

# Machine Learning: A Statistics and Optimization Perspective

Nan Ye

Mathematical Sciences School  
Queensland University of Technology

# What is Machine Learning?

# Machine Learning

- Machine learning turns data into insight, predictions and/or decisions.
- Numerous applications in diverse areas, including natural language processing, computer vision, recommender systems, medical diagnosis.

# A Much Sought-after Technology

## NYU “Deep Learning” Professor LeCun Will Head Facebook’s New Artificial Intelligence Lab

Posted Dec 9, 2013 by [Josh Constine \(@joshconstine\)](#)

863

SHARES



Yann LeCun

Timeline

About

By teaching a corn  
hopes to better u  
do too. So today t  
announced that o  
deep learning anc  
scientists, NYU's F  
lead its new artifici

## Baidu Hires Coursera Founder Andrew Ng to Start Massive Research Lab



## GOOGLE HIRES BRAINS THAT HELPED SUPERCHARGE MACHINE LEARNING



Geoffrey Hinton (right) Alex Krizhevsky, and Ilya Sutskever (left) will do machine learning

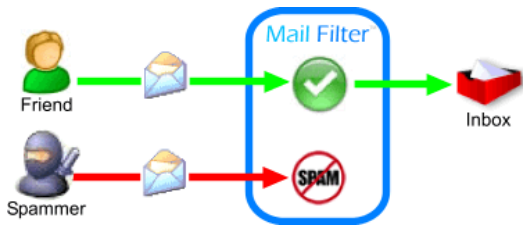
# Enabled Applications



Make reminders by talking to your phone.



Tell the car where you want to go to and the car takes you there.



Check your emails, see some spams in your inbox, mark them as spams, and similar spams will not show up.

## Recommended



### BACK TO THE FUTURE HOVER BOARD | SKATE

Braille Skateboarding  
522,458 views • 4 days ago



### BOY KIDNAPPED?!

RoccoPiazzaVlogs  
680,245 views • 1 week ago



### 20 Hidden Mistakes In Kids Movies That You Never

Screen Rant  
2,013,454 views • 4 days ago



### Designer- Panda (OFFICIAL SONG) Prod. By: Menace

Designer  
78,752,672 views • 4 months ago



### GIRL HAS PERFECT 360 FLIPS!! (Tre flip) | Skate

GarrettDavidGinner  
301,079 views • 2 weeks ago



### INSANE 7 YEAR OLD HOVERBOARD TRICKS AT

Tanner Fox  
12,651,467 views • 6 months ago

Video recommendations.





Play Go against the computer.

# Tutorial Objective

Essentials for crafting basic machine learning systems.

## **Formulate applications as machine learning problems**

*Classification, regression, density estimation, clustering...*

## **Understand and apply basic learning algorithms**

*Least squares regression, logistic regression, support vector machines, K-means,...*

## **Theoretical understanding**

*Position and compare the problems and algorithms in a unifying statistical framework*

## **Have fun...**

# Outline

- A statistics and optimization perspective
- Statistical learning theory
- Regression
- Model selection
- Classification
- Clustering

# Hands-on

- An exercise on using WEKA.

WEKA	Java	<a href="http://www.cs.waikato.ac.nz/ml/weka/">http://www.cs.waikato.ac.nz/ml/weka/</a>
H2O	Java	<a href="http://www.h2o.ai/">http://www.h2o.ai/</a>
scikit-learn	python	<a href="http://scikit-learn.org/">http://scikit-learn.org/</a>
CRAN	R	<a href="https://cran.r-project.org/web/views/MachineLearning.html">https://cran.r-project.org/web/views/MachineLearning.html</a>

- Some technical details are left as exercises. These are tagged with *(verify)*.

# A Statistics and Optimization Perspective

## Illustrations

- Learning a binomial distribution
- Learning a Gaussian distribution

# Learning a Binomial Distribution

I pick a coin with the probability of heads being  $\theta$ . I flip it 100 times for you and you see a dataset  $D$  of 70 heads and 30 tails, can you learn  $\theta$ ?

# Learning a Binomial Distribution

I pick a coin with the probability of heads being  $\theta$ . I flip it 100 times for you and you see a dataset  $D$  of 70 heads and 30 tails, can you learn  $\theta$ ?

## Maximum likelihood estimation

The likelihood of  $\theta$  is

$$P(D \mid \theta) = \theta^{70}(1 - \theta)^{30}.$$

Learning  $\theta$  is an optimization problem.

$$\begin{aligned}\theta_{ml} &= \arg \max_{\theta} P(D \mid \theta) \\ &= \arg \max_{\theta} \ln P(D \mid \theta) \\ &= \arg \max_{\theta} (70 \ln \theta + 30 \ln(1 - \theta)).\end{aligned}$$

$$\theta_{ml} = \arg \max_{\theta} (70 \ln \theta + 30 \ln(1 - \theta)).$$

Set derivative of log-likelihood to 0,

$$\frac{70}{\theta} - \frac{30}{1 - \theta} = 0,$$

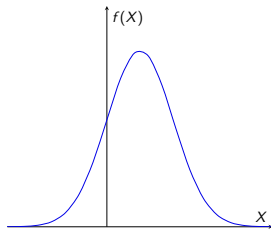
we have

$$\theta_{ml} = 70/(70 + 30).$$



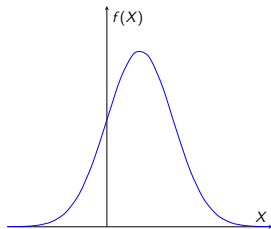
# Learning a Gaussian distribution

I pick a Gaussian  $N(\mu, \sigma^2)$  and give you a bunch of data  $D = \{x_1, \dots, x_n\}$  independently drawn from it. Can you learn  $\mu$  and  $\sigma$ .



# Learning a Gaussian distribution

I pick a Gaussian  $N(\mu, \sigma^2)$  and give you a bunch of data  $D = \{x_1, \dots, x_n\}$  independently drawn from it. Can you learn  $\mu$  and  $\sigma$ .



$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

## Maximum likelihood estimation

$$\begin{aligned}\ln P(D \mid \mu, \sigma) \\&= \ln \left( \frac{1}{\sigma\sqrt{2\pi}} \right)^n \exp \left( - \sum_i \frac{(x_i - \mu)^2}{2\sigma^2} \right) \\&= -n \ln(\sigma\sqrt{2\pi}) - \sum_i \frac{(x_i - \mu)^2}{2\sigma^2}.\end{aligned}$$

Set derivative w.r.t.  $\mu$  to 0,

$$- \sum_i \frac{x_i - \mu}{\sigma^2} = 0 \quad \Rightarrow \quad \mu_{ml} = \frac{1}{n} \sum_i x_i$$

Set derivative w.r.t.  $\sigma$  to 0,

$$-\frac{n}{\sigma} + \frac{(x_i - \mu)^2}{\sigma^3} = 0 \quad \Rightarrow \quad \sigma_{ml}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_{ml})^2.$$

# What You Need to Know...

Learning is...

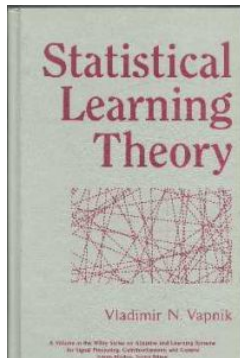
- Collect some data, e.g. coin flips.
- Choose a hypothesis class, e.g. binomial distribution.
- Choose a loss function, e.g. negative log-likelihood.
- Choose an optimization procedure, e.g. set derivative to 0.
- Have fun...

Statistics and optimization provide powerful tools for formulating and solving machine learning problems.

# Statistical Learning Theory

- The framework
- Applications in classification, regression, and density estimation
- Does empirical risk minimization work?

*There is nothing more practical than a good theory.*  
Kurt Lewin



*...at least in the problems of statistical inference.*  
Vladimir Vapnik

# Learning...

- H. Simon: Any process by which a system improves its performance.
- M. Minsky: Learning is making useful changes in our minds.
- R. Michalsky: Learning is constructing or modifying representations of what is being experienced.
- L. Valiant: Learning is the process of knowledge acquisition in the absence of explicit programming.

# A Probabilistic Framework

## Data

Training examples  $z_1, \dots, z_n$  are i.i.d. drawn from a *fixed* but *unknown* distribution  $P(Z)$  on  $\mathcal{Z}$ .

*e.g. outcomes of coin flips.*

## Hypothesis space $\mathcal{H}$

*e.g. head probability  $\theta \in [0, 1]$ .*

## Loss function

$L(z, h)$  measures the penalty for hypothesis  $h$  on example  $z$ .

$$\text{e.g. log-loss } L(z, \theta) = -\ln P(z \mid \theta) = \begin{cases} -\ln(\theta), & z = H, \\ -\ln(1 - \theta), & z = T. \end{cases}$$



## Expected risk

- The expected risk of  $h$  is  $\mathbb{E}(L(Z, h))$ .
- We want to find the hypothesis with minimum expected risk,

$$\arg \min_{h \in \mathcal{H}} \mathbb{E}(L(Z, h)).$$

## Empirical risk minimization (ERM)

Minimize empirical risk  $R_n(h) \stackrel{\text{def}}{=} \frac{1}{n} \sum_i L(z_i, h)$  over  $h \in \mathcal{H}$ .

*e.g. choose  $\theta$  to minimize  $-70 \ln \theta - 30 \ln(1 - \theta)$ .*

This provides a unified formulation for many machine learning problems, which differ in

- the data domain  $\mathcal{Z}$ ,
- the choice of the hypothesis space  $\mathcal{H}$ , and
- the choice of loss function  $L$ .

