



Principles and Methodology of Machine Learning

David Gianazza

Ecole Nationale de l'Aviation Civile
Bureau 008, bât. Z
gianazza@recherche.enac.fr

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

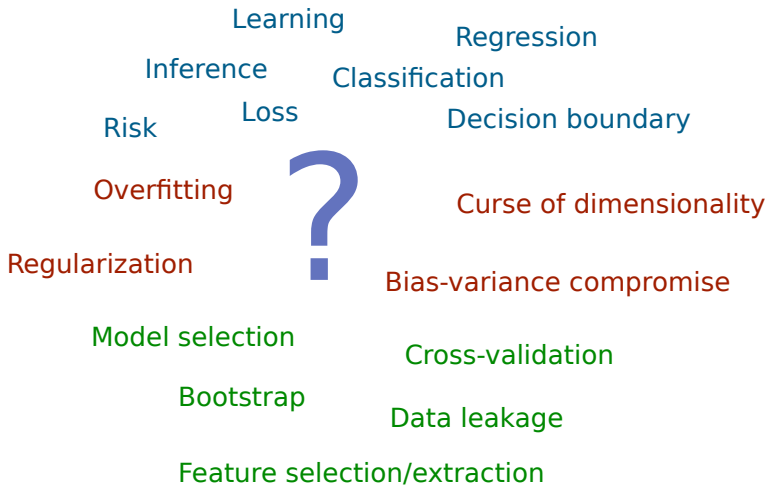
4. Illustration with linear models

5. Methodology

6. Unsupervised learning

7. Reinforcement learning

Machine Learning ??



Objectives of this unit (Principles and methodology)

- ▶ Acquire some theoretical background before applying methods
- ▶ Know basic ML notions (inference, risk, loss)
- ▶ Discover some ML pitfalls (overfitting, curse of dimensionality)
- ▶ Understand the context of application of ML methods
- ▶ Acquire the methodology of ML

Resources

Useful books



G. JAMES, D. WITTEN, T. HASTIE, AND R. TIBSHIRANI, *An introduction to statistical learning with applications in R*, Springer, 2013.



T. HASTIE, R. TIBSHIRANI, AND J. H. FRIEDMAN, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.



C. M. BISHOP ET AL., *Pattern recognition and machine learning*, vol. 4, springer New York, 2006.

Online courses

- ▶ Stanford MOOC on statistical learning

<https://lagunita.stanford.edu/courses/HumanitiesScience/StatLearning/Winter2014/about>

- ▶ Wikistat (in french) <http://wikistat.fr/>

ENAC e-campus

Principles and Methodology of Machine Learning

1. Introduction

- Learning from data
- Reminders on probabilities

2. Supervised learning

3. Principles of inference

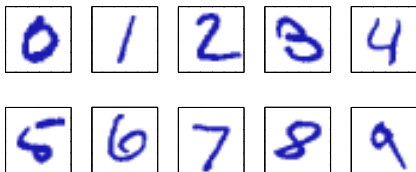
4. Illustration with linear models

5. Methodology


6. Unsupervised learning

7. Reinforcement learning

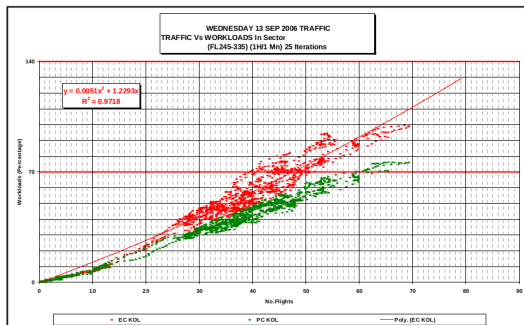
Example: handwritten digit recognition



Source image : C. Bishop, *Pattern Recognition and Machine Learning*

- ▶ Task : assign bitmap entries to the most plausible category ('0'-'9')
- ▶ Data : examples of known bitmap/digit associations
(e.g.  \longrightarrow '6')
- ▶ Objective: minimise the risk of prediction error

Example: curve fitting (traffic vs. workload)



Source image : Eurocontrol

- ▶ Task : compute air traffic controller workload for a given traffic
- ▶ Data : examples of known traffic/workload associations
- ▶ Objective: minimise the risk of prediction error

