Introduction to Machine Learning
(Part 1: Statistical Machine Learning)

Shou-de Lin

CSIE/GINM, NTU
sdlin@csie.ntu.edu.tw

Syllabus of a **Intro-**ML course ("Machine Learning", Andrew Ng, Stanford, Autumn 2009)

- **Supervised learning.** (7 classes) Supervised learning setup. LMS.
  - Logistic regression. Perceptron. Exponential family.
  - Generative learning algorithms. Gaussian discriminant analysis. Naive Bayes.
  - Support vector machines.
  - Model selection and feature selection.
  - Ensemble methods: Bagging, boosting, ECOC.
  - Evaluating and debugging learning algorithms.
- **Learning theory.** (3 classes)
  - Bias/variance tradeoff. Union and Chernoff/Hoeffding bounds.
  - VC dimension. Worst case (online) learning.
  - Practical advice on how to use learning algorithms.
- **Unsupervised learning.** (5 classes)
  - Clustering. K-means. EM. Mixture of Gaussians.
  - Factor analysis. PCA. MDS. pPCA.
  - Independent components analysis (ICA).
- **Reinforcement learning and control.** (4 classes)
  - MDPs. Bellman equations. Value iteration and policy iteration.
  - Linear quadratic regulation (LQR). LQG.
  - Q-learning. Value function approximation.
  - Policy search. Reinforce. POMDPs.

HT has done a great job teaching you "Advanced SL" and "Learning Theory", and my mission is to fill one missing piece in the puzzle.

Why teaching “Intro to ML”?

- When revealing that you have taken an ML course, people would more or less expect you to have already known something, E.g.
  - Naïve Bayes.
- There are some ML methods that are so commonly applied in research and real world that you will need to know a little bit about them. E.g.
  - K-means clustering
- There are some ML method that are too unbelievable and amazing to ignore . E.g.
  - EM framework.

To Bring you Back to the Earth

- **Statistical Machine Learning.** (2 hours)
  – A Bayesian view about ML
  – Generative learning model.
  – Gaussian discriminant analysis. Naïve Bayes
- **Unsupervised learning.** (3 hours)
  – Clustering: K-means.
  – EM.
- **Reinforcement learning** (0.5 hour)
  – Value iteration and policy iteration.
  – Q-learning & SARSA

Theoretical ML vs. Statistical ML

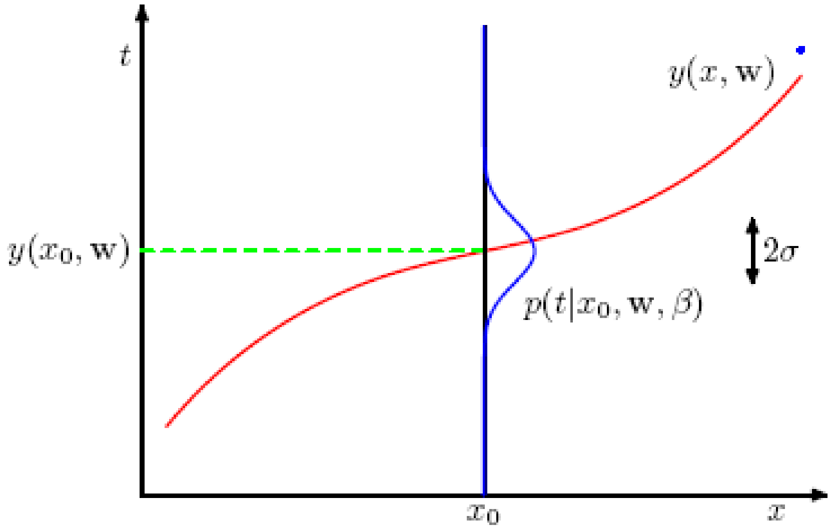
- What you have known: SL takes many (x,t) as inputs to train a learner f(x), then apply it to unseen x_k and predict it as $f(x_k)$
- For example (X is 3 dimensional):
  - Training { ([1,2,3], 0.1), ([2,3,4],0.2), ([3,4,5], 0.5)…}
  - Testing: [2,4,5] → 0.7
- However, uncertainty exist in the real world, therefore an error distribution (e.g. Gaussian) is usually added: t=f(x)+error. That says, it is possible to generate different results for same inputs, for example:
  - Training {([1,2,3],0.1), ([1,2,3],0.2),([1,2,3],0.1)…}
  - Testing: [1,2,3]=?

The Probabilistic Form of t

- The output t is a distribution caused by the error (assuming Gaussian) term:

  p(t|x,**W**,β)= N(t|y(x,**W**), β^{-1}), β is called a **precision parameter** which equals the inverse of the variance $1/\sigma^2$.