Most algorithms that we see later can be seen as special cases of ERM.

# Classification

predict a discrete class

**Digit recognition:** image to  $\{0, 1, \dots, 9\}$ .



**Spam filter:** email to  $\{\text{spam}, \text{not spam}\}$ .



Given  $D = \{(x_1, y_1), \dots, (x_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ , find a classifier  $f$  that maps an input  $x \in \mathcal{X}$  to a *class*  $y \in \mathcal{Y}$ .

We usually use the 0/1 loss

$$L((x, y), h) = \mathbb{I}(h(x) \neq y) = \begin{cases} 1, & h(x) \neq y, \\ 0, & h(x) = y. \end{cases}$$

ERM chooses the classifier with minimum classification error

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_i \mathbb{I}(h(x_i) \neq y_i).$$

# Regression

predict a numerical value

**Stock market prediction:** predict stock price using recent trading data



Given  $D = \{(x_1, y_1), \dots, (x_n, y_n)\} \subseteq \mathcal{X} \times \mathbb{R}$ , find a function  $f$  that maps an input  $x \in \mathcal{X}$  to a *value*  $y \in \mathbb{R}$ .

We usually use the quadratic loss

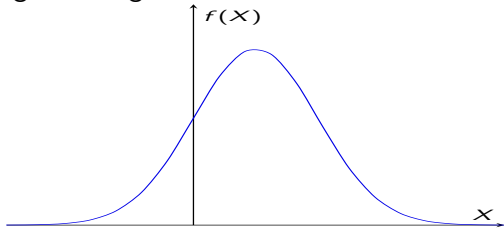
$$L((x, y), h) = (y - h(x))^2.$$

ERM is often called the method of least squares in this case

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_i (y_i - h(x_i))^2.$$

# Density Estimation

E.g. learning a binomial distribution, or a Gaussian distribution.



We often use the log-loss

$$L(x, h) = -\ln p(x | h).$$

ERM is MLE in this case.

# Does ERM Work?

## Estimation error

- How does the empirically best hypothesis  $h_n = \arg \min_{h \in \mathcal{H}} R_n(h)$  compare with the best in the hypothesis space? Specifically, how large is the estimation error  $R(h_n) - \inf_{h \in \mathcal{H}} R(h)$ ?
- **Consistency:** Does  $R(h_n)$  converge to  $\inf_{h \in \mathcal{H}} R(h)$  as  $n \rightarrow \infty$ ?



# Does ERM Work?

## Estimation error

- How does the empirically best hypothesis  $h_n = \arg \min_{h \in \mathcal{H}} R_n(h)$  compare with the best in the hypothesis space? Specifically, how large is the estimation error  $R(h_n) - \inf_{h \in \mathcal{H}} R(h)$ ?
- **Consistency:** Does  $R(h_n)$  converge to  $\inf_{h \in \mathcal{H}} R(h)$  as  $n \rightarrow \infty$ ?

If  $|\mathcal{H}|$  is finite, ERM is likely to pick the function with minimal expected risk when  $n$  is *large*, because then  $R_n(h)$  is close to  $R(h)$  for all  $h \in \mathcal{H}$ .

If  $|\mathcal{H}|$  is infinite, we can still show that ERM is likely to choose a near-optimal hypothesis if  $\mathcal{H}$  has finite complexity (such as VC-dimension).

## Approximation error

How good is the best hypothesis in  $\mathcal{H}$ ? That is, how large is the approximation error  $\inf_{h \in \mathcal{H}} R(h) - \inf_h R(h)$ ?

## Approximation error

How good is the best hypothesis in  $\mathcal{H}$ ? That is, how large is the approximation error  $\inf_{h \in \mathcal{H}} R(h) - \inf_h R(h)$ ?

Trade-off between estimation error and approximation error:

- Larger hypothesis space implies smaller approximation error, but larger estimation error.
- Smaller hypothesis space implies larger approximation error, but smaller estimation error.

## Optimization error

Is the optimization algorithm computing the empirically best hypothesis exact?

## Optimization error

Is the optimization algorithm computing the empirically best hypothesis exact?

While ERM can be efficiently implemented in many cases, there are also computationally intractable cases, and efficient approximations are sought. The performance gap between the sub-optimal hypothesis and the empirically best hypothesis is the optimization error.

# What You Need to Know...

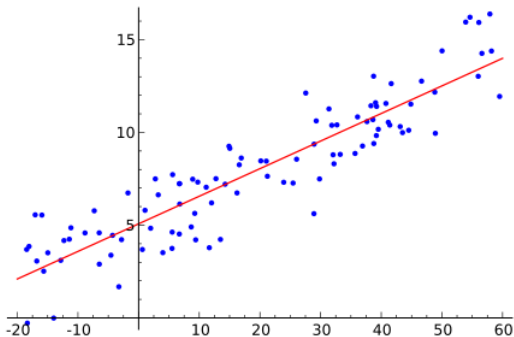
Recognise machine learning problems as special cases of the general statistical learning problem.

Understand that the performance of ERM depends on the approximation error, estimation error and optimization error.

# Regression

- Ordinary least squares
- Ridge regression
- Basis function method
- Regression function
- Nearest neighbor regression
- Kernel regression
- Classification as regression

# Ordinary Least Squares



Find a best fitting hyperplane for  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ .



OLS finds a hyperplane minimizing the sum of squared errors

$$\beta_n = \arg \min_{\beta \in \mathbb{R}^d} \sum_{i=1}^n (\mathbf{x}_i^T \beta - y_i)^2.$$

## A special case of function learning using ERM

- The input set is  $\mathcal{X} = \mathbb{R}^d$ , and the output set is  $\mathcal{Y} = \mathbb{R}$ .
- The hypothesis space are hyperplanes  $\mathcal{H} = \{\mathbf{x}^T \beta : \beta \in \mathbb{R}^d\}$ .
- Quadratic loss is used, as typically in regression.

**Empirically best hyperplane.** The solution to OLS is

$$\beta_n = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y},$$

where  $\mathbf{X}$  is the  $n \times d$  matrix with  $\mathbf{x}_i$  as the  $i$ -th row, and  $\mathbf{y} = (y_1, \dots, y_n)^T$ .

*The formula holds when  $\mathbf{X}^T \mathbf{X}$  is non-singular. When  $\mathbf{X}^T \mathbf{X}$  is singular, there are infinitely many possible values for  $\beta_n$ . They can be obtained by solving the linear systems  $(\mathbf{X}^T \mathbf{X})\beta = \mathbf{X}^T \mathbf{y}$ .*

*Proof.* The empirical risk is (ignoring a factor of  $\frac{1}{n}$ )

$$R_n(\beta) = \sum_{i=1}^n (\mathbf{x}_i^T \beta - y_i)^2 = \|\mathbf{X}\beta - \mathbf{y}\|_2^2.$$

Set the gradient of  $R_n$  to 0

$$\nabla R_n = 2\mathbf{X}^T(\mathbf{X}\beta - \mathbf{y}) = 0, \text{ (verify)}$$

we have

$$\beta_n = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}.$$

**Optimal hyperplane.** The hyperplane  $\beta^* = \mathbb{E}(XX^T)^{-1} \mathbb{E}(XY)$  minimizes the expected quadratic loss among all hyperplanes.

**Optimal hyperplane.** The hyperplane  $\beta^* = \mathbb{E}(XX^T)^{-1} \mathbb{E}(XY)$  minimizes the expected quadratic loss among all hyperplanes.

*Proof.* The expected quadratic loss of a hyperplane  $\beta$  is

$$\begin{aligned} R(\beta) &= \mathbb{E}((\beta^T X - Y)^2) \\ &= \mathbb{E}(\beta^T X X^T \beta - 2\beta^T X Y + Y^2) \\ &= \beta^T \mathbb{E}(X X^T) \beta - 2\beta^T \mathbb{E}(X Y) + \mathbb{E}(Y^2). \end{aligned}$$

Set the gradient of  $R$  to 0, we have

$$\nabla R(\beta) = 2 \mathbb{E}(X X^T) \beta - 2 \mathbb{E}(X Y) = 0 \quad \Rightarrow \quad \beta^* = \mathbb{E}(X X^T)^{-1} \mathbb{E}(X Y).$$

**Consistency.** We can show that least squares linear regression is consistent, that is,  $R(\beta_n) \xrightarrow{P} R(\beta^*)$ , by using the law of large numbers.

# Least Squares as MLE

- Consider the class of conditional distributions  $\{p_\beta(Y|X) : \beta \in \mathbb{R}^d\}$ , where

$$p_\beta(Y | X = \mathbf{x}) = N(Y; \mathbf{x}^T \beta, \sigma) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}\sigma} e^{-(Y - \mathbf{x}^T \beta)^2 / 2\sigma^2},$$

with  $\sigma$  being a constant.

- The (conditional) likelihood of  $\beta$  is

$$L_n(\beta) = p_\beta(y_1 | \mathbf{x}_1) \dots p_\beta(y_n | \mathbf{x}_n).$$

- Maximizing the likelihood  $L_n(\beta)$  gives the same  $\beta_n$  as given by the method of least squares. (verify)

# Ridge Regression

When collinearity is present, the matrix  $\mathbf{X}^T \mathbf{X}$  may be singular or close to singular, making the solution unreliable.