Learning

The process of getting an understanding of something by studying it or by experience (Cambridge dictionary)



Charles Sanders Peirce (1903): Three types of inference

Deduction:

- ▶ All the beans in this bag are white
- ▶ These beans are from this bag
- ▶ These beans are white

Abduction:

- ▶ All the beans in this bag are white
- ▶ These beans are white
- ▶ These beans are from this bag

Induction:

- ▶ These beans are from this bag
- ▶ These beans are white
- ▶ All the beans in this bag are white

Induction:

Inference in Machine Learning

In machine learning: Inference by induction

Definition of induction:

- ▶ Inference of a generalized conclusion from particular instances (Merriam-Webster dictionary)
- ▶ The process of discovering a general principle from a set of facts (Cambridge dictionary)

Machine learning finds generalizable predictive patterns from a set of instances

Learning from data

Learning and generalization

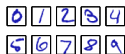
- ▶ **Observe** a phenomenon (collect data)
- ▶ **Fit a model** on the collected data and **assess its performances**
- ▶ **Generalize** (i.e. use the fitted model on fresh inputs)


Remarks:

- ▶ **We want models that generalize well**, not simply models that fit the observed data!
- ▶ **A model can only be as good as the data used to build it**

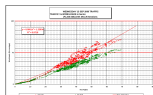
Several types of learning tasks

Handwritten digit recognition



- ▶ Assign input x (a bitmap ) to a category y (digit '6'). The target variable y is **discrete**.
- ▶ This is a **classification problem**

Workload prediction



- ▶ The target variable y for the output is **continuous** ($y \in \mathbb{R}$)
- ▶ This is a **regression problem**

Several types of learning tasks

- ▶ Supervised learning, using examples $\{(x_1, y_1), \dots, (x_N, y_N)\}$
 - ▶ Classification : y_n is a label $\mathcal{C}_1, \dots, \mathcal{C}_p$
E.g. digit recognition
 - ▶ Regression : $y_n \in \mathbb{R}$ or $y_n \in \mathbb{R}^P$
E.g. curve fitting, surface fitting
- ▶ Unsupervised learning, using examples $\{x_1, \dots, x_N\}$
 - ▶ Clustering: *cluster data according to similarity (distance)*
 - ▶ Density estimation: *What is the data distribution ?*
- ▶ Reinforcement learning *action* \rightarrow *reward*.
Objective: Maximize the reward cumulated over time

Several ways to use the data

► *Batch learning*

- Collect all examples
- Adjust the model on the examples
- Use the adjusted model to predict outputs for fresh instances

► *Online learning*

- Consider one instance x_n at a time
- Predict \hat{y}_n for this instance, using the current model
- Discover the true value y_n
- Refine the model, considering the error $\hat{y}_n - y_n$

Supervised learning

$$y = f(x) + \text{obs. noise} \quad ?$$

- ▶ f is unknown!
- ▶ We only have examples $(x_1, y_1), \dots, (x_N, y_N)$

f is approximated by $h \in \mathcal{H}$ (of known analytical expression)

$$y = h(x) + \epsilon \quad \text{where } h \in \mathcal{H}$$

We choose $h \in \mathcal{H}$, minimizing a **loss** $\ell(y, h(x))$

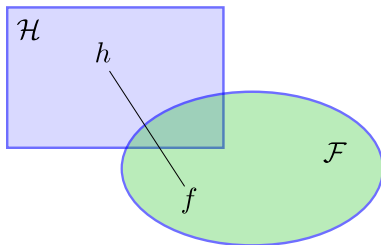
Loss: a cost associated to the error between

- ▶ the computed output $\hat{y} = h(x)$
- ▶ and the desired response y

Choice of \mathcal{H} ? Choice of ℓ ? What error should we minimize?

Vocabulary

- ▶ $x \in \mathcal{X}$: feature, explanatory variable, input
- ▶ $y \in \mathcal{Y}$: response or target variable, desired output, dependant variable, label (for classification)



- ▶ $h \in \mathcal{H}$: hypothesis (“model”)
- ▶ $f \in \mathcal{F}$: target (unknown)

What do we learn exactly?

What is the nature of f and h ?

Functions, probability distributions, boolean expressions, programs, or any other concept you might want to learn

Nature of x and y ?