-



The SL process under probability

- Given training data {**X,T**}, we want to determine the unknown parameter **W** and β so we will know the distribution of y.
- Assuming we observed N data points, then

$$p(T|X,W,\beta) = p(t_1|x_1,W,\beta) * p(t_2|x_2,W,\beta) \ldots * p(t_N|x_N,W,\beta)$$

$$= \prod_{n=1}^{N} \mathrm{N}(t_n \mid y(x_n,W),\beta^{-1}) \rightarrow \textit{likelihood} \text{ function}$$

$$\ln(p(T|X,W,\beta)) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,W)-t_n\}^2 + \frac{N}{2}(\ln\beta - \ln(2\pi)),$$

this is called log-likelihood function

Maximum Likelihood Estimation (MLE)

- Idea: trying to adjust the unknown parameters (i.e. W and β) to maximize the likelihood function or log-likelihood function

$$\ln(p(T|X,W,\beta)) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,W)-t_n\}^2 + \frac{N}{2}(\ln\beta - \ln(2\pi))$$

- Adjusting W to maximizing this log-likelihood function given Gaussian error function is equivalent to finding a W_{ML} that minimizing the mean-square error function

Maximum Likelihood Estimation for β

- First, we calculate W_{ML} that governs the mean of the distribution.
- Then we use W_{ML} in the likelihood function to determine the optimal β_{ML}

$$\frac{\partial \ln(p(T|X,W_{ML}, \beta))}{\partial \beta} = -\frac{1}{2}\sum_{n=1}^{N}\{y(x_n, W_{ML}) - t_n\}^2 + \frac{N}{2\beta} = 0$$

$$\Rightarrow \beta^{-1} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, W_{ML}) - t_n\}^2$$

A SL system using MLE

1. We first determine W as W_{ML} that minimizes the error function $\boxed{\frac{1}{2}\sum_{n=1}^{N}\{y(x_n,w)-t_n\}^2}$ → Tend to overfit
2. Using W_{ML} to find β as $\beta^{-1} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, W_{ML}) - t_n\}^2$
3. Prediction stage: Using W_{ML} and β to construct the distribution of t: $p(t|x,\mathbf{W},\beta) = N(t|y(x,W_{ML}), \beta_{ML}^{-1})$
4. Predict the value of an input x' by sampling t using the distribution in (3)

- The MLE approach consistently **underestimate the variance** of the data and can lead to **overfitting**

Bayesian Approach for Regression

- Why Bayesian Approach: some w’s are preferable than others
  - For example, the regularization prefers simple model (i.e. small w’s).
  - Consequently, p(w) cannot be treated as uniformly distributed

Bayes' Rule Review

$$P(W \mid T) = \frac{P(T \mid W) * P(W)}{P(T)}$$

$$P(W \mid X, T) = \frac{P(T \mid X, W) * P(W \mid X)}{P(T \mid X)}$$

$$P(W \mid X, T) \propto P(T \mid X, W) * P(W \mid X)$$

- P(W|X): prior probability
- P(Tl X,W): Likelihood probability (what MLE tries to optimize, argmax_w P(T|X,W))
- P(W|X,T) : posterior probability

Bayesian Curve Fitting

$$P(W \mid X, T) \propto P(T \mid X, W) * P(W \mid X)$$

- Likelihood probability (we have already done):

$$\ln(p(T|X,W,\beta)) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,W)-t_n\}^2 + \frac{N}{2}(\ln\beta - \ln(2\pi))$$

- Prior: Assuming independent of X, and is Gaussian with mean 0 and variance = 1/α

$$p(W \mid X) = (\frac{\alpha}{2\pi})^{\frac{M+1}{2}} e^{-\frac{\alpha}{2}w^T w}$$

- Then the log probability of posterior will be proportion to

$$-\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,W)-t_n\}^2 + \frac{N}{2}(\ln\beta - \ln(2\pi)) + \frac{M+1}{2}(\ln\alpha - \ln(2\pi)) - \frac{\alpha}{2}w^T w$$

Maximum Posterior Estimation (MAP)

$$-\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,W)-t_n\}^2+\frac{N}{2}(\ln\beta-\ln(2\pi))+\frac{M+1}{2}(\ln\alpha-\ln(2\pi))-\frac{\alpha}{2}w^Tw$$

- The best parameter set should maximize posterior probability instead of the likelihood probability.
- The MAP solution for the Gaussian noise and Gaussian Prior is to find a W that minimize

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,W)-t_n\}^2+\frac{\alpha}{2}w^Tw$$

