Report

Cuiyi

December 7, 2014

You can get this ppt from https://github.com/saiichi/LearnReport

Machine Learning: The Original Motivation

Traditional Software Process:

- Interview the experts,
- Create an algorithm that automates their process.

Machine Learning process:

- Collect input-output examples from the experts.
- Learn a function to map from the input to the output.

Supervised Learning

- Given: Training examples$(x_i, f(x_i))$ for some unknown function f.
- Find: A good approximation to f.
- Example Applications
  - Handwriting recognition
    - x data from pen motion
    - $f(x)$ letter of the alphabet
  - Disease Diagnosis
    - x: properties of patient(symptoms, lab tests)
    - $f(x)$: disease(or maybe, recommended therapy)
  - Face recognition
    - x: bitmap picture of person's face
    - $f(x)$: name of person
  - Spam detection
    - x: email message
    - $f(x)$: spam or not spam

Formal Setting

- Training examples are drawn independently at random according to unknown probability distribution $P(x, y)$.
- The learning algorithm analyzes the examples and produces a classifier f.
- Given a new data point (x, y) drawn from P, the classifier is given x and predicts $\hat{y} = f(x)$.
- The loss $\mathcal{L}(\hat{y}, y)$ is then measured.
- Goal of the learning algorithm: Find the f that minimizes the expected loss

Formal Version of Spam Detection

- $P(x, y)$: distribution of email message x and their true labels y("spam" or "not spam").
- training sample: a set of email messages that have been labeled by the user.
- learning algorithm: logistic regression and so on.
- f: the classifier output by the learning algorithm.
- test point: A new email message x.
- loss function $\mathcal{L}(\hat{y}, y)$

Three main approaches to ML

- Learn a classifier: a function f
- Learn a conditional distribution: a conditional distribution $P(y|x)$
- Learn the joint probability distribution: $P(x, y)$
- One example for each method
  - A classifier: The Perceptron algorithm.
  - A conditional distribution: Logistic regression.
  - A joint distribution: Linear discriminant analysis.

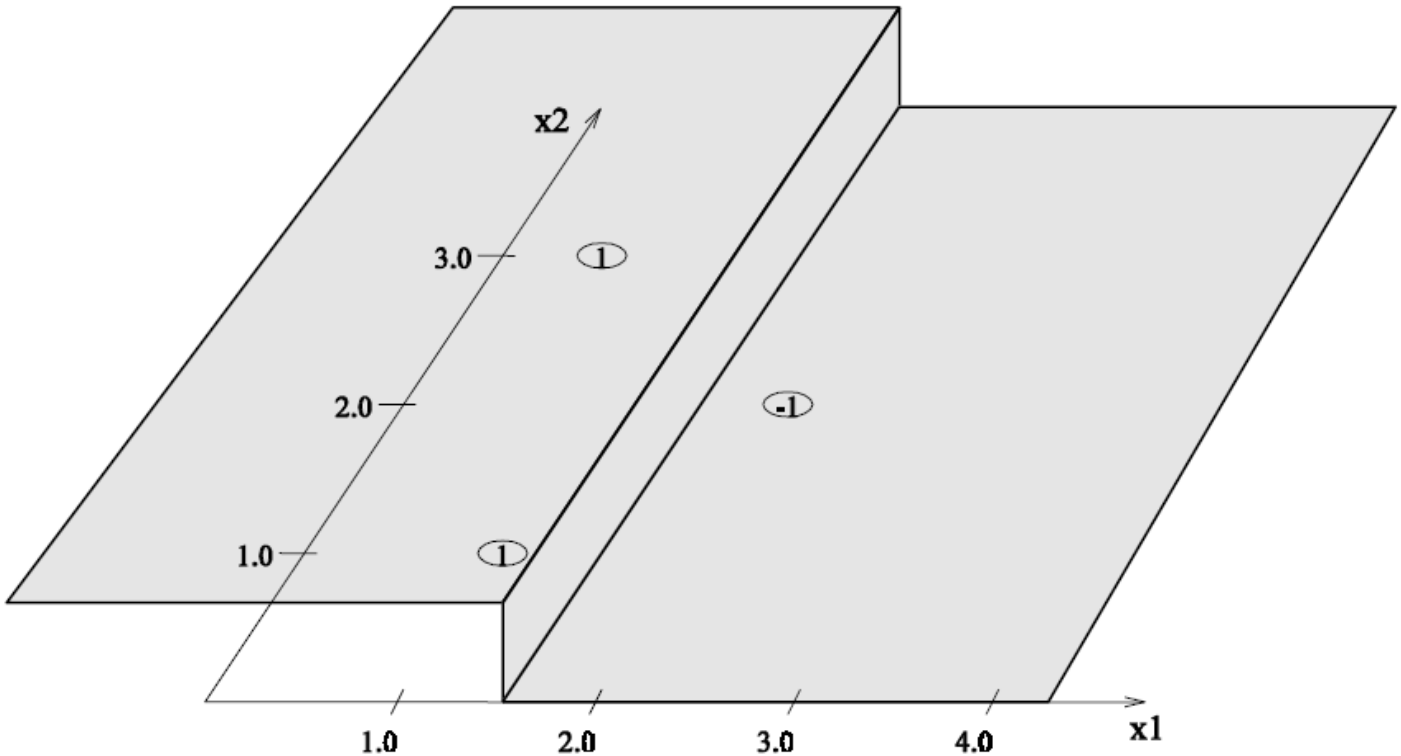
Linear Threshold Units

$$h_w(x) = g_w(w^T x) = \begin{cases} 1 & \text{if } w^T x \geq w_0 \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

- Where $x = (x_1, \ldots, x_n)$, $w = (w_1, \ldots, w_n)$.We assume that each feature x_j and each weight w_j is a real.
- For convenience, we transform $x = (x_1, \ldots, x_n)$ to $x = (-1, x_1, \ldots, x_n)$, then the parameter vector will then be $w = (w_0, w_1, \ldots, w_n)$
- Our goal is to find w

Geometrical View

- Consider three training examples:
  $(< 1.0, 1.0 >, +1), (< 1.5, 3.0 >, +1), (< 2.0, 2.0 >, -1)$
- We want a classifier that looks like the following:

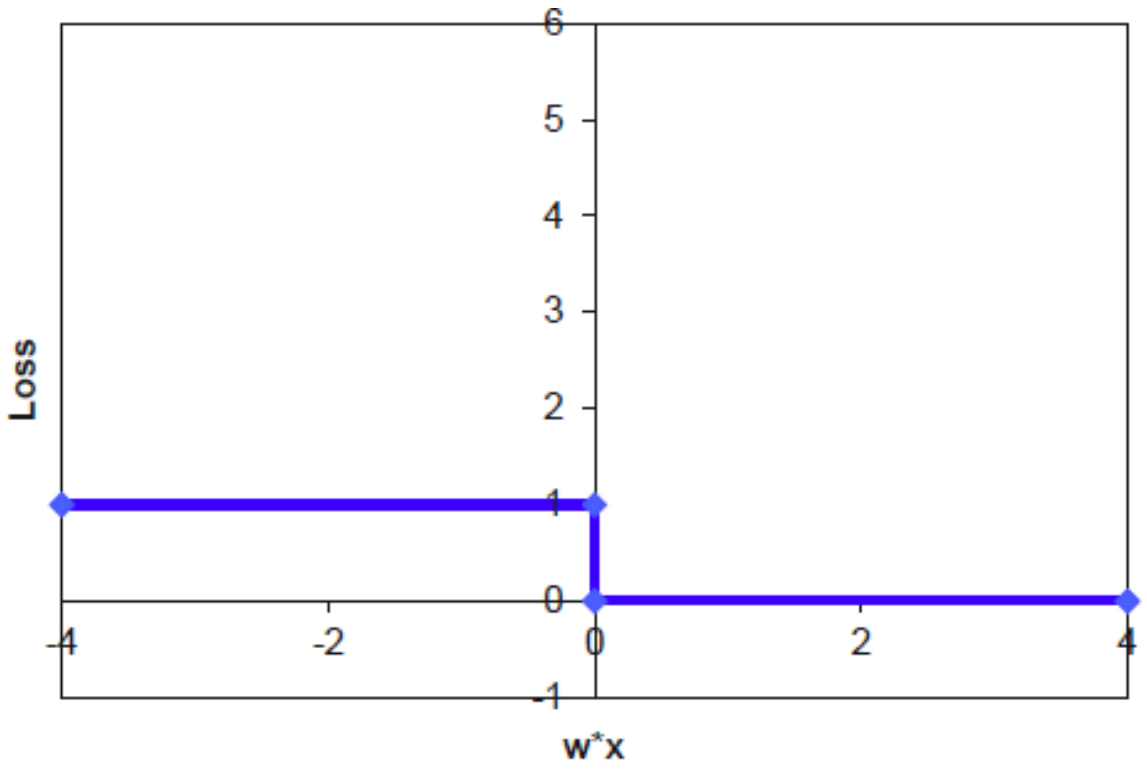


ML == Optimization

- Given:
  -A set of N training examples
  $\{x^{(1)}, \ldots, x^{(N)}\}$
- Find:
  -The weight vector w that minimizes the expected loss on the training data.

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(sgn(w^T x^{(i)}), y_i) \tag{2}$$

Problem: Step-wise Constant Loss Function



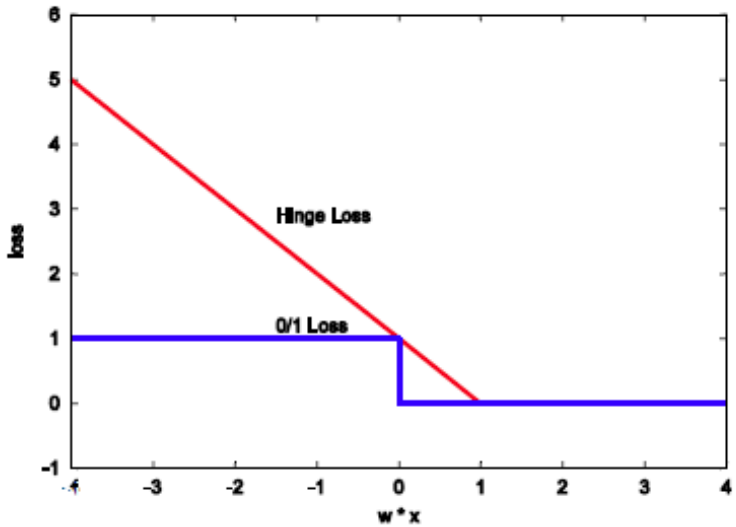
Derivative is either 0 or ∞

Approximating the expected loss by a smooth function

- Simplify the optimization problem by replacing the original objective function by a surrogate loss function. For example, consider the hinge loss:

$$\widetilde{J}(w) = \frac{1}{N} \sum_{N}^{i=1} \max(0, 1 - y_i w^T x^{(i)}) \tag{3}$$

- When $y = 1$



Gradient of the Hinge Loss

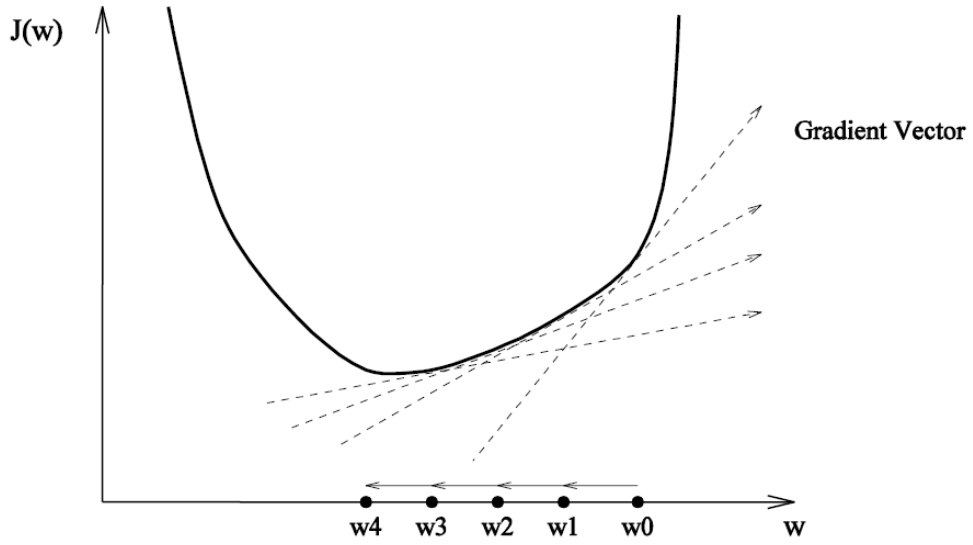
Let $\widetilde{J}_i(w) = \max(0, -y_i w^t x^{(i)})$

$$\frac{\partial \widetilde{J}(w)}{\partial w_k} = \frac{\partial}{\partial w_k}\left(\frac{1}{N}\sum_{i=1}^{N}\widetilde{J}_i(w)\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial w_k}\tilde{J}_i(w)$$

$$\sum_{i=1}^{N}\frac{\partial}{\partial w_k}\tilde{J}_i(w) = \frac{\partial}{\partial w_k}\max\left(0, -y_i\sum_j w_j x_j^{(i)}\right) \quad (4)$$

$$= \begin{cases} 0 & \text{if } w^T x \geq w_0 \\ -y x_k^{(i)} & \text{otherwise} \end{cases} \quad (5)$$

Minimizing $\tilde{J}$ by Gradient Descent Search



- Start with weight vector $w^{(0)}$
- Compute gradient $\nabla\tilde{J}(w^{(0)}) = \left(\frac{\partial\tilde{J}(w^{(0)})}{\partial w_0}, \frac{\partial\tilde{J}(w^{(0)})}{\partial w_1}, \ldots, \frac{\partial\tilde{J}(w^{(0)})}{\partial w_2}\right)$
- Compute $w^{(1)} = w^{(0)} - \eta\nabla\tilde{J}(w^{(0)})$
- Repeat until convergence