Real values, integers, boolean values, random variables, etc.

Depends on what you want to learn, and on your choice of models

How to learn from data?

Choose loss function ℓ

- ▶ according to a principle of inference
(maximum likelihood, maximum *a posteriori*, bayesian inference, minimum description length, *etc.*)
- ▶ or arbitrarily (e.g. loss matrix for cancer detection)

Choose model \mathcal{H} (E.g. linear model, neural network, etc)

Minimize risk (*i.e.* loss expectation)

$$\mathcal{R}(h) = \mathbb{E}_{X,Y}[\ell(Y, h(X))] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, h(x)) \mathbb{P}_{X,Y}(dx, dy)$$

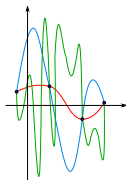
Problem: we only have a finite dataset $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$

Issues and pitfalls

We only have a finite dataset $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$

- Is there a universally applicable best model \mathcal{H} ?

No! See Wolpert's "No Free Lunch" theorems



$Generalization = Data + Knowledge$

Choose \mathcal{H} based on *a priori* assumptions and/or what you know of the problem

- **Overfitting** issues when minimizing **empirical risk**

$$\mathcal{R}_{emp}(h, S) = \frac{1}{|S|} \sum_{(x_n, y_n) \in S} \ell(y_n, h(x_n))$$

- **Curse of dimensionality**

Methods performance \searrow when $\dim \mathcal{X} \nearrow$

Workarounds?

Control overfitting

- ▶ Regularization

Example: Minimize $\mathcal{E}_\lambda(\mathbf{w}) = \sum_{n=1}^N (y_n - h_{\mathbf{w}}(x_n))^2 + \lambda \|\mathbf{w}\|^2$

- ▶ Model selection, using

- ▶ K -fold cross-validation, bootstrap

- ▶ Theoretical bounds on $|\mathcal{R}_{emp}(h, S) - \mathcal{R}(h)|$

See Vapnik's statistical learning theory

- ▶ Information criteria: Akaike's information criterion AIC, Schwartz's bayesian information criterion BIC, etc.

Mitigate risk

- ▶ Bayesian inference: Average over all hypotheses h in \mathcal{H} (weighted by probabilities) to compute output for a fresh input

Workarounds?

Reduce dimensionality of inputs

- ▶ Selection of relevant variables
Scoring, forward or backward selection, etc.
- ▶ Extraction of new variables
Principal component analysis, autoencoding neural networks, etc.

Explore subspaces of lower dimension

- ▶ Taking advantage of sparsity
- ▶ Using dictionaries of basis functions

Local interpolation (assuming f smooth enough)

E.g. K -nearest neighbors, classification and regression trees

Principles and Methodology of Machine Learning

1. Introduction

- Learning from data
- Reminders on probabilities

2. Supervised learning

3. Principles of inference

4. Illustration with linear models

5. Methodology

6. Unsupervised learning

7. Reinforcement learning

Reminders of probability theory

Discrete random variable X

Probability law (or distribution):

$$\mathbb{P}_X : x \rightarrow \mathbb{P}(X = x)$$

Example: let X be the sum obtained when rolling two dice

$$\mathbb{P}_X : 2 \rightarrow \frac{1}{36}$$

$$3 \rightarrow \frac{2}{36}$$

$$4 \rightarrow \frac{3}{36}$$

$$\vdots$$

$$12 \rightarrow \frac{1}{36}$$

Reminders of probability theory

Continuous random variable X

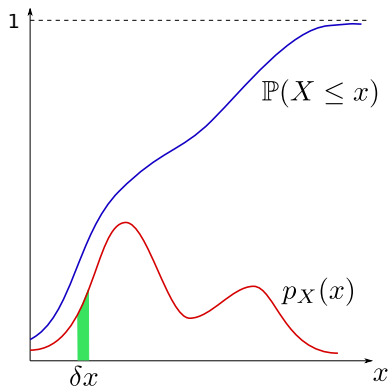
Probability density p_X (if it exists):

$$\forall t \quad p_X(t) \geq 0 \quad \text{and} \quad \int_{-\infty}^{+\infty} p_X(t) dt = 1$$