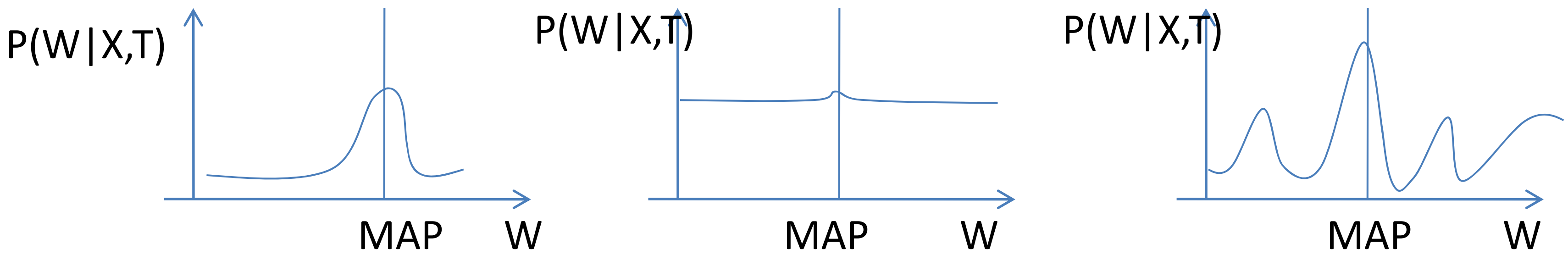
- Maximizing the posterior distribution is equivalent to minimizing the regularized sum-of-squares error function with the regularization parameter $\lambda=\alpha/\beta$

What we have discussed so far

1. Learning Phrase (MLE or MAP):
   – Finding W_{ML} that maximizes the likelihood function $p(T|X,W)$ ← → Finding W that minimize the square error of loss function, **or**
   – Finding W_{MAP} that maximizes the posterior function $P(W|T,X)$ ← → Finding W that minimize the regularized sum-of-squares loss function
2. Inference Phrase:
   – When an new x' comes in, using the determined W to predict the output y'

Potential Issues

- The problem of MLE: overfitting
- The problem of MAP: lose information



- Since in MAP we have learned P(W|X,T), why not using total probability theory

$$p(t \mid x, X, T) = \int_w p(t \mid x, W) * p(W \mid X, T) dW$$

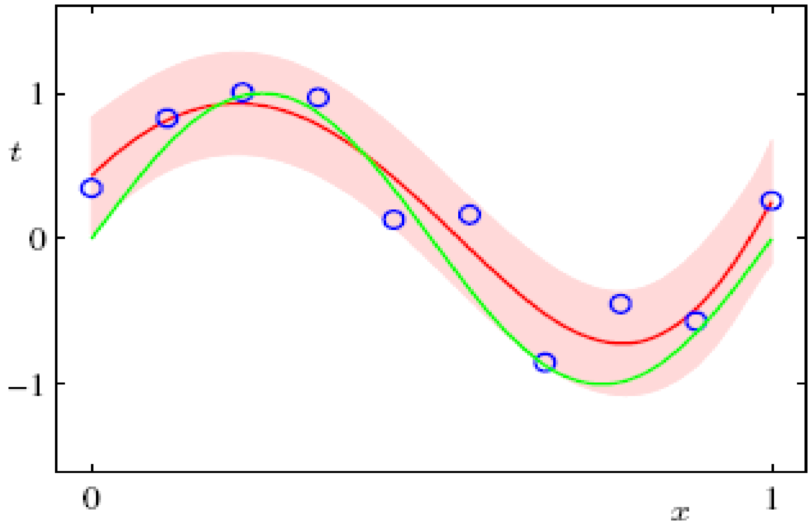
$$where \ p(t \mid x, w) = N(t \mid y(x, W), \beta^{-1})$$

The **predictive distribution** of t

$$p(t \mid x, X, T) = \int_w p(t \mid x, W) * p(W \mid X, T) dW$$

$$where\ p(t \mid x, w) = N(t \mid y(x, W), \beta^{-1})$$

- It can be proved that when the posterior and p(t|x,W) are Gaussian, then the predictive distribution p(t|x,X,T) is also Gaussian with mean m(x) and variance $s^2(x)$



