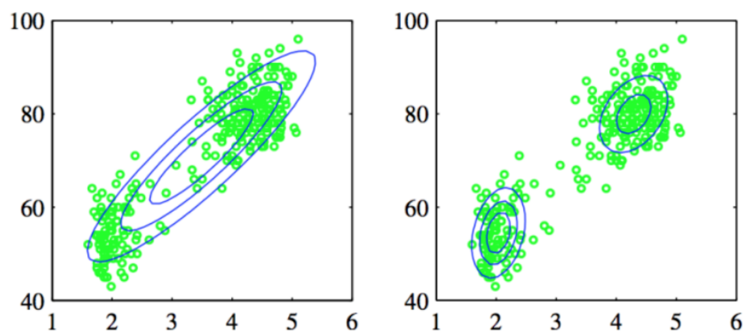


图 12: GMM

**methodology**

$$
\begin{aligned}
p(X) &= \sum_{k=1}^{K} p(X)p(X|k) \\
&= \sum_{k=1}^{K} \pi_k N(X|\mu_k, \sum\nolimits_k) \\
s.t. &\sum_{k=1}^{K} \pi_k = 1
\end{aligned}
\tag{27}
$$

Then adopt EM algorithm to do the optimization, i.e. work out the parameters $\mu_1, \sigma_1, ..., \mu_k, \sigma_k$.

# 13 Factor Analysis

**scenario** extract hidden common factors from variable groups.e.g. satisfaction survey, information concentration, weight calculation and comprehensive competitiveness study.


图 13: Factor Analysis

**methodology** assume $z \in \mathbb{R}^k$ is a latent random variable.

$$
\begin{gathered}
z \in \mathcal{N}(0,1) \\
\epsilon \in \mathcal{N}(0,\Psi) \\
x = \mu + \Lambda z + \epsilon
\end{gathered}
$$

Furthermore,

$$
\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} 0 \\ \mu \end{bmatrix}, \begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix} \right) \tag{28}
$$

$$
l(\mu,\Lambda,\Psi) = log \prod_{i=1}^{n} \frac{1}{(2\pi)^{d/2}|\Lambda\Lambda^T + \Psi|^{1/2}} \exp\left( -\frac{1}{2}(x^{(i)}-\mu)^T(\Lambda\Lambda^T+\Psi)^{-1}(x^{(i)}-\mu) \right) \tag{29}
$$

Then derive EM algorithm to maximize this formula.

# 14 Principal Components Analysis

**scenario** adopt this algorithm usually due to the following reasons:

- remove noise of data
- compress high dimensional data(also convinent for visualization)
- anomaly detection

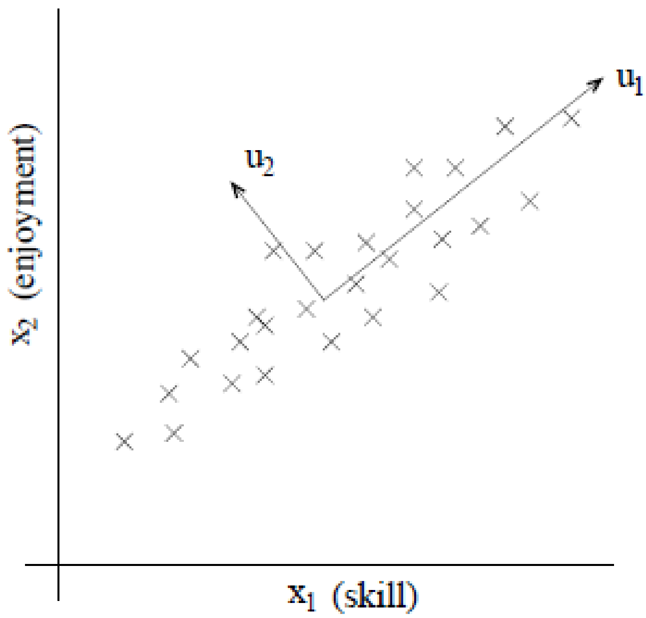


图 14: PCA

**methodology** normalizing each feature to have mean 0 and variance 1

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j} \tag{30}$$

Then do EVD for $XX_T$, choose the top k eigenvectors $u_1, u_2, ..., u_k$ to form projection matrix $\Sigma$.

# 15 Independent Components Analysis

**scenario** as long as data isn't Gaussian ,given enough data we can recover n independent sources that had generated it.