## Ridge regression

We add a *regularizer*  $\lambda \|\beta\|_2^2$  to OLS objective, where  $\lambda > 0$  is a fixed constant.

$$\beta_n = \arg \min_{\beta \in \mathbb{R}^d} \left( \sum_{i=1}^n (\mathbf{x}_i^T \beta - y_i)^2 + \overbrace{\lambda \|\beta\|_2^2}^{\text{quadratic}/\ell_2 \text{ regularizer}} \right).$$

## Empirically optimal hyperplane

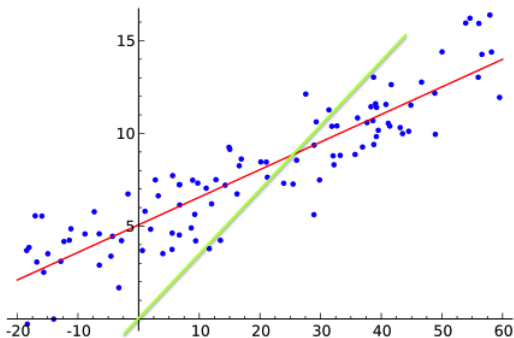
$$\beta_n = (\lambda I + \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}. \text{ (verify)}$$

The matrix  $\lambda I + \mathbf{X}^T \mathbf{X}$  is non-singular (verify), and thus there is always a unique solution.



# Regression with a Bias

So far we have only considered hyperplanes of the form  $y = \mathbf{x}^T \beta$ , which passes through the origin (green line).



Considering hyperplanes with a bias term, that is, hyperplanes of the form  $y = \mathbf{x}^T \beta + b$  is more useful (red line).

OLS with a bias solves

$$(b_n, \beta_n) = \arg \min_{b \in \mathbb{R}, \beta \in \mathbb{R}^d} \sum_{i=1}^n (\mathbf{x}_i^T \beta + \overbrace{b}^{\text{bias}} - y_i)^2.$$

OLS with a bias solves

$$(b_n, \beta_n) = \arg \min_{b \in \mathbb{R}, \beta \in \mathbb{R}^d} \sum_{i=1}^n (\mathbf{x}_i^T \beta + \overbrace{b}^{\text{bias}} - y_i)^2.$$

*Solution.* Reduce it to regression without a bias term by replacing  $\mathbf{x}^T \beta + b$  with  $(1 \quad \mathbf{x}^T) \begin{pmatrix} b \\ \beta \end{pmatrix}$ .

Ridge regression with a bias solves

$$(b_n, \beta_n) = \arg \min_{b \in \mathbb{R}, \beta \in \mathbb{R}^d} \left( \sum_{i=1}^n (\mathbf{x}_i^T \beta + b - y_i)^2 + \lambda \|\beta\|_2^2 \right).$$

Ridge regression with a bias solves

$$(b_n, \beta_n) = \arg \min_{b \in \mathbb{R}, \beta \in \mathbb{R}^d} \left( \sum_{i=1}^n (\mathbf{x}_i^T \beta + b - y_i)^2 + \lambda \|\beta\|_2^2 \right).$$

*Solution.* Reduce it to ridge regression without a bias term as follows.

Let  $\hat{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$ , and  $\hat{y}_i = y_i - \bar{y}$ , where  $\bar{\mathbf{x}} = \sum_{i=1}^n \mathbf{x}_i / n$ , and  $\bar{y} = \sum_{i=1}^n y_i / n$ , then

$$\beta_n = \arg \min_{\beta \in \mathbb{R}^d} \left( \sum_{i=1}^n (\hat{\mathbf{x}}_i^T \beta - \hat{y}_i)^2 + \lambda \|\beta\|_2^2 \right),$$

$$b_n = \bar{y} - \bar{\mathbf{x}}^T \beta_n. \text{ (verify)}$$

# Basis Function Method

We can use linear regression to learn complex regression functions

- Choose some *basis functions*  $g_1, \dots, g_k : \mathbb{R}^d \rightarrow \mathbb{R}$ .
- Transform each input  $\mathbf{x}$  to  $(g_1(\mathbf{x}), \dots, g_k(\mathbf{x}))$ .
- Perform linear regression on the transformed data.

# Basis Function Method

We can use linear regression to learn complex regression functions

- Choose some *basis functions*  $g_1, \dots, g_k : \mathbb{R}^d \rightarrow \mathbb{R}$ .
- Transform each input  $\mathbf{x}$  to  $(g_1(\mathbf{x}), \dots, g_k(\mathbf{x}))$ .
- Perform linear regression on the transformed data.

## Examples

- Linear regression: use basis functions  $g_1, \dots, g_d$  with  $g_i(\mathbf{x}) = x_i$ , and  $g_0(\mathbf{x}) = 1$ .
- Quadratic functions: use basis functions of the above form, together with basis functions of the form  $g_{ij}(\mathbf{x}) = x_i x_j$  for all  $1 \leq i \leq j \leq d$ .

# Regression Function

The minimizer of the expected quadratic loss is the regression function

$$h^*(x) = \mathbb{E}(Y \mid x).$$



# Regression Function

The minimizer of the expected quadratic loss is the regression function

$$h^*(x) = \mathbb{E}(Y \mid x).$$

*Proof.* The expected quadratic loss of a function  $h$  is

$$\mathbb{E}((h(X) - Y)^2) = \mathbb{E}_X(h(X)^2 - 2h(X)\mathbb{E}(Y \mid X) + \mathbb{E}(Y^2 \mid X)).$$

Hence we can set the value of  $h(x)$  independently for each  $x$  by choosing it to minimize the expression under expectation. This leads to  $h^*(x) = \mathbb{E}(Y \mid x)$ .

## $k$ nearest neighbor ( $k$ NN)

$k$ NN approximates the regression function using

$$h_n(\mathbf{x}) = \text{avg}(y_i \mid \mathbf{x}_i \in N_k(\mathbf{x})),$$

which is the average of the values of the set  $N_k(\mathbf{x})$  of the  $k$  nearest neighbors of  $\mathbf{x}$  in the training data.

## $k$ nearest neighbor ( $k$ NN)

$k$ NN approximates the regression function using

$$h_n(\mathbf{x}) = \text{avg}(y_i \mid \mathbf{x}_i \in N_k(\mathbf{x})),$$

which is the average of the values of the set  $N_k(\mathbf{x})$  of the  $k$  nearest neighbors of  $\mathbf{x}$  in the training data.

- Under mild conditions, as  $k \rightarrow \infty$  and  $n/k \rightarrow \infty$ ,  $h_n(x) \rightarrow h^*(x)$ , for any distribution  $P(X, Y)$ .
- (Curse of dimensionality) The number of samples required for accurate approximation is exponential in the dimension.
- $k$ NN is a *non-parametric* method, while linear regression is a *parametric* method.

# Kernel Regression

$$h_n(\mathbf{x}) = \sum_{i=1}^n K(\mathbf{x}, \mathbf{x}_i) y_i / \sum_{i=1}^n K(\mathbf{x}, \mathbf{x}_i),$$

where  $K(\mathbf{x}, \mathbf{x}')$  is a function measuring the similarity between  $\mathbf{x}$  and  $\mathbf{x}'$ , and is often called a kernel function.

# Kernel Regression

$$h_n(\mathbf{x}) = \sum_{i=1}^n K(\mathbf{x}, \mathbf{x}_i) y_i / \sum_{i=1}^n K(\mathbf{x}, \mathbf{x}_i),$$

where  $K(\mathbf{x}, \mathbf{x}')$  is a function measuring the similarity between  $\mathbf{x}$  and  $\mathbf{x}'$ , and is often called a kernel function.

## Example kernel functions

- Gaussian kernel  $K_\lambda(\mathbf{x}, \mathbf{x}') = \frac{1}{\lambda} \exp(-\frac{\|\mathbf{x}' - \mathbf{x}\|_2^2}{2\lambda})$ .
- $k$ NN kernel  $K_k(\mathbf{x}, \mathbf{x}') = \mathbb{I}(\|\mathbf{x}' - \mathbf{x}\| \leq \max_{\mathbf{x}'' \in N_k(\mathbf{x})} \|\mathbf{x}'' - \mathbf{x}\|)$ . Note that this kernel is data-dependent and non-symmetric.

# Binary Classification as Regression

- Label one class as  $-1$  and the other as  $+1$ .
- Fit a function  $f(\mathbf{x})$  using least squares regression.
- Given a test example  $\mathbf{x}$ , predict  $-1$  if  $f(\mathbf{x}) < 0$  and  $+1$  otherwise.

# What You Need to Know...

- Regression function.
- Parametric methods: Ordinary least squares, ridge regression, basis function method.
- Non-parametric methods:  $k$ NN, kernel regression.

# Model Selection



# Bias-Variance Tradeoff

The predicted value  $Y'$  at a fixed point  $\mathbf{x}$  can be considered a random function of the training set. Let  $Y$  be the true value at  $\mathbf{x}$ . The expected prediction error  $\mathbb{E}((Y' - Y)^2)$  is a property of the model class.

# Bias-Variance Tradeoff

The predicted value  $Y'$  at a fixed point  $\mathbf{x}$  can be considered a random function of the training set. Let  $Y$  be the true value at  $\mathbf{x}$ . The expected prediction error  $\mathbb{E}((Y' - Y)^2)$  is a property of the model class.

## Bias-variance decomposition

$$\overbrace{\mathbb{E}((Y' - Y)^2)}^{\text{expected prediction error}} = \overbrace{\mathbb{E}((Y' - \mathbb{E}(Y'))^2)}^{\text{variance}} + \overbrace{(\mathbb{E}(Y') - \mathbb{E}(Y))^2}^{\text{bias (squared)}} + \overbrace{\mathbb{E}((Y - \mathbb{E}(Y))^2)}^{\text{irreducible noise}},$$

*Proof.* Expand the RHS and simplify.

# Bias-Variance Tradeoff

The predicted value  $Y'$  at a fixed point  $\mathbf{x}$  can be considered a random function of the training set. Let  $Y$  be the true value at  $\mathbf{x}$ . The expected prediction error  $\mathbb{E}((Y' - Y)^2)$  is a property of the model class.