y and

$$m(x) = \beta \phi(x)^{\mathrm{T}} \mathbf{S} \sum_{n=1}^{N} \phi(x_n) t_n$$

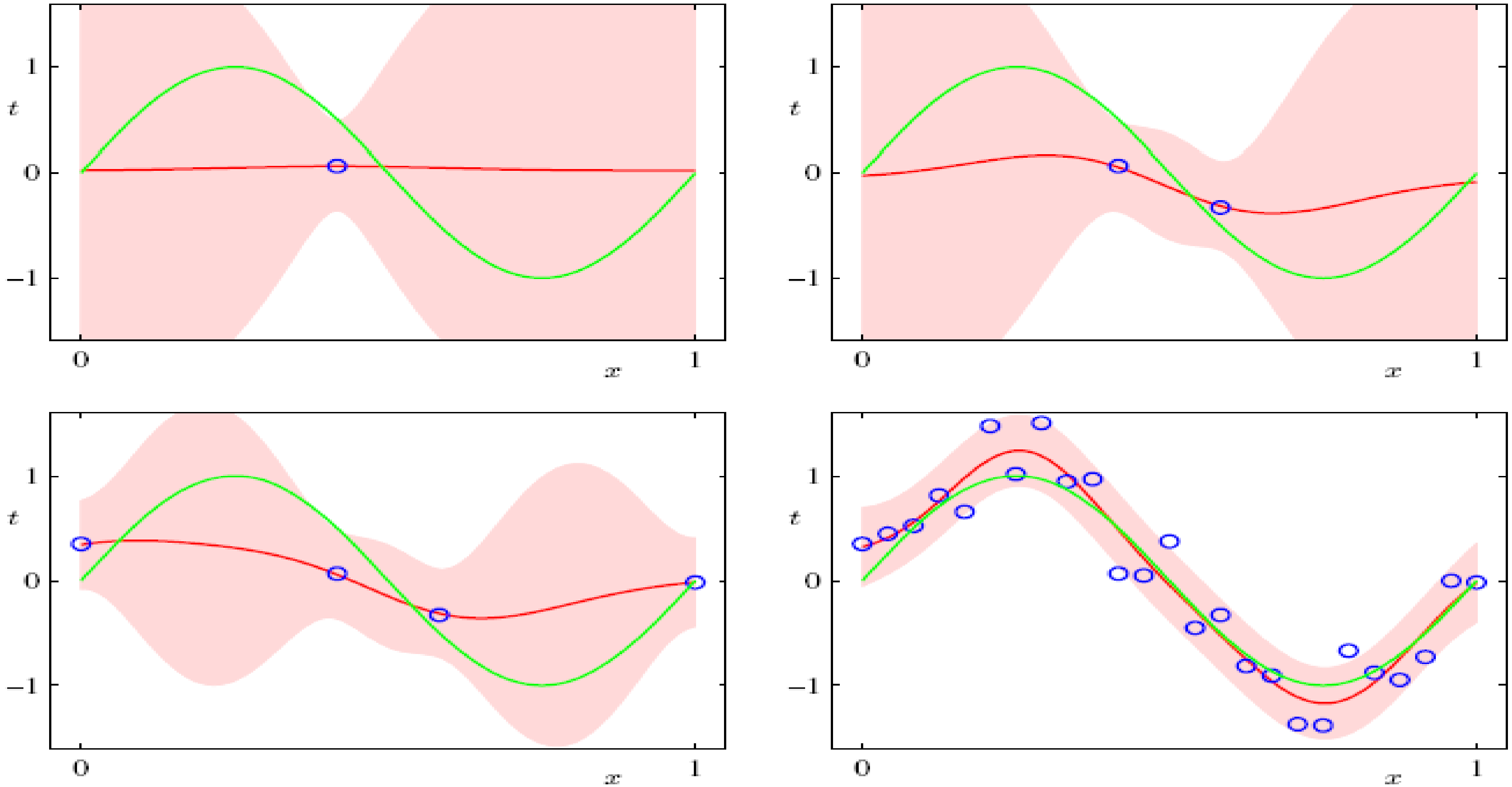
$$s^2(x) = \beta^{-1} + \phi(x)^{\mathrm{T}} \mathbf{S} \phi(x).$$