Logistic Regression

- Learn the conditional probability $P(y|x)$
- Let $p(y|x;w)$ be our estimate of $P(y|x)$, where w is a vector of adjustable parameters. Assume only two classes $y = 0$ and $y = 1$, and

$$p(y = 1|x; w) = \frac{\exp w^T x}{1 + \exp w^T x}$$

$$p(y = 0|x; w) = 1 - p(y = 1|x; w)$$

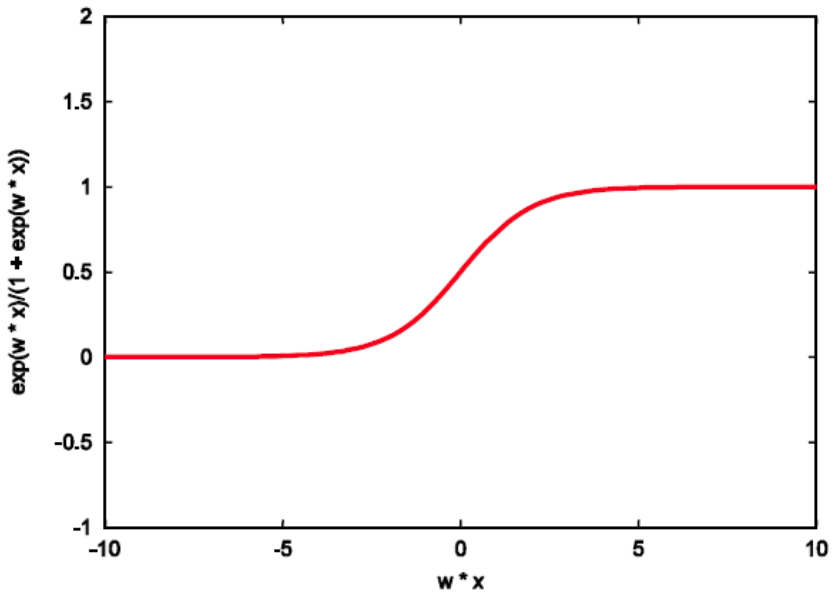
- It is easy to show that this is equivalent to

$$\log \frac{p(y = 1|x; w)}{p(y = 0|x; w)} = w^T x$$

- In other words, the log odds of class 1 is a linear function of x

Why the exp function

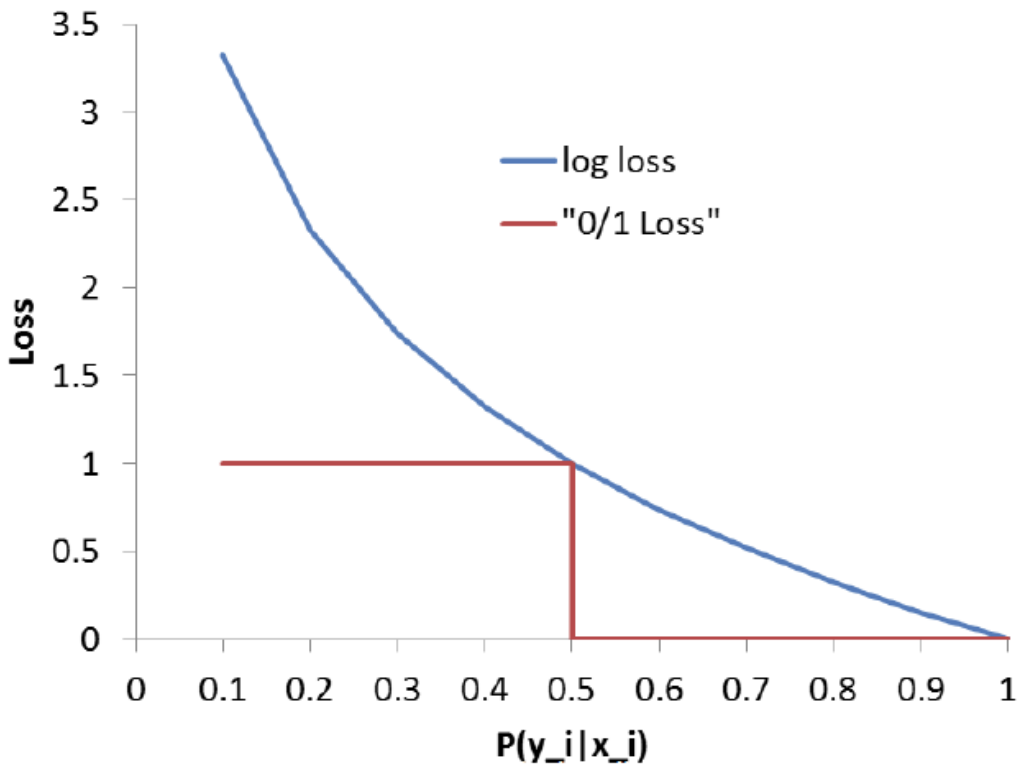
One reason: linear function has a range from $-\infty$ to $+\infty$ and we need to force it to be positive and sum to 1 in order to be a probability:



Choosing the Loss Function

For probabilistic models, we use the log loss:

$$\mathcal{L}(\hat{P}(y|x), y) = \begin{cases} -\log \hat{P}(y = 1|x^{(i)}) & \text{if } y_i = 1 \\ -\log \hat{P}(y + 0|x^{(i)}) & \text{if } y_i = 0 \end{cases}$$



Discriminate Method Review

1. Logistic Regression
   Assume $y = 0$ or $y = 1$,and $h_w(x) = \frac{1}{1-e^{w^T x}}$
2. Perceptron Learning Algorithm
   $h_w(x)$ is a threshold function:

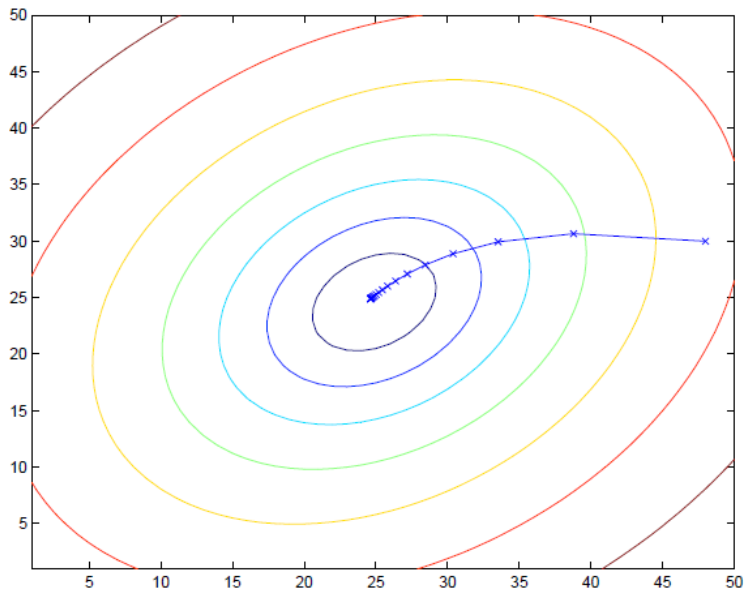
$$h_w(x) = g_w(w^T x) = \begin{cases} 1 & \text{if } w^T x \geq w_0 \\ -1 & \text{otherwise} \end{cases}$$

Gradient descent Review

Batch Gradient descent

Repeat for each parameter :

$$w_i := w_i + \alpha \frac{\partial}{\partial w_i} Loss(h_w(x), y)$$



Gradient descent Review(2)

The normal equations:
For a function $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ define the derivative of f with respect to A to be:

$$\nabla_A f(A) = \begin{pmatrix} \frac{\partial}{\partial A_{11}} & \cdots & \frac{\partial}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial A_{m1}} & \cdots & \frac{\partial}{\partial A_{mn}} \end{pmatrix}$$

For examples, suppose $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ is a 2-by-2 matrix,and:

$$f(A) = \frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}$$

Then we have:

$$\nabla_A f(A) = \begin{pmatrix} \frac{3}{2} & 10A_{12} \\ A_{22} & A_{21} \end{pmatrix}$$

Define of **trace** operator. For an n-by-n matrix A:

$$trA = \sum_{i=1}^{n} A_{ii}$$

Gradient descent Review(3)

For two matrices A and B,such that ABC is square,we have:

$$trAB = trBA$$

As corollaries:

$$trABC = trCAB = trBCA$$

Here A and B is square matrices and a is real number:

$$\begin{aligned} trA &= trA^T \\ tr(A+B) &= trA + trB \\ traA &= atrA \end{aligned}$$

Gradient descent Review(4)

Some facts without proof:

$$\begin{aligned} \nabla_A trAB &= B^T \\ \nabla_{A^T} f(A) &= (\nabla_A f(A))^T \\ \nabla_A trABA^TC &= CAB + C^TAB^T \end{aligned}$$

Revisit Least Squares:
define **design matrix**X to be the m-by-n matrix:

$$X = \begin{pmatrix} x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \dots & x_n^{(m)} \end{pmatrix}$$

Gradient descent Review(5)

define $\vec{y}$ be m-dimensional vector:

$$\vec{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

Since $h_w(x^{(i)}) = (x^{(i)})^T w$:

$$\begin{aligned} Xw - \vec{y} &= \begin{pmatrix} (x^{(1)})^T w \\ \vdots \\ (x^{(m)})^T w \end{pmatrix} - \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix} \\ &= \begin{pmatrix} h_w(x^{(1)}) - y^{(1)} \\ \vdots \\ h_w(x^{(m)}) - y^{(m)} \end{pmatrix} \end{aligned}$$

Gradient descent Review(5)

Using the face that for $\vec{z}$,we have $\vec{z}^T\vec{z} = \Sigma_i \vec{z}_i^2$:

$$\begin{aligned}\frac{1}{2}(Xw-\vec{y})^T(Xw-\vec{y}) &= \frac{1}{2}\sum_{i=1}^{m}(h_w(x^{(i)})-y^{(i)})^2 \\ &= J(w)\end{aligned}$$

$$\begin{aligned}\nabla_w J(w) &= \nabla_w \frac{1}{2}(Xw-\vec{y})^T(Xw-\vec{y}) \\ &= \frac{1}{2}\nabla_w tr(w^TX^TXw - w^TX^T\vec{y} - \vec{y}^TXw + \vec{y}^T\vec{y}) \\ &= \frac{1}{2}\nabla_w(trw^TX^TXw - 2tr\vec{y}^TXw) \\ &= \frac{1}{2}(X^TXw + X^TXw - 2X^Tw) \\ &= X^TXw - X^T\vec{y}\end{aligned}$$

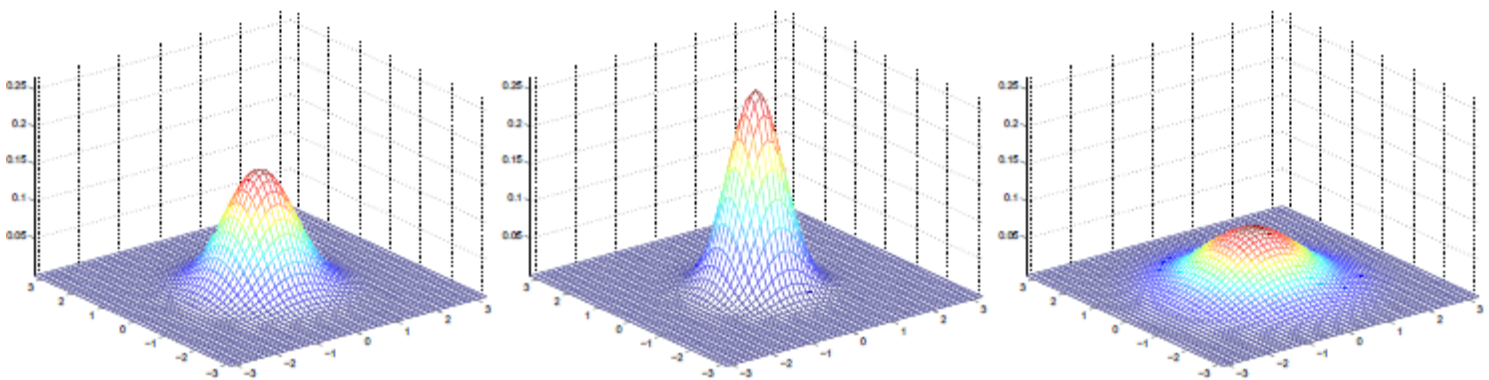
What are Generative Learning algorithms

- Consider a classification problem: distinguish between benign tumors and malignant tumors.
- Given a training set, an algorithm like logistic regression or the perceptron algorithm (basically) tries to find a straight line that is, a decision boundarythat separates the benign tumor and malignant tumor. Algorithms that try to learn $P(y|x)$ directly are called **discriminative learning algorithms**.
- Different approach: Fist we can build a model of what benign tumors look like. Then we build a seperate model of what malignant tumors look like. Finally, to classify a new sample,we can match the new sample with each model,to see whether one model matches better than one other model. Algorithms that try to learn $P(x|y)$(and $P(y)$) are called **generative learning algorithms**.

The multivariate normal distribution

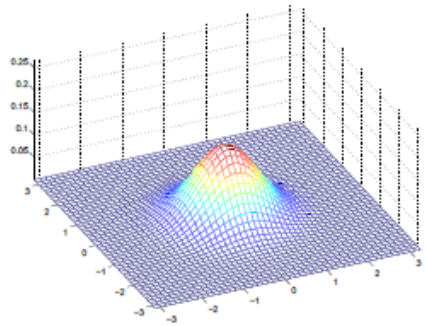
Suppose these is a multivariate normal distribution is parameterized by a **mean vector** $\mu \in \mathbb{R}^n$ and a **covariance matrix** $\Sigma \in \mathbb{R}^{n \times n}$,its density is given by:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu) \right)$$

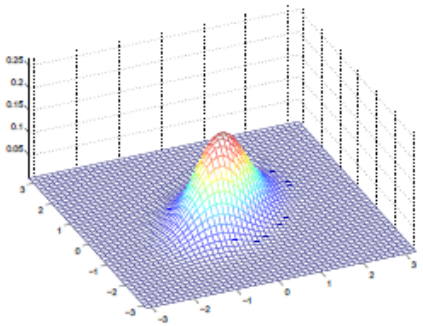


$\Sigma = I$ $\Sigma = 0.6I$; $\Sigma = 2I$.

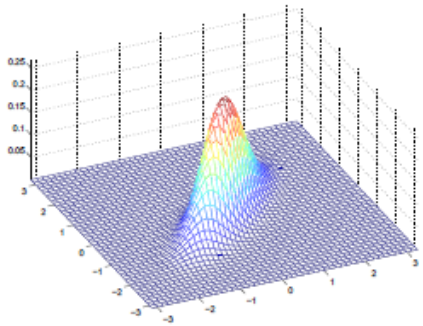
The multivariate normal distribution(2)



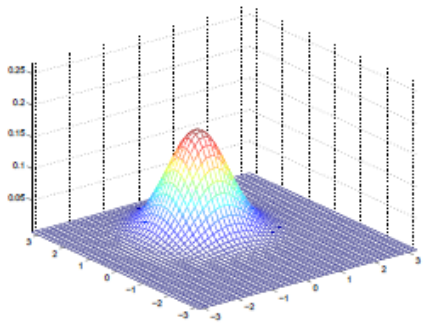
$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



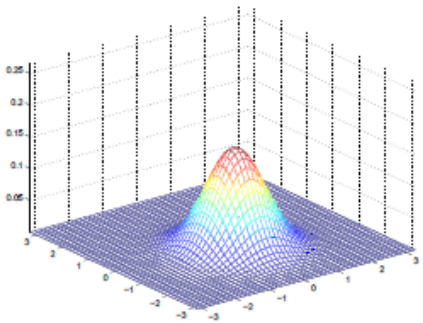
$$\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$



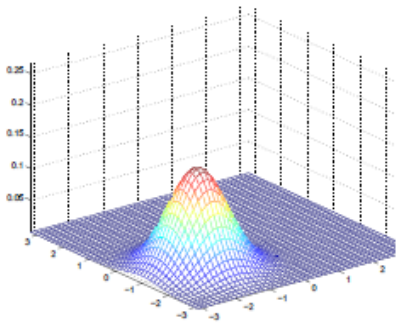
$$\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



$$\mu = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix}$$



$$\mu = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix}$$

The Gaussian Discriminant Analysis model(1)

GDA Model is:

$$\begin{aligned} y &\sim Bernoulli(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

Writing out the distribution:

$$p(y) = \phi^y(1-\phi)^y$$

$$p(x|y=0) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right)$$

$$p(x|y=1) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)$$

The Gaussian Discriminant Analysis model(2)

The log-likelihood is:

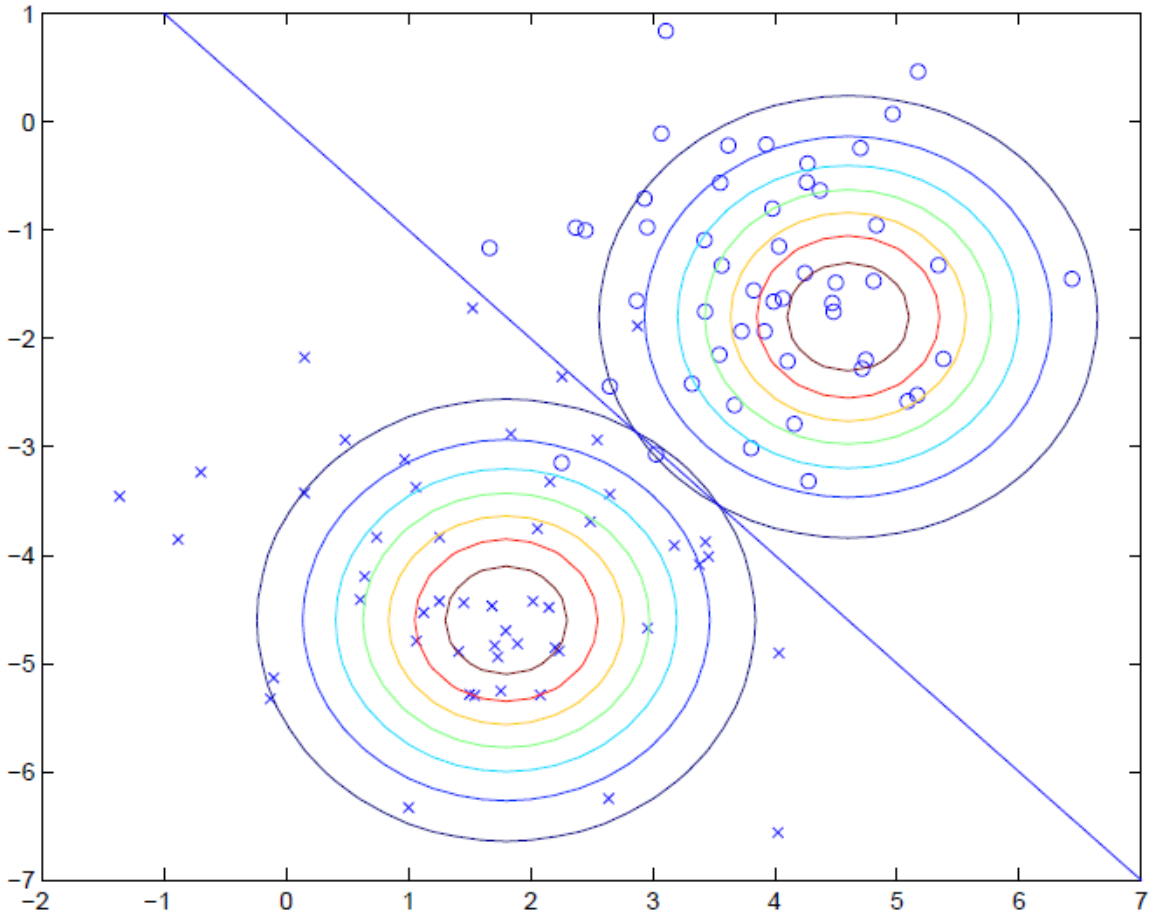
$$\begin{aligned} l(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \log \prod_{i=1}^{m} p(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi) \end{aligned}$$

By maximizing l with respect to the parameters,we find the maximum likelihood estimate of parameters to be:

$$\begin{aligned} \phi &= \frac{1}{m} \sum_{i:y^{(i)}=1}^{m} y^{(i)} \\ \mu_0 &= \frac{\sum_{i:y^{(i)}=0}^{m} x^{(i)}}{\sum_{i:y^{(i)}=0}^{m} y^{(i)}} \\ \mu_1 &= \frac{\sum_{i:y^{(i)}=1}^{m} x^{(i)}}{\sum_{i:y^{(i)}=1}^{m} y^{(i)}} \\ \Sigma &= \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T \end{aligned}$$

The Gaussian Discriminant Analysis model(3)

Pictorially, what the algorithm is doing can be seen in as follows:



Discussion: GDA and logistic regression

If we view $p(y = 1|x; \phi, \mu_0, \mu_1, \Sigma)$ as a function of x, there is a fact that:

$$p(y = 1|x; \phi, \Sigma, \mu_0, \mu_1) = \frac{1}{1 + \exp(-w^T x)}$$

This is exactly the form that logistic regression used to model $p(y = 1|x)$.
In fact in binary division problem:

$$p(x|y = 1) \sim ExpFamily(x, \eta) \Rightarrow p(y = 1|x) = Sigmoid$$

When would we prefer one model over another?

- GDA makes stronger modelling 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.

Bayes' formula

Suppose that we have the prior probability $P(\omega_j)$ and the conditional densities $p(x|\omega_j)$, the posterior probability can be written as:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)},$$

where

$$p(x) = \sum_{j=1}^{c} p(x|\omega_j)P(\omega_j).$$

Bayes' formula can be expressed informally in English:

$$posterior = \frac{likelihood \times prior}{evidence}.$$

Bayesian Decision Theory Continuous Features

Let $\omega_1, \ldots, \omega_c$ be the finite set of c states of nature ("categories") and $\alpha_1, \ldots, \alpha_a$ be the finite set of a possible action. The Loss function $\lambda(\alpha_i|\omega_j)$ describe the loss incurred for taking action α_i when the state of nature is ω_j. The posterior probability $P(\omega_j|x)$ can be compute from $P(\omega_j|x)$ by Bayes'formula:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)}$$

where the evidence is now

$$p(x) = \sum_{j=1}^{c} p(x|\omega_j)P(\omega_j)$$

Bayesian Decision Theory Continuous Features(2)

Suppose that we observe a particular x and that we contemplate taking action α_i.

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)P(\omega_j|x)$$

In decision-theoretic terminology, an expected loss is called a risk, and $R(\alpha_i|x)$ is called the **conditional risk**. If we define loss function to be

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad i, j = 1, \ldots, c.$$

Condition risk will be

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)P(\omega_j|x) = 1 - P(\omega_i|x)$$

Classifiers, Discriminant Functions and Decision Surfaces

For 0-1 Loss function case, we can simplely let the discriminant functions to be $g_i(x) = P(\omega_i|x)$. More generally, if we replace every $g_i(x)$ by $f(g_i(x))$, where $f(\cdot)$ is a monotonically increasing fuction, the resulting classification is unchanged.

$$g_i(x) = P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(x|\omega_j)P(\omega_j)}$$

$$g_i(x) = p(x|\omega_i)P(\omega_i)$$

$$g_i(x) = \log p(x|\omega_i) + \log P(\omega_i)$$

Discriminant Functions for the Normal Density

if $p(x|\omega_i) \sim \mathcal{N}(\mu_i, \Sigma_i)$,

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) - \frac{d}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_i| + \log P(\omega_i)$$

Case: $\Sigma_i = \sigma^2 I$
Ignore terms which are independent of i, we obtain:

$$\begin{aligned} g_i(x) &= -\frac{\|x-\mu_i\|^2}{2\sigma^2} + \log P(\omega_i) \\ &= -\frac{1}{2\sigma^2}(x^Tx - 2\mu_i^Tx + \mu_i^T\mu_i) + \log P(\omega_i) \end{aligned}$$

Since x^Tx is the same for all i's, we obtain

$$g_i(x) = w_i^Tx + w_{i0}$$

where

$$w_i = \frac{1}{\sigma^2}\mu_i, \quad w_{i0} = -\frac{1}{2\sigma^2}\mu_i^T\mu_i + \log P(\omega_i)$$

Discriminant Functions for the Normal Density(2)

For two class, we can set $g_i(x) = g_j(x)$:

$$w^T(x - x_0) = 0,$$

where,

$$w = \mu_i - \mu_j$$

and,

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|} \log \frac{P(\omega_i)}{P(\omega_j)}(\mu_i - \mu_j)$$

Discriminant Functions for the Normal Density(3)

Case $\Sigma_i = \Sigma$:

$$g_i(x) = w^T x + w_0$$

where

$$w_i = \Sigma^{-1}\mu_i, \quad w_{i0} = -\frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i + \log P(\omega_i)$$

Case $\Sigma_i = ?$

$$g_i = x^T W_i x + w_i^T x + w_{i0}$$

where,

$$W_i = -\frac{1}{2}\Sigma_i^{-1}, \quad w_i = \Sigma_i^{-1}\mu_i$$

and,

$$w_{i0} = -\frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i - \frac{1}{2}\log|\Sigma_i| + \log P(\omega_i)$$

Bayesian estimation

In Bayesian estimation, we consider parameter θ to be a random variable. Any information we might have about θ prior to observing the samples is assumed to be contained in a *known* prior density $p(\theta)$. Observation of the samples converts this to a posterior density $p(\theta|\mathcal{D})$
Knowing that our goal is to compute $p(x|\mathcal{D})$, we do this by:

$$p(x|\mathcal{D}) = \int p(x, \theta|\mathcal{D}) d\theta$$

then:

$$p(x|\mathcal{D}) = \int p(x|\theta) p(\theta|\mathcal{D}) d\theta$$

Bayesian estimation(2)

Suppose μ is the only unknown parameter, and

$$p(x|\mu) \sim \mathcal{N}(\mu, \sigma^2).$$

Our prior knowledge we might have about μ can be expressed by a known prior density $p(\mu)$.

$$p(\mu) \sim \mathcal{N}(\mu_0, \sigma_0^2)$$

To obtain the posterior probability:

$$\begin{aligned} p(\mu|\mathcal{D}) &= \frac{p(\mathcal{D}|\mu)p(\mu)}{\int p(\mathcal{D}|\mu)p(\mu)} \\ &= \alpha \prod_{k=1}^{n} p(x_k|\mu)p(\mu) \end{aligned}$$

Bayesian estimation(3)

Since $p(x_k|\mu) \sim \mathcal{N}(\mu, \sigma^2)$ and $p(\mu) \sim \mathcal{N}(\mu_0, \sigma_0^2)$:

$$\begin{aligned} p(\mu|\mathcal{D}) &= \alpha \prod_{k=1}^{n} \overbrace{\frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x_k-\mu}{\sigma}\right)^2\right]}^{p(x_k|\mu)} \overbrace{\frac{1}{\sqrt{2\pi}\sigma_0} \exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right]}^{p(\mu)} \\ &= \alpha' \exp\left[-\frac{1}{2}\left(\sum_{k=1}^{n}\left(\frac{\mu-x_k}{\sigma}\right)^2 + \left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right)\right] \\ &= \alpha'' \exp\left[-\frac{1}{2}\left[\left(\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2}\right)\mu^2 - 2\left(\frac{1}{\sigma^2}\sum_{k=1}^{n} x_k + \frac{\mu_0}{\sigma_0^2}\right)\mu\right]\right], \qquad (\end{aligned}$$

Now we know $p(\mu|\mathcal{D})$ is a Gaussian distribution, If we write $p(\mu|\mathcal{D} \sim \mathcal{N}(\mu_n \sigma_n^2))$,we obtain:

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\bar{x_n} + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

$$\sigma_n^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}$$

Now we need to compute the $p(x|\mathcal{D})$:

$$\begin{aligned} p(x|\mathcal{D}) &= \int p(x|\mu)p(\mu|\mathcal{D})\, d\mu \\ &= \int \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_n}{\sigma_n}\right)^2\right] d\mu \\ &= \frac{1}{2\pi\sigma\sigma_n} \exp\left[-\frac{1}{2}\frac{(x-\mu_n)^2}{\sigma^2+\sigma_n^2}\right] f(\sigma,\sigma_n), \end{aligned}$$

Hence,

$$p(x|\mathcal{D}) \sim \mathcal{N}(\mu_n, \sigma^2 + \sigma_n^2)$$

Margin(1)

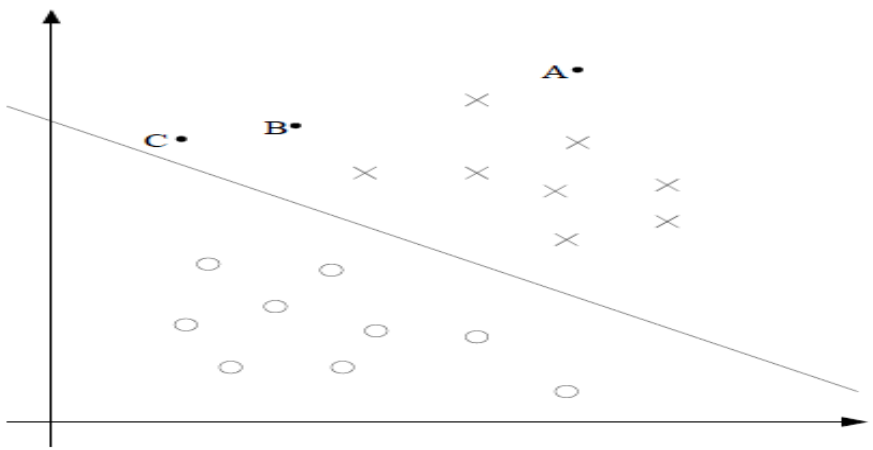
Previous:

Consider logistic regression,we compute $w^T x$ then we:

- Predict "1" iff $w^T x \geq 0$.
- Predict "0" iff $w^T x < 0$.

The larger $w^T x$ is,the larger also is $h_w(x) = p(y|x; w)$:

- if $w^T x \gg 0$ then we are very "confident" predicting "1".
- if $w^T x \ll 0$ then we are very "confident" predicting "0".



Margin(2)

To make our discussion of SVMs easier,we need to change our notation:
we use $y \in 1, -1$ to denote class label, and change classifier as:

$$h_{w,b}(x) = g(w^T x + b)$$

Here g is a threshold function.
Given a training example $(x^{(i)}, y^{(i)})$,we define **functional margin** of (w, b) with respect to the training example.

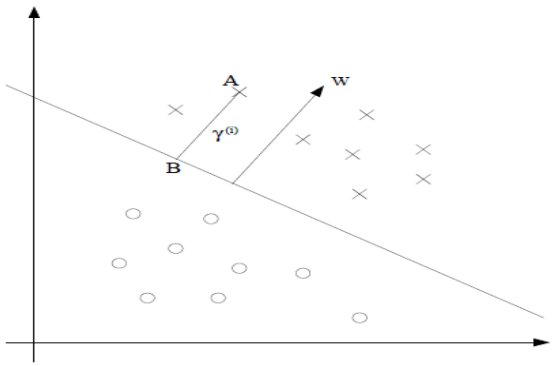
$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$$

A large functional margin represents a confident and a correct prediction.But by exploiting our freedom to scale w and b, we can make the functional margin arbitrarily large without really changing anything meaningful.
Given a training set S,we also define the function margin of (w, b) with respect to S:

$$\hat{\gamma} = \min_{i=1,\ldots,m} \hat{\gamma}^{(i)}$$

Margin(3)

Lets talk about **geometric margins**. Consider the picture below:



Suppose A is $x^{(i)}$,then B is $x^{(i)} - \gamma^{(i)} \cdot (w/\|w\|)$,B lies on the decision boundary,Hence:

$$w^T\left(x^{(i)} - \gamma^{(i)} \cdot \frac{w}{\|w\|}\right) + b = 0$$

Solving for $\gamma^{(i)}$ yields:

$$\gamma^{(i)} = \left(\frac{w}{\|w\|}\right)^T x^{(i)} + \frac{b}{\|w\|}$$

Margin(4)

More generally,define geometric margins of (w, b) with respect to a training example $(x^{(i)}, y^{(i)})$ to be

$$\gamma^{(i)} = y^{(i)} \left(\left(\frac{w}{\|w\|} \right)^T x^{(i)} + \frac{b}{\|w\|} \right)$$

Given a training set S, we also define the geometric margin of (w, b) with respect to S to be the smallest of the geometric margins on the individual training examples:

$$\gamma = \min_{i=1,\ldots,m} \gamma^{(i)}$$

The optimal margin classifier

Given a training set,We try to find a decision boundary that maximizes the margin. We can pose the following optimization problem

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq \gamma, i = 1, \ldots, m \\ & \|w\| = 1. \end{aligned}$$

The "$\|w\|$" constraint is a nasty(non-convex) constraint. Transform to a nicer one:

$$\begin{aligned} \max_{\gamma,w,b} \quad & \frac{\hat{\gamma}}{\|w\|} \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma}, i = 1, \ldots, m \end{aligned}$$

We get rid of "$\|w\| = 1$" which we don't like. But the downside is that we now have a nasty(non-convex) objective function.

