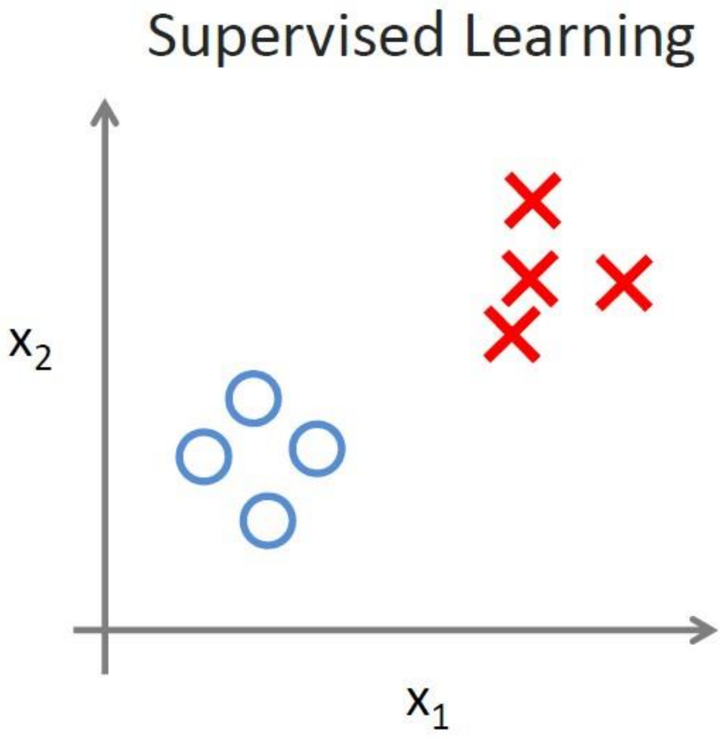
Supervised Learning Methods

lanyanyan@ict.ac.cn

What is Supervised Learning?



Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Regression Methods

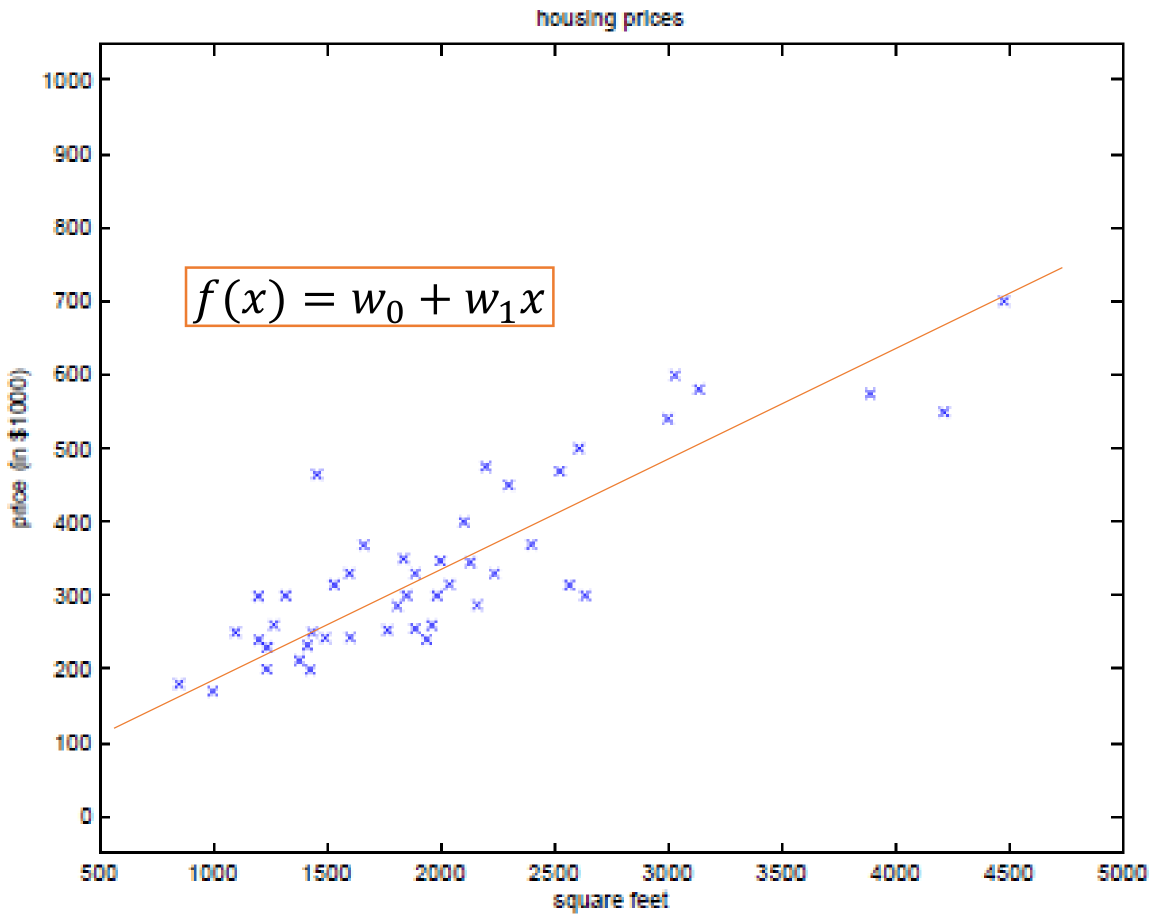
Regression

- Input: n i.i.d. training samples $\left(x^{(i)}, y^{(i)}\right) \in X \times R, i = 1, \cdots, n$
- Hypothesis: $f \in \mathcal{F}$
- Loss function: $L(f; x, y) = (f(x) - y)^2$
- Expected Risk: $\int (f(x) - y)^2 dP(x, y)$

Linear Regression with one Variable

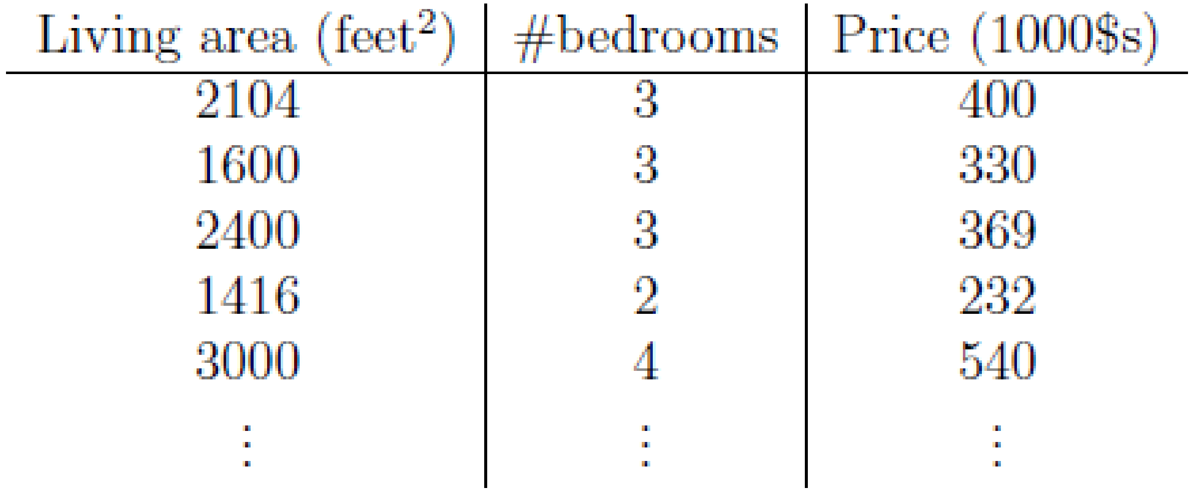
- Example:
  - Suppose we have a dataset giving the living area and prices of 47 houses from Portland, Oregon:

Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



Mutivariate Linear Regression

- Both living area and number of bedroom are given:



Living area (feet^2)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
$\vdots$	$\vdots$	$\vdots$

- $x = (x_1, x_2)$, where x_1 stands for the living area and x_2 is the number of the bedroom.
- Linear Function: $f(x) = w_0 + w_1 x_1 + w_2 x_2 = \vec{w}^T x$

Closed Form of Solution

- Optimization Problem $\min_{\vec{w}} J(\vec{w}) = \sum_{i=1}^{n} \left(\vec{w}^T x^{(i)} - y^{(i)}\right)^2$
- Gradient:

$$\frac{\partial J(\vec{w})}{\partial w_j} = \sum_{i=1}^{n} 2\left(\vec{w}^T x^{(i)} - y^{(i)}\right) x_j^{(i)} = 0$$

- Closed form of unique form:

$$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$$

where $X^T = (x^{(1)}, \cdots, x^{(n)})$, $\vec{y}^T = (y^{(1)}, \cdots, y^{(n)})$.