## Bias-variance decomposition

$$\overbrace{\mathbb{E}((Y' - Y)^2)}^{\text{expected prediction error}} = \overbrace{\mathbb{E}((Y' - \mathbb{E}(Y'))^2)}^{\text{variance}} + \overbrace{(\mathbb{E}(Y') - \mathbb{E}(Y))^2}^{\text{bias (squared)}} + \overbrace{\mathbb{E}((Y - \mathbb{E}(Y))^2)}^{\text{irreducible noise}},$$

*Proof.* Expand the RHS and simplify.

## Bias-variance tradeoff

In general, as model complexity increases (i.e., the hypothesis becomes more complex), variance tends to increase, and bias tends to decrease.

# Bias-variance Tradeoff in $k$ NN

## Assumption

Suppose  $Y | X \sim N(f(X), \sigma)$  for some function  $f$  and some fixed  $\sigma$ . In addition, suppose  $\mathbf{x}_1, \dots, \mathbf{x}_n$  are fixed.

# Bias-variance Tradeoff in $k$ NN

## Assumption

Suppose  $Y | X \sim N(f(X), \sigma)$  for some function  $f$  and some fixed  $\sigma$ . In addition, suppose  $\mathbf{x}_1, \dots, \mathbf{x}_n$  are fixed.

## Bias and variance

At  $\mathbf{x}$ ,  $Y' = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i$  is predicted and the true value is  $Y$ .

$$bias = \mathbb{E}(Y') - \mathbb{E}(Y) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} f(\mathbf{x}_i) - f(\mathbf{x}),$$

$$variance = \mathbb{E}((Y' - \mathbb{E}(Y'))^2) = \sigma^2/k.$$

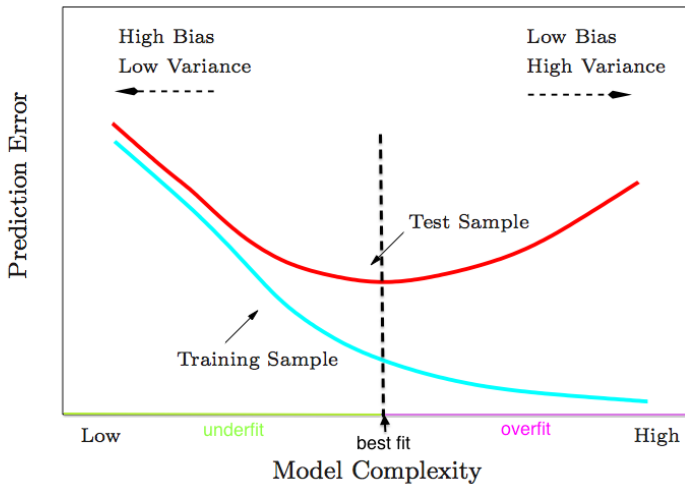
$$\begin{aligned} \text{bias} &= \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} f(\mathbf{x}_i) - f(\mathbf{x}), \\ \text{variance} &= \sigma^2/k. \end{aligned}$$

### $\frac{1}{k}$ as a complexity measure

This is because with smaller  $\frac{1}{k}$ ,  $h_n(\mathbf{x}) = \text{avg}(y_i \mid \mathbf{x}_i \in N_k(\mathbf{x}))$  is closer to a constant, and thus model complexity is smaller.

### Bias-variance trade-off

As  $\frac{1}{k}$  increases (or as model complexity increases), bias is likely to decrease, and variance increases.



# Model Selection

Assume model complexity is controlled by some parameter  $\theta$ . How to pick the best value among candidates  $\theta_0, \dots, \theta_m$ ?



# Model Selection

Assume model complexity is controlled by some parameter  $\theta$ . How to pick the best value among candidates  $\theta_0, \dots, \theta_m$ ?

## Using a development set


- Split available data into a training set  $\mathcal{T}$  and a development set  $\mathcal{D}$ .
- For each  $\theta_i$ , train a model on  $\mathcal{T}$ , and test it on  $\mathcal{D}$ .
- Choose the parameter with best performance.

A lot of data is needed, while the amount may be limited.

## **$K$ -fold cross validation**

- Split the training data into  $K$  folds.
- For each  $\theta_i$ , train  $K$  models, with each trained on  $K - 1$  folds and tested on the remaining fold.
- Choose the parameter with best average performance.

Computationally more expensive than using a development set.



Cross validation could help you find the function that you seek.

Machine Learning A Cappella - Overfitting Thriller!



Udacity



Subscribe

45,724

31,516

# What You Need to Know...

The expected prediction error is a function of the model complexity.

The bias-variance decomposition implies that minimizing the expected prediction error requires careful tuning of the model complexity.

Using a development set and cross-validation are two basic methods for model selection.

# Recap

- A statistics and optimization perspective  
*Learning a binomial and a Gaussian*
- Statistical learning theory  
*Data, hypotheses, loss function, expected risk, empirical risk...*
- Regression  
*Regression function, parametric regression, non-parametric regression...*
- Model selection  
*Bias-variance tradeoff, development set, cross-validation...*
- **Classification**
- **Clustering**

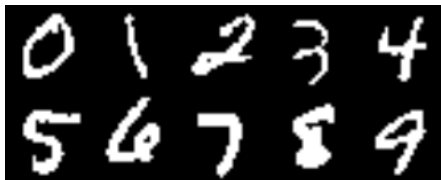
# Classification

- Bayes optimal classifier
- Nearest neighbor classifier
- Naive Bayes classifier
- Logistic regression
- The perceptron
- Support vector machines

# Recall

## Classification

Output a function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  where  $\mathcal{Y}$  is a finite set.



## 0/1 loss

$$L((x, y), h) = \mathbb{I}(y \neq h(x)) = \begin{cases} 1, & y \neq h(x), \\ 0, & y = h(x). \end{cases}$$

## Expected/True risk

$$R(h) = \mathbb{E}(L((X, Y), h)).$$

# Bayes Optimal Classifier

The expected 0/1 loss is minimized by the Bayes optimal classifier

$$h^*(x) = \arg \max_{y \in \mathcal{Y}} P(y | x).$$



# Bayes Optimal Classifier

The expected 0/1 loss is minimized by the Bayes optimal classifier

$$h^*(x) = \arg \max_{y \in \mathcal{Y}} P(y | x).$$

*Proof.* The expected 0/1 loss of a classifier  $h(x)$  is

$$\begin{aligned} \mathbb{E}(L((X, Y), h)) &= \mathbb{E}_X \mathbb{E}_{Y|X}(I(Y \neq h(X))) \\ &= \mathbb{E}_X P(Y \neq h(X) | X). \end{aligned}$$

Hence we can set the value of  $h(x)$  independently for each  $x$  by choosing it to minimize the expression under expectation. This leads to  $h^*(x) = \arg \max_{y \in \mathcal{Y}} P(y | x)$ .

The Bayes optimal classifier is

$$h^*(x) = \arg \max_{y \in \mathcal{Y}} P(y \mid x).$$

However,  $P(Y \mid x)$  is unknown...

**Idea.** Estimate  $P(y \mid x)$  from data.

# Nearest Neighbor Classifier

Approximate  $P(y \mid x)$  using the label distribution in  $\{y_i \mid x_i \in N_k(x)\}$ , where  $N_k(x)$  consists of the  $k$  nearest examples of  $x$  (with respect to some distance measure), and predict the majority label

$$h_n(x) = \text{majority}\{y_i \mid x_i \in N_k(x)\}.$$

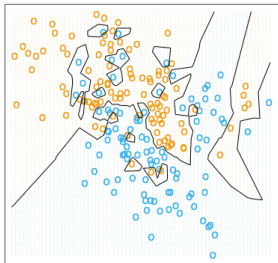
# Nearest Neighbor Classifier

Approximate  $P(y \mid x)$  using the label distribution in  $\{y_i \mid x_i \in N_k(x)\}$ , where  $N_k(x)$  consists of the  $k$  nearest examples of  $x$  (with respect to some distance measure), and predict the majority label

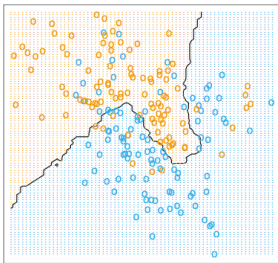
$$h_n(x) = \text{majority}\{y_i \mid x_i \in N_k(x)\}.$$

- Under mild conditions, as  $k \rightarrow \infty$  and  $n/k \rightarrow \infty$ ,  $h_n(x) \rightarrow h^*(x)$ , for any distribution  $P(X, Y)$ .
- (Curse of dimensionality) The number of samples required for accurate approximation is exponential in the dimension.

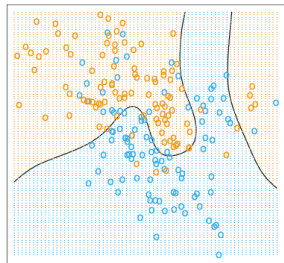
1-NN classifier



15-NN classifier



Bayes optimal classifier



# Naive Bayes Classifier (NB)

## Model

- $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_d$ , where each  $\mathcal{X}_i$  is a finite set.
- A model  $p(X, Y)$  satisfies the independence assumption

$$p(x_1, \dots, x_d \mid y) = p(x_1 \mid y) \dots p(x_d \mid y).$$

# Naive Bayes Classifier (NB)

## Model

- $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_d$ , where each  $\mathcal{X}_i$  is a finite set.
- A model  $p(X, Y)$  satisfies the independence assumption

$$p(x_1, \dots, x_d \mid y) = p(x_1 \mid y) \dots p(x_d \mid y).$$

## Classification

An example  $\mathbf{x} = (x_1, \dots, x_d)$  is classified as

$$y = \arg \max_{y' \in \mathcal{Y}} p(y' \mid \mathbf{x}).$$

This is equivalent to

$$y = \arg \max_{y' \in \mathcal{Y}} p(y', \mathbf{x}) = \arg \max_{y' \in \mathcal{Y}} p(y') p(x_1 \mid y') \dots p(x_d \mid y'),$$

by the independence assumption.