: given by

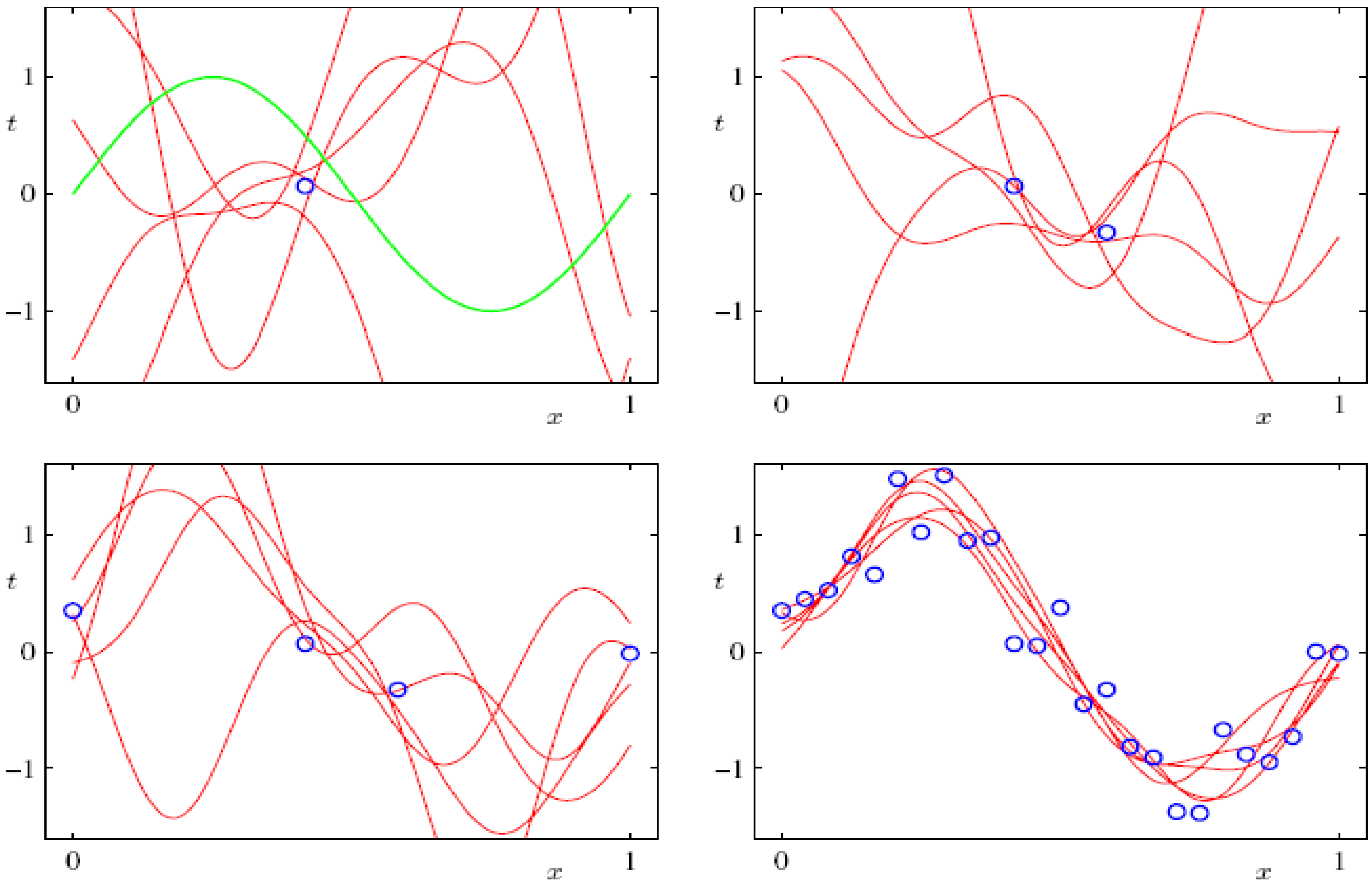
$$\mathbf{S}^{-1} = \alpha \mathbf{I} + \beta \sum_{n=1}^{N} \phi(x_n) \phi(x)^{\mathrm{T}}$$

Example of predictive distribution

- Green: true function. Red line: mean of the predicted function . Red zone: one variance from mean.



Y(x,w) from sampling posterior distributions over w



The benefit of Statistical Learning

- Because it can not only produce the output, but the distribution of the outputs.
  - The distribution tells us more about the data, including how confident the system has about its prediction.
  - It can be used to generate the dataset.

We have talked about Regression, so how about Classification?

Two Classification Strategies

Strategy 1: two-stage methods

Classification can be broken down into two stages

- Inference stage: for each C_k, using its own training data to learn a model for $p(C_k|X)$
- Decision stage: Use $p(C_k|X)$ and the loss matrix to make optimal class assignment

Strategy 2: One-shot methods (or Discriminant model)

Using all training data to learn a function that directly maps inputs x into the output class

Two Models for Strategy 1 (1/2)

- Model 1: Generative Model
  - First solve the inference problem of determining $p(x|C_k)$ for **each class C_k individually.**
  - Separately infer the prior class probabilities $p(C_k)$.
  - Use Bayes' theorem to find the posterior class probabilities $p(C_k|x)$
    $$p(C_k \mid x) = \frac{p(x \mid C_k)\, p(C_k)}{p(x)}$$
  - note that the denominator can be generated as $p(x)=\Sigma\, p(x|C_k)p(C_k)$
  - Finally use $p(C_k|x)$ and decision theory to find the best class assignment.
- This is called generative model since we can learn $p(x)$ and $p(C_k,x)$

