Chapter 1: Machine Learning introduction

contents

What are Machine Learning

Definition of Machine Learning

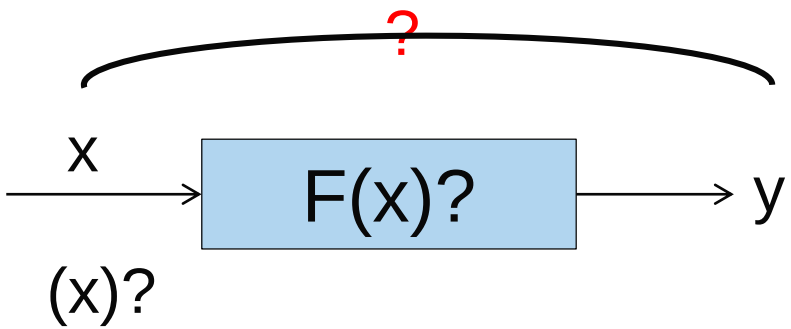
1. **A computer program is said to learn from experience E， if：**
    1. its performance at tasks in T,
    2. as measured by P,
    3. improves with experience E.　　T,
2. **we define machine learning as a set of methods**
    1. automatically detect patterns in data
    2. use the uncovered patterns to predict future data
    3. perform other kinds of decision making under uncertainty

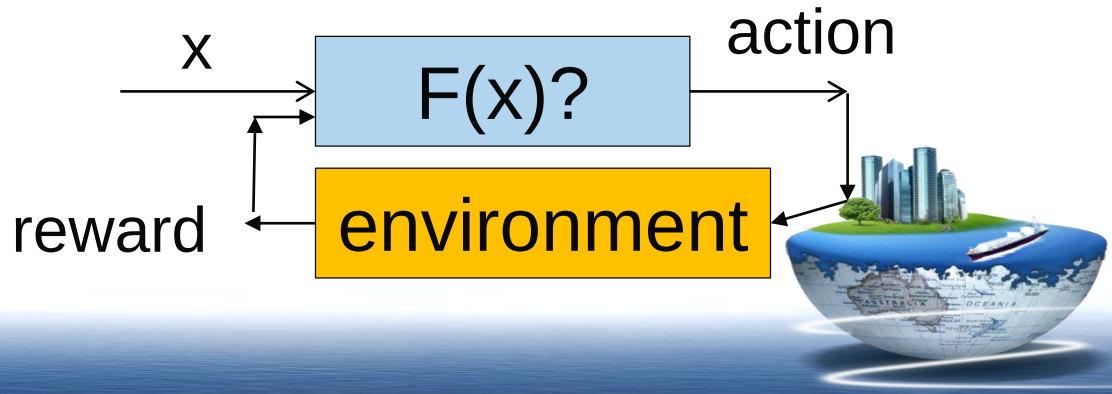
Feature of ML from a probabilistic perspective

1. we treat all unknown quantities as random variables
2. it is the optimal approach to decision making under uncertainty
3. probabilistic modeling is the language used by
   - most other areas of science and engineering
4. it provides a unifying framework between these fields
5. it is through this view that we can connect
   - what we do in machine learning to every other computational science whether that be in
     - stochastic optimisation, control theory, operations research,
     - econometrics, information theory, statistical physics or bio-statistics

Types of Machine Learning method

- Supervised Learning :
  - Strongly Supervised Learning
  - Weakly Supervised Learning
- unsupervised Learning.
  - This is sometimes called **knowledge discovery**.
- reinforcement learning

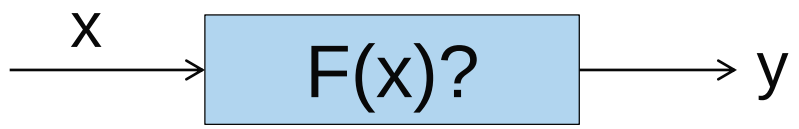




Supervised Learning

Concepts of Supervised Learning

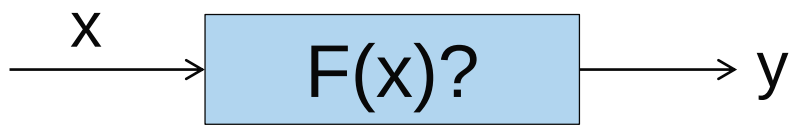
- the task T
  - to learn a mapping *f* from inputs $x \in X$ to outputs $y \in Y$
  - The inputs *x* are also called the **features**
  - The output y are also known as the **labels**



- The experience E is given in the form of
  - a set of input-output pairs D={ (x1,y1), (x1,y2),…(xN,yN)}
  - The set D is called **training set**
- measure P depends on the type of output we are predicting

Supervised Learning include two problems

- Classification : output y is one of a finite discrete number, $y \in \{1, 2, \ldots, N\}$
  - The problem of predicting the class label given an input is also called **pattern recognition**.
  - If there are just two classes, $y \in \{0, 1\}$ or $y \in \{-1, +1\}$, it is called **binary classification**.
- Regression: output y is of one or more continuous variables, $y \in [R1,R2]$



Supervised Learning

Classification problem

Example of Supervised Learning

- Three types of Iris flowers:

Setosa

Versicolor

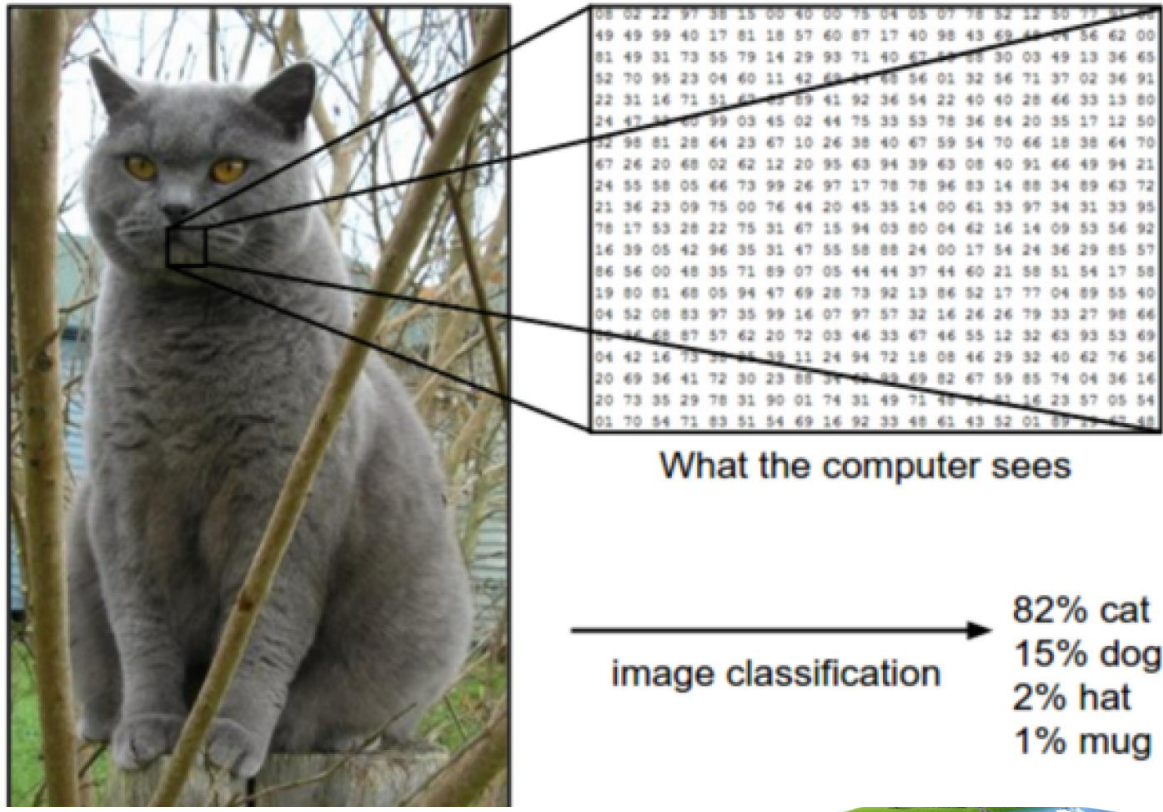
Virginica

- classifying Iris flowers into their 3 subspecies→
  - $Y = \{1,2,3\}$

index	sepal length	sepal width	petal length	petal width	label
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
…					
50	7.0	3.2	4.7	1.4	Versicolor
…					
149	5.9	3.0	5.1	1.8	Virginica