## Learning (MLE)

The maximum likelihood Naive Bayes model is  $\hat{p}(X, Y)$  given by

$$\begin{aligned}\hat{p}(y) &= n_y/n, \\ \hat{p}(x_i | y) &= n_{y,x_i}/n_y,\end{aligned}$$

where  $n_y$  is the number of times class  $y$  appears in the training set, and  $n_{y,x_i}$  is the number of times attribute  $i$  takes value  $x_i$  when the class label is  $y$ . (verify)



## Learning (MLE)

The maximum likelihood Naive Bayes model is  $\hat{p}(X, Y)$  given by

$$\begin{aligned}\hat{p}(y) &= n_y/n, \\ \hat{p}(x_i | y) &= n_{y,x_i}/n_y,\end{aligned}$$

where  $n_y$  is the number of times class  $y$  appears in the training set, and  $n_{y,x_i}$  is the number of times attribute  $i$  takes value  $x_i$  when the class label is  $y$ . (verify)

## Issues

- Independence assumption unlikely to be satisfied.
- The counts  $n_y$  may be 0, making the estimates undefined.
- The counts may be very small, leading to unstable estimates.

## Laplace correction

$$\hat{p}(y) = (n_y + c_0) / \sum_{y \in \mathcal{Y}} (n_{y'} + c_0),$$

$$\hat{p}(x_i | y) = (n_{y, x_i} + c_1) / \sum_{x'_i \in \mathcal{X}_i} (n_{y, x'_i} + c_1),$$

where  $c_0 > 0$  and  $c_1 > 0$  are user-chosen constants. Laplace correction makes NB more stable, but still relies on strong independence assumption.

# Logistic Regression (LR)

## Model

- $\mathcal{X} = \mathbb{R}^d$ .
- Logistic regression estimates conditional distributions of the form

$$p(y \mid \mathbf{x}, \theta) = \exp(\mathbf{x}^T \theta_y) / \sum_{y' \in \mathcal{Y}} \exp(\mathbf{x}^T \theta_{y'}),$$

where  $\theta_y = (\theta_{y1}, \dots, \theta_{yd}) \in \mathbb{R}^d$ , and  $\theta$  is the concatenation of  $\theta_y$ 's.

# Logistic Regression (LR)

## Model

- $\mathcal{X} = \mathbb{R}^d$ .
- Logistic regression estimates conditional distributions of the form

$$p(y \mid \mathbf{x}, \theta) = \exp(\mathbf{x}^T \theta_y) / \sum_{y' \in \mathcal{Y}} \exp(\mathbf{x}^T \theta_{y'}),$$

where  $\theta_y = (\theta_{y1}, \dots, \theta_{yd}) \in \mathbb{R}^d$ , and  $\theta$  is the concatenation of  $\theta_y$ 's.

## Classification

An example  $\mathbf{x}$  is classified as

$$y = \arg \max_{y' \in \mathcal{Y}} p(y' \mid \mathbf{x}, \theta).$$

## Learning

- Training is often done by maximizing regularized log-likelihood

$$L(\theta) = \log \prod_{i=1}^n p(y_i \mid \mathbf{x}_i, \theta) - \lambda \|\theta\|_2^2.$$

That is, the parameter estimate is

$$\theta_n = \arg \max_{\theta} L(\theta).$$

- $L(\theta)$  is a concave function, and can be optimized using standard numerical methods (such as L-BFGS).

# Comparing $k$ NN, NB and LR

All approximates the Bayes-optimal classifier

- Learn an approximation  $\tilde{P}(X, Y)$  to  $P(X, Y)$  (or learn an approximation  $\tilde{P}(Y | X)$  to  $P(Y | X)$ ).
- Choose the function  $h_n(x) = \arg \max_y \tilde{P}(y | x)$ .

$k$ NN and logistic regression estimate  $P(Y | X)$ , while naive Bayes estimates  $P(X, Y)$ .

$k$ NN is a non-parametric method, while naive Bayes and logistic regression are both parametric methods.

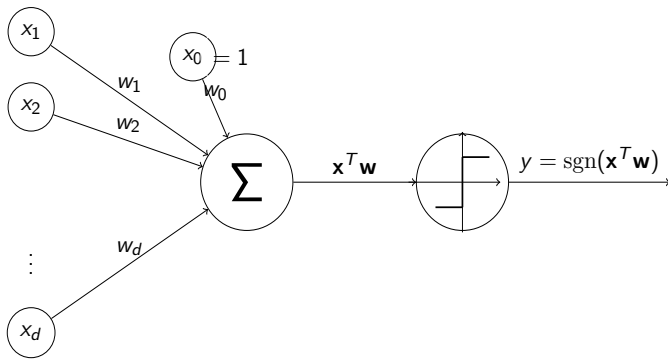
# Application in Digit Recognition

Assume each digit image is a binary image, represented as a vector with the pixel values as its elements.

## Applying the algorithms

- $k$ NN: use the Euclidean distance between the examples.
- NB can be applied because each example is a discrete vector.
- LR can be applied because each example is a continuous vector.

# The Perceptron

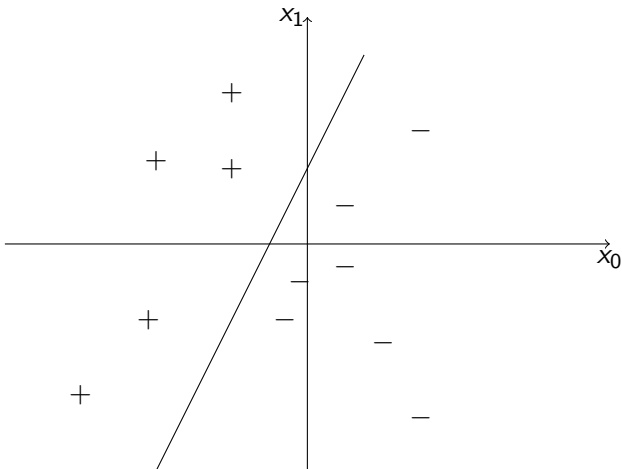


A perceptron maps an input  $\mathbf{x} \in \mathbb{R}^{d+1}$  to

$$h(\mathbf{x}) = \text{sgn}(\mathbf{x}^T \mathbf{w}) = \begin{cases} 1, & \mathbf{x}^T \mathbf{w} > 0, \\ 0, & \mathbf{x}^T \mathbf{w} = 0, \\ -1, & \mathbf{x}^T \mathbf{w} < 0. \end{cases}$$

Here  $\mathbf{x} = (1, x_1, \dots, x_d)$  includes a dummy variable 1.





A perceptron corresponds to a linear decision boundary (i.e., the boundary between the regions for examples of the same class).

It is NP-hard to minimize the empirical 0/1 loss of a perceptron, that is, given a training set  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ , it is NP-hard to solve

$$\min_{\mathbf{w}} \frac{1}{n} \mathbb{I}(\text{sgn}(\mathbf{x}_i^T \mathbf{w}) \neq y_i)$$

*Idea: Can we use  $(\mathbf{x}_i^T \mathbf{w} - y_i)^2$  as a surrogate loss for the 0/1 loss?*

# Least Squares May Fail

## Recall: Binary classification as regression

- Label one class as  $-1$  and the other as  $+1$ .
- Fit a function  $f(\mathbf{x})$  using least squares regression.
- Given a test example  $\mathbf{x}$ , predict  $-1$  if  $f(\mathbf{x}) < 0$  and  $+1$  otherwise.

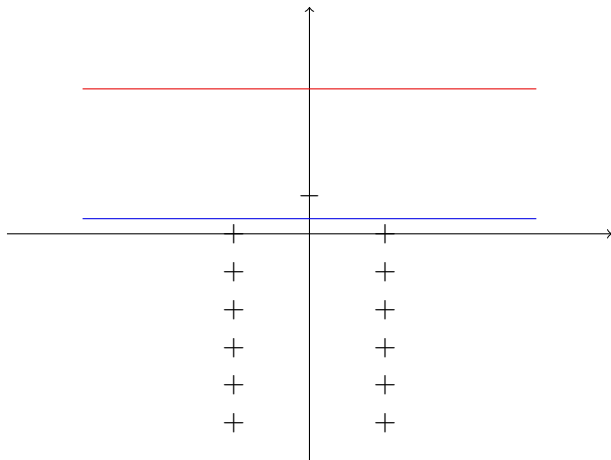
# Least Squares May Fail

## Recall: Binary classification as regression

- Label one class as  $-1$  and the other as  $+1$ .
- Fit a function  $f(\mathbf{x})$  using least squares regression.
- Given a test example  $\mathbf{x}$ , predict  $-1$  if  $f(\mathbf{x}) < 0$  and  $+1$  otherwise.

## Issue

Least squares fitting may not find a separating hyperplane (i.e., a hyperplane which puts the positive and negative examples on different sides of it) even there is one.



The decision boundary learned using least squares fitting (red line) wrongly classifies the negative example, while there exists separating hyperplanes (like the blue line).

# Perceptron Algorithm

**Require:**  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \in \mathbb{R}^{d+1} \times \{-1, +1\}$ ,  $\eta \in (0, 1]$ .