Least Mean Squares Algorithm

- Optimization Problem $\min_{\vec{w}} J(\vec{w}) = \sum_{i=1}^{n} \left(\vec{w}^T x^{(i)} - y^{(i)}\right)^2$
- Gradient Decent:

$$\frac{\partial J(\vec{w})}{\partial w_j} = \sum_{i=1}^{n} 2\left(\vec{w}^T x^{(i)} - y^{(i)}\right) x_j^{(i)}$$

- Update rule:

$$\boxed{w_j := w_j + 2\alpha \sum_{i=1}^{n} \left(\vec{w}^T x^{(i)} - y^{(i)}\right) x_j^{(i)}}$$

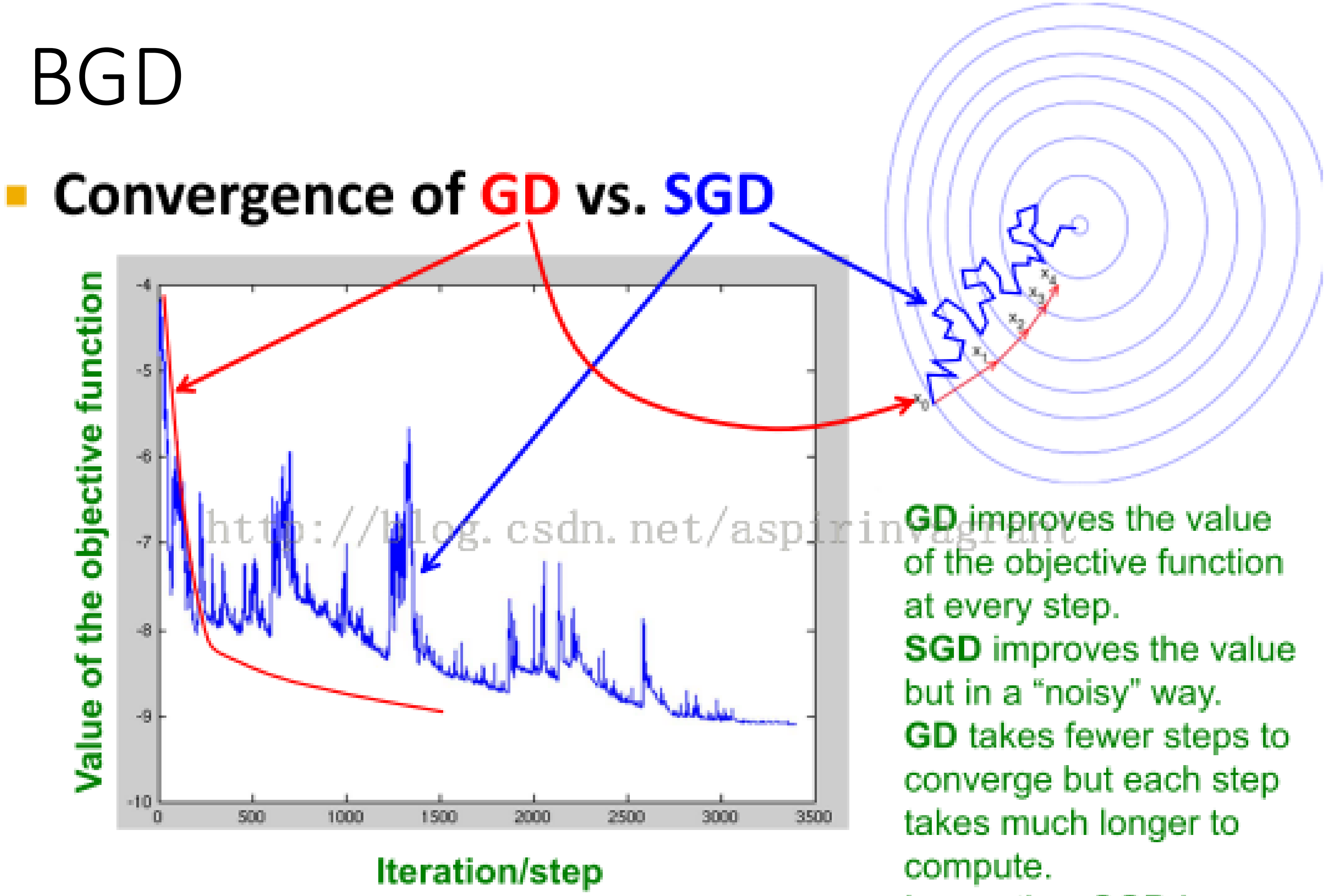
Batch Gradient Descent

$$\boxed{w_j := w_j + 2\alpha \left(\vec{w}^T x^{(i)} - y^{(i)}\right) x_j^{(i)}}$$

Stochastic Gradient Descent

SGD vs. BGD

- **Convergence of GD vs. SGD**



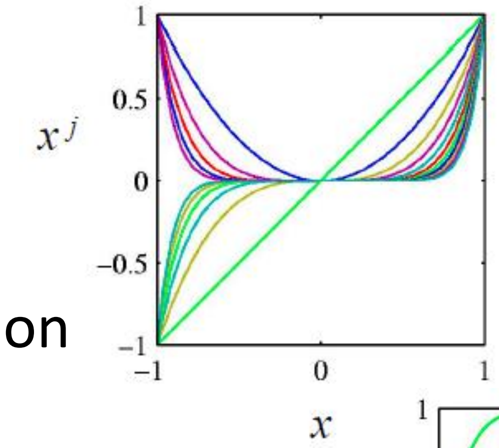
GD improves the value of the objective function at every step.
SGD improves the value but in a “noisy” way.
GD takes fewer steps to converge but each step takes much longer to compute.
In practice, **SGD** is much faster!

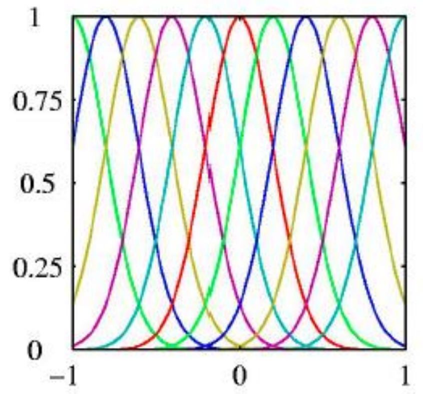
Linear Regression with Nonlinear Basis

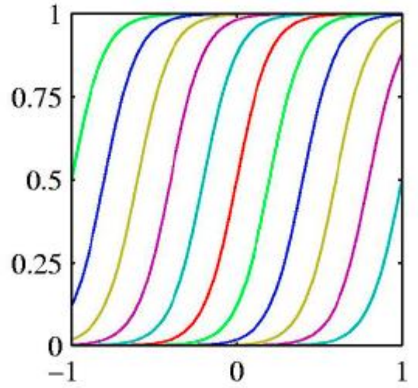
- Consider linear combination of nonlinear basis functions of the input variables:

$$f(\vec{w}, x) = w_0 + \sum_{j=1}^{M} w_j \phi_j(x) = \vec{w}^T \phi(x)$$

  - $\phi = (1, \phi_1, \cdots, \phi_M)$
- Examples of basis functions:
  - $\phi(x) = (1, x, x^2, \cdots, x^M)$, polynomial basis function
  - $\phi_j(x) = \exp(-\frac{(x-\mu_j)^2}{2s^2})$, Gaussian basis function
  - $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right), \sigma(a) = \frac{1}{1+\exp(-a)}$, Sigmoid basis function







Closed Form of Solution for Nonlinear Basis

- Optimization Problem: $$\min_{\vec{w}} J(\vec{w}) = \sum_{i=1}^{n} \left(\vec{w}^T \phi(x^{(i)}) - y^{(i)}\right)^2$$
- Gradient: $$\frac{\partial J(\vec{w})}{\partial w_j} = \sum_{i=1}^{n} 2\left(\vec{w}^T \phi(x^{(i)}) - y^{(i)}\right) x_j^{(i)} = 0$$
- Closed form of unique form:

$$\vec{w}^* = (\Phi^T \Phi)^{-1} \Phi^T \vec{y}$$

where $\Phi = \begin{pmatrix} \phi_0(x^{(1)}) & \cdots & \phi_M(x^{(1)}) \\ \vdots & \vdots & \vdots \\ \phi_0(x^{(n)}) & \cdots & \phi_M(x^{(n)}) \end{pmatrix}$, $\vec{y}^T = (y^{(1)}, \cdots, y^{(n)})$.