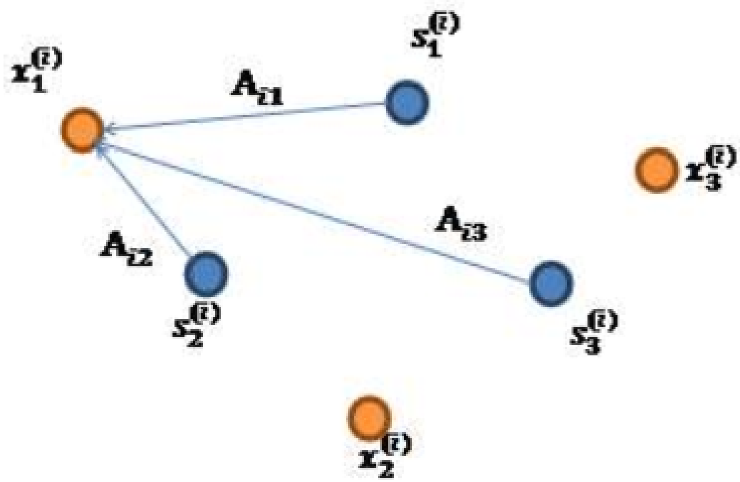


图 15: ICA

**methodology** repeated observations gives us a dataset with m records at time 1,2,...,n $\{x^{(i)}; i = 1, ..., n\}$, and we have d independent sources $\{s^{(i)}; i = 1, ..., d\}$. Assume $A \in \mathbb{R}^{m*d}$

$$\begin{bmatrix} x_1^{(1)} & .. & x_1^{(n)} \\ | & .. & | \\ x_m^{(1)} & .. & x_m^{(n)} \end{bmatrix} = A \begin{bmatrix} s_1^{(1)} & .. & s_1^{(n)} \\ | & .. & | \\ s_d^{(1)} & .. & s_d^{(n)} \end{bmatrix} \tag{31}$$

$$x = As \tag{32}$$

To solve for s, define $W = A^{-1}$(means generalized inverse if $m \neq d$). Suppose that the distribution of each source $s_j$ is given by a density $p_s$ and let $p_s = g'(s)$, where $g(s) = 1/(1 + e^{-s})$. Since $x = W^{-1}s$, (time fixed)joint distribution of the x is given by

$$p(x) = \prod_{i=1}^{m} p(x_i) = \prod_{i=1}^{m} p_s(\omega_j^T x)|W| \tag{33}$$

Then we do maximum likelihood estimation for W.

$$l(W) = \sum_{i=1}^{n} \left( \sum_{j=1}^{m} log\ g'(\omega_j^T x^{(i)}) + log|W| \right) \tag{34}$$

Concretly,

$$W := W + \alpha \left( \begin{bmatrix} 1 - 2g(\omega_1^T x^{(i)}) \\ 1 - 2g(\omega_2^T x^{(i)}) \\ \cdots \\ 1 - 2g(\omega_d^T x^{(i)}) \end{bmatrix} x^{(i)T} + (W^T)^{-1} \right) \tag{35}$$

Notice that in order for ICA to work, it requires at least one different recording for each signal you want to unmix. So if you have two musical instruments playing together in a room, and want to unmix them to get separate recordings of each individual instrument, you' ll need two different recordings of the mixture to work with (like a stereo microphone). If you have three instruments playing together, you' ll need three microphones to separate out all three original signals, etc.

# 16 Markov Decision Processes

**scenario** It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.

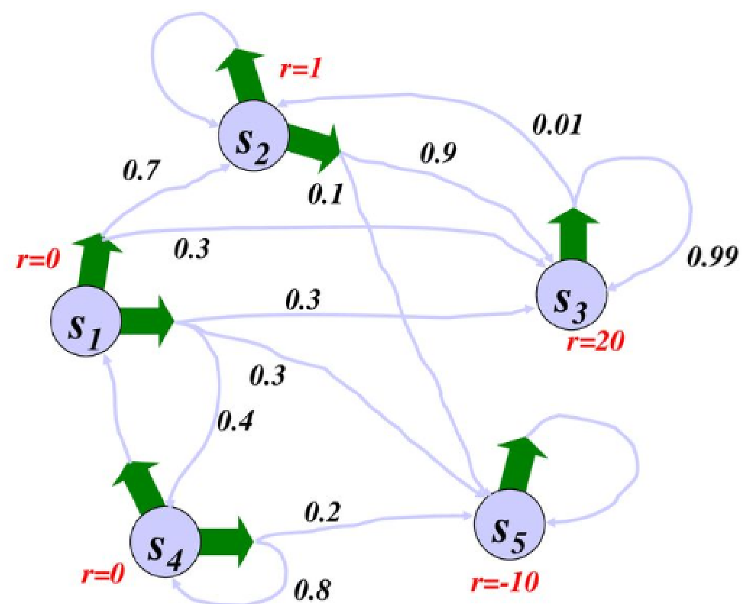


图 16: MDP

**methodology** Present a Markov decision process with tuple $(S, A, \{P_{sa}\}, \gamma, R)$, where S is a set of states, A is a set of actions, $P_{sa}$ are the state transition probabilities, $\gamma \in [0, 1)$ is called the discount factor, $R : S \times A \to R$ is the reward function.