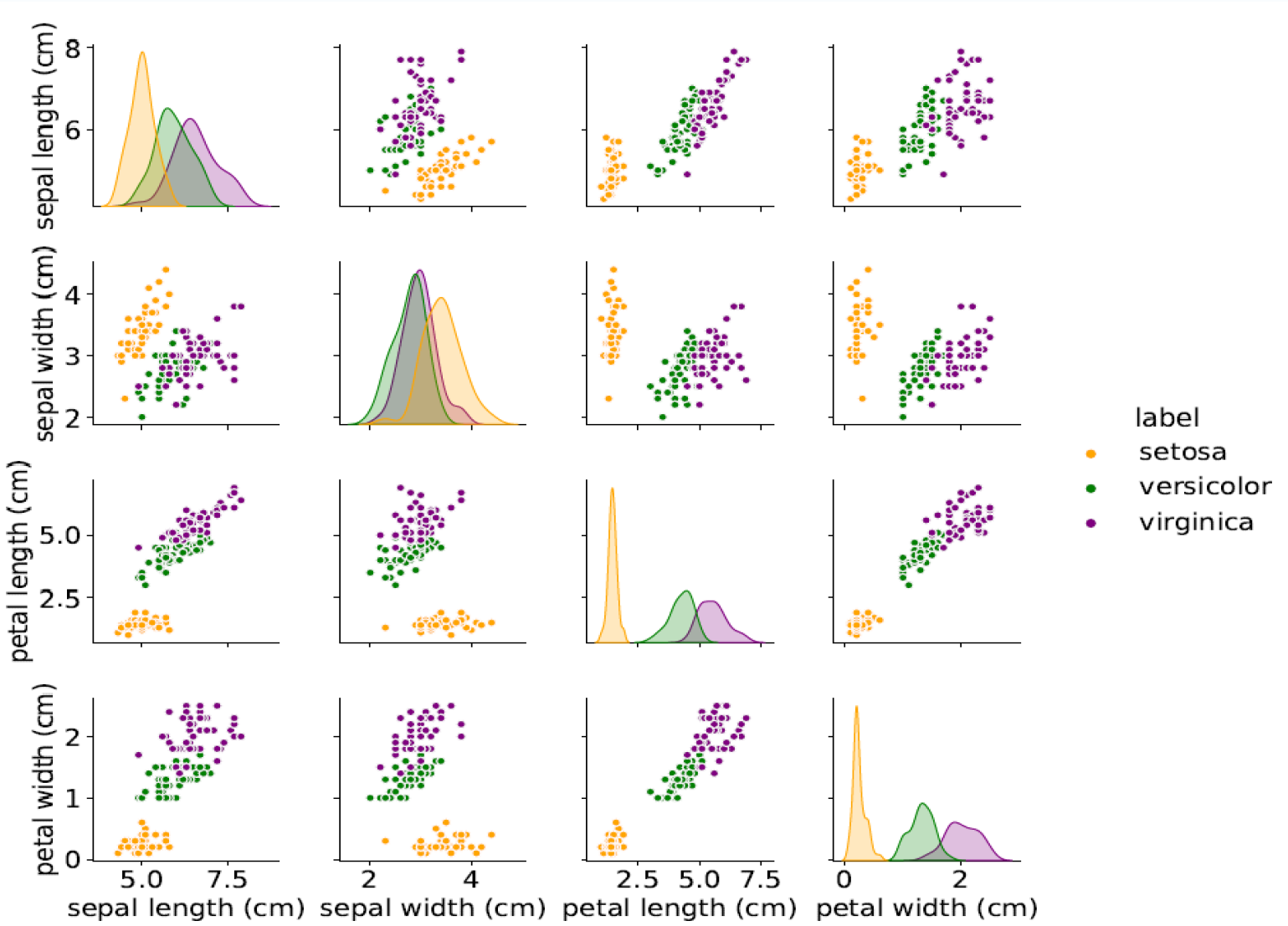
Example of image classification

- classifying images into 4 categories
  - the set of images is a very high-dimensional space:
    - ➓ a color image with C = 3 channels (e.g., RGB)
    - ➓ D1 × D2 pixels,
    - ➓ $X = R^D$,
    - ➓ D = C × D1 × D2.
    - ➓ pixel intensity with an integer from range {0,1,..,255}



Exploratory data analysis

- Iris data
  - a pairwise scatter plot
- the marginal distribution
  - of each feature for each class
  - On the diagonal
- For higher-dimensional data
  - it is common to first perform **dimensionality reduction**

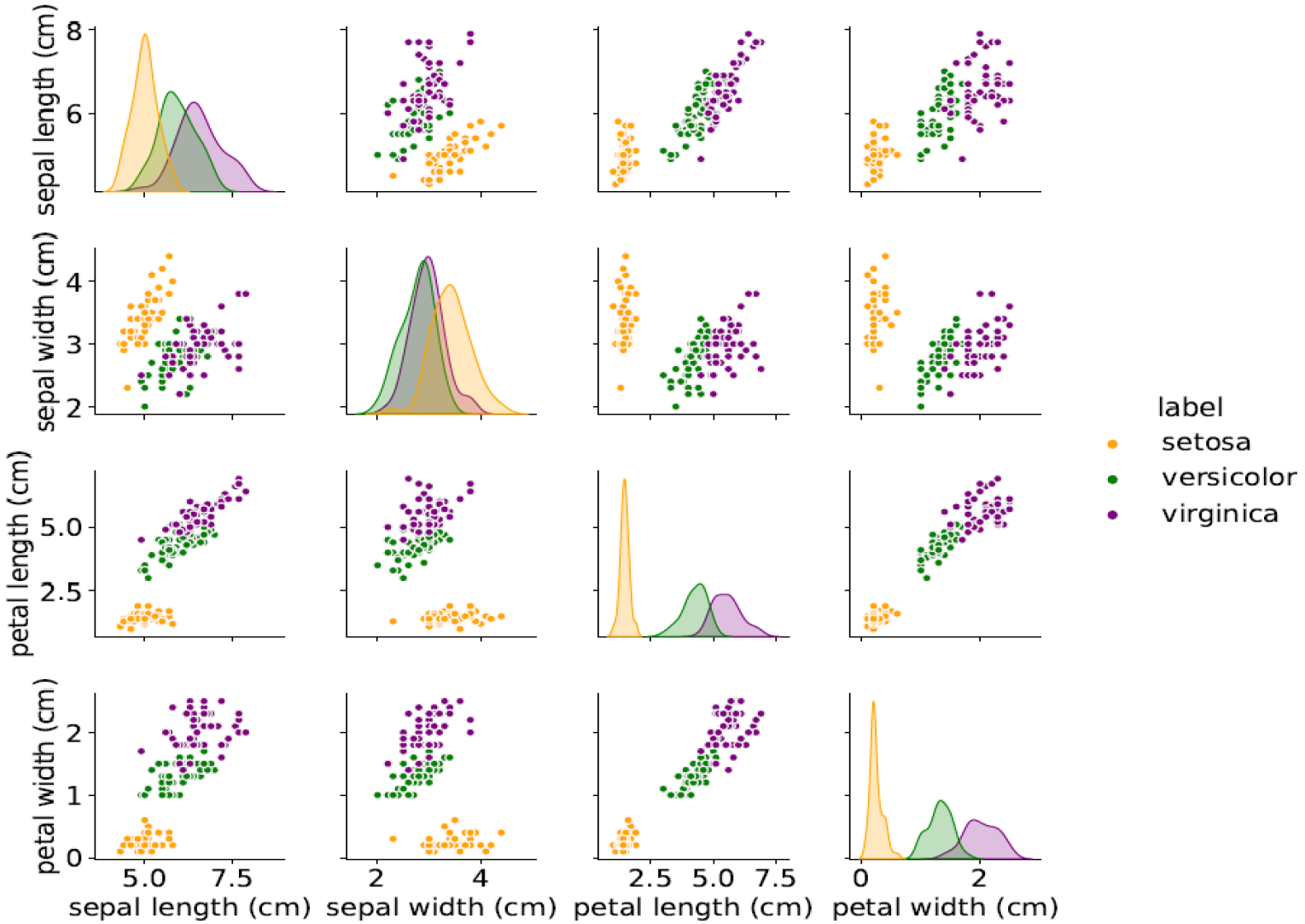


learning a classifier

- we can find that
  - the Setosa class is easy to distinguish from the other two classes
- a very simple a classifier*:*