Probability:

$$\mathbb{P}(X \leq x) = \int_{-\infty}^x p_X(t) dt$$

Density and probability



$$\mathbb{P}(X \leq x) = \int_{-\infty}^x p_X(t) dt$$

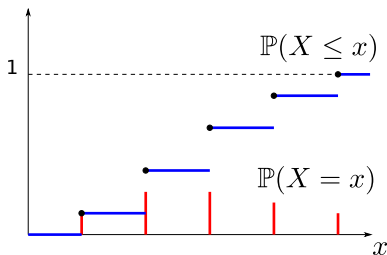
$$\forall x \quad \mathbb{P}(X = x) = 0$$

$$\mathbb{P}(X \in]x, x + \delta x]) = \int_x^{x+\delta x} p_X(t) dt$$

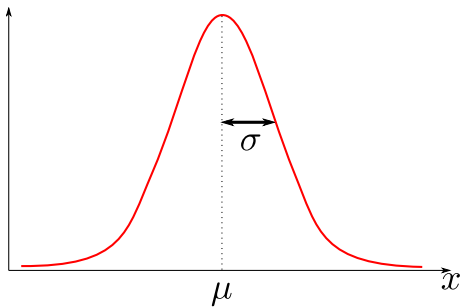
Density and probability

For a discrete variable:

$$\begin{aligned}\mathbb{P}(X \leq x) &= \sum_{x_i \leq x} \mathbb{P}(X = x_i) = \sum_i \mathbb{P}(X = x_i) \mathbb{1}_{]-\infty, x]}(x_i) \\ &= \int_{-\infty}^x \underbrace{\mathbb{P}(X = t) \mathbb{1}_{]-\infty, x]}(t)}_{\text{density } p_X(t)} dt\end{aligned}$$



Gaussian law $\mathcal{N}(\mu, \sigma^2)$



$$p_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{1}{2} \frac{(x - \mu)^2}{\sigma^2} \right\}$$

Parameters: mean μ , variance σ^2

Expectation

- ▶ Discrete variable:

$$\mathbb{E}(f) = \sum_x f(x)P(x)$$

- ▶ Variable with density:

$$\mathbb{E}(f) = \int_{\mathcal{X}} f(x)p(x)dx$$

Variance, covariance

Covariance :

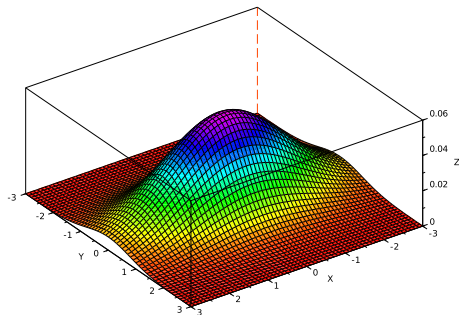
$$\begin{aligned} \text{Cov}(x, y) &= \mathbb{E} \left[\left(x - \mathbb{E}(x) \right) \left(y - \mathbb{E}(y) \right) \right] \\ &= \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y] \end{aligned}$$

Variance : $\text{Var}(x) = \text{Cov}(x, x)$

Covariance matrix (when x and y are vectors)

$$\begin{aligned} \text{Cov}(x, y) &= \mathbb{E} \left[\left(x - \mathbb{E}(x) \right) \left(y^T - \mathbb{E}(y^T) \right) \right] \\ &= \mathbb{E}[xy^T] - \mathbb{E}[x]\mathbb{E}[y^T] \end{aligned}$$

Multivariate Gaussian density



$$p(\mathbf{x}) = (2\pi)^{-\frac{P}{2}} |\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right\}$$

- ▶ μ is a vector
- ▶ Σ is a covariance matrix

Reminders of probability theory

Rules of probability:

► Sum rule $\mathbb{P}(X) = \sum_Y \mathbb{P}(X, Y)$

► Product rule $\mathbb{P}(X, Y) = \mathbb{P}(Y|X) \mathbb{P}(X)$

Independence: $\mathbb{P}(X, Y) = \mathbb{P}(X) \mathbb{P}(Y)$

Bayes' theorem:

$$\mathbb{P}(Y|X) = \frac{\mathbb{P}(X|Y) \mathbb{P}(Y)}{\mathbb{P}(X)}$$

Exercise 1.1

Heights (X) of female (F) and male (M) students in a school:

X	165	166	167	168	169	170	171	172	173	174	175	Tot.
F	1	4	7	10	6	4	3	1	2	1	1	40
M	1	0	3	2	5	8	9	13	10	5	4	60
Tot.	2	4	10	12	11	12	12	14	12	6	5	100

1. What is the joint probability that a student is of height $X = 170$ **and** is a F?
2. What is the probability that a student is of height $X = 170$?
3. What is the conditional probability that a student is of height $X = 170$ **knowing that** she is a F?
4. What is the probability that a student is a F?
5. What is the probability that a student is a F **knowing that** $X = 170$?