A policy is any function $\pi : S \to A$ mapping from the states to the actions. We say that we are executing some policy $\pi$ if, whenever we are in state s, we take action a = $\pi$(s). We also define the value function for a policy $\pi$ according to

$$V^{\pi}(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + ...|s_0 = s, \pi] \tag{36}$$

Policy fixed, $V^{\pi}$ satisfies the ***Bellman equation***:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s')V^{\pi}(s') \tag{37}$$

Then define the optimal value function according to

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{38}$$

To optimize the value function, we adopt

- value iteration:

  1. For each state s, initialize $V(s)$:=0.

  2. Repeat until convergence{

  For every state, update $V(s) := R(s) + max_{a \in A} \gamma \sum_{s'} P_{sa}(s')V(s')$}

- or policy iteration:

  1. Initialize $\pi$ randomly.

  2. Repeat until convergence{

  (a) Let $V := V^{\pi}$.

  (b) For each state s, let $\pi(s) := arg\ max_{a \in A} P_{sa}(s')V(s')$}

If you want to learn a model for an MDP, you can first use the accumulated experience in the MDP to update our estimates for $P_{sa}$ and $R$, then run value iteration or policy itertion.

**Continous MDP** discretize space into finite states for problems up to 3D, which usually works well.

An alternative method, called value function approximation, is to approximate $V^{\star}$ directly, without resorting to discretization.

- Using a simulator

  Fixed some forms for $s_t, a_t$ to generlize $s_{t+1}$, like

  $$s_{t+1} = As_t + Ba_t \tag{39}$$

  and use data generlized by simulator to estimate parameters.

- Fitted value iteration

  Our goal is to perform the update:

$$
\begin{aligned}
V(s) &:= R(s) + \gamma \max_a \int_{s'} P_{sa}(s')V(s')ds' \\
&= R(s) + \gamma \max_a E_{s' \sim P_{sa}}[V(s')]
\end{aligned} \tag{40}
$$

  assume action space A is small and dicrete and V(s) can be presented by linear combination of some appropriate feature mapping of the states., namely $V(s) = \theta^T \phi(s)$. Then we come up with $y^{(i)}$ wihch is an estimate of $R(s^{(i)}) + \gamma \max_a E_{s' \sim P_{s^{(i)}a}}[V(s')]$ then update $\theta$ using supervised learning.

# 17 Decision Tree

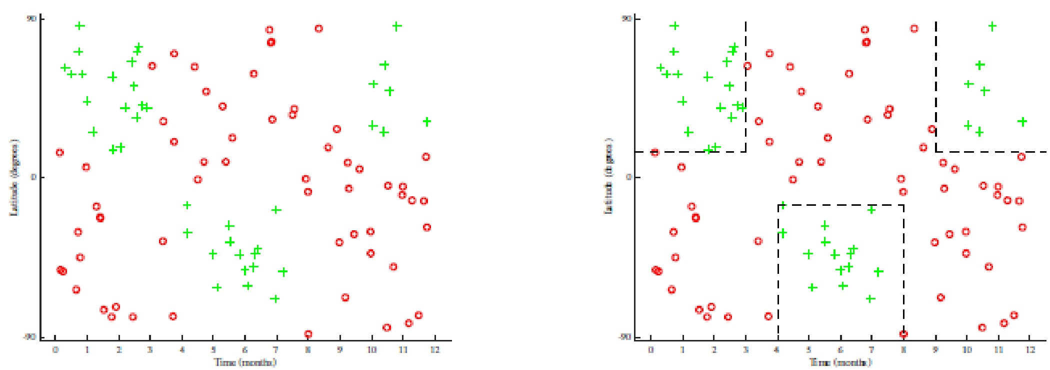

图 17: DT

# 18 Neural Network

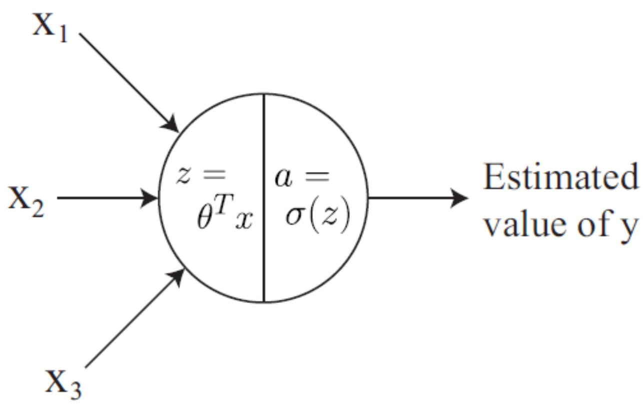


图 18: NN