- Can also be solved by SGD

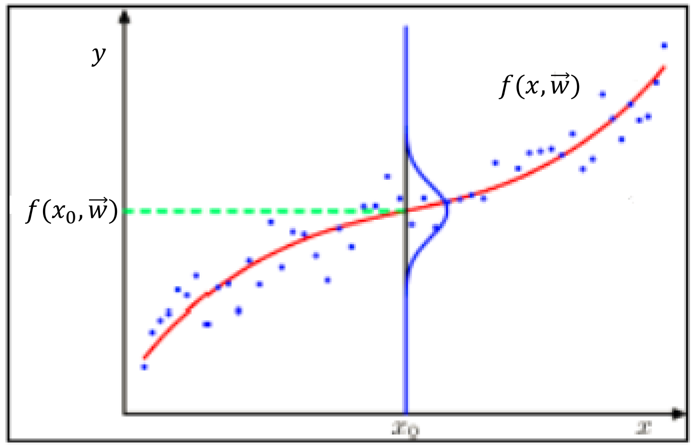
Probabilistic Explanation: MLE

- Minimizing sum of square error is the same as maximum likelihood solution under a Gaussian noise model
- Assume that y is a scalar given by deterministic function f with additive Gaussian noise $y = f(x, \vec{w}) + \epsilon$, ϵ is a zero mean Gaussian with precision β

$$p(y|x, \vec{w}, \beta) = N(y|f(x, \vec{w}), \beta^{-1})$$

- Training Data: $\left(x^{(i)}, y^{(i)}\right), i = 1, \cdots, n$
- Likelihood:

$$\prod_{i=1}^{n} N(y^{(i)}|\vec{w}^T x^{(i)}, \beta^{-1})$$



Log-Likelihood Function

- Log-Likelihood

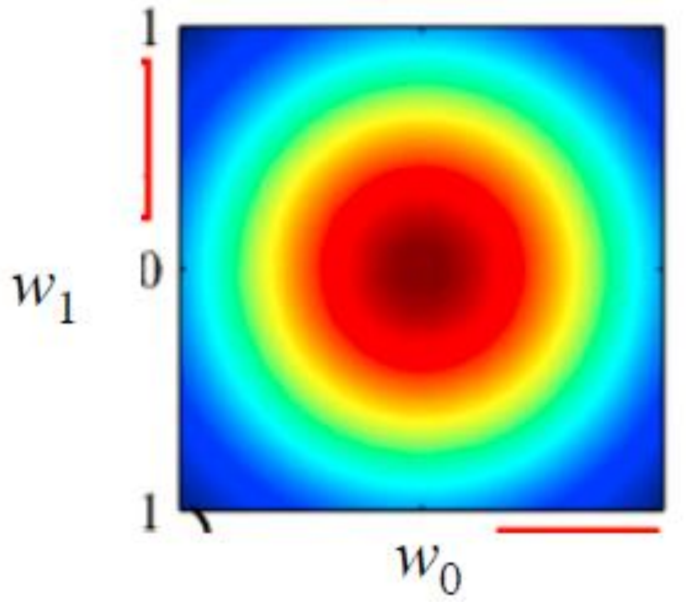
$$\sum_{i=1}^{n} \ln N(y^{(i)} | \vec{w}^T x^{(i)}, \beta^{-1}) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln 2\pi - 2\beta J(\vec{w})$$

where, $J(\vec{w}) = \sum_{i=1}^{n} \left(\vec{w} x^{(i)} - y^{(i)}\right)^2$ *sum of square error*

- Maximizing likelihood is equivalent to minimizing sum of square error

Regularized LMS as MAP

- Optimization Problem: $$\min_{\vec{w}} \sum_{i=1}^{n} \left(\vec{w}^T x^{(i)} - y^{(i)}\right)^2 + \lambda \vec{w}^T \vec{w}$$
- Closed form of solution: $$\vec{w}^* = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T \vec{y}$$
- Likelihood: $\prod_{i=1}^{n} N(y^{(i)} | \vec{w}^T x^{(i)}, \beta^{-1})$
- Prior: $p(\vec{w}) = N(\vec{w} | 0, \lambda^{-1} I)$ multivariate Gaussian
  - The variance of the weights are all equal to λ^{-1}, and co-variances are zero



Posterior Distribution of Parameters

- Bayes Rule: $p(\vec{w}|\vec{y}) = \frac{p(\vec{y}|\vec{w})p(\vec{w})}{p(\vec{y})}$
  - Likelihood function $p(\vec{y}|X, \vec{w}, \beta) = \prod_{i=1}^{n} N(y^{(i)}|\vec{w}^T x^{(i)}, \beta^{-1})$,
  - Prior $p(\vec{w}) = N(\vec{w}|0, \lambda^{-1}I)$,
- Posterior is also Gaussian
- Log of Posterior:

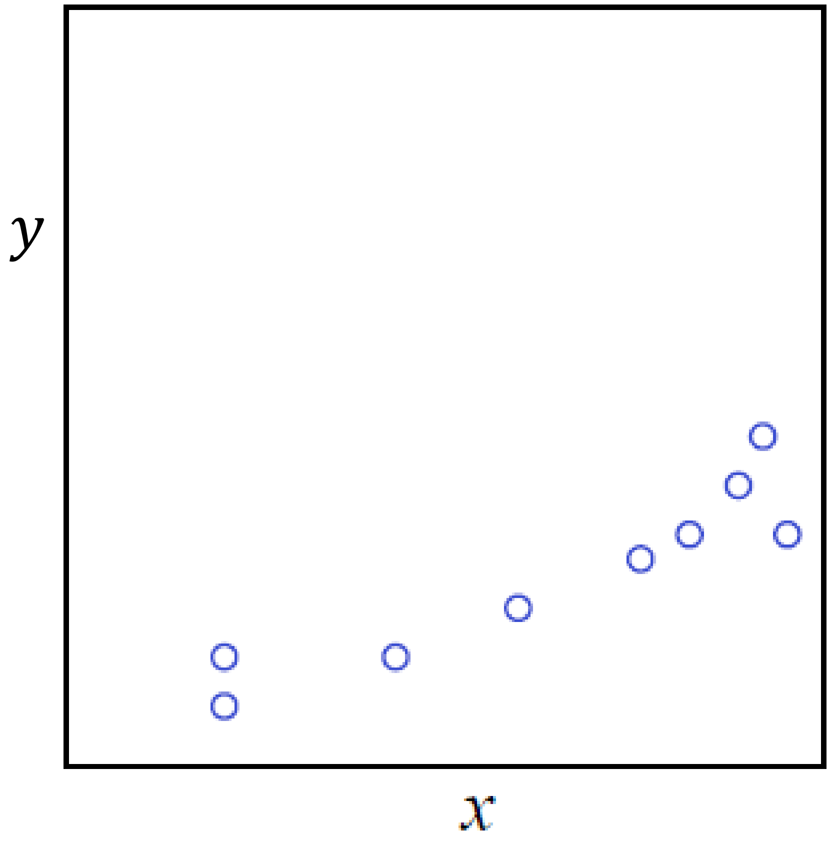
$$\boxed{\ln p(\vec{w}|\vec{y}) = -\beta \sum_{i=1}^{n} \left(y^{(i)} - \vec{w}^T x^{(i)}\right)^2 - \alpha \vec{w}^T \vec{w} + constant}$$

- Maximization of posterior distribution is equivalent to minimization of sum-of-squares error:

$$\boxed{\min_{\vec{w}} \sum_{i=1}^{n} \left(\vec{w}^T x^{(i)} - y^{(i)}\right)^2 + \lambda \vec{w}^T \vec{w}, \lambda = \frac{\alpha}{\beta}.}$$

Example: Straight Line Fit

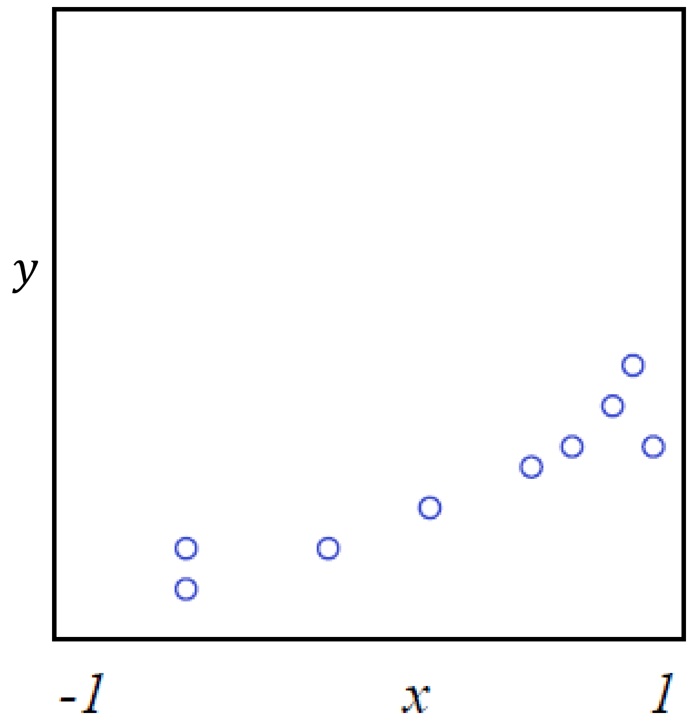
- Single input variable x
- Single target variable y
- Goal is to fit linear model $f(\vec{w}, x) = w_0 + w_1 x$
- Goal of linear regression is to recover $\vec{w} = (w_0, w_1)$ given the samples