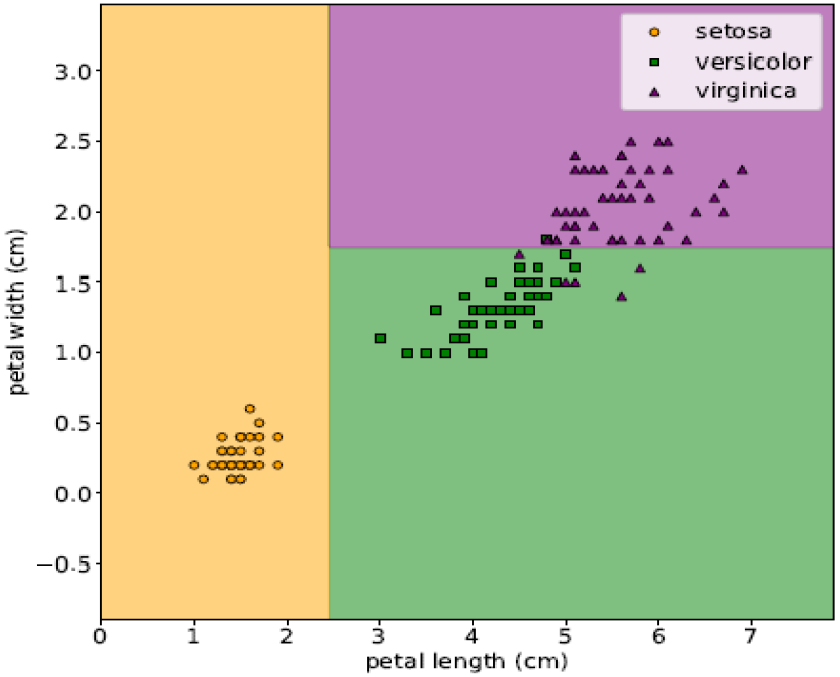
$$f(x,\theta) = \begin{cases} \text{Setosa} & \text{if petal length} < 2.45 \\ \text{Versicolor or Virginica} & \text{otherwise} \end{cases}$$

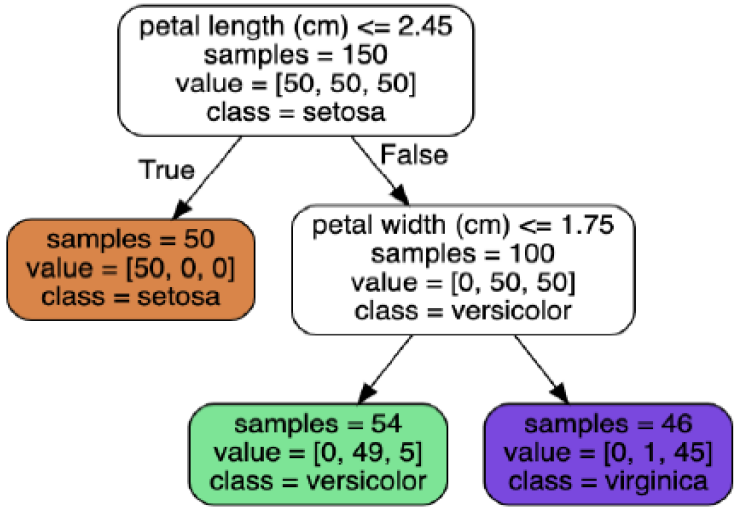
- decision boundary
  - the petal length = 2.45



Further distinguish the other two classes

- a decision tree of depth 2 applied to the Iris data
  - using just the petal length and petal width features





misclassification rate

- A common way to measure performance on classification models
  - Define of the misclassification rate

$$L(\theta) = \frac{1}{N}\sum_{n=1}^{N} I(y_n \neq f(x_n, \theta))$$

- indicator function

$$I(e) = \begin{cases} 1 & if\ e\ is\ true \\ 0 & if\ e\ is\ false \end{cases}$$

empirical risk

- define
  - the average loss of the predictor on the training set

$$L(\theta) = \frac{1}{N}\sum_{n=1}^{N} l(y_n \neq f(x_n, \theta))$$

- zero-one loss

$$l_{01}(y, \hat{y}) = I(y \neq \hat{y})$$

loss matrix for Iris classification

		Estimate		
		Setosa	Versicolor	Virginica
Truth	Setosa	0	1	1
	Versicolor	1	0	1
	Virginica	10	10	0

empirical risk minimization

- One way to define the problem of model fitting or training model
  - to find a setting of the parameters that minimizes the empirical risk on the training sete

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, L(\theta) = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} l(y_n \neq f(x_n, \theta))$$

- our true goal
  - to minimize the expected loss on future data that we have not yet seen.
  - we want to generalize, rather than just do well on the training set.

Uncertainty

- In many cases, not be able to perfectly predict the exact output given the input
  - due to lack of knowledge of the input-output mapping (**epistemic uncertainty or model uncertainty**)
  - due to intrinsic stochasticity in the mapping (**aleatoric uncertainty or data uncertainty**).
- We can capture our uncertainty using the conditional probability distribution:

$$p(y = c|\boldsymbol{x}; \boldsymbol{\theta}) = f_c(\boldsymbol{x}; \boldsymbol{\theta})$$

  - where f : $\boldsymbol{X} \rightarrow [0, 1]^C$ maps inputs to a probability distribution over the C possible output labels.
  - $0 \leq f_c \leq 1$ for each c, and $\sum_{c=1}^{C} f_c = 1$

softmax function

- To avoid restriction as the probability distribution, We can use the softmax function

$$\mathrm{S}(\boldsymbol{a}) = \left[\frac{e^{a_1}}{\sum_{c=1}^{C} e^{a_1}}, \cdots, \frac{e^{a_C}}{\sum_{c=1}^{C} e^{a_C}}\right]$$

  - where S(a) : $\mathbf{R}^{\mathrm{C}} \rightarrow [0, 1]^{\mathrm{C}}$ maps inputs to a probability distribution over the C possible output labels
  - $0 \leq S(a)_c \leq 1$ for each c, and $\sum_{c=1}^{C} S(a)_c = 1$

- if a = f(x; θ), We thus define the overall model as follows:

$$p(y = c|\boldsymbol{x}; \boldsymbol{\theta}) = S_c(f(\boldsymbol{x}; \boldsymbol{\theta}))$$

Maximum likelihood estimation

- likelihood function: $p(y_n = c|\boldsymbol{x_n}; \boldsymbol{\theta}) = f_c(\boldsymbol{x_n}; \boldsymbol{\theta})$

- A common loss function for fitting probabilistic models
  - **negative log probability** $l(y, f(\boldsymbol{x}; \boldsymbol{\theta})) = -\log p(y, f(\boldsymbol{x}; \boldsymbol{\theta}))$

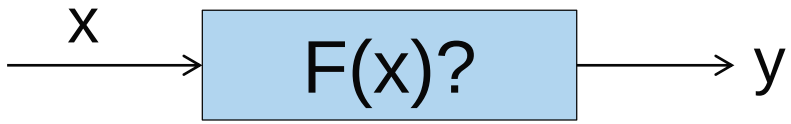
- **negative log likelihood**
  - The average negative log probability (empirical risk):

$$\mathrm{NLL}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N} \log p(y_n, f(\boldsymbol{x_n}; \boldsymbol{\theta}))$$

- To minimize this loss equal to compute **the maximum likelihood estimate**:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, NLL(\theta) = \underset{\theta}{\operatorname{argmax}} \frac{1}{N}\sum_{n=1}^{N} logp(y_n, f(x_n, \theta))$$

Classification methods



1. *discriminant function* **methods :** F(x) ?