**Ensure:** Weight vector  $\mathbf{w}$ .

Randomly or smartly initialize  $\mathbf{w}$ .

**while** there is any misclassified example **do**

    Pick a misclassified example  $(\mathbf{x}_i, y_i)$ .

$\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$ .

## Why the update rule $\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$ ?

- $\mathbf{w}$  classifies  $(\mathbf{x}_i, y_i)$  correctly if and only if  $y_i \mathbf{x}_i^T \mathbf{w} > 0$ .
- If  $\mathbf{w}$  classifies  $(\mathbf{x}_i, y_i)$  wrongly, then the update rule moves  $y_i \mathbf{x}_i^T \mathbf{w}$  towards positive, because

$$y_i \mathbf{x}_i^T (\mathbf{w} + \eta y_i \mathbf{x}_i) = y_i \mathbf{x}_i^T \mathbf{w} + \eta \|\mathbf{x}_i\|^2 > y_i \mathbf{x}_i^T \mathbf{w}.$$

## Perceptron convergence theorem

If the training data is linearly separable (i.e., there exists some  $\mathbf{w}^*$  such that  $y_i \mathbf{x}_i^T \mathbf{w}^* > 0$  for all  $i$ ), then the perceptron algorithm terminates with all training examples correctly classified.



## Perceptron convergence theorem

If the training data is linearly separable (i.e., there exists some  $\mathbf{w}^*$  such that  $y_i \mathbf{x}_i^T \mathbf{w}^* > 0$  for all  $i$ ), then the perceptron algorithm terminates with all training examples correctly classified.

*Proof.* Suppose  $\mathbf{w}^*$  separates the data. We can scale  $\mathbf{w}^*$  such that  $|\mathbf{x}_i^T \mathbf{w}^*| \geq \|\mathbf{x}_i\|_2^2$ .

If  $\mathbf{w}$  classifies  $(\mathbf{x}_i, y_i)$  wrongly, then it will be updated to  $\mathbf{w}' = \mathbf{w} + \eta y_i \mathbf{x}_i$ .

We show that  $\|\mathbf{w}^* - \mathbf{w}'\|_2^2 \leq \|\mathbf{w}^* - \mathbf{w}\|_2^2 - \eta R^2$ , where  $R = \min_i \|\mathbf{x}_i\|_2$ . This implies that only finitely many updates is possible.

The inequality can be shown as follows.

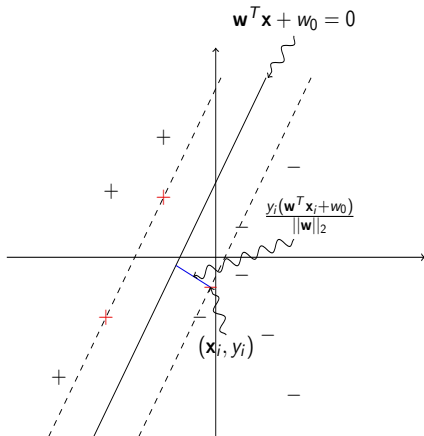
$$\begin{aligned}\|\mathbf{w}^* - \mathbf{w}'\|_2^2 &= \|\mathbf{w}^* - \mathbf{w} - \eta y_i \mathbf{x}_i\|_2^2 \\&= \|\mathbf{w}^* - \mathbf{w}\|_2^2 - 2\eta y_i \mathbf{x}_i^T (\mathbf{w}^* - \mathbf{w}) + \eta^2 y_i^2 \|\mathbf{x}_i\|_2^2 \\&\leq \|\mathbf{w}^* - \mathbf{w}\|_2^2 - 2\eta (|\mathbf{x}_i^T \mathbf{w}^*| + |\mathbf{x}_i^T \mathbf{w}|) + \eta \|\mathbf{x}_i\|_2^2 \\&\leq \|\mathbf{w}^* - \mathbf{w}\|_2^2 - 2\eta |\mathbf{x}_i^T \mathbf{w}^*| + \eta \|\mathbf{x}_i\|_2^2 \\&\leq \|\mathbf{w}^* - \mathbf{w}\|_2^2 - \eta |\mathbf{x}_i^T \mathbf{w}^*| \\&\leq \|\mathbf{w}^* - \mathbf{w}\|_2^2 - \eta R^2.\end{aligned}$$

## Issues

- When the data is separable, the hyperplane found by the perceptron algorithm depends on the initial weight and is thus arbitrary.
- Convergence can be very slow, especially when the gap between the positive and negative examples is small.
- When the data is not separable, the algorithm does not stop, but this can be difficult to detect.

# Support Vector Machines (SVMs)

## Separable data

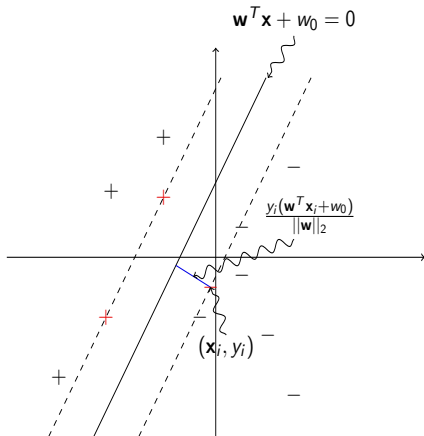


### Geometric intuition

Find a separating hyperplane with maximal margin (i.e., the minimum distance from the points to it).

# Support Vector Machines (SVMs)

## Separable data



### Geometric intuition

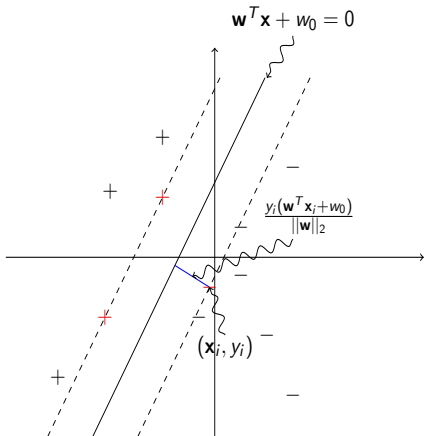
Find a separating hyperplane with maximal margin (i.e., the minimum distance from the points to it).

### Algebraic formulation

$$\begin{aligned} & \max_{M, \mathbf{w}, w_0} M \\ & \text{subject to } \frac{y_i(\mathbf{w}^T \mathbf{x}_i + w_0)}{\|\mathbf{w}\|_2} \geq M, i = 1, \dots, n. \end{aligned}$$

# Support Vector Machines (SVMs)

## Separable data



### Geometric intuition

Find a separating hyperplane with maximal margin (i.e., the minimum distance from the points to it).

### Algebraic formulation

$$\begin{aligned} & \max_{M, \mathbf{w}, w_0} M \\ & \text{subject to } \frac{y_i (\mathbf{w}^T \mathbf{x}_i + w_0)}{\|\mathbf{w}\|_2} \geq M, i = 1, \dots, n. \end{aligned}$$

### Equivalent formulation (add $M\|\mathbf{w}\|_2 = 1$ )

$$\begin{aligned} & \min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|_2^2 \\ & \text{subject to } y_i (\mathbf{w}^T \mathbf{x}_i + w_0) \geq 1, i = 1, \dots, n. \end{aligned}$$

# Soft-margin SVMs

## Non-separable data

### Algebraic formulation

$$\min_{\mathbf{w}, w_0, \xi_1, \dots, \xi_n} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_i \xi_i$$

subject to  $y_i(\mathbf{w}^T \mathbf{x}_i + w_0) \geq 1 - \xi_i, i = 1, \dots, n,$

$\xi_i \geq 0, i = 1, \dots, n.$

- $C > 0$  is a user chosen constant.
- Introducing  $\xi_i$  allows  $(\mathbf{x}_i, y_i)$  to be misclassified with a penalty of  $C\xi_i$  in the original objective function  $\frac{1}{2} \|\mathbf{w}\|_2^2$ .

An SVM always have a unique solution that can be found efficiently.

## SVM as minimizing regularized hinge loss

Soft-margin SVMs can be equivalently written as

$$\min_{\mathbf{w}, w_0} \frac{1}{2C} \|\mathbf{w}\|_2^2 + \sum_i \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + w_0)),$$

where  $\max(0, 1 - y(\mathbf{w}^T \mathbf{x} + w_0))$  is the hinge loss

$$L_{\text{hinge}}((\mathbf{x}, y), h) = \max(0, 1 - yh(\mathbf{x}))$$

of the classifier  $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$ , and upper bounds the 0/1 loss

$$L_{0/1}((\mathbf{x}, y), h) = \mathbb{I}(y \neq \text{sgn}(h(\mathbf{x})))$$

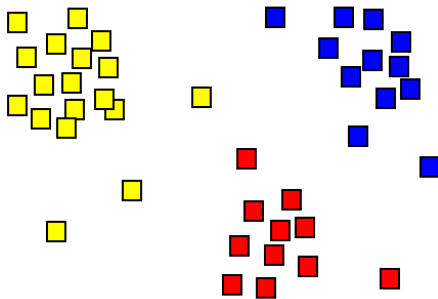
# What You Need to Know...

- Bayes optimal classifier.
- Probabilistic classifiers: NN, NB and LR.
- Perceptrons and SVMs.



# Clustering

# Clustering



Clustering is unsupervised function learning which learns a function from  $\mathcal{X}$  to  $[K] = \{1, \dots, K\}$ . The objective generally depends on some similarity/distance measure between the items, so that similar items are grouped together.