Exercise 1.1

Heights (X) of female (F) and male (M) students in a school:

X	165	166	167	168	169	170	171	172	173	174	175	Tot.
F	1	4	7	10	6	4	3	1	2	1	1	40
M	1	0	3	2	5	8	9	13	10	5	4	60
Tot.	2	4	10	12	11	12	12	14	12	6	5	100

What is the joint probability that a student is of height $X = 170$ **and** is a F?

What is the probability that a student is of height $X = 170$?

Exercise 1.1

Heights (X) of female (F) and male (M) students in a school:

X	165	166	167	168	169	170	171	172	173	174	175	Tot.
F	1	4	7	10	6	4	3	1	2	1	1	40
M	1	0	3	2	5	8	9	13	10	5	4	60
Tot.	2	4	10	12	11	12	12	14	12	6	5	100

What is the conditional probability that a student is of height $X = 170$ **knowing that** she is a F?

What is the probability that a student is a F?

Exercise 1.1

Heights (X) of female (F) and male (M) students in a school:

X	165	166	167	168	169	170	171	172	173	174	175	Tot.
F	1	4	7	10	6	4	3	1	2	1	1	40
M	1	0	3	2	5	8	9	13	10	5	4	60
Tot.	2	4	10	12	11	12	12	14	12	6	5	100

What is the probability that a student is a F **knowing that** $X = 170$?

The Bayes' theorem

The Bayes' theorem can be rewritten as follows:

$$\mathbb{P}(Y|X) = \frac{\mathbb{P}(X|Y) \mathbb{P}(Y)}{\sum_Y \mathbb{P}(X|Y) \mathbb{P}(Y)}$$

$\mathbb{P}(Y)$ Prior (initial degree of belief in of Y)

$\mathbb{P}(Y|X)$ Posterior (degree of belief having accounted for X)

$\mathbb{P}(X|Y)$ Conditional probability of X , knowing Y

In Statistics: Likelihood of model of parameter Y to have produced data X

$p(X) = \sum_Y \mathbb{P}(X|Y) \mathbb{P}(Y)$ Normalization factor

Posterior \propto **likelihood** \times **prior**

Example

$$\mathbb{P}(Y|X) = \frac{\mathbb{P}(X|Y) \mathbb{P}(Y)}{\sum_Y \mathbb{P}(X|Y) \mathbb{P}(Y)}$$

- ▶ $X \in \{true, false\}$ is the result of a drug test
- ▶ $Y \in \{\mathcal{C}_1, \mathcal{C}_2\}$ is a category (user, or non-user)

Assume

- ▶ $\mathbb{P}(true | user) = 0.99$, $\mathbb{P}(false | non-user) = 0.99$
- ▶ 0.5% of drug-users in the population

$$\begin{aligned} \mathbb{P}(user|true) &= \frac{\mathbb{P}(true|user) \mathbb{P}(user)}{\mathbb{P}(true|user) \mathbb{P}(user) + \mathbb{P}(true|non-user) \mathbb{P}(non-user)} \\ &= \frac{0.99 * .0005}{0.99 * .0005 + 0.01 * 0.995} \approx 0.33 \end{aligned}$$

Bayes' theorem when X is continuous and Y discrete

Assuming densities p_X exist and $p_{X|Y=C_j}$ exist

$$\mathbb{P}(Y = C_j | X = x) = \frac{p_{X|Y=C_j}(x) \mathbb{P}(Y = C_j)}{p_X(x)}$$

With concise notations:

$$\mathbb{P}(C_j | x) = \frac{p(x | C_j) \mathbb{P}(C_j)}{p(x)} = \frac{p(x | C_j) \mathbb{P}(C_j)}{\sum_k p(x | C_k) \mathbb{P}(C_k)}$$