2. Probabilistic *generative models:* $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$

3. Probabilistic *discriminative models:* p(y / x)?

Supervised Learning

Regression

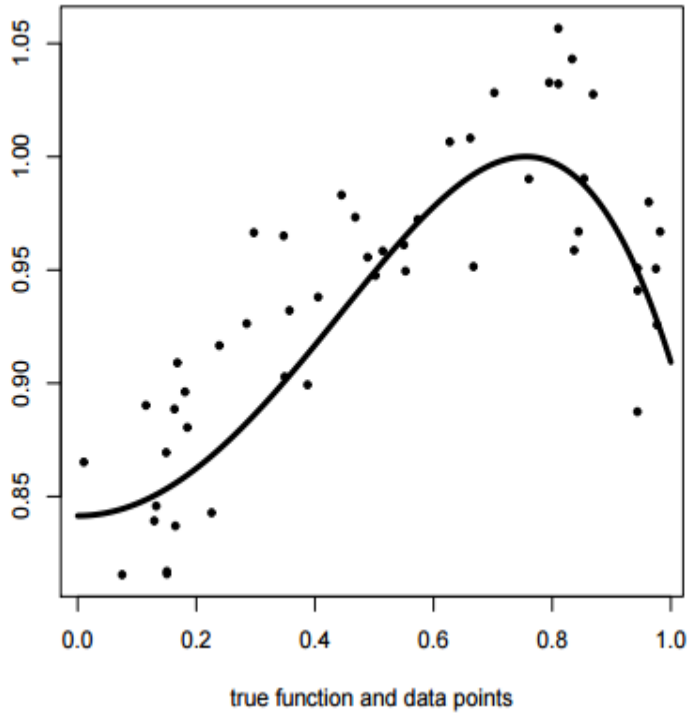
idea of regression

- Regression is to predict a real-valued quantity y∈R
  - It is very similar to classification.
  - since the output is real-valued, we need to use a different loss function.
- the most common loss function : **quadratic loss, or l_2 loss:**

$$l_2(y, \hat{y}) = (y - \hat{y})^2$$

- empirical risk is equal to the **mean squared error**:

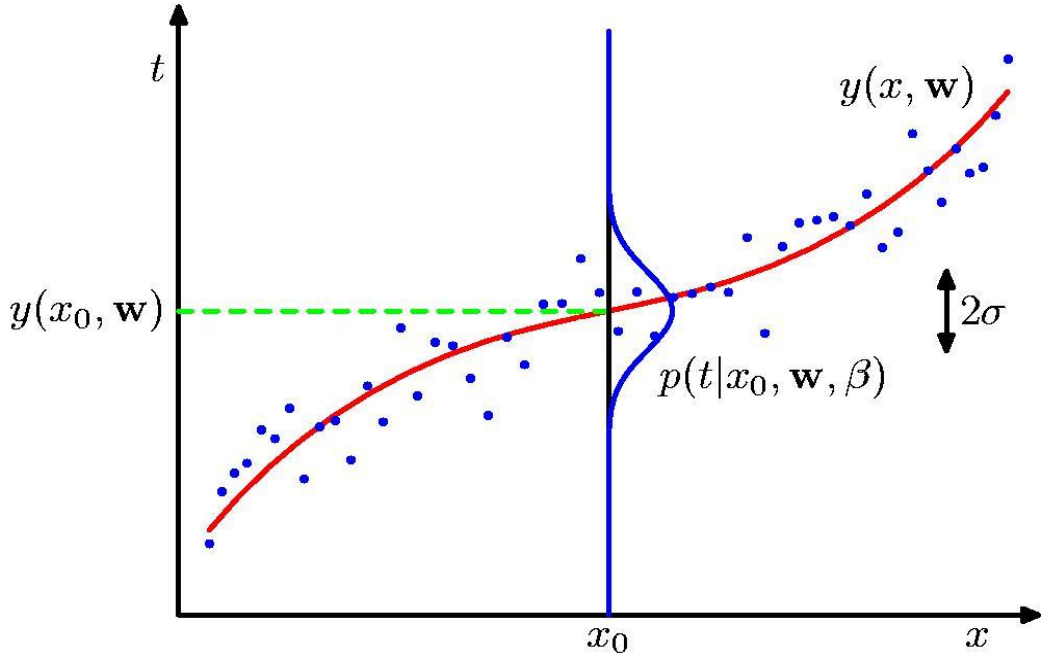
$$\mathrm{MSE}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}(y - \hat{y})^2$$



probabilistic perspective mean square error

- it is common to assume the output distribution is a Gaussian in regression

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = N(y|f(\boldsymbol{x};\boldsymbol{\theta}),\sigma^2)$$



- the negative log likelihood becomes

$$\mathrm{NLL}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N}\log\left[\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{1}{2}} exp\left(-\frac{1}{2\sigma^2}\left(y_n - f(x_n,\theta)\right)^2\right)\right] = \frac{1}{2\sigma^2}MSE(\theta) + \mathrm{const}$$

Linear Regression

- The form of linear regression
  - $\emptyset(x)$: base functions (**feature extractor**)

$$f(x, \theta) = \mathbf{w}^T \phi(x) = \sum_{1 \le i \le D} w_i \varphi(x_i)$$

- multiple linear regression

$$f(\mathbf{x}, \theta) = \mathbf{w}^T \mathbf{x} = \sum_{1 \le i \le D} w_i x_i$$

- The polynomial regression

$$f(x, \theta) = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d$$

deep neural network

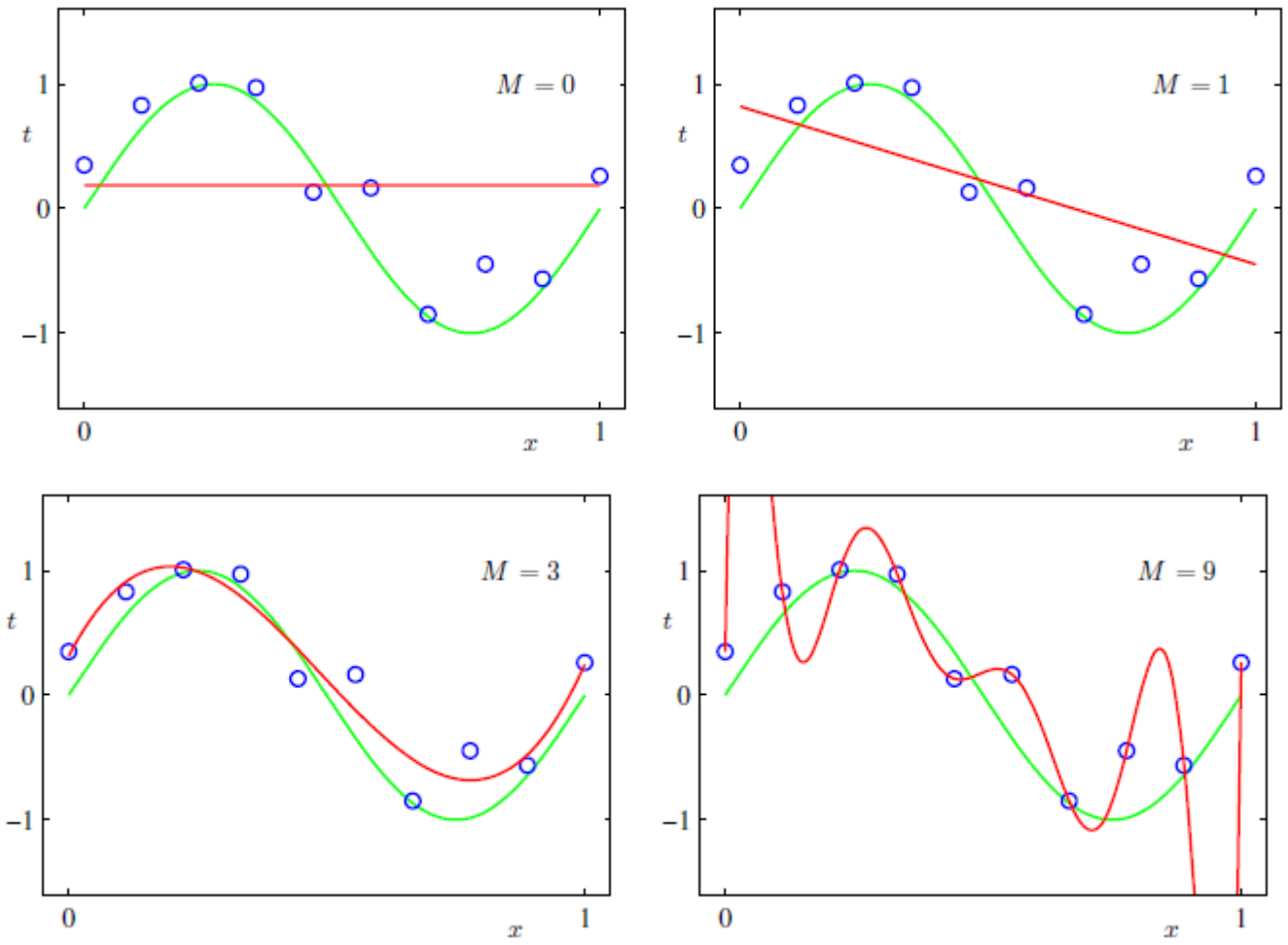
- let the feature extractor ∅(x) have its own set of parameters :