The optimal margin classifier(2)

Since we can add an arbitrary scaling constraint on w and b without changing anything.We introduce a scaling constraint:

$$\hat{\gamma} = 1$$

Plugging this into our problem above,and noting that maximizing $\hat{\gamma}/\|w\| = 1/\|w\|$ is the same thing as minimizing$\|w\|^2$, we now have the following optimization problem:

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}\|w\|^2 \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, \ldots, m \end{aligned}$$

Now,we have an optimization problem with a convex quadratic objective and only linear constraints.This optimization problem can be solved using commercial quadratic programming (QP) code.

Lagrange duality

Consider problem of the following form:

$$\begin{aligned} \min_{w} \quad & f(w) \\ s.t. \quad & h_i(w) = 0, i = 1, \ldots, l. \end{aligned}$$

Define **Lagrangian** to be

$$\mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

Here,the β_i's are called the **Lagrange multipliers**. Set $\mathcal{L}$'s partial derivatives to zero:

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0; \frac{\partial \mathcal{L}}{\partial \beta_i} = 0$$

We can solve for w and β

Lagrange duality(2)

Consider the following, which we call the **primal** optimization problem:

$$\begin{aligned} \min_{w} \quad & f(w) \\ s.t. \quad & g_i(w) \leq 0, i = 1, \ldots, k \\ & h_i(w) = 0, i = 1, \ldots, l \end{aligned}$$

Define **generalized Lagrangian**

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w).$$

Here the α_i's and β_i's are the Lagrange multipliers.Consider the quantity:

$$\theta_{\mathcal{P}} = \max_{\alpha, \beta : \alpha_i \geq 0} f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

Here,the "$\mathcal{P}$" subscript stands for "primal".

Lagrange duality(3)

Let some w be given. If w violates any of the primal constraints,then it's easy to verify that $\theta_{\mathcal{P}} = \infty$. Conversely, if the constraints are indeed satisfied for a particular value of w,then $\theta_{\mathcal{P}} = f(w)$.Hence,

$$\theta_{\mathcal{P}}(w) = \begin{cases} f(w) & \text{if } w \text{ satisfied primal conatraints} \\ \infty & \text{otherwise} \end{cases}$$

If we consider the minimization problem

$$\min_w \theta_{\mathcal{P}}(w) = \min_w \max_{\alpha,\beta,\alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

It is the same problem of primal problem.Define optimal value of the objective to be $p^* = \min_w \theta_{\mathcal{P}}(w)$

Lagrange duality(4)

Define:

$$\theta_{\mathcal{D}}(\alpha, \beta) = \min_{w} \mathcal{L}(w, \alpha, \beta).$$

Here,the "$\mathcal{D}$" subscript stands for "dual".
Now the **dual** optimization problem:

$$\max_{\alpha,\beta:\alpha_i \geq 0} \theta_{\mathcal{D}}(\alpha, \beta) = \max_{\alpha,\beta:\alpha_i \geq 0} \min_{w} \mathcal{L}(w, \alpha, \beta)$$

define the optimal value of the dual problems objective to be
$d^* = \max_{\alpha,\beta:\alpha_i \geq 0} \theta_{\mathcal{D}}(\alpha, \beta)$.
It can easily be shown that

$$d^* = \max_{\alpha,\beta:\alpha_i \geq 0} \min_{w} \mathcal{L}(w, \alpha, \beta) \leq \min_{w} \max_{\alpha,\beta:\alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = p^*$$

However under certain conditions,we will have $d^* = p^*$

Lagrange duality(5)

Let's see what these conditions are:

- f and the g_i's are convex;
- h_i's are affine;
- g_i's are (strictly) feasible; this means that there exits some w so that $g_i(w) < 0$ for all i.

w^*,α^* and β^* satisfy the **Karush-Kuhn-Tucker (KKT) conditions**, which are as follows:

$$\begin{aligned}
\frac{\partial}{\partial w_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, i = 1, \ldots, n \\
\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, i = 1, \ldots, l \\
\alpha_i^* g_i(w^*) &= 0, i = 1, \ldots, k \\
g_i(w^*) &\leq 0, i = 1, \ldots, k \\
\alpha^* &\geq 0, i = 1, \ldots, k
\end{aligned}$$

The optimal margin classifier(3)

we posed the following (primal) optimization problem for finding the optimal margin classifier:

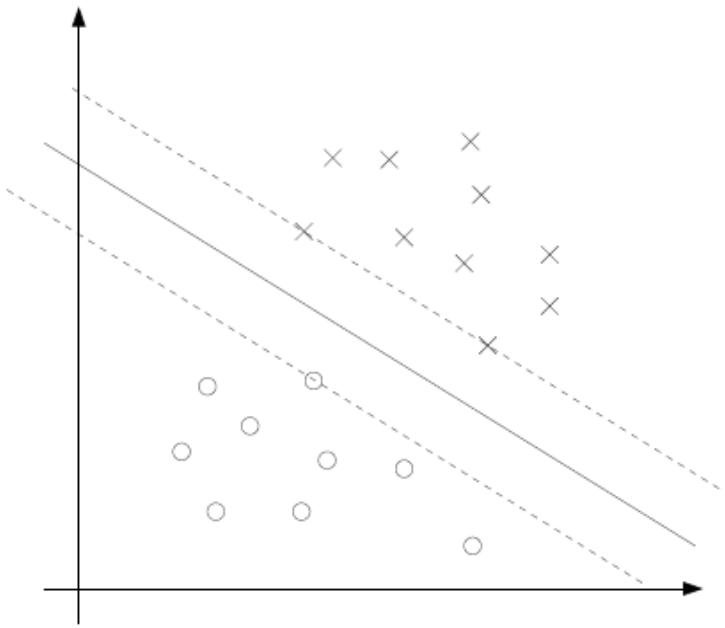
$$\begin{aligned} \min_{\gamma,w,\beta} \quad & \frac{1}{2}\|w\|^2 \\ s.t \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, \ldots, m \end{aligned}$$

We can write the constraints as:

$$g_i(x) = 1 - y^{(i)}(w^T x^{(i)} + 1) \leq 0, i = 1, \ldots, m$$

Note that from the KKT condition,we will have $\alpha_i > 0$ only for that training example that have function margin exactly equal to one

The optimal margin classifier(4)



The points with the smallest margins are exactly the ones closest to the decision boundary;here, these are the three points which are called the **support vectors** in this problem.

The optimal margin classifier(5)

Construct Lagrangian for our problem:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m}[y^{(i)}(w^T x^{(i)} + b) - 1]$$

Note that there are only α_i but no β_i Lagrange multipliers. To minimize $\mathcal{L}(w, b, \alpha)$ for fix α:

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \stackrel{\text{set}}{=} 0$$

This implies that:

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \tag{6}$$

The optimal margin classifier(6)

As for the derivation with respect to b,we obtain:

$$\frac{\partial}{\partial b}\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m}\alpha_i y^{(i)} = 0 \tag{7}$$

Plugging w back into the Lagrangian:

$$\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle$$

We obtain the following dual problem:

$$\begin{aligned}
\max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle \\
s.t. \quad & \alpha_i \geq 0, \quad i = 1,\ldots,m \\
& \sum_{i=1}^{m}\alpha_i y^{(i)} = 0
\end{aligned}$$

The optimal margin classifier(7)

Having found w^*,then we can get b^* as:

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T} + \min_{i:y^{(i)}=1} w^{*T} x^{(i)}}{2}$$

Suppose we've fit our model's parameters to a training set ,and now wish to make a prediction at a new point x.We would calculate $w^T x + b$,and do prediction.But using (1), this quantity can also be written:

$$w^T x + b = \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b$$

Using (2) we saw that the α_i's will all be zero except for the support vector.

Kernels

Define **Kernel** to be:

$$K(x, z) = \phi(x)^T \phi(z)$$

we let ϕ denote the **feature mapping**, which maps from the original input value (**attributes**) to the features in higher dimensional.
Let's see an example. Suppose $x, z \in \mathbb{R}^n$, and consider:

$$K(x, z) = (x^T z)^2 = \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) = \phi(x)^T \phi(z)$$

Thus, we can see feature mapping ϕ is given by (shown here for the case of $n = 3$)

$$\phi(x) = (x_1 x_1, x_1 x_2, \ldots, x_3 x_3)^T$$

Kernels(2)

Another example:

$$\begin{aligned} K(x,z) &= (x^T z + c)^2 \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) + \sum_{i=1}^{n} (\sqrt{2c}x_i)(\sqrt{2c}z_i) + c^2 \end{aligned}$$

feature mapping:(Again shown for $n = 3$)

$$\phi(x) = (x_1x_1, x_1x_2, \ldots, x_3x_3, \sqrt{2c}x_1, \sqrt{2c}x_2, \sqrt{2c}x_3, c)^T$$

More broadly, the kernel $K(x,z) = (x^T z + c)^d$ corresponds to a feature mapping to an $\binom{n+d}{d}$ feature space.
Map attribute to infinity dimension **Gaussian kernel**:

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$$

Kernels(3)

More broadly, given some function K, how can we tell if it's a valid kernel; i.e., can we tell if there is some feature mapping ϕ so that $K(x, z) = \phi(x)^T \phi(z)$ for all x, z?
Consider some finite set of m points $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, and let a m-by-m matrix K be defined as

$$K_{ij} = K(x^{(i)}, y(i))$$

K is called the **Kernel matrix**.
Now, if K is a valid Kernel, then

$$K_{ij} = \phi(x^{(i)})^T \phi(x^{(j)}) = \phi(x^{(j)})^T \phi(x^{(i)}) = K_{ji}$$

Hence, K must be symmetric.

Kernels(4)

More over, letting $\phi_k(x)$ denote the k-th coordinate of the vector $\phi(x)$,we find that for any vector z, we have

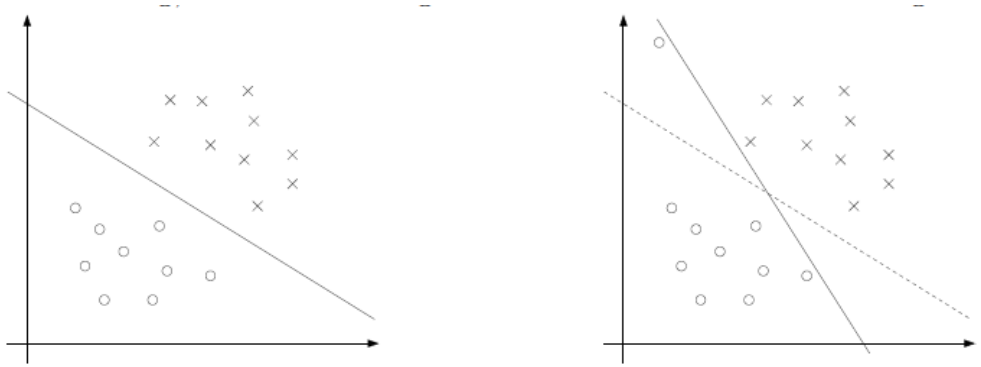
$$\begin{aligned} z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\ &= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\ &= \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)})^T \phi_k(x^{(j)}) z_j \\ &= \sum_k \sum_i \sum_j z_i \phi_k(x^{(i)})^T \phi_k(x^{(j)}) z_j \\ &= \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \geq 0 \end{aligned}$$

This shows that K is positive semi-definite

Kernels(5)

Theorem (Mercer). Let $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be given. Then for K to be a valid (Mercer) Kernel, it is necessary and sufficient that for any $\{x^{(1)}, \ldots, x^{(m)}\}, (m < \infty)$,the corresponding kernel matrix is symmetric positive semi-definite.

Regularization and the non-separable case



To make the algorithm work for non-linearly separable datasets as well as be less sensitive to outliers, we reformulate our optimization (l_1 **regularization**):

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ s.t \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, \ldots, m \\ & \xi_i \geq 0, i = 1, \ldots, m \end{aligned}$$

Regularization and the non-separable case(2)

As before, we can form the Lagrangian:

$$\mathcal{L}(w, b, \xi, \alpha, \gamma) = \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i \left(y^{(i)} (x^T w + b) - 1 + \xi_i \right) - \sum_{i=1}^{m} \gamma_i \xi_i$$

Here, the α_i's and γ_i's are our Lagrangian multipliers.
After some work,we obtain the dual form of the problem:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ s.t. \quad & 0 \leq \alpha_i \geq C, i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned}$$

The SMO algorithm

Consider trying to solve the unconstrained optimization problem

$$\max_{\alpha} W(\alpha_1, \alpha_2, \ldots, \alpha_m).$$

The algorithm here is called **coordinate ascent**:

Loop until convergence: {

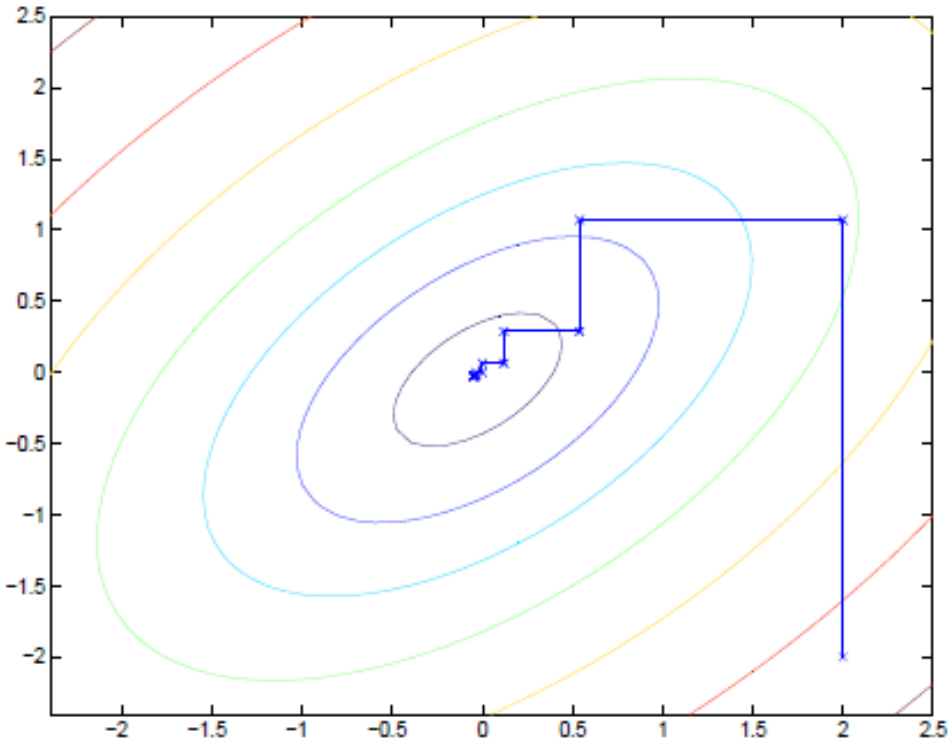
For $i = 1, \ldots, m$, {

$\alpha_i := \arg\max_{\hat{\alpha}_i} W(\alpha_1, \ldots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_m)$.