Two Approaches for Strategy 1 (2/2)

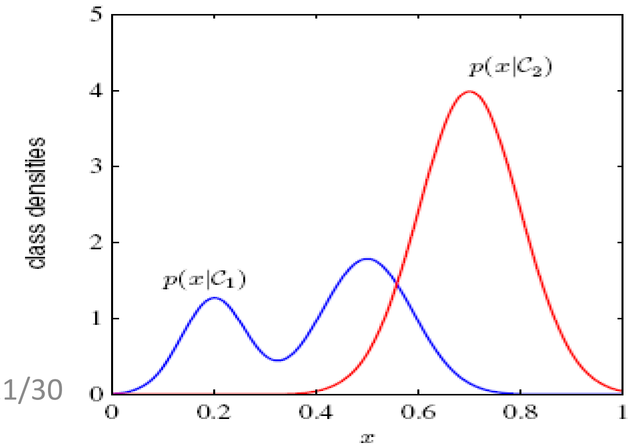
- Model 2: Discriminative Model
  - Directly learn $p(C_k|x)$ from data (know nothing about $p(x|C_k)$, and $p(x)$)
  - Logistic regression is a typical example.

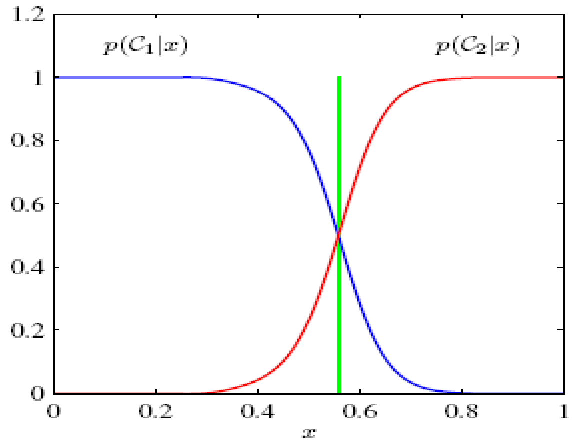
Classification Models

- Generative Model: learning $P(C_k|X)$ using Bayes Rule
  - First solve the inference problem of determining $p(x|C_k)$ and $p(C_k)$ for each class C_k individually**.**
  - Use Bayes' rule to find the posterior class probabilities $p(C_k|x)$
- Discriminative Model: learning $P(C_k|X)$ directly from data
  - Then apply decision theory to decide which C is the best assignment for x (e.g. Logistic Regression)
- Discriminant Model: Learn a function that directly maps inputs x into the output class
  - Linear discriminant function: learning linear functions to separate the classes
    - Least Squares
    - Fisher's linear discriminant
    - Perceptron Algorithm

Generative vs. Discriminative Model

- Generative model
  - Pros: P(x) can be used to generate samples of inputs, which is useful for knowledge discovery & data mining (e.g. outlier detection and novelty detection).
  - Cons: very demanding since it has to find the joint distribution of Ck and x. Need a lot training data.
- Discriminative Model
  - Pros: can be learned with fewer data
  - Cons: cannot learn the detail structure of the data





Generative vs. Discriminant Model (1/3)

- A discriminant approach learns a discriminant function and use it for decision making. It does not learn $P(C_k|x)$.
- However, $P(C_k|x)$ is useful in many aspects
  1. It can be combined with the cost function to produce the final decision. If the cost function changes, we don't need to re-train the whole model as a discriminant model does.
  2. It can be used to determine the reject region.
     - $P(C_{HT}|x)= 0.1$, $P(C_{PJ}|x)= 0.05$
     - $P(C_{HT}|x)= 0.7$, $P(C_{PJ}|x)= 0.8$

Generative vs. Discriminant Model (2/3)

- Generative Model takes care of the class prior P(y) explicitly.
  - E.g.: in cancer prediction, only a small amount of data (e.g. 0.1 %) are positive.
  - A normal classifier will guess negative and receive 99.9% accuracy.
  - Using $P(C_k|x)$ and $P(C_k)$ allow us to ignore the inference from the prior during learning.