Data Generation

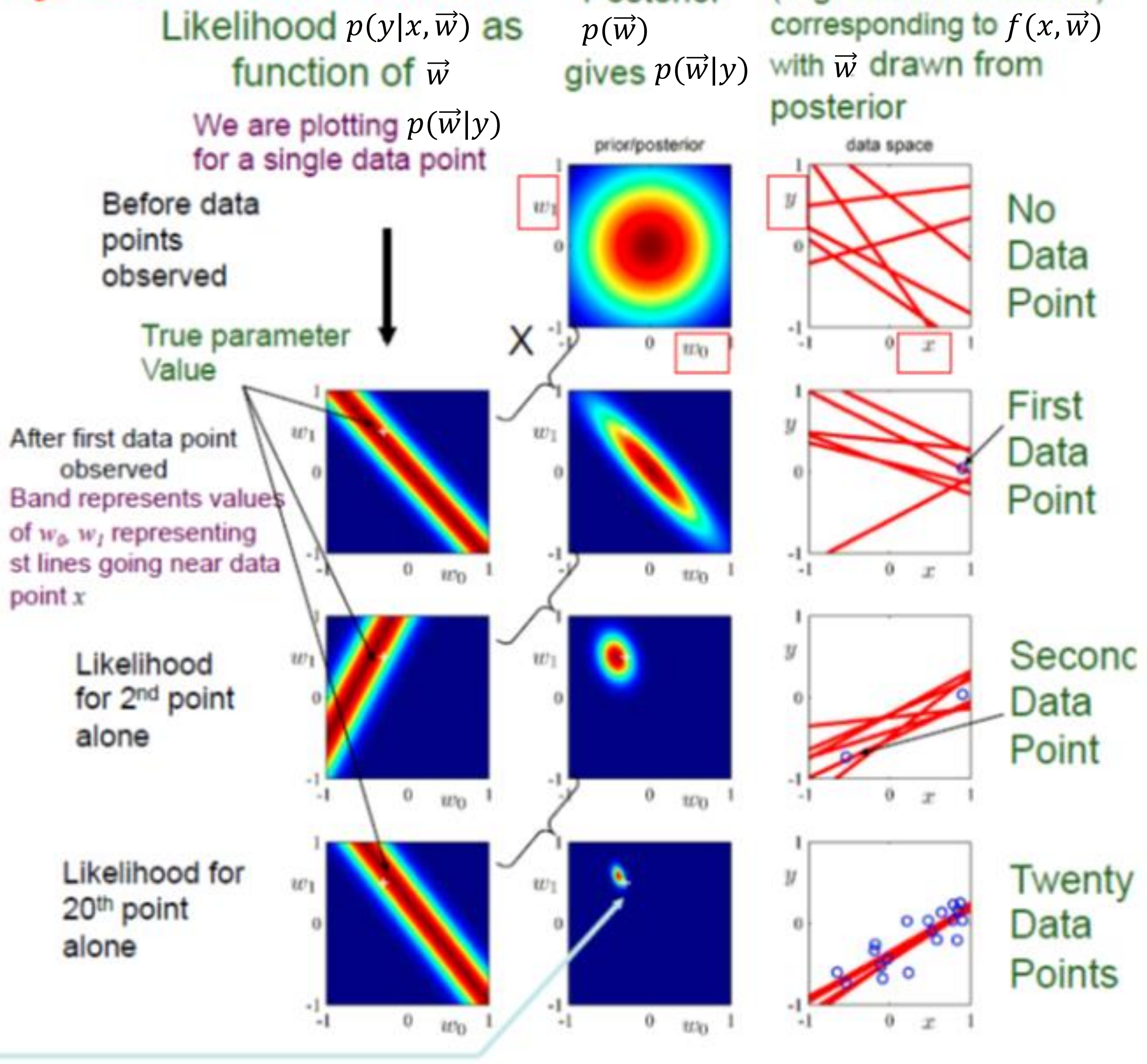
- Synthetic data generated from $f(x, \vec{w}) = w_0 + w_1 x$ with parameters $w_0 = -0.3, w_1 = 0.5$
  - First choose $x^{(i)}$ from $U(-1,1)$, then evaluate $f(x^{(i)}, \vec{w})$
  - Add Gaussian noise with $\sigma = 0.2$ to obtain target $y^{(i)}$, i.e. $\beta = \left(\frac{1}{0.2}\right)^2 = 25$
- For prior over $\vec{w}$ we choose $\alpha = 2$

$$p(\vec{w}|\alpha) = N(\vec{w}|0, \alpha^{-1}I)$$



Sequential Bayesian Learning

- Since there are only two parameters
  - We can plot prior and posterior distributions in parameter space
- We look at sequential update of posterior



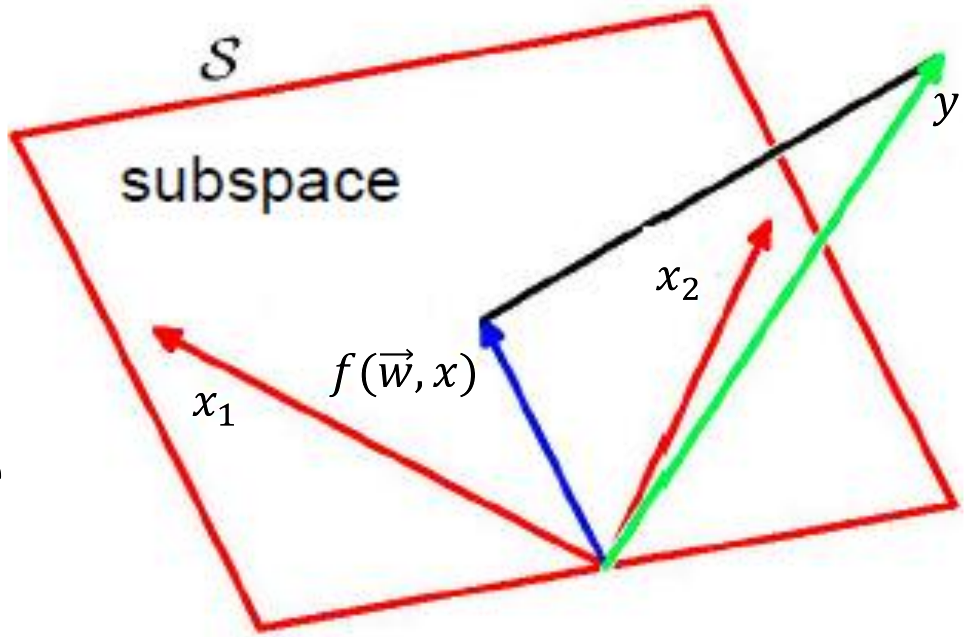
With infinite points posterior is a delta function centered at true parameters (white cross)

MLE vs. MAP

- MLE: $\hat{\theta}_{MLE} = \arg\max_{\theta} P(D|\theta)$
- MAP: $\hat{\theta}_{MAP} = \arg\max_{\theta} P\,(\theta|D) = \arg\max_{\theta} P(D|\theta)P(\theta)$
- MLE can be viewed as MAP with flat prior
- MLE is a frequency idea, while MAP is totally Bayesian
- Analogy to ERM vs. SRM
- More data will make MLE fit better but easy to overfitting
- MAP can produce the result of regularization through adding the prior of the parameter, which can also be viewed as an approach of model selection

Geometry of LMS

- $y = \left(y^{(1)}, \cdots, y^{(n)}\right)$ lies in a n dimensional space.
- If the dimension of $\vec{w}$, denoted by m is small than n, then x_js are in subspace $\mathcal{S}$ of dimension m.