}

}

The SMO algorithm(2)



The ellipses in the figure are the contours of a quadratic function that we want to optimize.

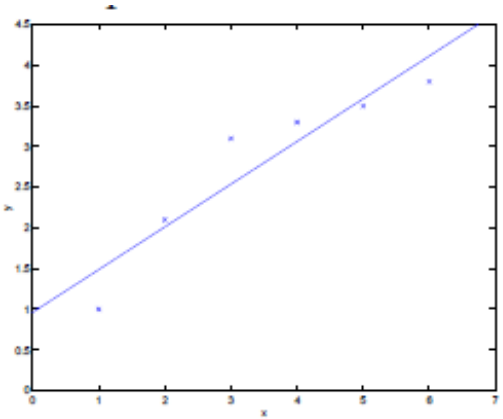
The SMO algorithm(3)

Repeat till convergence {

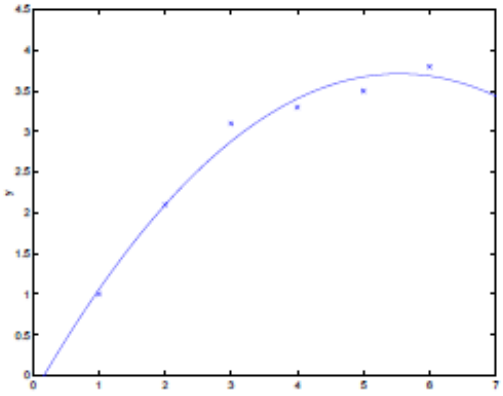
1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
2. Reoptimize $W(\alpha)$ with respect to α_i and α_j, while holding all the other α_k's $(k \neq i, j)$ fixed.

}

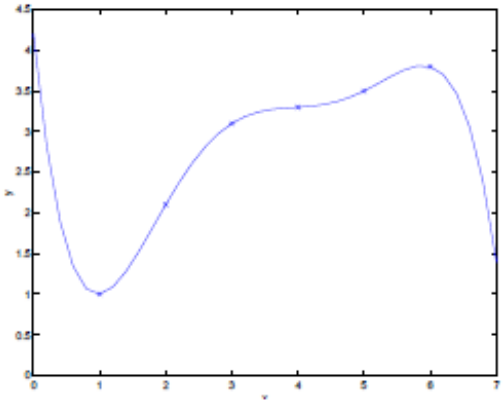
Bias/variance tradeoff



$y = \theta_0 + \theta_1 x$



$y = \theta_0 + \theta_1 x + \theta_2 x^2$



$y = \theta_0 + \theta_1 x + \cdots \theta_5 x^5$

- The linear model suffers from large bias, and may underfit the data.
- The 5th order polynomial model suffers from large variance, and may overfit the data.

Preliminaries

Lemma. (The union bound). Let $A_1, A_2, \ldots, A_k$ be k different events (that may not be independent). Then

$$P(A_1 \cup \ldots \cup A_k) \leq P(A_1) + \ldots + P(A_k)$$

Lemma. (Hoeffding inequality) Let $Z_1, \ldots, Z_m$ be the m independent and identically distributed (iid) random variables drawn from a $\text{Bernoulli}(\phi)$ distribution. I.e., $P(Z_i = 1) = \phi$, and $P(Z_i = 0) = 1 - \phi$. Let $\hat{\phi} = (1/m) \sum_{i=1}^{m} Z_i$ be the mean of these random variables, and let any $\gamma > 0$ be fixed. Then

$$P(|\phi - \hat{\phi}| > \gamma) \leq 2 \exp(-2\gamma^2 m)$$

Preliminaries(2)

We assume we are given a training set $S = \{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ of size m, where the training examples $(x^{(i)}, y^{(i)})$ are drawn iid from some probability distribution $\mathcal{D}$. For a hypothesis h, we define the **training error** (also called the **empirical risk** or **empirical error** in learning theory) to be

$$\hat{\varepsilon}(h) = \frac{1}{m} \sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\}$$

Here $1\{condition\}$ means, if condition is true then $1\{condition\} = 1$, otherwise $1\{condition\} = 0$.
Define **generalization error** to be

$$\varepsilon(h) = P_{(x,y)\sim\mathcal{D}}(h(x) \neq y)$$

Empirical risk minimization (ERM)

We define the **hypothesis class** $\mathcal{H}$ used by a learning algorithm to be the set of all classifiers considered by it. For linear classification, $\mathcal{H} = \{h_w : h_w(x) = 1\{w^T x \geq 0\}, w \in \mathbb{R}^{n+1}\}$.
Empirical risk minimization can now be thought of as a minimization over the class of functions $\mathcal{H}$:

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{\varepsilon}(h)$$

Let's consider the case of finite $\mathcal{H} = \{h_1, \ldots, h_k\}$. Take any one, fixed, $h_i \in \mathcal{H}$. Consider a Bernoulli random variable $Z_j = 1\{h_i(x^{(i)}) \neq y^{(j)}\}$, then the training error can be written

$$\hat{\varepsilon}(h_i) = \frac{1}{m} \sum_{j=1}^{m} Z_j$$

Since our training set was drawn idd from $\mathcal{D}$

$$\varepsilon(h_i) = E[Z_j] = P(Z_j = 1)$$

Empirical risk minimization (ERM)(2)

We can apply the Hoeffding inequality, and obtain

$$P(|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) \leq 2\exp(-2\gamma^2 m)$$

Let A_i denote the event that $|\varepsilon(h_i) - \varepsilon(\hat{h}_i)| > \gamma$. Using the union bound, we have

$$\begin{aligned} P(\exists h \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(A_1 \cup \ldots \cup A_k) \\ &\leq \sum_{i=1}^{k} P(A_i) \\ &\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 m) \\ &= 2k\exp(-2\gamma^2 m) \end{aligned}$$

Empirical risk minimization (ERM)(3)

Subtract both sides from 1, we obtain

$$\begin{aligned} P(\neg\exists h \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(\forall \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| \leq \gamma) \\ &\geq 1 - 2k\exp(-2\gamma^2 m) \end{aligned}$$

So, with probability at least $1 - 2k\exp(-2\gamma^2 m)$ we have that $\varepsilon(h)$ will be within γ of $\varepsilon(\hat{h})$ for all $h \in \mathcal{H}$. This is called a **uniform convergence result**.
Given γ and some $\delta > 0$, by setting $\delta = 2k\exp(-2\gamma^2 m)$,we find that if

$$m \geq \frac{1}{2\gamma^2}\log\frac{2k}{\delta},$$

then with probability at least $1 - \delta$, we have that $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ for all $h \in \mathcal{H}$.
The training set size m that a certain method or algorithm requires in order to achieve a certain level if performance is also called the algorithm's **sample complexity**.

Empirical risk minimization (ERM)(4)

Similarly,we can also hold m and δ fixed and solve for γ.

$$|\varepsilon(\hat{h}) - \varepsilon(h)| \leq \gamma \leq \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$$

Define $h^* = \arg\min_{h \in \mathcal{H}} \varepsilon(h)$. We have:

$$\varepsilon(\hat{h}) \leq \hat{\varepsilon}(\hat{h}) + \gamma \leq \hat{\varepsilon}(h^*) + \gamma \leq \varepsilon(h^*) + 2\gamma$$

Theorem. Let $|\mathcal{H}| = k$, and let any m, δ be fixed. Then with probability at least $1 - \delta$, we have that

$$\varepsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \varepsilon(h) \right) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$$

Empirical risk minimization (ERM)(5)

Given a set $S = \{x^{(1)}, \ldots, x^{(d)}\}$, we say that $\mathcal{H}$ **shatters** S if $\mathcal{H}$ can realize any labelling on S. I.e., if for any set of labels $\{y^{(1)}, \ldots, y^{(d)}\}$, there exists some $h \in \mathcal{H}$ so that $h(x^{(i)}) = y^{(i)}$ for all $i = 1, \ldots, d$.

Given a hypothesis class $\mathcal{H}$, we can define its **Vapnik-Chervonenkis dimension**, written $VC(\mathcal{H})$, to be the size of the largest set that is shattered by $\mathcal{H}$.

Theorem. Let $\mathcal{H}$ be given, and let $d = VC(\mathcal{H})$. Then with probability at least $1 - \delta$, we have that for all $h \in \mathcal{H}$,

$$|\epsilon(h) - \hat{\epsilon}(h)| \leq O\left(\sqrt{\frac{d}{m}\log\frac{m}{d} + \frac{1}{m}\log\frac{1}{\delta}}\right).$$

Empirical risk minimization (ERM)(6)

Thus, with probability at least $1-\delta$, we also have that:

$$\epsilon(\hat{h}) \leq \epsilon(h^*) + O\left(\sqrt{\frac{d}{m}\log\frac{m}{d} + \frac{1}{m}\log\frac{1}{\delta}}\right).$$

Corollary. For $|\epsilon(h) - \hat{\epsilon}(h)| \leq \gamma$ to hold all $h \in \mathcal{H}$ with probability at least $1-\delta$, it suffices that $m = O_{\gamma,\delta}(d)$.

In Mixture densities we have some assumptions.

1. The samples come from a known number of classes.
2. The prior probabilities $P(\omega_j)$ for each class are known, $j = 1, \ldots, c$.
3. The forms for the class-conditional probability densities $p(x|\omega_j, \theta_j)$ are known, $j = 1, \ldots, c$.
4. The values for the c parameter vectors. $\theta_1, \ldots, \theta_c$ are unknown.
5. The category labels are unknown.

Samples are assumed to be obtained by selecting state if nature ω_j with probability $P(\omega_j)$ and then selecting an x according to the probability law $p(x|\omega_j, \theta_j)$. Thus probability density function for the samples is given by

$$p(x|\theta) = \sum_{j=1}^{c} p(x|\omega_j, \theta_j) P(\omega_j)$$

Definition: A density $p(x|\theta)$ is said to be *identifiable* if $\theta \neq \theta'$ implies that there exists an x such that $p(x|\theta) \neq p(x|\theta')$.
Or put another way, a density $p(x|\theta)$ is *not identifiable* if we cannot recover a unique θ.
A example for unidentifiable:

$$\begin{aligned} P(x|\theta) &= \frac{1}{2}\theta_1^x(1-\theta_1)^{1-x} + \frac{1}{2}\theta_2^x(1-\theta_2)^{1-x} \\ &= \begin{cases} \frac{1}{2}(\theta_1+\theta_2) & \text{if } x=1; \\ 1-\frac{1}{2}(\theta_1+\theta_2) & \text{if } x=0. \end{cases} \end{aligned}$$

Suppose, for example, that we know for our data that P(x $= 1$—) $= 0.6$, and hence that $P(x=0|\theta) = 0.4$. Then we know the function $P(x|\theta)$, but we cannot determine θ, and hence cannot extract the component distributions. The most we can say is that $\theta_1 + \theta_2 = 1.2$. Thus, here we have a case in which the mixture distribution is completely unidentifiable.

Suppose now that we are given a set $\mathcal{D} = \{x_1, \ldots, x_n\}$ of n unlabeled samples drawn independently from mixture density.

$$p(x|\theta) = \sum_{j=1}^{c} p(x|\omega_j, \theta_j) P(\omega_j)$$

where the full parameter vector θ is fixed but unknown. The likelihood of the observed samples is, by definition, the joint density

$$p(\mathcal{D}|\theta) \equiv \prod_{k=1}^{n} p(x_k|\theta)$$

The maximum-likelihood estimate $\hat{\theta}$ is that value of θ that maximizes $p(\mathcal{D}|\theta)$.

The Log maximum-likelihood:

$$l = \sum_{k=1}^{n} \log p(x_k|\theta)$$

and

$$\begin{aligned}
\nabla_{\theta_i} l &= \sum_{k=1}^{n} \frac{1}{p(x_k|\theta)} \nabla_{\theta_i} \left(\sum_{j=1}^{c} p(x_k|\omega_j, \theta_j) P(\omega_j) \right) \\
&= \sum_{k=1}^{n} \frac{1}{p(x_k|\theta)} \nabla_{\theta_i} p(x_k|\omega_i) P(\omega_i)
\end{aligned}$$

Here we assume that θ_i and θ_j are independent if $i \neq j$.

According the Bayes' formula,

$$P(\omega_i|x_k,\theta) = \frac{p(x_k|\omega_i,\theta_i)P(\omega_i)}{p(x_k|\theta)}$$

Hence,

$$\nabla_{\theta_i} l = \sum_{k=1}^{n} P(\omega_i|x_k,\theta)\nabla_{\theta_i} \log p(x_k|\omega_i,\theta_i)$$

and $P(\omega_i)$ can be estimate by $P(\omega_i|x_k)$:

$$p(\omega_i) = \frac{1}{n}\sum_{k=1}^{n} P(\omega_i|x_k,\theta)$$

Mixtures of Gaussians and the EM algorithm

Suppose that we are given a training set $\{x^{(1)}, \ldots, x^{(m)}\}$, we wish to model the data by specifying a joint distribution $p(x^{(1)}, z^{(i)}) = p(x^{(i)}|z^{(i)})$. Here,

$$\begin{aligned} z^{(i)} \in \{1, \ldots, k\} &\sim \text{Multinomial}(\phi), \\ x^{(i)}|z^{(i)} = j &\sim \mathcal{N}(\mu_j, \Sigma_j) \end{aligned}$$

This is call the **mixture of Gaussians** model, and $z^{(i)}$'s are **latent** random variables. To estimate ϕ, μ and Σ, we can write down the likelihood of our data.

$$\begin{aligned} l(\phi, \mu, \Sigma) &= \sum_{i=1}^{m} \log p(x^{(i)}; \phi, \mu, \Sigma) \\ &= \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)}|z^{(i)}; \mu, \Sigma) p(z^{(i)}; \phi) \end{aligned}$$

Mixtures of Gaussians and the EM algorithm(2)

Note that if we knew what the $z^{(i)}$'s were, the maximum likelihood problem would have been easy. Specifically, we could then write down the likelihood as

$$l(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}|z^{(i)}; \mu, \Sigma) + \log p(z^{(i)}; \phi)$$

Maximizing this with respect to ϕ, μ and Σ gives the parameters.

$$\begin{aligned}
\phi_j &= \frac{1}{m} \sum_{i=1}^{m} 1\{z^{(i)} = j\} \\
\mu_j &= \frac{\sum_{i=1}^{m} 1\{x^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}} \\
\Sigma_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}
\end{aligned}$$

Mixtures of Gaussians and the EM algorithm(3)

The EM algorithm is an iterative algorithm that has two main step.