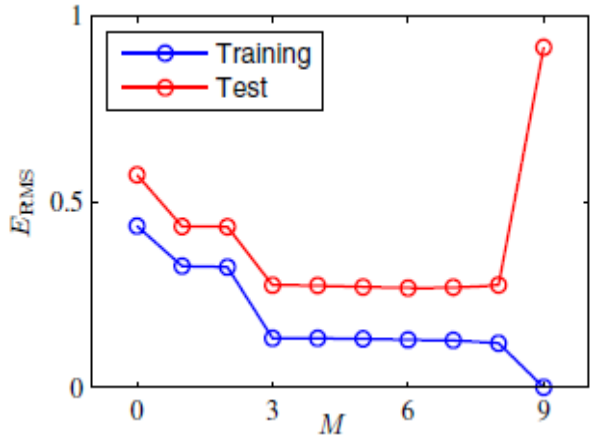
$$f(x, \omega, V) = \omega^T \phi(x, V)$$

- the key idea behind deep neural networks
  - recursively decompose Φ(x;V) into a composition of simpler functions.
    - ❿The resulting model then becomes a stack of L nested functions:

$$f(\mathbf{x}, \theta) = f_L(f_{L-1}(\ldots(f_1(x))\ldots))$$

Overfitting and generalization





No free lunch theorem

- All models are wrong, but some models are useful
- There are the large variety of models, it is natural to wonder which one is best.
  - Unfortunately, there is no single best model that works optimally for all kinds of problems
- a set of models that works well in one domain may work poorly in another.

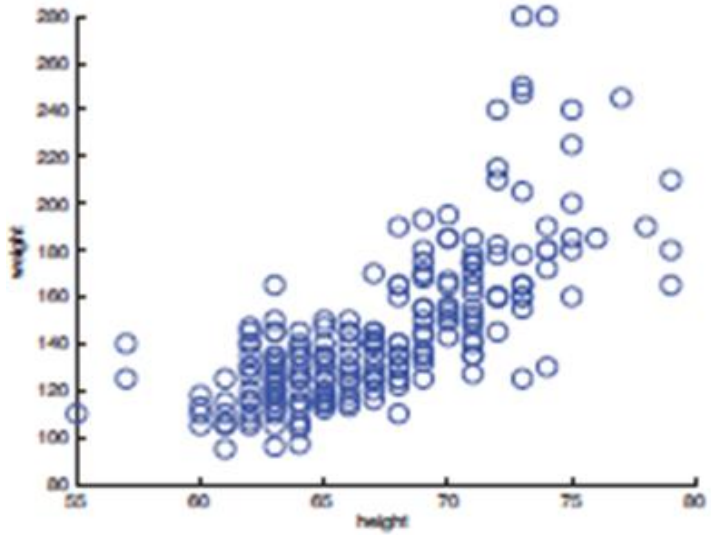
How to pick suitable model

- The best way to pick a suitable model
  - based on domain knowledge,
  - trial and error
  - Bayesian methods

Unsupervised Learning

Unsupervised Learning

- An arguably much more interesting task
  - to try to “make sense of” data
  - That is, we just get observed “inputs” $D = \{x_n : n = 1 : N\}$ without any “outputs” y_n.
- From a probabilistic perspective, the task of unsupervised learning
  - fitting an unconditional model of the form p(x),
  - The model can generate new data x

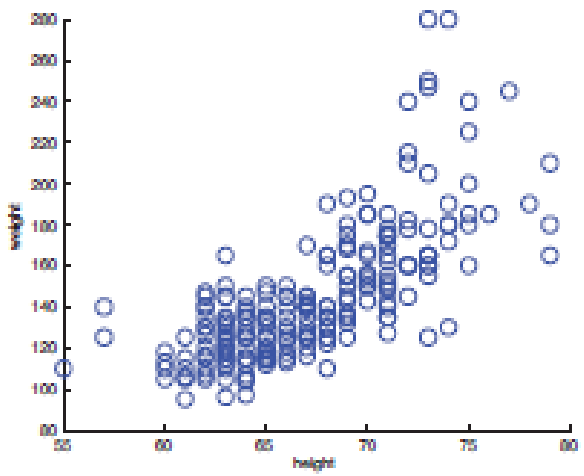


Feature of Unsupervised Learning

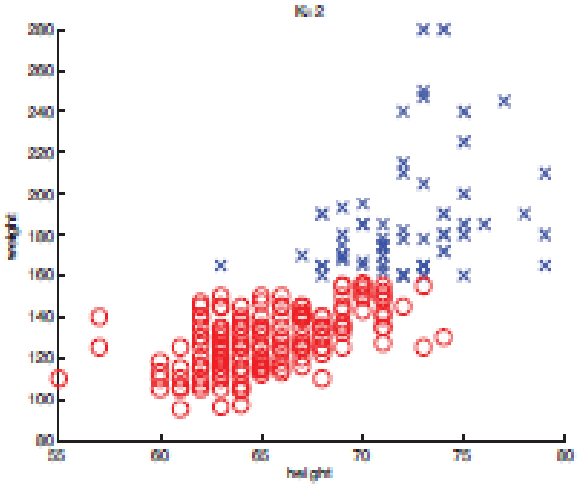
- It avoids the need
  - to collect large labeled datasets for training
  - to learn how to partition the world into often arbitrary categories.
- It forces the model
  - to “explain” the high-dimensional inputs, rather than just the low-dimensional outputs.
- This allows us to learn richer models of “how the world works”.

Clustering

- One task of unsupervised learning is the problem of finding clusters in data.
  - The goal is to partition the input into regions that contain “similar” points.
- there is no correct number of clusters
  - we need to consider the tradeoff between model complexity and fit to the data
- As an example :
  - People’s HeightWeight dataset
  - the data points without any class labels



(a)



(b)

Discovering latent factors

- it is often useful to reduce the dimensionality of data
  - which captures the “essence” of the data.
- One approach to this problem is to assume
  - high-dimensional data $x_n \in R^D$ was generated by low-dimensional **latent factors** $z_n \in R^K$
  - We can represent the model diagrammatically: $z_n \rightarrow x_n$
- we use a linear model
  - factor analysis: $p(x_n|z_n;\boldsymbol{\theta}) = N(x_n|wz_n+\mu,\sum)$
  - principal components analysis (PCA): $p(x_n|z_n;\boldsymbol{\theta}) = N(x_n|wz_n+\mu, \sigma^2\mathbf{I})$
- nonlinear extensions : $p(x_n|z_n;\boldsymbol{\theta}) = N(x_n|f(z_n,\boldsymbol{\theta}), \sigma^2\mathbf{I})$, where $f(z,\boldsymbol{\theta})$: nonlinear
  - **variational autoencoder**: an approximate method

PCA Image Demo

- **a) 25 randomly chosen 64 × *64 pixel images .***
- ***(b) The mean*** and the first three principal component basis vectors (eigenfaces).
- Figure generated by PCA ImageDemo.

(a)

(b)

Discovering graph structure

- For a set of correlated variables, we discover which ones are most correlated with which others.
  - This can be represented by a graph *G*
- We learn this graph structure from data
- there are two main applications for learning sparse graphs:
  - to discover new knowledge,
  - to get better joint probability density estimators.
- A example is predicting traffic jams on the freeway.

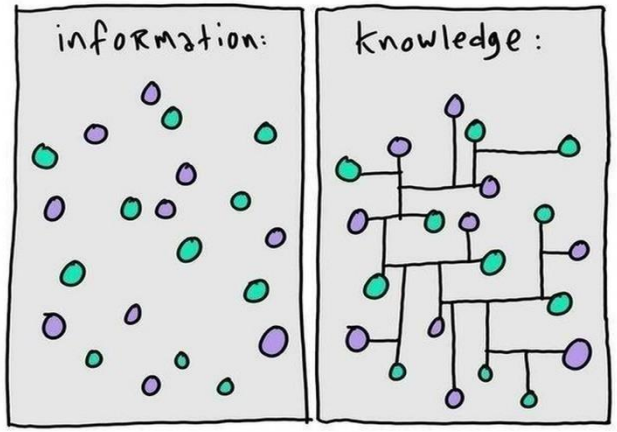


Image inpainting

- **The goal is** to “fill in” holes
- for example, we denoise the image, as well as impute the pixels hidden behind the occlusion.
- we can build a joint probability model of the pixels
  - given a set of clean images,
  - inferring the unknown variables (pixels) given the known variables (pixels).