Generative vs. Discriminant Model (3/3)

- Generative model are better in terms of combining several models:
  - Assuming in the previous example, we have two types of information for each photo:
    - The image features (X_i)
    - The social information (X_s)
- It might be more effective and meaningful to build separate models $P(C_k|X_i)$, $P(C_k|X_s)$ for these two sets of features.
- Generative allows us to combine these models as: $P(C_k|X_i,X_s)$

$$p(C_k \mid x_i, x_s) \propto P(x_i, x_s \mid C_k)P(C_k) \propto$$

Naïve bayes assumption

$$P(x_i \mid C_k)P(x_s \mid C_k)P(C_k) \propto \frac{P(C_k \mid x_i)P(C_k \mid x_s)}{P(C_k)}$$

Naïve Baye Assumption

- Recall in Bayesian Setup, we have $p(C_k \mid x) = \dfrac{p(x \mid C_k)\, p(C_k)}{p(x)}$
- If we assume features of an instance are independent **given the class** (***conditionally independent***).

$$P(X \mid C) = P(X_1, X_2, \cdots X_n \mid C) = \prod_{i=1}^{n} P(X_i \mid C)$$

- Therefore, we then only need to know P(X_i |C) for each possible pair of a feature-value and class.
- If C and all X_i are binary, this requires specifying only 2*n* parameters:
  - P(X_i=true | C=true) and P(X_i=true | C=false) for each X_i
  - P(X_i=false | C) = 1 – P(X_i=true | C)
- Compared to specifying 2^n parameters without any independence assumptions.

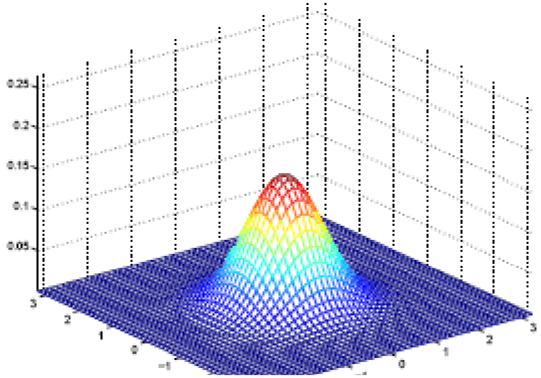
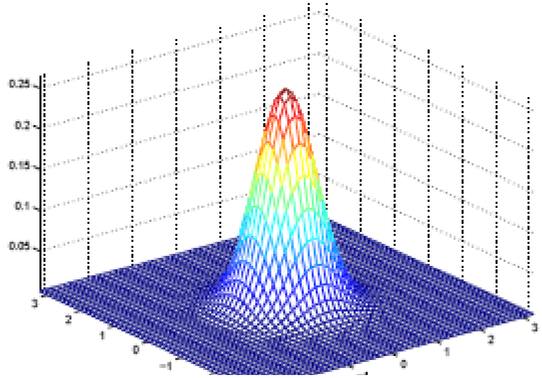
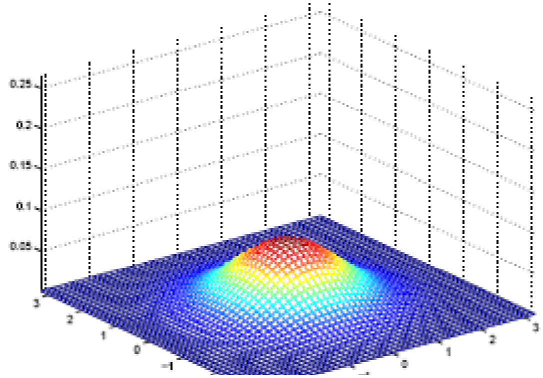
Gaussian Discriminant Analysis (GDA)

- This is another generative model.
- GDA assumes p(x|y) is distributed according to a Multivariate Normal Distribution (MND).
- An MND in n-dimensions is parameterized by a **mean vector** $\mu \in R^n$ and a covariance matrix $\Sigma \in R^{nxn}$, also written as $N(\mu, \Sigma)$. Its density is:

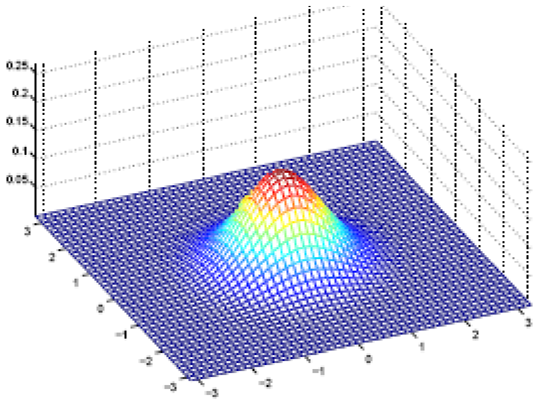
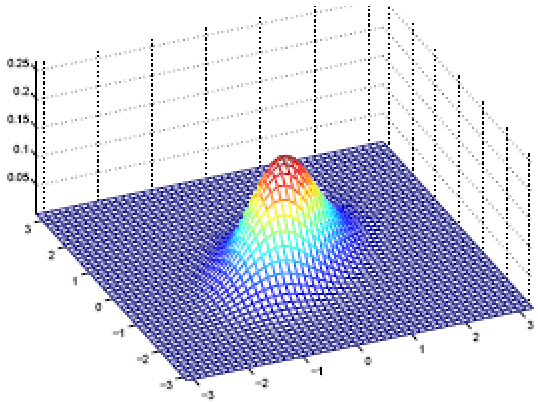
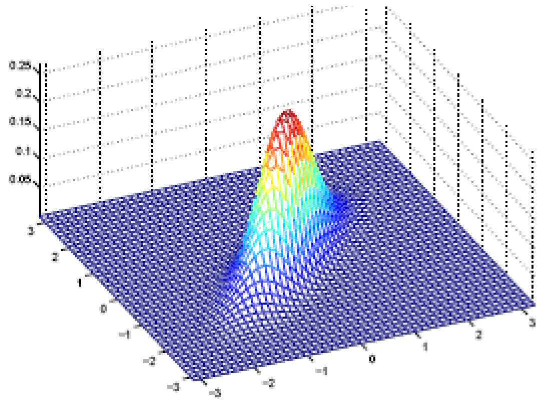
$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Examples for 2-D Multivariate Normal Distribution