Repeat until convergence: {

(E-step) For each i, j, set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

(M-step) Update the parameters:

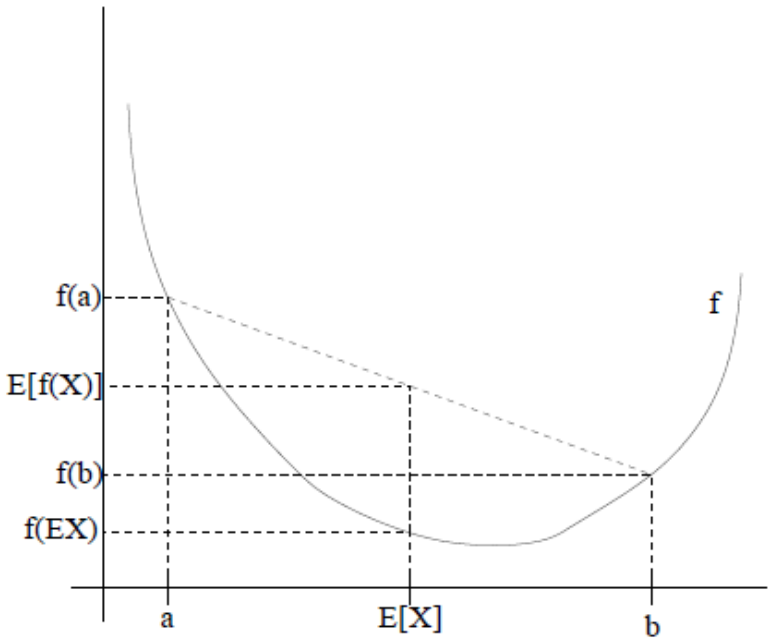
$$\begin{aligned}
\phi_j &:= \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}, \\
\mu_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}, \\
\Sigma_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}
\end{aligned}$$

Jensens inequality

Theorem. Let f be a convex(concave) function, and let X be a random variable. Then:

$$E[f(X)] \geq f(EX) \quad \text{(Concave case is } E[f(X)] \leq f(EX) \text{)}$$

Moreover, if f is strictly convex(concave), then $E[f(X)] = f(EX)$ holds true if and only if $X = E[X]$ with probability 1.



The EM algorithm

Suppose we have a training set $\{x^{(i)}, \ldots, x^{(m)}\}$. We wish to fit the parameters of a model p(x,z) to the data, where the likelihood is given by

$$l(\theta) = \sum_{i=1}^{m} \log p(x;\theta) = \sum_{i=1}^{m} \log \sum_{z} p(x,z;\theta).$$

For each i, let Q_i be some distribution over the z's($\sum_z Q_i(z) = 1, Q_i(z) \geq 0$. Consider:

$$\begin{aligned}
\sum_i \log p(x^{(i)};\theta) &= \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)};\theta) && (8)\\
&= \sum_i \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^i)} && (9)\\
&= \sum_i \log E_{z^{(i)}\sim Q_i}[\frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}] && (10)\\
&\geq \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})} && (11)
\end{aligned}$$

The EM algorithm(2)

To make the bound tight,we need Jensen's inequality to hold equality.we require that

$$\frac{p(x^{(i)}, z^{(i)})}{Q_i(z^{(i)})} = c$$

for some constant c that not depend on $z^{(i)}$

$$Q_i(z^{(i)}) \propto p(x^{(i)}, z^{(i)}; \theta)$$

Actually,since $\sum_z Q_i(z^{(i)}) = 1$:

$$\begin{aligned} Q_i(z^{(i)}) &= \frac{p(x^{(i)}, z^{(i)}; \theta)}{\sum_z p(x^{(i)}, z; \theta)} \\ &= \frac{p(x^{(i)}, z^{(i)}; \theta)}{p(x^{(i)}; \theta)} = p(z^{(i)}|x^{(i)}; \theta) \end{aligned}$$

The EM algorithm(3)

Generalized EM algorithm:

Repeat until convergence {

(E-step) For each i, set

$$Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)};\theta).$$

(M-step) Set

$$\theta := \arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}.$$

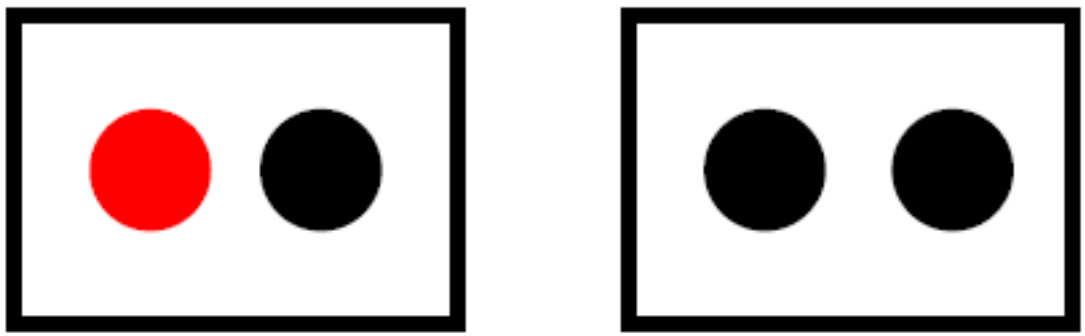
}

If we define

$$J(Q,\theta) = \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}.$$

the EM can also be viewed a coordinate ascent on J.

The envelope quiz



- red ball is goal
- You randomly picked an envelop randomly took out a ball and it was black
- Should you choose this envelope or the other envelope?

The envelope quiz(2)

- Probabilistic inference
  - Joint distribution on $E \in \{1,0\}, B \in \{r,b\}$: $P(E,B) = P(E)P(B|E)$
  - $P(E=1) = P(E=0) = \frac{1}{2}$
  - $P(B=r|E=1) = \frac{1}{2}, P(B=r|E=0) = 0$
  - The graphical model:

- Statistical decision theory: switch if $P(E=1|B=b) < \frac{1}{2}$
- $P(E=1|B=b) = \frac{P(B=b|E=1)P(E=1)}{P(B=b)} = \frac{1}{3}$

The probability explanation for ML

- The world is reduced to a set of random variables $x_1, \ldots, x_d$
  - e.g. $(x_1, \ldots, x_{d-1})$ is a feature vector, $x_d \equiv y$ is the class label.
- Inference: given joint distribution $p(x_1, \ldots, x_d)$, compute $p(X_Q|X_E)$ where $X_Q \cup X_E \subseteq \{x_1, \ldots, x_d\}$
  - e.g. $Q = \{x_d\}, E = \{x_1, \ldots, x_{d-1}\}$, by the definition of conditional

$$p(x_d|x_1, \ldots, x_{d-1}) = \frac{p(x_1, \ldots, x_d)}{\sum_v p(x_1, \ldots, x_{d-1}, x_d = v)}$$

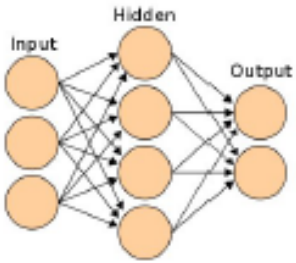
- Learning estimate $p(x_1, \ldots, x_d)$ from training data $\{x^{(1)}, \ldots, x^{(m)}\}$

What are graphical models?

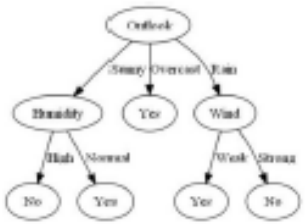
- Graphical model = joint distribution $p(x_1, \ldots, x_d)$
  - Bayesian network
  - Markov random field
- Inference = $p(X_Q|X_E)$, in general $X_Q \cup X_E \subseteq \{x_1, \ldots, x_d\}$
- If $p(x_1, \ldots, x_d) not given, estimate it from data$
  - parameter and structure learning

What are graphical models?(2)

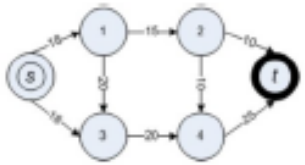
- Graphical model is the study of probabilistic models
- Just because there are nodes and edges doesn't mean it's a graphical model
- These are not graphical models:



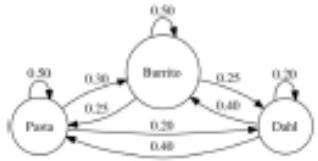
neural network



decision tree



network flow



HMM template
(but HMMs are!)

Directed graphical models

- Also called Bayesian networks
- A directed graph has nodes $x_1, \ldots, x_d$, some of them connected by directed edges $x_i \to x_j$
- A cycle is a directed path $x_1 \to \ldots \to x_k$ where $x_1 = x_k$
- A directed acyclic graph (DAG) contains no cycles

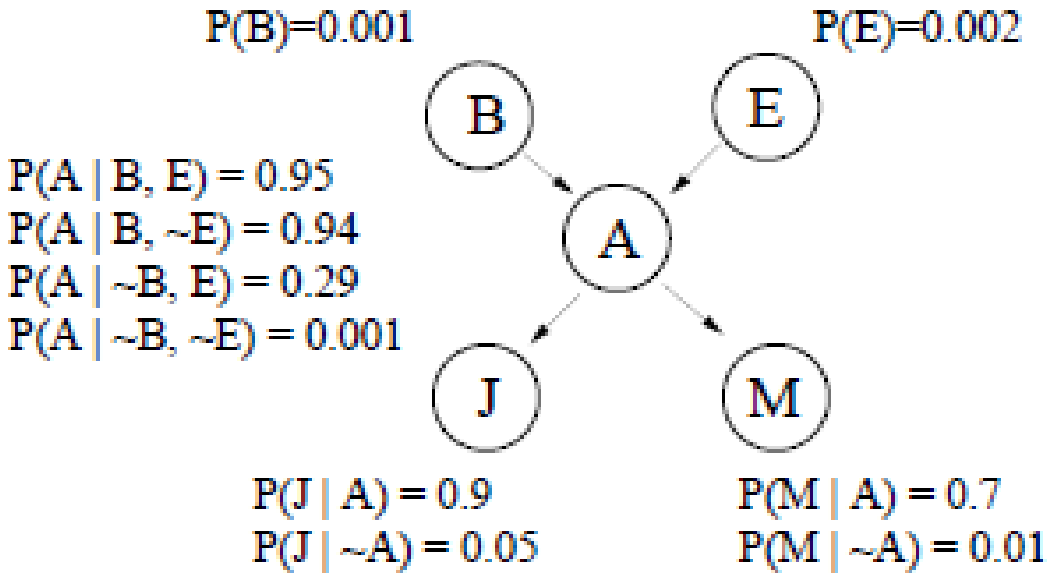
Directed graphical models(2)

- A Bayesian network on the DAG is a family of distributions satisfying

$$p|p(x_1, \ldots, x_d) = \prod_i p(x_i|Pa(x_i))$$

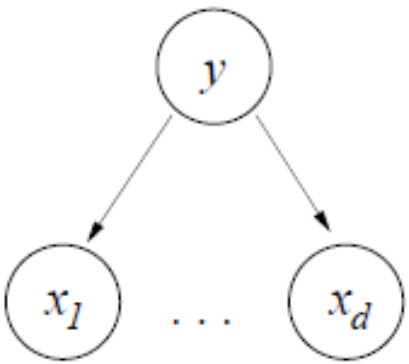
  where $Pa(x_i)$ is the set of parents of x_i
- $p(x_i|Pa(x_i))$ is the conditional probability distribution (CPD) at x_i
- By specifying the CPDs for all i, we specify a joint distribution $p(x_1, \ldots, x_d)$

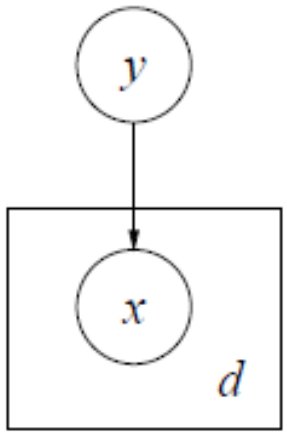
Example:Burglary, Earthquake, Alarm, John and Marry



$$P(B, \sim E, A, J, \sim M) = P(B)P(\sim E)P(A|B, \sim E)P(J|A)P(\sim M|A)$$

Example: Naive Bayes





- $p(y, x_1, \ldots, x_d) = p(y) \prod_{i=1}^{d} p(x_i|y)$
- Plate representation on the right
- $p(y)$ multinomial
- $p(x_i|y)$ depends on the feature type.

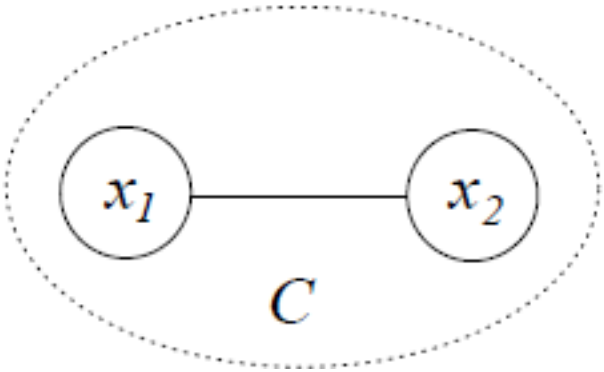
Undirected graphical models

- Also known as Markov Random Fields
- A clique C in an undirected graph is a set of fully connected nodes (full of loops!)
- Define a nonnegative potential function $\psi_C : X_C \to \mathbb{R}_+$
- An undirected graphical model is a family of distributions satisfying

$$\left\{ p | p(X) = \frac{1}{Z} \prod_C \psi_C(X_c) \right\}$$

- $Z = \int \prod_C \psi_C(X_C) dX$

A Tiny Markov Random Field



- $x_1, x_2 \in \{-1, 1\}$
- A single clique $\psi_C(x_1, x_2) = e^{ax_1x_2}$
- $p(x_1, x_2) = \frac{1}{Z} e^{ax_1x_2}$
- $Z = (2e^a + 2e^{-a})$
- $p(1, 1) = p(-1, -1) = \frac{e^a}{2e^a + 2e^{-a}}$

Topic Model

- A topic is defined as a probability distribution over terms or a cluster of weighted terms.
- A document is defined as a set of words generated from a mixture of latent topics.

Various topic modeling methods, such as PLSI, LDA, LSI, NMF, and RLSI have been proposed and successfully applied to different applications.

Probabilistic Topic Models

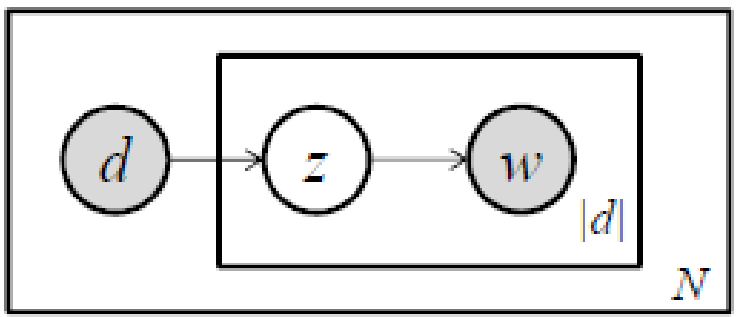
Suppose that $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ is a set of documents with size N and $\mathcal{V}$ is a set of terms or words with size M, i.e., the vocabulary. A document $d \in \mathcal{D}$ consists of $|d|$ words from the vocabulary, denoted as $d = (w_1, w_2, \ldots, w_{|d|})$. Suppose that there are K topics in the document collection.
PLSI is one of the widely used probabilistic topic models. One can generate the documents in the collection in the following way.

- select a document d from the collection with probability $P(d)$
- select a latent topic z with probability $P(z|d)$
- generate a word w with probability $P(w|z)$

Here $z \in \{z_1, \ldots, z_K\}$ is a latent variable representing a topic.