Self-supervised learning

- we create **proxy supervised tasks** from unlabeled data.
- For example
  - to learn to predict a color image from a grayscale image,
  - to mask out words in a sentence and then try to predict them given the surrounding context
- The hope
  - $x'_1 = f(x_2, \boldsymbol{\theta})$, where x_2 is the observed input and x'_1 is the predicted output,
    - ❿ it will learn useful features from the data
  - that can then be used in standard, downstream supervised tasks

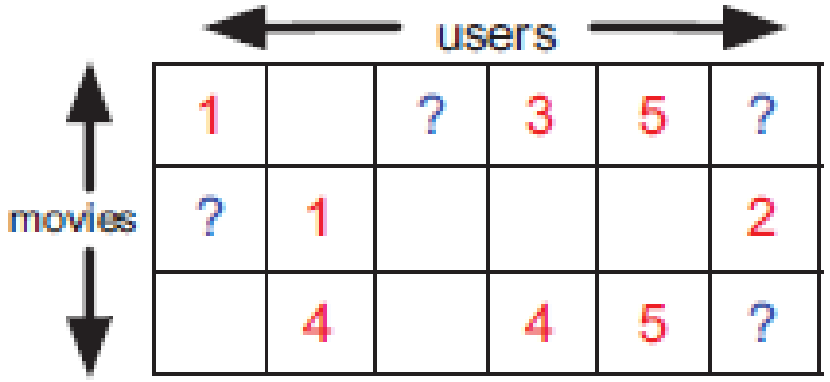
Matrix completion

- Sometimes we have missing data, whose values are unknown.
- For example,
  - we conducted a survey, some people not answered certain questions.
  - we might have various sensors, some of which fail.
- The corresponding design matrix will then have “holes” in it;
- The **goal** is to infer plausible values for the missing entries.

Collaborative filtering

- predicting which movies people will want to watch
  - based on how they have rated movies which they have already seen.
- we have a ***rating*** matrix **X** (1 is dislike and 5 is like)
- most of the entries in **X will be unknown, since most users will not have rated** most movies.

- Training data is in red
- test data is denoted by ?



Market basket analysis

	item 1	item j	item n
Transaction i		1	

- *Many items are purchased together (e.g., bread and butter),*
- Given a bit vector, representing a subset of items that the consumer has bought,
- the goal is to predict which other bits are likely for the consumer to buy.
- It is common to solve such tasks using
  - frequent itemset mining, which create association rules
  - a joint density model $p(x1, . . . , x_n)$ *to the bit vectors,*

Evaluating unsupervised learning

- it is very hard to evaluate the quality of the output of an unsupervised learning method
  - because there is no ground truth
- A common method for evaluating unsupervised models

$$L(\boldsymbol{\theta}, \boldsymbol{D}) = -\frac{1}{|D|}\sum_{x \in D} \log p(x, \theta)$$

- This treats the problem of unsupervised learning as one of **density estimation**.
- The idea is that a good model
  - assigns high probability to regions of data space where the data samples come from,
  - implicitly assigns low probability to the regions where the data does not come from.

An alternative evaluation metric

- to use the learned unsupervised representation as features or input
  - to a downstream supervised learning method.
  - using much less labeled data than when working with the original features.

reinforcement Learning

reinforcement Learning

- Agent can obervate environment
- Agent can take actions
  - that affect the environment
- Agent can know how it’s actions affect the environment
  - Through reward function

Parametric vs non-parametric models

Parametric models

- For probabilistic models of the form *p(y|x) or p(x)*
- What is parametric models:
  - It has a fixed number of parameters,
  - Example, Gaussian distribution: $f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp(\frac{(1-\mu)^2}{2x^2})$
- Advantage and disadvantage:
  - it has the advantage of often being faster to use,
  - the disadvantage of making stronger assumptions about the nature of the data distributions.

non-parametric models

- For probabilistic models of the form *p(y|x) or p(x)*
- What is non-parametric models:
  - the number of parameters grow with the amount of training data,
- Advantage and disadvantage :
  - it is more flexible,
  - often computationally intractable for large datasets.

A simple non-parametric classifier

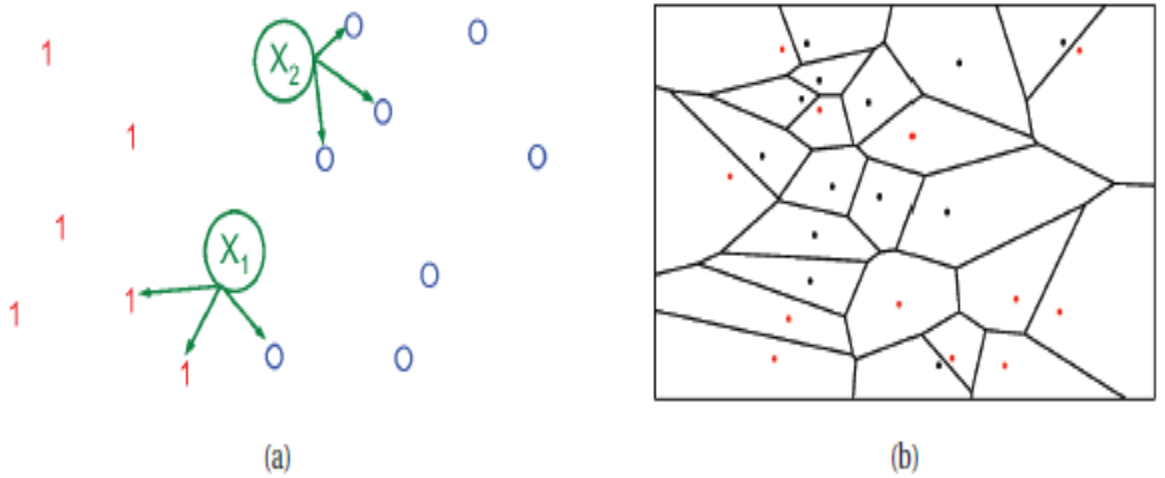
K-Nearest-Neighbors(KNN)

- "looks at" the *K points in the training set nearest to the test* ***x,***

$$p(y=c \mid x, D, K) = \frac{1}{K} \sum_{i \in N_K(x)} I(y_i = c) \qquad I(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases}$$

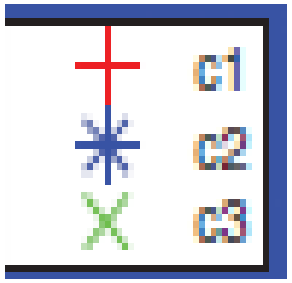
- We also need to decide how to measure "nearness".
- If the inputs are numeric, we might just use Euclidean distance.
- Big question: How should we choose K ?