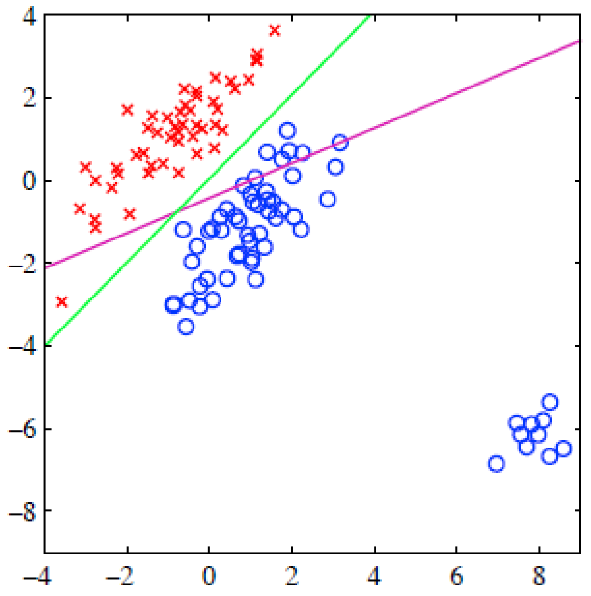
- $x = (x^{(1)}, \cdots, x^{(n)})$, then $f(\vec{w}, x)$ is a n dimensional vector of elements $f\left(\vec{w}, x^{(i)}\right)$.
- Solution $\vec{w}$ corresponds to $f(\vec{w}, x)$ that lies in subspace $\mathcal{S}$ that is closest to y:
  - Corresponds to orthogonal projection of y onto $\mathcal{S}$

10 min Break

Classification Methods

Classification Problem

- Input: n i.i.d. training samples $(x_i, y_i) \in X \times C, i = 1, \cdots, n,$
- Hypothesis: $f \in \mathcal{F}$
- Loss function:$L(f; x, y) = I_{\{f(x) \neq y\}}$
- Expected Risk:$\int I_{\{f(x) \neq y\}} dP(x, y) = P\{f(x) \neq y\}$



- One solution is to directly apply regression methods to it, but it will suffer from *serious outlier problem*!

Discriminant Function: Perceptron

- Binary Classification: $y = 1$ or -1
- $f(x, \vec{w}) = sgn(\vec{w}^T x) = \begin{cases} 1, \vec{w}^T x \geq 0 \\ -1, \vec{w}^T x < 0 \end{cases}$
- Target Coding Scheme
  - $y = 1$ for class C_1, and $y = -1$ for class C_2
- Loss function:$L(f; x, y) = I_{\{f(x) \neq y\}}$
  - Piecewise constant function of $\vec{w}$ with discontinuities (unlike regression)
  - No closed form solution (No derivatives exist for non smooth functions)

F. Rosenblatt

Perceptron Criterion

- Seek $\vec{w}$ such that:
  - $x \in C_1$, we will have $\vec{w}^T x \geq 0$
  - $x \in C_2$, we will have $\vec{w}^T x < 0$
- Using $y \in \{+1, -1\}$, all patterns need to satisfy:

$$\vec{w}^T xy > 0$$

- For each misclassified sample, *Perceptron Criterion* tries to minimize

$$\widehat{E_p}(\vec{w}) = -\sum_{i \in M} \vec{w}^T x^{(i)} y^{(i)}$$

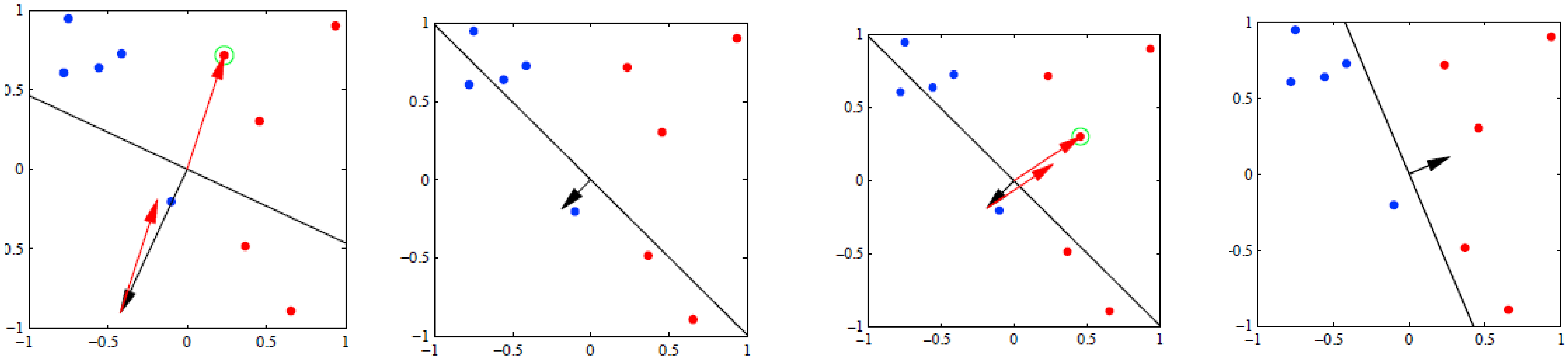
where M denotes set of all misclassified patterns.

Perceptron Algorithm

- Loss Function: $\widehat{E_p}(\vec{w}) = -\sum_{i \in M} \vec{w}^T x^{(i)} y^{(i)}$
- Stochastic Gradient Descent
    - Change in weight is given by $\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta \nabla \hat{E}_P(\vec{w}) = \vec{w}^{(t)} + \eta x^{(i)} y^{(j)}$
    - η is learning rate, t indexes the steps
- The algorithm
    - Cycle through the training patterns in turn
    - If correctly classified for Class C_1, add to weight vector
    - If incorrectly classified for Class C_2, subtract from weight vector

Perceptron Learning Illustration

- Two-dimensional feature space (x_1, x_2) and two classes
- Weight vector in black, green -1, red +1
- Green point is misclassified, which is added to weight vector



Disadvantages of Perceptron

- Does not converge if classes are not linearly separable
- Does not provide probabilistic output

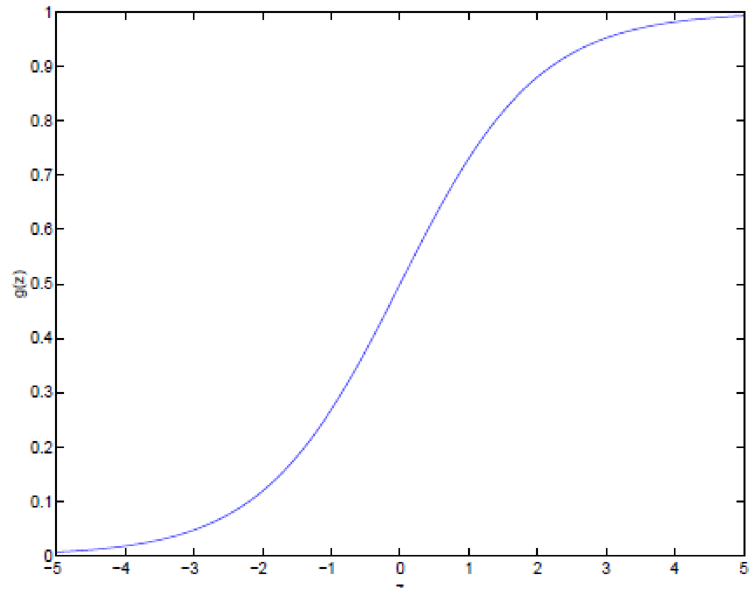
Discriminative Model: Logistic Regression

- The posterior probability $p(y|x)$:

$$P(y=1|x) = f(x, \vec{w}) = g(\vec{w}^T x) = \frac{1}{1+\exp(-\vec{w}^T x)}$$

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

Logistic function
Sigmoid function



- Properties of g(z)
  - $g(z)$ tends towards 1 as $z \to \infty$
  - $g(z)$ tends towards 0 as $z \to -\infty$
  - $g(z)$ is always bounded between 0 and 1
  - $g'(z) = g(z)(1-g(z))$

Maximum Likelihood Estimator

- Probability Distribution $P(y|x,\vec{w}) = \left(f(x,\vec{w})\right)^{y}\left(1 - f(x,\vec{w})\right)^{1-y}$ *Bernoulli*
- Likelihood: $$L(\vec{w}) = \prod_{i=1}^{n} P(y^{(i)}|x^{(i)},\vec{w}) = \prod_{i=1}^{n}\left(f\left(x^{(i)},\vec{w}\right)\right)^{y^{(i)}}\left(1 - f\left(x^{(i)},\vec{w}\right)\right)^{1-y^{(i)}}$$
- Maximize the log likelihood: $$l(\vec{w}) = \log L(\vec{w}) = \sum_{i=1}^{n} y^{(i)} \log f(x^{(i)},\vec{w}) + \left(1 - y^{(i)}\right)\log(1 - f(x^{(i)},\vec{w}))$$
- Gradient: $$\frac{\partial l(\vec{w})}{\partial w_j} = \left(y^{(i)} - f\left(x^{(i)},\vec{w}\right)\right)x_j^{(i)}, \forall(x^{(i)}, y^{(i)})$$
- SGD: $$w_j := w_j + \alpha\left(y^{(i)} - f\left(x^{(i)},\vec{w}\right)\right)x_j^{(i)}$$

Multi-Class Logistic Regression

- Softmax Function instead of logistic sigmoid: $$\boxed{P(C_k|\vec{w}, x) = \frac{\exp(\overrightarrow{w_k}^T x)}{\sum_{j=1}^{K} \exp(\overrightarrow{w_j}^T x)}}$$

where $\overrightarrow{w_1}, \cdots, \overrightarrow{w_K}$ are a set of weight vectors to be learned.