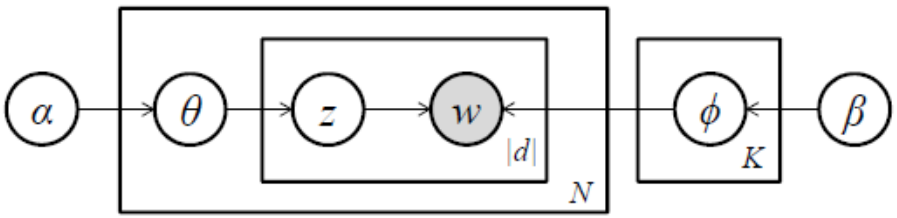
Probabilistic Topic Models(2)

The parameters of $P(d)$, $P(w|z)$, and $P(z|d)$ can be estimated by EM algorithm.



Probabilistic Topic Models(3)

LDA Model:



1. for each topic $k = 1, \ldots, K$
   (a) draw word distribution ϕ_k according to $\phi_k|\beta \sim \mathrm{Dir}(\beta)$
2. for each document d in the collection
   (a) draw topic distribution θ according to $\theta|\alpha \sim \mathrm{Dir}(\alpha)$
   (b) for each word w in the document d
      i. draw a topic z according to $z|\theta \sim \mathrm{Mult}(\theta)$
      ii. draw a word w according to $w|z \sim \mathrm{Mult}(\phi_z)$

Non-probabilistic Topic Models

Non-probabilistic topic models are usually obtained by matrix factorization.Suppose that $\mathcal{D}$ is a set of document with size N, and $\mathcal{V}$ is a vocabulary with size M.The document collection $\mathcal{D}$ is represented as an $M \times N$ matrix $D = (d_1, \ldots, d_N)$.
Suppose that there are K topics represented as an $M \times K$ term-topic matrix $U = (u_1, \ldots, u_K)$. $V^T = (v_1, \ldots, v_N)$ is an $K \times N$ topic-document matrix.
Our goal is to factor matrix D by U and V^T.

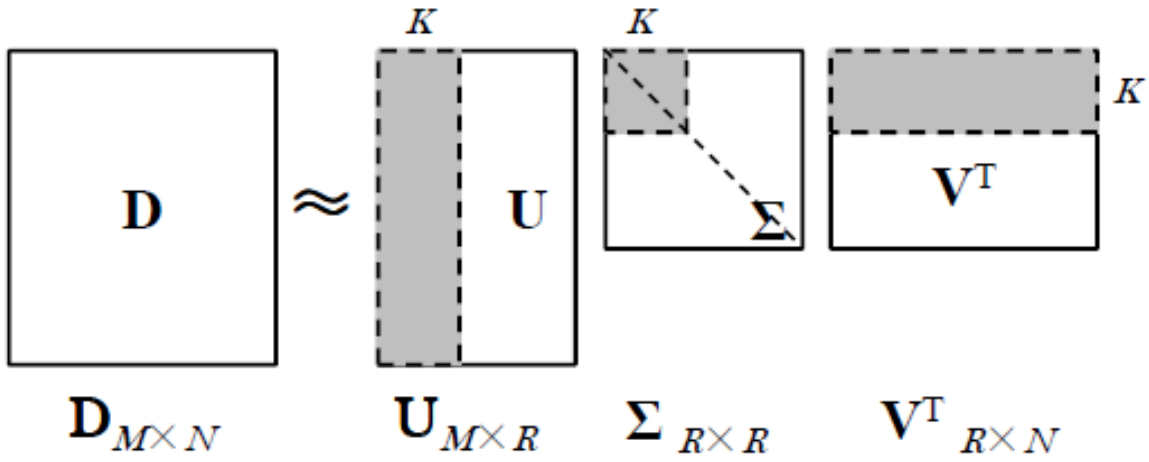
$$D \approx UV^T$$

Non-probabilistic Topic Models(2)

LSI assumes that the K columns of matrix U as well as the K columns of matrix V are orthonormal. LSI amounts to minimizing the following objective function with the orthonormal constraints.

$$\min_{U,V} \|D - U\Sigma V^T\|_F$$

$$s.t. \quad U^T \times U = I, V \times V^T = I, \text{and } \Sigma \text{ is diagonal}$$



What is entropy

In thermodynamics, entropy (usual symbol S) is a measure of the number of specific ways in which a thermodynamic system may be arranged, commonly understood as a measure of disorder.

$$\Delta S = \int \frac{dQ_{rev}}{T}$$

In information theory, **Entropy** is a measure of unpredictability of information content. Shannon defined the entropy H of a discrete random variable X with possible values $x_1, ..., x_n$ and probability mass function P(X) as:

$$H(X) = E[I(X)] = E[-\ln(P(X))]$$

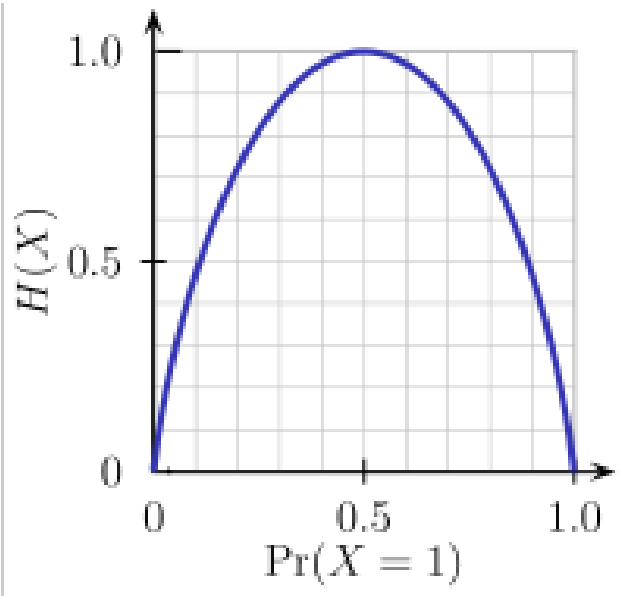
Here I is the information content of X.

What is entropy(2)

When taken from a finite sample, the entropy can explicitly be written as

$$H(X) = \sum_i P(x_i) I(x_i) = -\sum_i P(x_i) \log_b P(x_i)$$

Common values of b are 2, e, and 10, and the unit of entropy is bit for $b = 2$, nat for $b = e$, and dit (or digit) for $b = 10$.



Whis is entropy(3)

According the definition of $H(X)$, we can define **Joint Entropy** as:

$$H(X,Y) = -\sum_{x,y} p(x,y) \log p(x,y)$$

and define **Conditional Entropy** to measure unpredictability of random variable Y given random variable X as:

$$H(Y|X) = -\sum_{x,y} p(x,y) \log p(y|x)$$

The entropy or the amount of information revealed by evaluating (X,Y) (that is, evaluating X and Y simultaneously) is equal to the information revealed by conducting two consecutive experiments: first evaluating the value of Y, then revealing the value of X given that you know the value of Y. This may be written as

$$H(X,Y) = H(X|Y) + H(Y) = H(Y|X) + H(X) \qquad (12)$$

Whis is entropy(4)

We can get (7) by:

$$
\begin{aligned}
& H(X,Y) - H(X) \\
= & -\sum_{x,y} p(x,y) \log p(x,y) + \sum_{x} p(x) \log p(x) \\
= & -\sum_{x,y} p(x,y) \log p(x,y) + \sum_{x} \left(\sum_{y} p(x,y) \right) \log p(x) \\
= & -\sum_{x,y} p(x,y) \log p(x,y) + -\sum_{x,y} p(x,y) \log p(x) \\
= & -\sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)} \\
= & -\sum_{x,y} p(x,y) \log p(y|x) = H(Y|X)
\end{aligned}
$$

Whis is entropy(5)

Another useful measure of entropy that works equally well in the discrete and the continuous case is the **Relative Entropy** of a distribution. It is also called as the **Kullback-Leibler Divergence**.

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} = E_{p(x)} \log \frac{p(x)}{q(x)}$$

To some extent, Relative Entropy can be the measure of the distance between two random variables.
Mutual Information can be defined as:

$$I(X,Y) = \sum_{x,y} \log \frac{p(x,y)}{p(x)p(y)}$$

What is entropy(6)

Let's consider

$$
\begin{aligned}
& H(Y) - I(X,Y) \\
= & -\sum_{y} p(y)\log p(y) - \sum_{x,y} \log \frac{p(x,y)}{p(x)p(y)} \\
= & -\sum_{y}\left(\sum_{x} p(x,y)\right)\log p(y) - \sum_{x,y} p(x,y)\log \frac{p(x,y)}{p(x)p(y)} \\
= & -\sum_{x,y} p(x,y)\log p(y) - \sum_{x,y} p(x,y)\log \frac{p(x,y)}{p(x)p(y)} \\
= & -\sum_{x,y} p(x,y)\log \frac{p(x,y)}{p(x)} \\
= & -\sum_{x,y} p(x,y)\log p(y|x) = H(Y|X)
\end{aligned}
$$

According the formula (7):

$$I(X,Y) = H(X) + H(Y) - H(X,Y)$$

Motivating example

Suppose we want a model translate English word *in* to French.
Let p denote the model, and $p(f)$ denote the probability that f is chose as a translation of *in*.($f \in \{$dans,en,à,au cours de,pendant$\}$)
Now, we can impose our first constraint on our model p:

$$p(\text{dans}) + p(\text{en}) + p(\text{à}) + p(\text{au cours de}) + p(\text{pendant}) = 1$$

With no empirical justification, the most in intuitively model is

$$p(\text{dans}) = p(\text{en}) = p(\text{à}) = p(\text{au cours de}) = p(\text{pendant}) = \frac{1}{5}$$

If we add an additional constraint

$$p(\text{dans}) + p(\text{en}) = \frac{1}{3}$$

subject to the constraints

$$p(\text{dans}) = p(\text{en}) = \frac{3}{20}; p(\text{à}) = p(\text{au cours de}) = p(\text{pendant}) = \frac{7}{30}$$

Training data

Let's do some form formalization, suppose we have collected a large number of samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$.

In the example we have been considering, each sample would consist of a phrase x containing the words surrounding *in*, together with the translation y of *in*.

We can summarize the training example in terms of its **empirical probability distribution** $\widetilde{p}$, define by

$$\widetilde{p}(x, y) = \frac{1}{N} \times \text{number of times that } (x, y) \text{ occurs in sample.}$$

Features and constraints

Now, let's consider the definition of **feature function**

$$f(x,y) = \begin{cases} 1 & \text{if } x,y \text{ satisfy some condition} \\ 0 & \text{otherwise} \end{cases}$$

In our example we have employed some constraints, but we could also consider statistics which depend on x.
For instance, we might notice that, in the training example, if *April* is the word following *in*, then the translation of *in* is *en*.
Now we can use feature function to express the event that *in* translates as *en* when *April* is the following word.

$$f(x,y) = \begin{cases} 1 & \text{if } y = en \text{ and } \textit{April} \text{ follows } \textit{in} \\ 0 & \text{otherwise} \end{cases}$$

Features and constraints(2)

The expected value of f with respect to the empirical distribution $\widetilde{p}(x, y)$

$$\widetilde{p}(f) = \sum_{x,y} \widetilde{p}(x, y) f(x, y) \tag{13}$$

The expected value of f with respect to real distribution $p(x, y)$

$$p(f) = \sum_{x,y} p(x, y) f(x, y) = \sum_{x,y} p(x) p(y|x) f(x, y)$$

For convenience, we usually use $\widetilde{p}(x)$ instead of $p(x)$

$$p(f) = \sum_{x,y} \widetilde{p}(x) p(y|x) f(x, y) \tag{14}$$

Features and constraints(3)

We constrain this expected value to be the same as the expected value of f in the training sample. That is, we require

$$p(f) = \widetilde{p}(f). \tag{15}$$

Combining (8), (9) and (10), we obtain

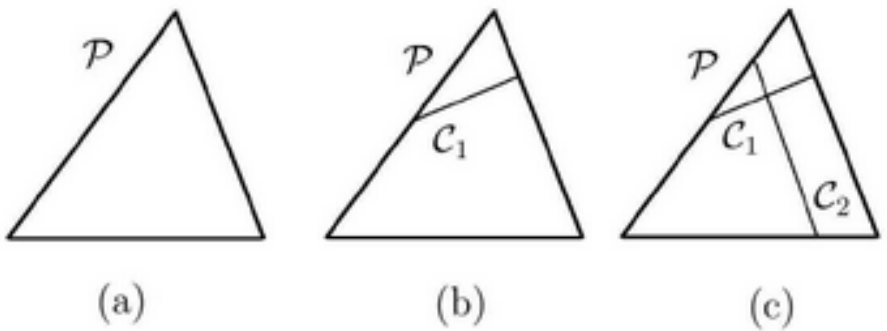
$$\sum_{x,y} \widetilde{p}(x)p(y|x)f(x,y) = \sum_{x,y} \widetilde{p}(x,y)f(x,y). \tag{16}$$

We call the requirement (11) a **constraint equation** or simply a **constraint**.

The Maximum Entropy Principle

Suppose that we are given n feature function $\{f_1, \ldots, f_n\}$, which determine statistics we feel are import in modeling the process. We would like our model to accord with these conditions. That is, we would like p to lie in the subset $\mathcal{C}$ of $\mathcal{P}$ defined by

$$\mathcal{C} = \{p \in \mathcal{P} | p(f_i) = \widetilde{p}(f_i) \quad i \in \{1, 2, \ldots, n\}\} \tag{17}$$



The Maximum Entropy Principle(2)

Among the models $p \in \mathcal{C}$, the maximum entropy philosophy dictates that we select the distribution which is most uniform, which means we choose p maximizing the entropy

$$H(p) = -\sum_{x,y} p(x,y)p(y|x)\log p(y|x) \tag{18}$$

With these definition in hand, we can present the principle of maximum entropy

$$p^* = \arg\max_{p \in \mathcal{C}} H(p) \tag{19}$$

It can be shown that p^* is well defined; that is, there is always a unique model p^* with maximum entropy in any constrained set $\mathcal{C}$

Optimization problem

The constrained optimization problem at hand is to find

$$p^* = \arg\max_{p \in \mathcal{C}} \left(-\sum_{x,y} \widetilde{p}(x) p(y|x) \log p(y|x) \right) \tag{20}$$

subject to

$$p(y|x) \geq 0 \quad \text{for all } x, y \tag{21}$$

$$\sum_y p(y|x) = 1 \quad \text{for all } x \tag{22}$$

$$\widetilde{p}(f_i) = p(f_i) \quad \text{for } i \in \{1, 2, \ldots, n\} \tag{23}$$

Optimization problem(2)

To solve this optimization problem, introduce the Lagrangian

$$\mathcal{L}(p,\Lambda) = -H(p) + \lambda_0 \left(1 - \sum_y p(y|x)\right) + \sum_{i=1}^{n} \lambda_i \left(\widetilde{p}(f_i) - p(f_i)\right), \tag{24}$$

where $\Lambda = \{\lambda_0, \lambda_1, \ldots, \lambda_n\}$.
Then we obtain the **primary problem**

$$\min_{p \in \mathcal{C}} \max_{\Lambda} \mathcal{L}(p,\Lambda) \tag{25}$$

For $-H(p)$ is convex function, the primary problem is equivalent to its **dual problem**

$$\max_{\Lambda} \min_{p \in \mathcal{C}} \mathcal{L}(p,\Lambda) \tag{26}$$

Optimization problem(3)