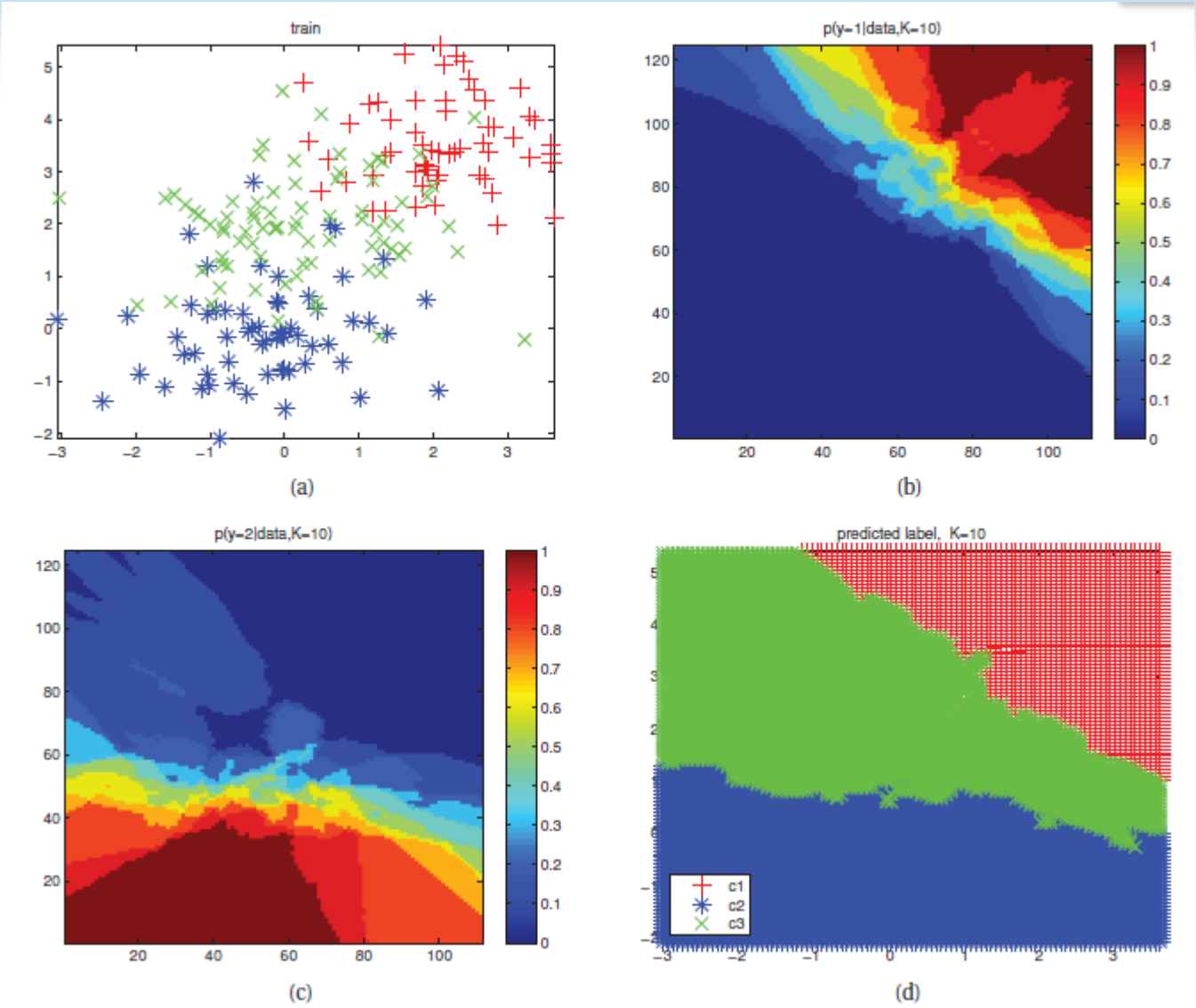
Example of KNN



- (a) K=3
- (b) K=1

Example of KNN

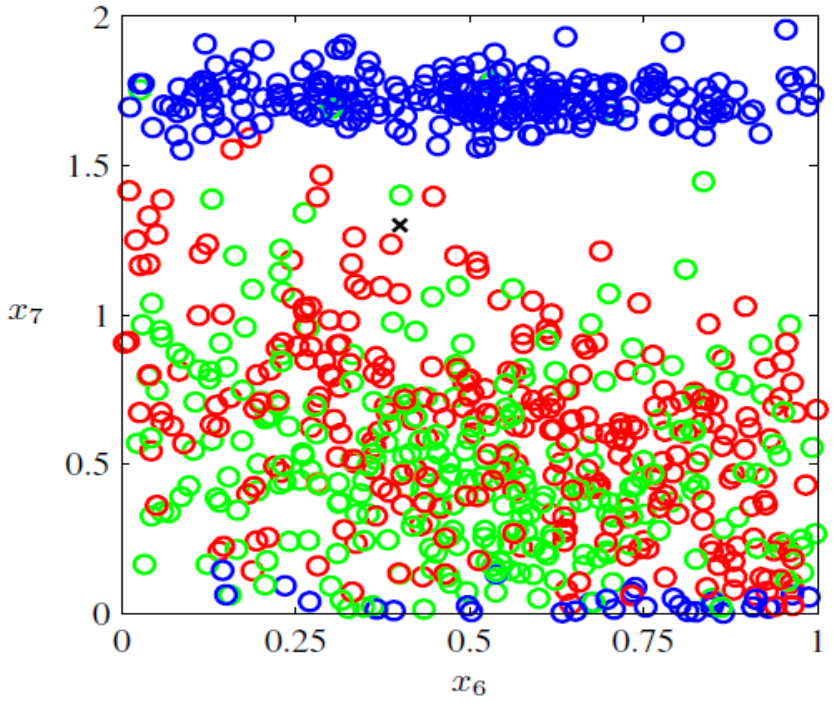


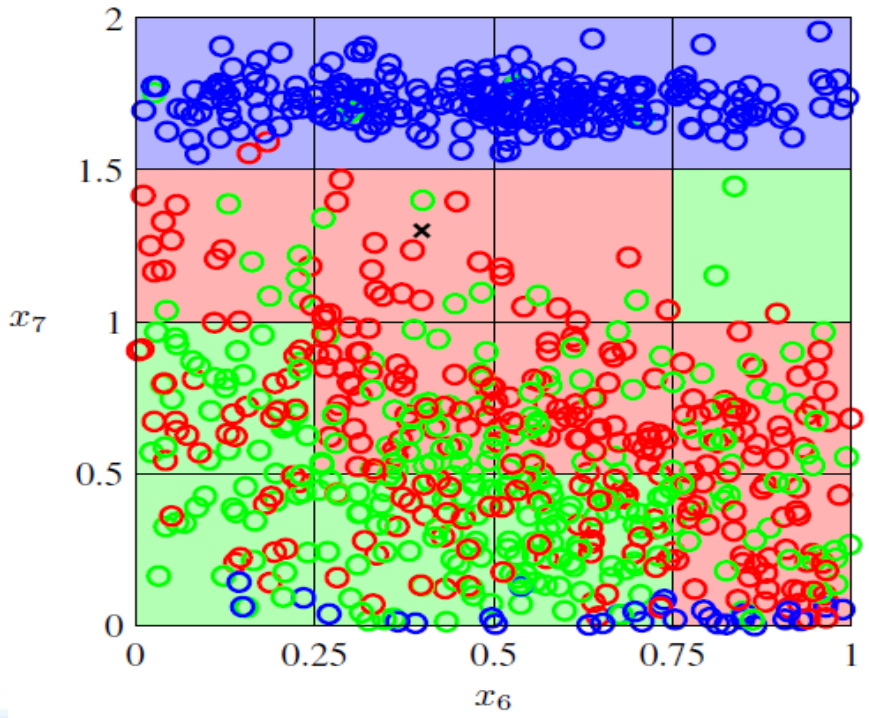


The curse of dimensionality

Analysis a classification problem

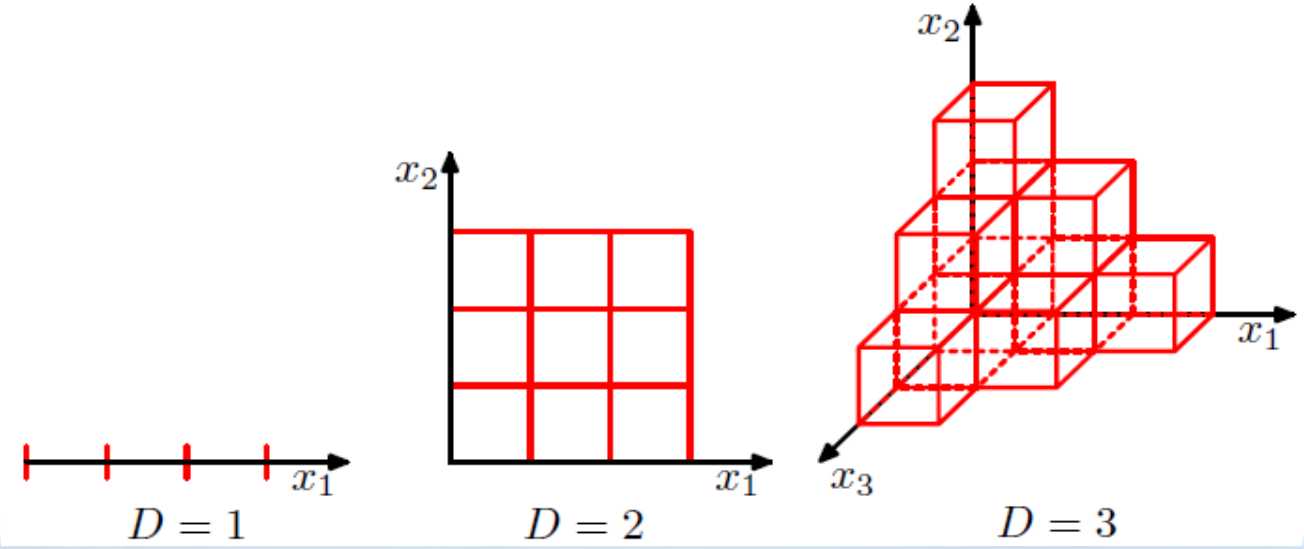
- Our goal： to classify the new point ‘×’ *by* KNN
- The intuition： the identity of ‘×’ should be determined
  - more strongly by nearby points from the training set.