# $K$ -means Clustering

Given observations  $\mathbf{x}_1, \dots, \mathbf{x}_n$ , find a  $K$ -clustering  $f$  (a surjective from  $[n]$  to  $[K]$ ) to minimize the cost

$$\sum_{i=1}^n \|\mathbf{x}_i - \mathbf{c}_{f(i)}\|^2,$$

where  $\mathbf{c}_k$  is the centroid of cluster  $k$ , that is, the average of  $\mathbf{x}_i$ 's with  $f(i) = k$ .

## ***K*-means algorithm**

Randomly initialize  $\mathbf{c}_{1:K}$ , and set each  $f(i) = 0$ .

**repeat**

*(Assignment)* Set each  $f(i)$  to be the index of the  $\mathbf{c}_j$  closest to  $\mathbf{x}_i$ .

*(Update)* Set each  $\mathbf{c}_i$  as the centroid of cluster  $i$  given by  $f$ .

**until**  $f$  does not change

## ***K*-means algorithm**

Randomly initialize  $\mathbf{c}_{1:K}$ , and set each  $f(i) = 0$ .

**repeat**

*(Assignment)* Set each  $f(i)$  to be the index of the  $\mathbf{c}_j$  closest to  $\mathbf{x}_i$ .

*(Update)* Set each  $\mathbf{c}_i$  as the centroid of cluster  $i$  given by  $f$ .

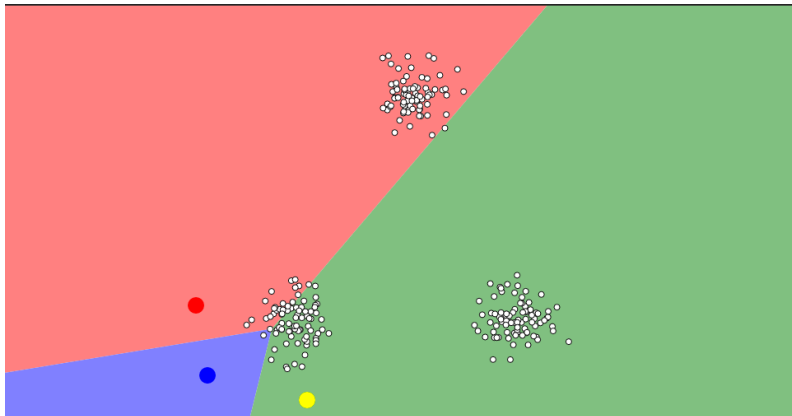
**until**  $f$  does not change

### Initialization

- Forgy method: randomly choose  $K$  observations as centroids, and initialize  $f$  as in the assignment step.
- Random Partition: assign a random cluster to each example.

Random partition is preferable.

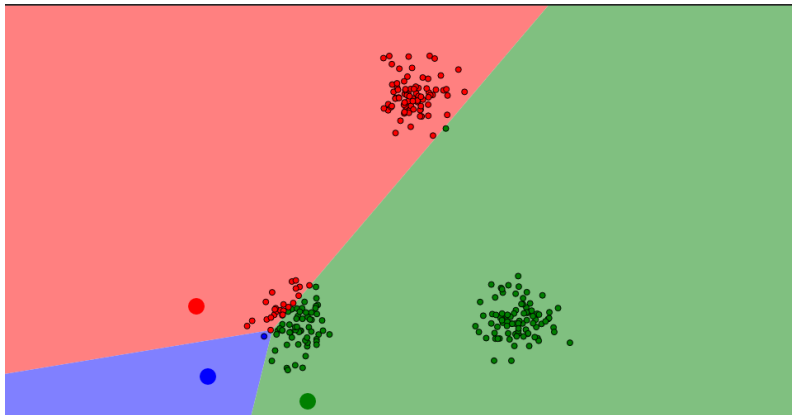
# Illustration



Initialize centroids

Generated using <http://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

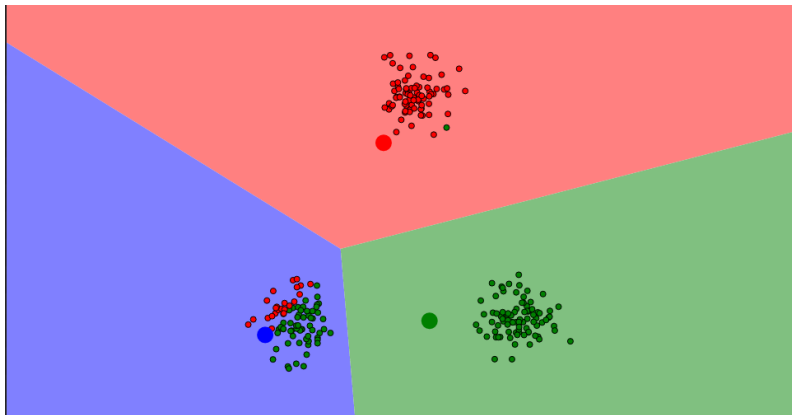
# Illustration



Iter 1: Assignment

Generated using <http://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

# Illustration

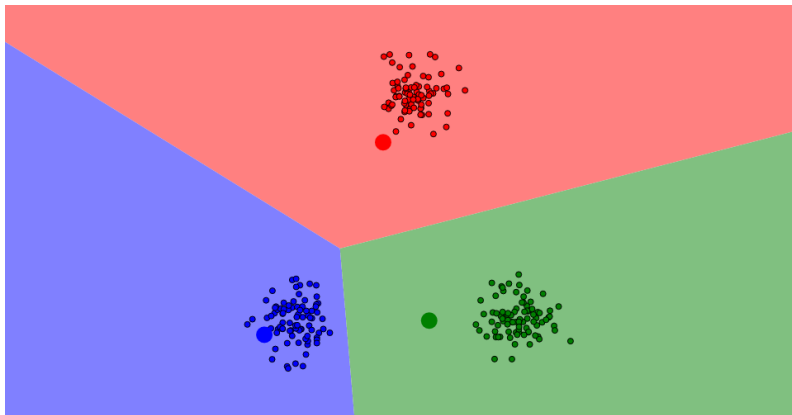


Iter 1: Update

Generated using <http://www.naftaliharris.com/blog/visualizing-k-means-clustering/>



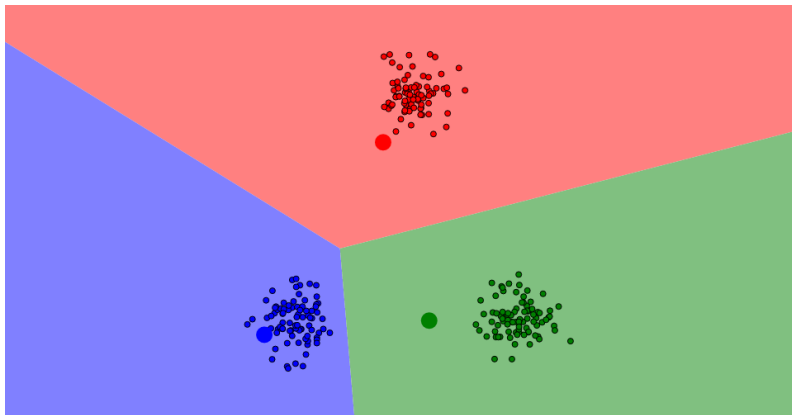
# Illustration



Iter 2: Assignment

Generated using <http://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

# Illustration



Iter 2: Update (converged)

Generated using <http://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

# K-means as Coordinate Descent

K-means is a coordinate descent algorithm for the cost function

$$C(f, \mathbf{c}_{1:K}) = \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{c}_{f(i)}\|^2.$$

Specifically, we have (verify)

(Assignment)  $f \leftarrow \arg \min_{f'} C(f', \mathbf{c}_{1:K})$ , where  $f'$  is a  $K$ -clustering.

(Update)  $\mathbf{c}_{1:K} \leftarrow \arg \min_{\mathbf{c}'_{1:K}} C(f, \mathbf{c}'_{1:K})$ ,

## Convergence

- The cost decreases at each iteration before termination.
- The cost converges to a local minimum by the monotone convergence theorem.
- Convergence may be very slow, taking exponential time in some cases, but such cases do not seem to arise in practice.

## Convergence

- The cost decreases at each iteration before termination.
- The cost converges to a local minimum by the monotone convergence theorem.
- Convergence may be very slow, taking exponential time in some cases, but such cases do not seem to arise in practice.

## Dealing with poor local minimum

- Restart multiple times and pick the minimum cost clustering found.

## Convergence

- The cost decreases at each iteration before termination.
- The cost converges to a local minimum by the monotone convergence theorem.
- Convergence may be very slow, taking exponential time in some cases, but such cases do not seem to arise in practice.

## Dealing with poor local minimum

- Restart multiple times and pick the minimum cost clustering found.

*K-means gives hard clusterings. Can we give probabilistic assignments to clusters?*

# Soft Clustering with Gaussian Mixtures

## Assumption

- Each cluster is represented as a Gaussian

$$N(\mathbf{x}; \mu_k, \Sigma_k) = \frac{1}{\sqrt{|2\pi\Sigma_k|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1}(\mathbf{x} - \mu_k)\right)$$

- Cluster  $k$  has weight  $w_k \geq 0$ , with  $\sum_{k=1}^K w_k = 1$ .

Equivalently, we assume the probability of observing  $\mathbf{x}$  and it is in cluster  $z = k$  is

$$p(\mathbf{x}, z = k \mid \theta) = w_k N(\mathbf{x}; \mu_k, \Sigma_k),$$

where  $\theta = \{w_{1:K}, \mu_{1:K}, \Sigma_{1:K}\}$ .



Equivalently, we assume the probability of observing  $\mathbf{x}$  and it is in cluster  $z = k$  is

$$p(\mathbf{x}, z = k \mid \theta) = w_k N(\mathbf{x}; \mu_k, \Sigma_k),$$

where  $\theta = \{w_{1:K}, \mu_{1:K}, \Sigma_{1:K}\}$ .