Consider the minimum problem in dual problem:

$$\Psi(\Lambda) = \min_{p \in \mathcal{C}} \mathcal{L}(p, \Lambda) = \mathcal{L}(p_\Lambda, \Lambda), \tag{27}$$

where

$$p_\Lambda = \arg\max_{p \in \mathcal{C}} \mathcal{L}(p, \Lambda). \tag{28}$$

The we use **Lagrangian multiplier method** to obtain p_Λ

$$\frac{\partial \mathcal{L}(p, \Lambda)}{\partial p(y|x)} = \widetilde{p}(x)(\log p(y|x) + 1) - \lambda_0 - \sum_{i=1}^{n} \widetilde{p}(x) f_i(x, y) \tag{29}$$

Optimization problem(4)

Let $\frac{\partial \mathcal{L}(p,\Lambda)}{\partial p(y|x)}$ be zero, we obtain

$$\log p(y|x) + 1 - \lambda_0 - \sum_{i=1}^{n} \lambda_i f_i(x,y) = 0, \tag{30}$$

then

$$p(y|x) = e^{\lambda_0 - 1} \cdot e^{\sum_{i=1}^{n} \lambda_i f_i(x,y)}. \tag{31}$$

According to constraint (17), we obtain

$$p_\Lambda = \frac{1}{Z_\Lambda(x)} e^{\sum_{i=1}^{n} \lambda_i f_i(x,y)} \tag{32}$$

where

$$Z_\Lambda(x) = \sum_{y} e^{\sum_{i=1}^{n} \lambda_i f_i(x,y)} \tag{33}$$

Optimization problem(5)

Consider the maximum problem in dual problem:

$$\max_{\Lambda} \Psi(\Lambda) \tag{34}$$

We suppose

$$\Lambda^* = \arg\max_{\Lambda} \Psi(\Lambda), \tag{35}$$

then we can solve the optimization problem by $p^* = p_\Lambda$.
We know

$$\begin{aligned}
\Psi(\Lambda) &= \mathcal{L}(p_\Lambda, \Lambda) \\
&= \sum_{x,y} \widetilde{p}(x) p_\Lambda(y|x) \log p_\Lambda(y|x) \\
&+ \sum_{i=1}^{n} \lambda_i \left(\widetilde{p}(f_i) - \sum_{x,y} \widetilde{p}(x) p_\Lambda(y|x) f_i(x,y) \right) \\
&= \sum_{i=1}^{n} \lambda_i \widetilde{p}(f_i) + \sum_{x,y} \widetilde{p}(x) p_\Lambda(y|x) \left(\log p_\Lambda(y|x) - \sum_{i=1}^{n} \lambda_i f_i(x,y) \right)
\end{aligned}$$

Optimization problem(6)

According to (26), we obtain

$$\log p_\Lambda(y|x) = \sum_{i=1}^{n} \lambda_i f_i(x,y) - \log Z_\Lambda(x) \tag{36}$$

substitute it to $\Psi(\Lambda)$,

$$\begin{aligned}
\Psi(\Lambda) &= \sum_{i=1}^{n} \lambda_i \widetilde{p}(f_i) - \sum_{x,y} \widetilde{p}(x) p_\Lambda(y|x) \log Z_\Lambda(x) \\
&= \sum_{i=1}^{n} \lambda_i \widetilde{p}(f_i) - \sum_{x} \widetilde{p}(x) \log Z_\Lambda(x) \sum_{y} p_\Lambda(y|x) \\
&= \sum_{i=1}^{n} \lambda_i \widetilde{p}(f_i) - \sum_{x} \widetilde{p}(x) \log Z_\Lambda(x)
\end{aligned}$$

Maximum likelihood

Consider MLE:

$$\mathcal{L}_{\widetilde{p}}(p) = \log \prod_{x,y} p(y|x)^{\widetilde{p}(x,y)} = \sum_{x,y} \widetilde{p}(x,y) \log p(y|x) \tag{37}$$

According (30) we obtain,

$$\begin{aligned}
\mathcal{L}_{\widetilde{p}}(p_\Lambda) &= \sum_{x,y} \widetilde{p}(x,y) \left(\sum_{i=1}^{n} \lambda_i f_i(x,y) - \log Z_\Lambda(x) \right) \\
&= \sum_{x,y} \widetilde{p}(x,y) \sum_{i=1}^{n} \lambda_i f_i(x,y) - \sum_{x,y} \widetilde{p}(x,y) \log Z_\Lambda(x) \\
&= \sum_{i=1}^{n} \lambda_i \left(\sum_{x,y} \widetilde{p}(x,y) f_i(x,y) \right) - \sum_{x,y} \widetilde{p}(x,y) \log Z_\Lambda(x) \\
&= \sum_{i=1}^{n} \lambda_i \widetilde{p}(f_i) - \sum_{x} \widetilde{p}(x) \log Z_\Lambda(x)
\end{aligned}$$

Computing the Parameters

Some Parameters Optimization method in common use like **Gradient descent**:

$$
\begin{aligned}
\frac{\partial \Psi}{\partial \lambda_i} &= \frac{\partial}{\partial \lambda_i}\left(\sum_{i=1}^{n} \lambda_i \widetilde{p}(f_i) - \sum_x \widetilde{p}(x) \log Z_\Lambda(x)\right) \\
&= \widetilde{p}(f_i) - \sum_x \widetilde{p}(x) \frac{1}{Z_\Lambda(x)} \frac{\partial}{\partial \lambda_i}\left(\sum_y e^{\sum_{i=1}^{n} \lambda_i f_i(x,y)}\right) \\
&= \widetilde{p}(f_i) - \sum_x \widetilde{p}(x) \frac{1}{Z_\Lambda(x)} \sum_y e^{\sum_{i=1}^{n} \lambda_i f_i(x,y)} f_i(x,y) \\
&= \widetilde{p}(x) - \sum_x \widetilde{p}(x) \sum_y p_\Lambda(y|x) f_i(x,y)
\end{aligned}
$$

GIS algorithm

Step1 Initial parameters. set $\Lambda = 0$

Step2 Calculate $E_{\widetilde{p}(f_i)} = \sum_{i=1}^{n} \widetilde{p}(x,y) f_i(x,y)$, for $i = 1, 2, \ldots, n$.

Step3 Iteration, update parameters:

Calculate $E_{p_\Lambda}(f_i)$, $i = 1, 2, \ldots, n$.
FOR $i = 1, 2, \ldots, n$ **DO**
$\{$
$\lambda_i := \lambda_i + \eta \log \frac{E_{\widetilde{p}}(f_i)}{E_{p_\Lambda}(f_i)}$
$\}$

Step4 Check convergence.

IIS algorithm

Step1 Initial parameters, set $\Lambda := 0$

Step2 Iteration, update parameters:

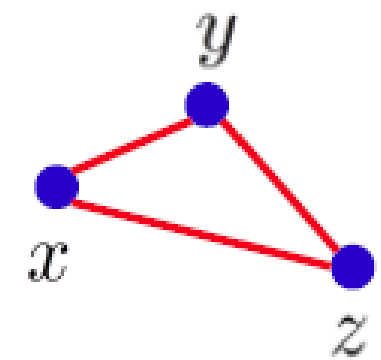
FOR $i = 1, 2, \ldots, n$ **DO**
{
Solve equations
$\sum_{x,y} \widetilde{p}(x) p(y|x) f_i(x,y) e^{\delta_i \sum_{i=1}^{n} f_i(x,y)} = \widetilde{p}(f_i)$
Let $\lambda_i := \lambda_i + \delta_i$
}

Step3 Check convergence.

Euclidean distance and cosine

Strict **distance** satisfies four constraints while divergence may no satisfy.

- **Non-negativity:** $\forall x, y, \quad d(x, y) \geq 0$
- **Non-degeneracy:** $d(x, y) = 0 \Leftrightarrow x = y$
- **Symmetry:** $\forall x, y, \quad d(x, y) = d(y, x)$
- **Triangularity:** $\forall x, y, z \quad d(x, z) \leq d(x, y) + d(y, z)$



Euclidean distance and cosine(2)

In the **Euclidean space** $\mathbb{R}^n$,the distance between two point can be given by **Minkowski distance**.(When p=2, it is equivalent the **Euclidean distance** and when p=1, it is equivalent to **Manhattan distance**) Other distance, based on norms, are sometimes used instead.

- 1-norm distance $= \sum_{i=1}^{n} |x_i - y_i|$
- 2-norm distance $= \left(\sum_{i=1}^{n} |x_i - y_i|^2\right)^{1/2}$
- p-norm distance $= \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$
- ∞ norm distance $= \lim_{p\to\infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$

If we care about the direction of the data rather than the magnitude, then using the **cosine distance** is a common approach.

$$\cos\theta = \frac{A \cdot B}{\|A\|\|B\|}$$

Euclidean distance and cosine(3)

Why is Euclidean distance not a good metric in most situation?

- Vector length often is not determinant.
- Most Problems are in high-dimensional.
- Most of the volume of a high-dimensional orange is in the skin, not the pulp.

Kullback-Leibler Divergence

Kullback-Leibler Divergence defines as

$$KL(p||p') = \int p(x) \log \frac{p(x)}{p'(x)} dx \tag{38}$$

- ☺ Compatible with maximum likelihood.
- ☺ Invariant under input transformation.
- ☹ Doesnt satisfy symmetry and triangularity.
- ☹ Sensitive to outliers (due to log and ratio).

f-Divergence

Definition:

$$F(p||p') = \int p'(x) f\left(\frac{p(x)}{p'(x)}\right) dx \tag{39}$$

where f is convex function such as $f(1) = 0$
Let $f\left(\frac{p(x)}{p'(x)}\right) = \frac{p(x)}{p'(x)} \log \frac{p(x)}{p'(x)}$, then yield the KL-divergence.

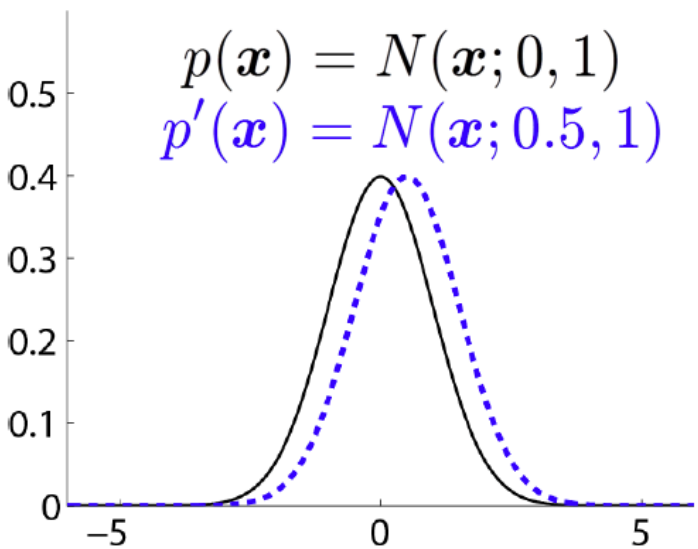
To avoid the log function, let us use $f = \left(\frac{p(x)}{p'(x)} - 1\right)^2$. We obtain the **Pearson (PE) Divergence**

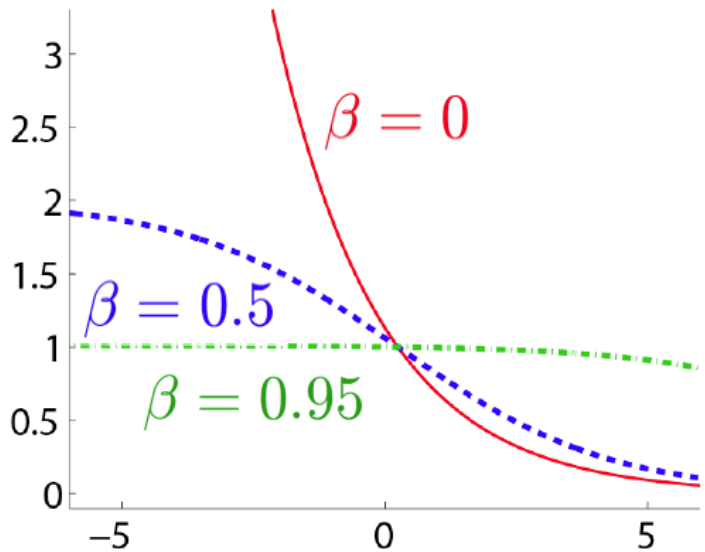
$$PE(p||p') = \int p'(x) \left(\frac{p(x)}{p'(x)} - 1\right)^2 \tag{40}$$

Relative Density Ratio

Density ratio $\frac{p(x)}{p'(x)}$ can diverge to infinity. However, Relative density ratio is always bounded:

$$\frac{p(x)}{\beta p(x) + (1-\beta)p'(x)} < \frac{1}{\beta} \quad 0 \leq \beta < 1 \tag{41}$$





Relative Pearson (rPE) Divergence

$$rPE(p||p') = PE(p||p_\beta) = \int p_\beta(x) \left(\frac{p(x)}{p_\beta(x)} - 1 \right)^2 dx \qquad (42)$$

where

$$0 \leq \beta < 1 \quad p_\beta(x) = \beta p(x) + (1-\beta)p'(x)$$

- ☺ Compatible with least-squares.
- ☺ Invariant under input transformation.
- ☺ Robust against outliers.
- ☹ Doesnt satisfy symmetry and triangularity.
- ☹ Not clear how to choose β.

How to choose

braycurtis(u, v)	Computes the Bray-Curtis distance between two 1-D arrays.
canberra(u, v)	Computes the Canberra distance between two 1-D arrays.
chebyshev(u, v)	Computes the Chebyshev distance.
cityblock(u, v)	Computes the City Block (Manhattan) distance.
correlation(u, v)	Computes the correlation distance between two 1-D arrays.
cosine(u, v)	Computes the Cosine distance between 1-D arrays.
dice(u, v)	Computes the Dice dissimilarity between two boolean 1-D arrays.
euclidean(u, v)	Computes the Euclidean distance between two 1-D arrays.
hamming(u, v)	Computes the Hamming distance between two 1-D arrays.
jaccard(u, v)	Computes the Jaccard-Needham dissimilarity between two boolean 1-D arrays.
kulsinski(u, v)	Computes the Kulsinski dissimilarity between two boolean 1-D arrays.
mahalanobis(u, v, VI)	Computes the Mahalanobis distance between two 1-D arrays.
matching(u, v)	Computes the Matching dissimilarity between two boolean 1-D arrays.
minkowski(u, v, p)	Computes the Minkowski distance between two 1-D arrays.
rogerstanimoto(u, v)	Computes the Rogers-Tanimoto dissimilarity between two boolean 1-D arrays.
russellrao(u, v)	Computes the Russell-Rao dissimilarity between two boolean 1-D arrays.
seuclidean(u, v, V)	Returns the standardized Euclidean distance between two 1-D arrays.
sokalmichener(u, v)	Computes the Sokal-Michener dissimilarity between two boolean 1-D arrays.
sokalsneath(u, v)	Computes the Sokal-Sneath dissimilarity between two boolean 1-D arrays.
sqeuclidean(u, v)	Computes the squared Euclidean distance between two 1-D arrays.
wminkowski(u, v, p, w)	Computes the weighted Minkowski distance between two 1-D arrays.
yule(u, v)	Computes the Yule dissimilarity between two boolean 1-D arrays.