- y can be viewed as discrete variable that takes one of K values
- Represent as 1 of K scheme:
  - Represent y as a K-dimensional vector $\vec{y}$
  - If $y = 3$, then we represent it as $\vec{y} = (0,0,1,0, \cdots, 0)$
  - Such vectors satisfy $\sum_{y=1}^{K} P(y|\vec{w}, x) = 1$
- Probability Distribution: $$\boxed{P(\vec{y}|\mu) = \prod_{y=1}^{K} \mu_y^{\vec{y}_y}}$$ where $\mu_y = P(y|\vec{w}, x)$

Generalized Bernoulli

Multi-Class Likelihood Function

$$P\left(\overrightarrow{y^{(1)}}, \cdots, \overrightarrow{y^{(n)}} \middle| \overrightarrow{w_1}, \cdots, \overrightarrow{w_K}, x^{(1)}, \cdots, x^{(n)}\right)$$
$$= \prod_{i=1}^{n} \prod_{k=1}^{K} P\left(C_k \middle| \overrightarrow{w_k}, x^{(n)}\right) = \prod_{i=1}^{n} \prod_{k=1}^{K} \mu_{nk}^{\overrightarrow{y^{(n)}}_k}$$

where

$$\mu_{nk}^{\overrightarrow{y^{(n)}}_k} = \frac{\exp(\overrightarrow{w_k}^T x^{(n)})}{\sum_{j=1}^{K} \exp(\overrightarrow{w_j}^T x^{(n)})}$$

Optimization for Multi-Class LR

- Optimization Problem:

$$\min E = -\ln P\left(\overrightarrow{y^{(1)}}, \cdots, \overrightarrow{y^{(n)}} \middle| \overrightarrow{w_1}, \cdots, \overrightarrow{w_K}, x^{(1)}, \cdots, x^{(n)}\right)$$

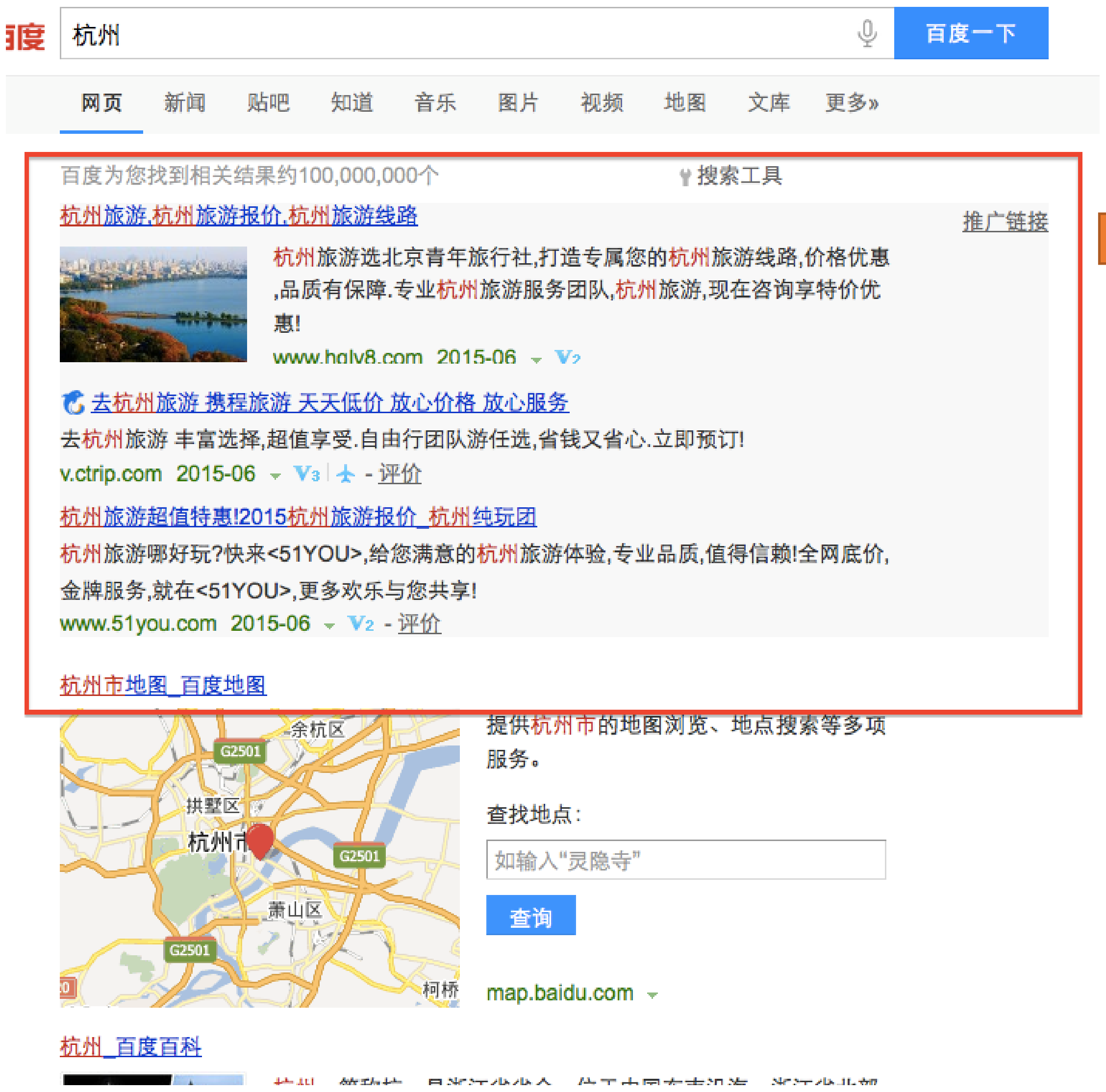
$$\Rightarrow \quad \boxed{\min -\sum_{i=1}^{n}\sum_{k=1}^{K} \overrightarrow{y^{(n)}}_k \ln \mu_{nk}}$$

Cross Entropy Loss Function

- Gradient:

$$\boxed{\nabla_{\overrightarrow{w_j}} E = -\sum_{i=1}^{n}\left(\mu_{nk} - \overrightarrow{y^{(n)}}_k\right) x^{(n)}}$$

Application of Logistic Regression



杭州

网页 新闻 贴吧 知道 音乐 图片 视频 地图 文库 更多»

百度为您找到相关结果约100,000,000个 搜索工具

杭州旅游,杭州旅游报价,杭州旅游线路 推广链接

杭州旅游选北京青年旅行社,打造专属您的杭州旅游线路,价格优惠,品质有保障.专业杭州旅游服务团队,杭州旅游,现在咨询享特价优惠!

www.hqlv8.com 2015-06

去杭州旅游 携程旅游 天天低价 放心价格 放心服务

去杭州旅游 丰富选择,超值享受.自由行团队游任选,省钱又省心.立即预订!

v.ctrip.com 2015-06 - 评价

杭州旅游超值特惠!2015杭州旅游报价_杭州纯玩团

杭州旅游哪好玩?快来<51YOU>,给您满意的杭州旅游体验,专业品质,值得信赖!全网底价,金牌服务,就在<51YOU>,更多欢乐与您共享!

www.51you.com 2015-06 - 评价

杭州市地图_百度地图

提供杭州市的地图浏览、地点搜索等多项服务。

查找地点:

如输入"灵隐寺"