The distribution  $p(\mathbf{x} \mid \theta) = \sum_k w_k N(\mathbf{x}; \mu_k, \Sigma_k)$  is called a Gaussian mixture.

## Computing a soft clustering

$$p(Z = k \mid \mathbf{x}, \theta) = \frac{p(\mathbf{x}, Z = k \mid \theta)}{\sum_{k'=1}^K p(\mathbf{x}, Z = k' \mid \theta)}.$$

## Learning a Gaussian mixture

Given the observations  $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , we choose  $\theta$  by maximizing the log-likelihood  $L(D \mid \theta) = \sum_{i=1}^n \ln p(\mathbf{x}_i \mid \theta)$

$$\max_{\theta} L(D \mid \theta).$$

This is solved using the EM algorithm.

# The EM Algorithm

Let  $z_i$  be the random variable representing the cluster from which  $x_i$  is drawn from. EM starts with some initial parameter  $\theta^{(0)}$ , and repeats the following steps:

**Expectation step:**  $Q(\theta \mid \theta^{(t)}) = \mathbb{E}\left(\sum_i \ln p(\mathbf{x}_i, z_i \mid \theta) \mid D, \theta^{(t)}\right),$

**Maximization step:**  $\theta^{(t+1)} = \arg \max_{\theta} Q(\theta \mid \theta^{(t)}).$

In words...

- E-step: Expectation of the log-likelihood for the complete data, w.r.t. the conditional distribution  $p(z_{1:n} \mid D, \theta^{(t)})$ .
- M-step: Maximization of the expectation

# The EM Algorithm

Let  $z_i$  be the random variable representing the cluster from which  $\mathbf{x}_i$  is drawn from. EM starts with some initial parameter  $\theta^{(0)}$ , and repeats the following steps:

**Expectation step:**  $Q(\theta \mid \theta^{(t)}) = \mathbb{E}\left(\sum_i \ln p(\mathbf{x}_i, z_i \mid \theta) \mid D, \theta^{(t)}\right),$

**Maximization step:**  $\theta^{(t+1)} = \arg \max_{\theta} Q(\theta \mid \theta^{(t)}).$

Data completion interpretation

- E-step: Create complete data  $(\mathbf{x}_i, z_i)$  with weight  $p(z_i \mid \mathbf{x}_i, \theta^{(t)})$ , for each  $\mathbf{x}_i$  and each  $z_i \in [K]$ .
- M-step: Perform maximum likelihood estimation on the complete data set.

## EM algorithm is iterative likelihood maximization

EM algorithm iteratively improves the likelihood function,

$$L(D \mid \theta^{(t+1)}) \geq L(D \mid \theta^{(t)}).$$

## EM algorithm is iterative likelihood maximization

EM algorithm iteratively improves the likelihood function,

$$L(D \mid \theta^{(t+1)}) \geq L(D \mid \theta^{(t)}).$$

*Proof.* With some algebraic manipulation, we have

$$\begin{aligned} & Q(\theta^{(t+1)} \mid \theta^{(t)}) - Q(\theta^{(t)} \mid \theta^{(t)}) \\ &= L(D \mid \theta^{(t+1)}) - L(D \mid \theta^{(t)}) - \sum_i KL(p(Z_i \mid \mathbf{x}_i, \theta^{(t)}) \mid p(Z_i \mid \mathbf{x}_i, \theta^{(t+1)})), \end{aligned}$$

where  $KL(q \parallel q')$  is the KL-divergence  $\sum_x q(x) \ln \frac{q(x)}{q'(x)}$ .

The result follows by noting that the LHS is non-negative by the choice of  $\theta^{(t+1)}$  in the M-step, and the nonnegativeness of the KL-divergence.

$$\begin{aligned}
& Q(\theta^{(t+1)} \mid \theta^{(t)}) - Q(\theta^{(t)} \mid \theta^{(t)}) \\
&= \mathbb{E} \left( \sum_i \ln p(\mathbf{x}_i, z_i \mid \theta^{(t+1)}) \mid D, \theta^{(t)} \right) - \mathbb{E} \left( \sum_i \ln p(\mathbf{x}_i, z_i \mid \theta^{(t)}) \mid D, \theta^{(t)} \right) \\
&= \mathbb{E} \left( \sum_i (\ln p(\mathbf{x}_i \mid \theta^{(t+1)}) + \ln p(z_i \mid \mathbf{x}_i, \theta^{(t+1)}) \right. \\
&\quad \left. - \ln p(\mathbf{x}_i \mid \theta^{(t)}) - \ln p(z_i \mid \mathbf{x}_i, \theta^{(t)} \mid D, \theta^{(t)})) \right) \\
&= \sum_i \ln p(\mathbf{x}_i \mid \theta^{(t+1)}) - \sum_i \ln p(\mathbf{x}_i \mid \theta^{(t)}) - \mathbb{E} \left( \ln \frac{p(z_i \mid \mathbf{x}_i, \theta^{(t)})}{p(z_i \mid \mathbf{x}_i, \theta^{(t)})} \mid D, \theta^{(t)} \right) \\
&= L(D \mid \theta^{(t+1)}) - L(D \mid \theta^{(t)}) - \sum_i KL(p(Z_i \mid \mathbf{x}_i, \theta^{(t)}) \mid p(Z_i \mid \mathbf{x}_i, \theta^{(t+1)})).
\end{aligned}$$

# Update Equations for Gaussian Mixtures

## Scalar covariance matrices

Assume each covariance matrix  $\Sigma_k$  is a scalar matrix  $\sigma_k^2 I_d$ . Let  $w_k^{(t,i)} = p(z_i = k \mid \mathbf{x}_i, \theta^{(t)})$ . Then given  $\theta^{(t)}$ ,  $\theta^{(t+1)}$  can be computed using

$$w_k^{(t+1)} = \sum_i w_k^{(t,i)} / n,$$

$$\mu_k^{(t+1)} = \sum_i w_k^{(t,i)} \mathbf{x}_i / \sum_i w_k^{(t,i)},$$

$$(\sigma_k^{(t+1)})^2 = \sum_i w_k^{(t,i)} \sum_j (\mathbf{x}_{ij} - \mu_{kj}^{(t+1)})^2 / (d \sum_i w_k^{(t,i)}).$$



### *Data completion interpretation*

- Split example  $\mathbf{x}_i$  into  $K$  complete examples  $(\mathbf{x}_i, 1), \dots, (\mathbf{x}_i, K)$ , where example  $(\mathbf{x}_i, k)$  has weight  $w_k^{(t,i)}$ .
- Apply maximum likelihood estimation to the complete data
  - $w_k^{(t+1)}$  is total weight of examples in cluster  $k$ .
  - $\mu_k^{(t+1)}$  is the mean  $\mathbf{x}$  value of the (weighted) examples in cluster  $k$ .
  - $\sigma_k^{(t+1)}$  is the standard deviation of all the attributes of the (weighted) examples in cluster  $k$ .

## Diagonal covariance matrices

Assume each covariance matrix  $\Sigma_k$  is a diagonal matrix with diagonal entries  $\sigma_{k1}^2, \dots, \sigma_{kd}^2$ . Let  $w_k^{(t,i)} = p(z_i = k \mid \mathbf{x}_i, \theta^{(t)})$ . Then given  $\theta^{(t)}$ ,  $\theta^{(t+1)}$  can be computed using

$$w_k^{(t+1)} = \sum_i w_k^{(t,i)} / n,$$

$$\mu_k^{(t+1)} = \sum_i w_k^{(t,i)} \mathbf{x}_i / \sum_i w_k^{(t,i)},$$

$$(\sigma_{kj}^{(t+1)})^2 = \sum_i w_k^{(t,i)} (\mathbf{x}_{ij} - \mu_{kj}^{(t+1)})^2 / \sum_i w_k^{(t,i)}.$$

## Arbitrary covariance matrices

Let  $w_k^{(t,i)} = p(z_i = k \mid \mathbf{x}_i, \theta^{(t)})$ . Then given  $\theta^{(t)}$ ,  $\theta^{(t+1)}$  can be computed using

$$w_k^{(t+1)} = \sum_i w_k^{(t,i)} / n,$$

$$\mu_k^{(t+1)} = \sum_i w_k^{(t,i)} \mathbf{x}_i / \sum_i w_k^{(t,i)},$$

$$\Sigma_k^{(t+1)} = \sum_i w_k^{(t,i)} (\mathbf{x}_i - \mu_k^{(t+1)}) (\mathbf{x}_i - \mu_k^{(t+1)})^T / \sum_i w_k^{(t,i)}.$$

# What You Need to Know...

- The clustering problem.
- Hard clustering with  $K$ -means algorithm.
- Soft clustering with Gaussian mixture.

# Density Estimation

- Maximum likelihood estimation.
- Naive Bayes.
- Logistic regression.

# This Tutorial...

Essentials for crafting basic machine learning systems.

## **Formulate applications as machine learning problems**

*Classification, regression, density estimation, clustering*

## **Understand and apply basic learning algorithms**

*Least squares regression, logistic regression, support vector machines, K-means,...*

## **Theoretical understanding**

Position and compare the problems and algorithms in the unifying framework of statistical learning theory.

# Beyond This Course

- Representation: dimensionality reduction, feature selection,...
- Algorithms: decision tree, artificial neural network, Gaussian processes,...
- Meta-learning algorithms: boosting, stacking, bagging,...
- Learning theory: generalization performance of learning algorithms
- *Many other exciting stuff...*

# Further Readings