- $\Sigma = I$ $\qquad\qquad$ $\Sigma = 0.6I$ $\qquad\qquad$ $\Sigma = 2I$

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}.$$

The Model for GDA (1/2)

- p(x|y) is MND, p(y=0)=Φ, p(y=1)=1-Φ

$$
\begin{aligned}
p(x|y=0) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right) \\
p(x|y=1) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)
\end{aligned}
$$

(assuming different y shares the same Σ)

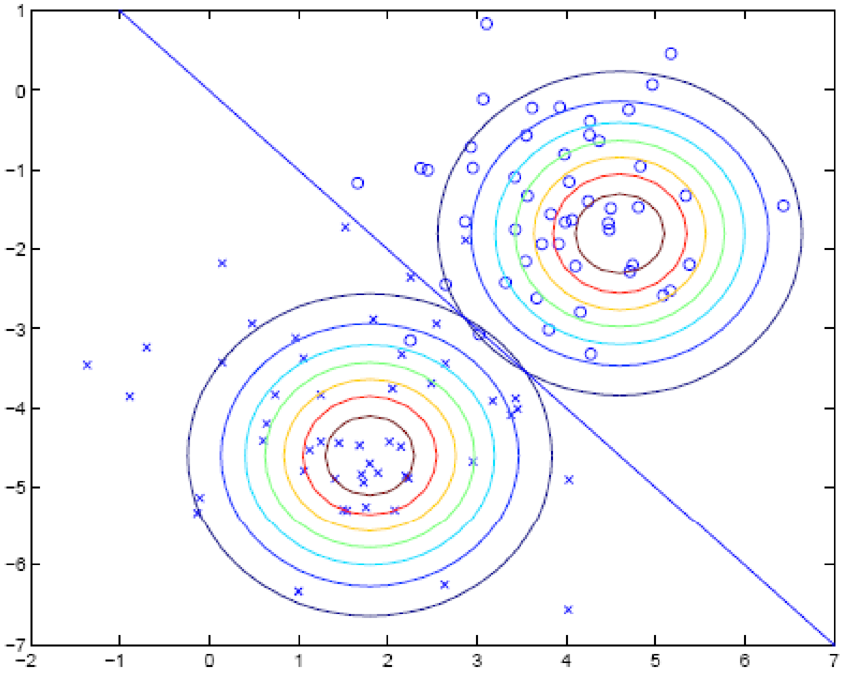
- The log-likelyhood of the data is

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

The Model for GDA (2/2)

- Using maximum likelihood estimate (MLE), we can obtain

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



Discussion: GDA vs. Logistic Regression

- In GDA, $p(y|x)$ is of the form $1/(1+\exp(-\theta^T x))$, where θ is a function of φ, Σ, μ.
  - This is exactly the form of logistic regression to model $p(y|x)$. That says, if $p(x|y)$ is multivariate gaussian, then $p(y|x)$ follows a logistic function.
  - However, the converse is not true. This implies that GDA makes **stronger** modeling assumptions about the data than LR does.
- Training on the same dataset, these two algorithms will produce different decision boundaries.
  - If $p(x|y)$ is indeed Gaussian, then GDA will get better results. That says, if x is some sort of the mean value of something whose size is not small, then based on central-limit-theorem, GDA should perform very well.
  - If $p(x|y=1)$ and $p(x|y=0)$ are both Poisson, then $P(y|x)$ will be logistic. In this case, LR can work better than GDA.
  - If we are sure the data is non-Gaussian, we should use LR than GDA