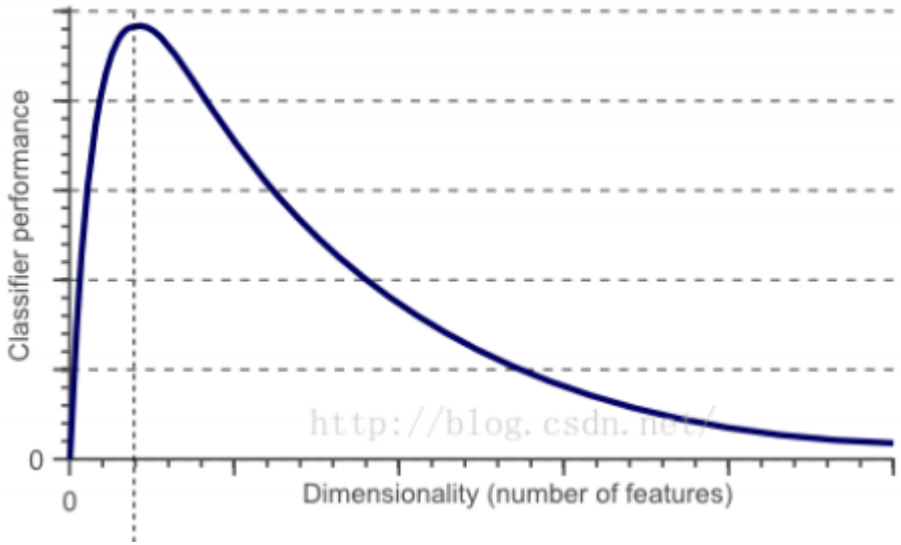
the most severe problem

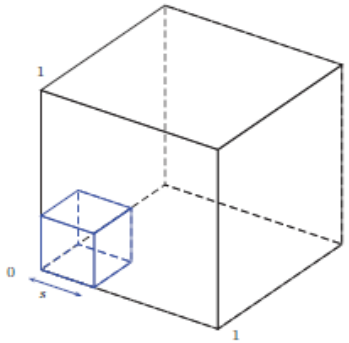
- Consider the input spaces of higher dimensionality.
- the number of such cells grows exponentially
- The most severe problem
  - we would need an exponentially large quantity of training data in order to ensure that the cells are not empty.



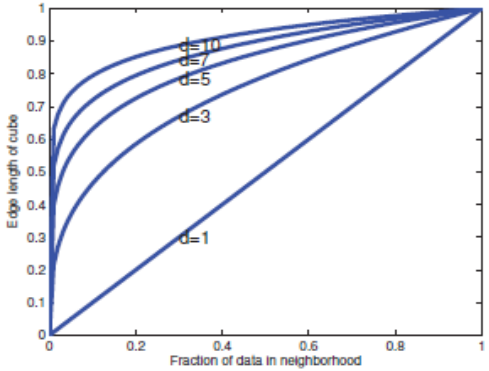
The curse of dimensionality

- KNN classifiers do not work well with high dimensional inputs.
- Suppose we estimate the density of class labels around **x**
  - **“growing” a** hyper-cube around **x** until it contains a desired fraction *of the datas.*





(a)



(b)

Combat curse of dimensionality

- The main way
  - to make some assumptions about the the nature of the data distribution

    ⑩(***p(y|x) or p(x)***)
  - These assumptions, known as **inductive bias,**
  - **They** are often embodied in the form of a parametric model

Parametric models

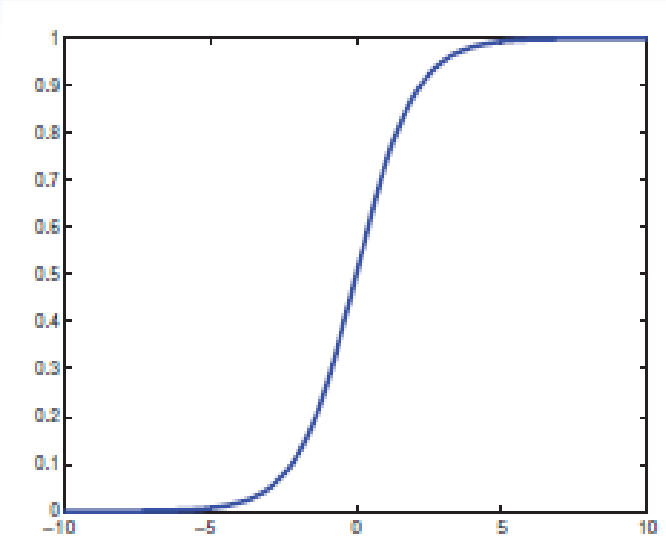
a parametric model

- **a statistical model with a fixed number of parameters**
- two widely used examples of a parametric model
  - **Linear regression**
  - **Logistic regression**

Logistic Regression

sigmoid function

$$sigm(x) = \frac{1}{1+\exp(-x)} = \frac{e^x}{1+e^x}$$



- Sigm() : **logistic function** or **sigmoid function**.
- x>0.5 → y=1;
- x<-0.5 → y=0
- -0.5<x<0.5 y=0.5

Logistic regression

- We can use linear regression for the classification problem
- change the form for hypotheses: y=sigm($w^T x$)
- we replace the Gaussian distribution for *y with a* ***Bernoulli distribution***.

$$p(y \mid x, w) = Ber(y \mid \mathrm{sigm}(\mathrm{w}^{\mathrm{T}}\mathrm{x}))$$

- where ***p(y = 1|x) = sigm($w^T x$)***

  p(y = 0|x) = 1- p(y = 1|x)

Model selection

Model selection

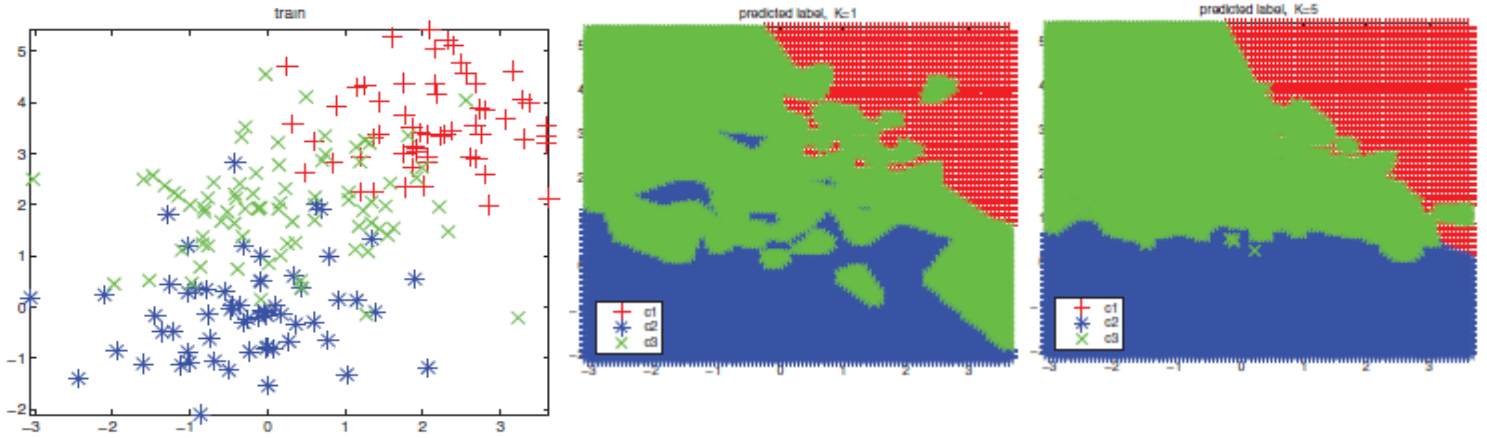
- if we have a variety of models of different complexity
  - e.g., linear or logistic regression models with different degree polynomials
  - or KNN classifiers with different values of *K*
- How should we pick the right one?

A natural approach of Model selection

- to compute the **misclassification rate** for each method

$$err(f, D) = \frac{1}{N}\sum_{i=1}^{N} I(f(x_i) \neq y_i)$$

- Example:



- We see that increasing *K increases our error rate on the training set*
- we can get minimal error on the training set by using *K = 1*
- However, what we care about is **generalization error**

generalization error

- **generalization error:** the misclassification rate over future data
- This can be approximated by
  - Computing the misclassification rate on a large independent *test set*.
- ***an obvious way to pick K :***
  - *to pick* the value with the minimum error on the test set .
- Unfortunately, when training the model, we don’t have access to the test set

cross validation

- partitioning the training set into two: the one for training, the second for testing, called the **validation set**
- **cross validation (CV):**