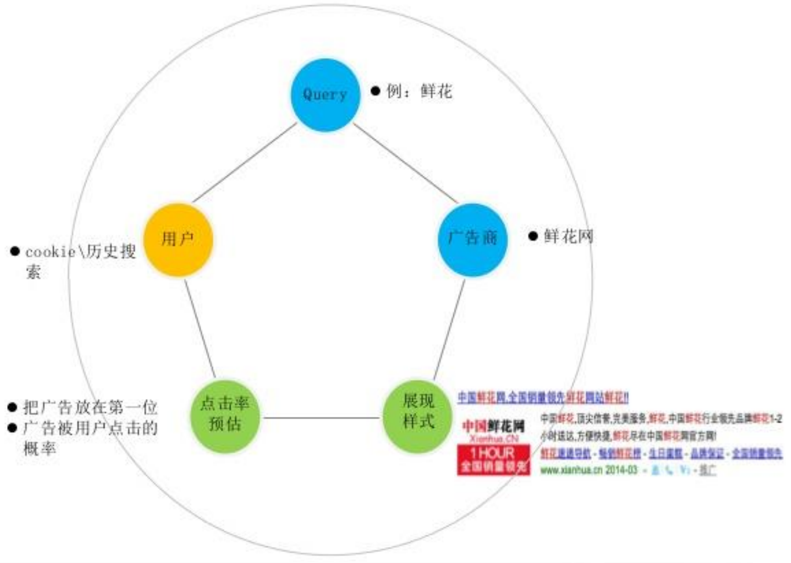
查询

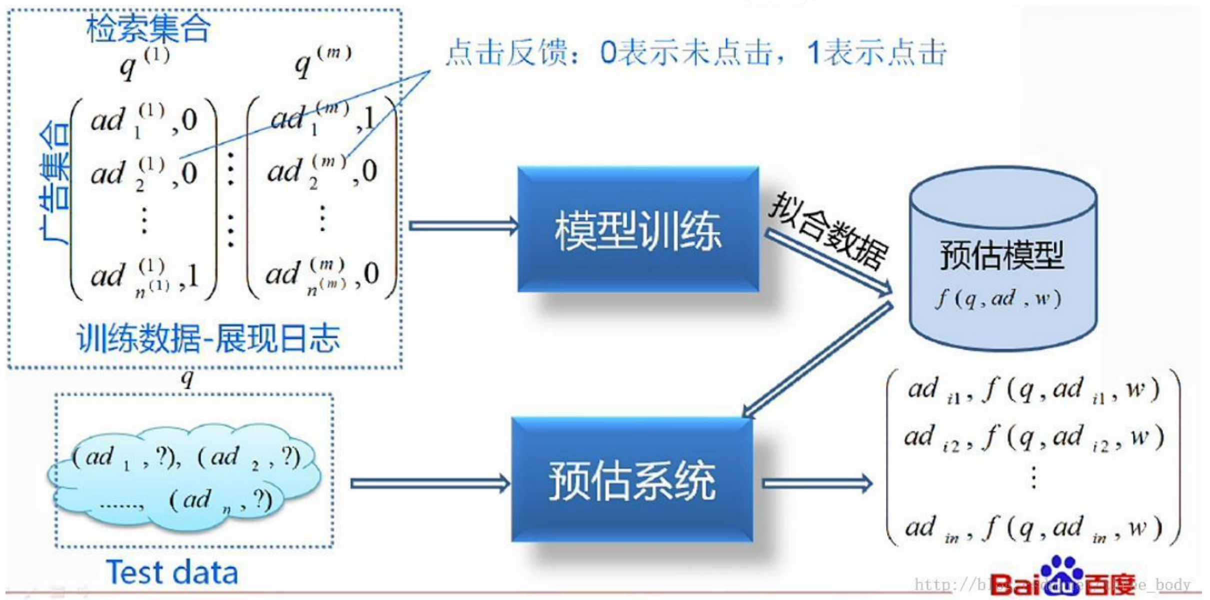
map.baidu.com

杭州_百度百科

核心

点击率（CTR）预估问题





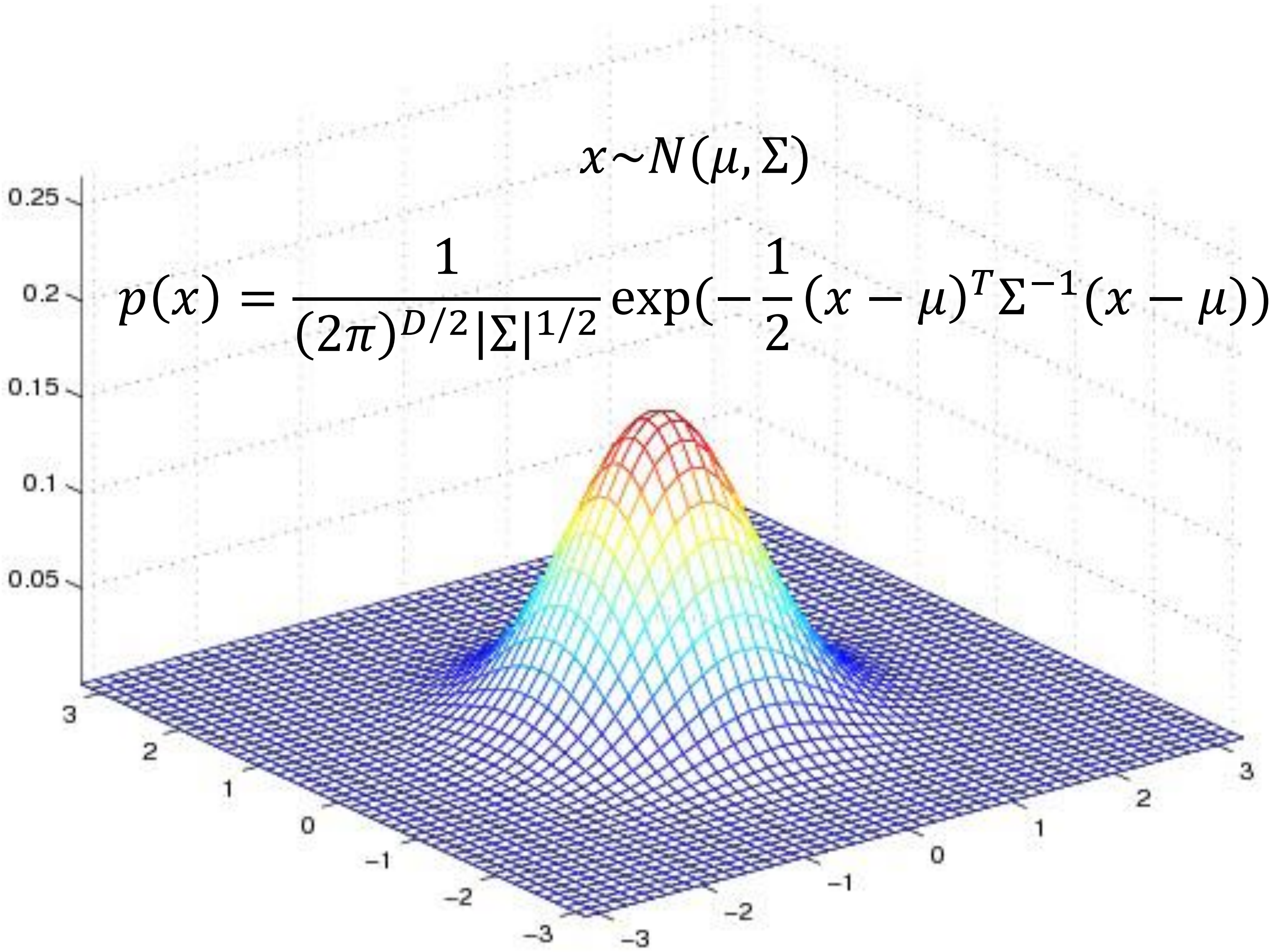
Generative: Gaussian Discriminative Analysis

- Bernoulli: the distribution for a single binary variable $x \in \{0,1\}$ governed by a single continuous parameter $\beta \in [0,1]$ which represents the probability of $x = 1$:
$$P(x|\beta) = \beta^x (1-\beta)^{1-x}$$
- Binomial: gives the probability of observing m occurrences of $x = 1$ in a set of N samples from a Bernoulli distribution:
$$P(m|N,\beta) = \binom{n}{m} \beta^m (1-\beta)^{N-m}$$
- Multinomial: multivariate generalization of binomial, gives the distribution over counts m_k for K-state discrete variable to be in state k:
$$P(m_1, \cdots, m_K|N, \beta) = \binom{N}{m_1, \cdots, m_K} \prod_{k=1}^{K} \beta_k^{m_k}$$

Multivariate Normal Distribution

$$x \sim N(\mu, \Sigma)$$

$$p(x) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$$



Gaussian Discriminative Analysis

- GDA models $p(x|y)$ with a multivariate normal distribution

$$\boxed{y \sim Bernoulli(\beta); x|y=0 \sim N(\mu_0, \Sigma); x|y=1 \sim N(\mu_1, \Sigma)}$$

- Log Likelihood:

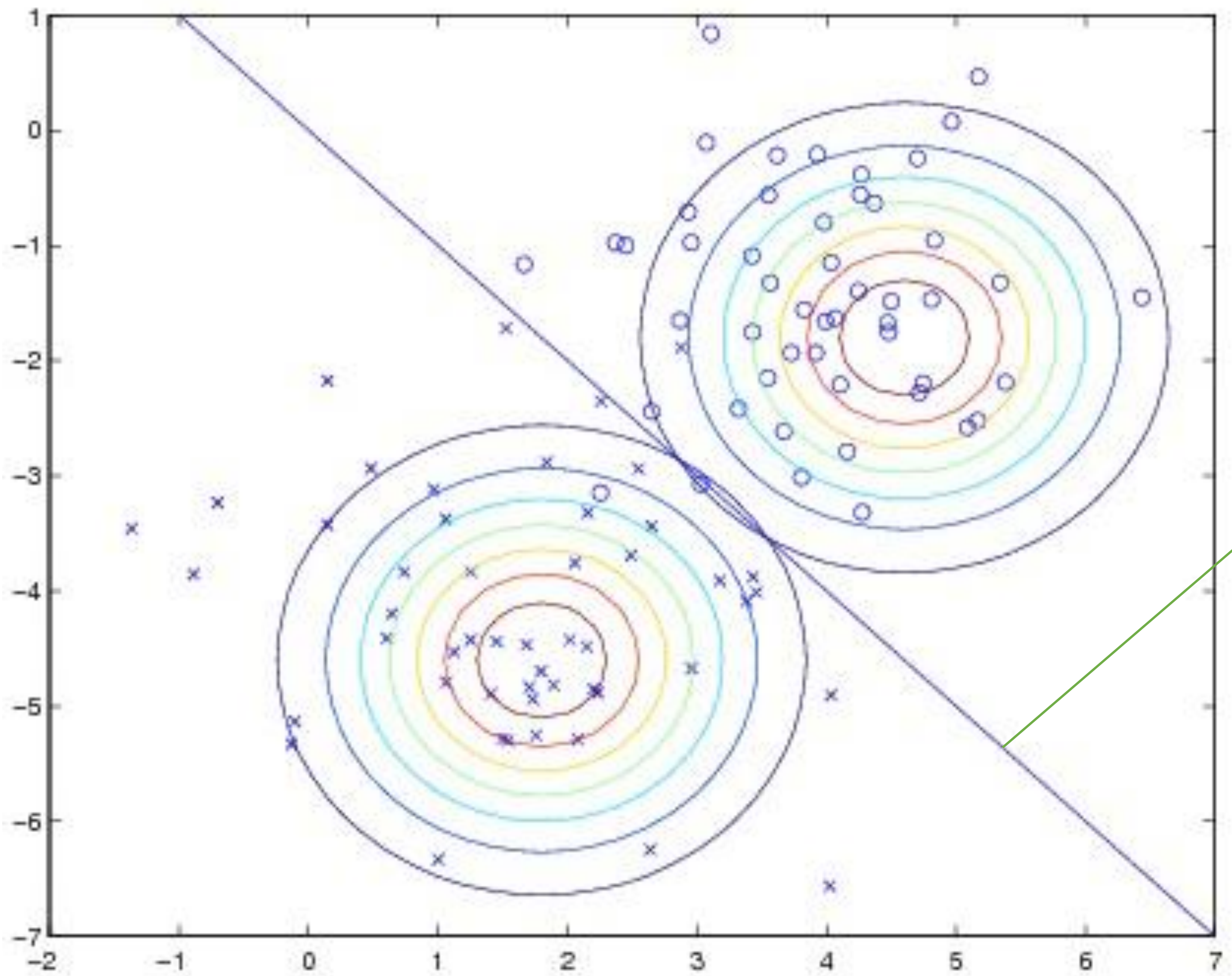
$$L(\beta, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{N} p(x^{(i)}, y^{(i)}; \beta, \mu_0, \mu_1, \Sigma)$$

$$= \log \prod_{i=1}^{N} p\left(y^{(i)}; \beta\right) p(x^{(i)} | y^{(i)}; \beta, \mu_0, \mu_1, \Sigma)$$

- MLE:

$$\boxed{\beta = \frac{1}{N} \sum_{i=1}^{N} I_{\{y^{(i)}=1\}}; \mu_k = \frac{\sum_{i=1}^{N} I_{\{y^{(i)}=k\}} x^{(i)}}{\sum_{i=1}^{N} I_{\{y^{(i)}=k\}}}, k = \{0,1\}; \Sigma = \frac{1}{N} \sum_{i=1}^{N} \left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^T}$$

Illustration of GDA



$P(y = 1|x) = 0.5$

GDA vs. LR

- GDA Makes stronger modeling assumption, is more data efficient if the assumption is correct.
- Logistic Regression makes weaker assumption, and is significantly more robust to deviations from modeling assumption.
- In practice, LR is used more often than GDA.

In GDA, the feature vectors x are continuous, real valued vectors.

How about discrete valued ones?

Generative Model: Naïve Bayes

- Example of Spam Filtering:
  - Classifying emails to spam (y=1) or non-spam (y=0)
  - Feature vector

$$\mathbf{x} = \begin{array}{ll} \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ \vdots \\ 0 \end{array}\right] & \begin{array}{l} \text{a} \\ \text{aardwolf} \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{array} \end{array}$$

    - $x \in (0,1)^D$, D is the size of the set of words (vocabulary)
    - Word selection: include words that appear in the emails but not in a dictionary; exclude very high frequency words; etc.
    - $x_j's$ can take more general values in $\{1,2,\cdots,k_j\}$, then the model $p(x_j|y)$ is multinomial rather than binomial.
    - $2^D - 1$ parameter needed! Cost much since D is usually extremely large!

Generative Model: Naïve Bayes

- To model $p(x|y)$, we assume that $x_j's$ are conditionally independent given y. This is *Naïve Bayes Assumption*.

$$p(x_1, \cdots, x_D|y) = \prod_{i=1}^{D} p(x_i|y)$$

- Given a training data $\left(x^{(i)}, y^{(i)}\right), i = 1, \cdots, N$, the joint log likelihood of the data:

$$L = \sum_{i=1}^{N} \sum_{j=1}^{D} \{\log p\left(x_j^{(i)} \middle| y^{(i)}\right) + \log p(y^{(i)})\}$$

Estimation and Prediction

- MLE:

$$p(x_j = 1 | y = 1) = \frac{\sum_{i=1}^{N} I_{\{x_j^{(i)}=1 \wedge y^{(i)}=1\}}}{\sum_{i=1}^{N} I_{\{y^{(i)}=1\}}}$$

$$p(x_j = 1 | y = 0) = \frac{\sum_{i=1}^{N} I_{\{x_j^{(i)}=1 \wedge y^{(i)}=0\}}}{\sum_{i=1}^{N} I_{\{y^{(i)}=0\}}}$$

$$p(y = 1) = \frac{\sum_{i=1}^{N} I_{\{y^{(i)}=1\}}}{N}$$

Can be generalized to multi-classification problem!

- Prediction

$$p(y = 1|x) = \frac{p(x|y = 1)p(y = 1)}{p(x)}$$

$$= \frac{\prod_{j=1}^{D} p(x_j|y = 1)p(y = 1)}{\prod_{j=1}^{D} p(x_j|y = 1)p(y = 1) + \prod_{j=1}^{D} p(x_j|y = 0)p(y = 0)}$$

Smoothing

- If you haven’t seen some word before in your finite training set, the Naïve Bayes classifier does not know how to make a decision.
- To estimate the mean of multinomial random variable x taking values $\{1, \cdots, k\}$, given N observations $\left\{x^{(1)}, \cdots, x^{(N)}\right\}$, the maximum likelihood estimators are:

$$p(x = j) = \frac{\sum_{i=1}^{N} I_{\{x^{(i)}=j\}}}{N}, j = 1, \cdots, k$$

- Laplace Smoothing:

$$p(x = j) = \frac{\sum_{i=1}^{N} I_{\{x^{(i)}=j\}} + 1}{N + k}, \text{j} = 1, \cdots, k$$

- NB with Laplace Smoothing: $k = 2$

NB vs. LR

- Asymptotic Comparison (training data→infinity)
  - When model assumption is correct
    - NB and LR produce identical classifier
  - When model assumption is not correct
    - LR is less biased, since it does not assume conditional, independence
    - Therefore expected to outperform NB
- Non-Asymptotic Comparison (Ng. and Jordan 2002)
  - Converge rate of parameter estimation: how many training example needed to assume good estimators?
    - NB order $\log D$
    - LR order D
  - NB converges much more quickly to its asymptotic estimators.

Exercise: Naïve Bayes Classifier

- Use the following data to train a NB classifier, and decide the label y of instance $x = (2, S)^T$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
x_2	S	M	M	S	S	S	M	M	L	L	L	M	M	L	L
y	-1	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1

  - In the table, x_1 and x_2 are features valued from $A_1 = \{1,2,3\}$ and $A_2 = \{S, M, L\}$, $y \in \{1, -1\}$.

Homework

- Practice of Logistic Regression:
  - http://cs229.stanford.edu/materials.html
  - Problem Set 1 (pdf) Data: q1x.dat, q1y.dat, q2x.dat, q2y.dat
  - Binary Classification

풍성한 마음이 {秋} 머무르는 계절
이번 풍요로운 추석 고마운 분들과
웃음 풍년 맞이하시기 바랍니다.
더도 덜도 말고 한가위만 같아라.

Q&A

lanyanyan@ict.ac.cn