Similarly, we can also write:

$$p(x | C_j) = \frac{\mathbb{P}(C_j | x) p(x)}{\mathbb{P}(C_j)} = \frac{\mathbb{P}(C_j | x) p(x)}{\int \mathbb{P}(C_j | x) p(x) dx}$$

Bayes' theorem when both X and Y are continuous

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

Exercise 1.2

Town with 1 business school (BS) and 1 engineering school (ES)

- ▶ 90% of BS students wear informal business dress (jacket, etc)
- ▶ 10% of BS students wear casual dress (T-shirt, jeans, etc)
- ▶ 20% of ES students wear business dress (b)
- ▶ 80% of ES students wear casual dress (c)

BS : 100 students

ES : 900 students

Assuming you meet a student wearing business dress, what are the odds that he is from the business school ?

Answer

$$\mathbb{P}(\text{BS}|b) = ?$$

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

4. Illustration with linear models

5. Methodology

6. Unsupervised learning

7. Reinforcement learning

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

- Classification
- Regression
- Bayesian decision theory

3. Principles of inference

4. Illustration with linear models

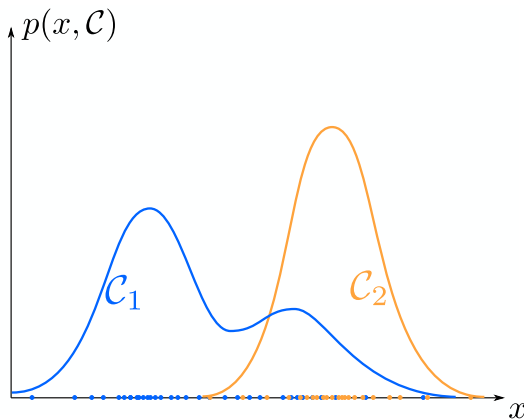
5. Methodology

6. Unsupervised learning

Types of learning tasks

- ▶ Supervised learning, using examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$
 - ▶ **Classification:** y_n is a label $\mathcal{C}_1, \dots, \mathcal{C}_p$
E.g. digit recognition
 - ▶ **Regression:** $y_n \in \mathbb{R}$ or $y_n \in \mathbb{R}^P$
E.g. curve fitting, surface fitting
- ▶ Unsupervised learning, using examples $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
 - ▶ **Clustering:** *cluster data according to similarity (distance)*
 - ▶ **Density estimation:** *What is the data distribution ?*
- ▶ Reinforcement learning *action* \rightarrow *reward*.
Objective: Maximize the reward cumulated over time

A classification problem

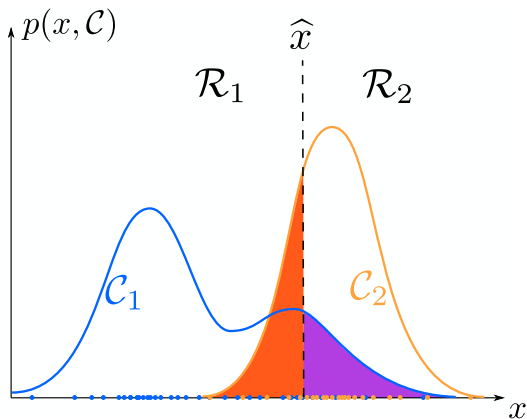


Feature $x \in \mathbb{R}$: dosage of specific antigene in blood

Response $y \in \{C_1, C_2\}$: No disease / disease

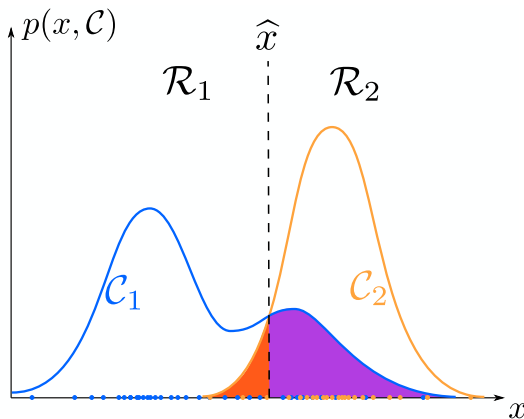
Assume we want to minimize the number of misclassifications

Minimizing the misclassification rate



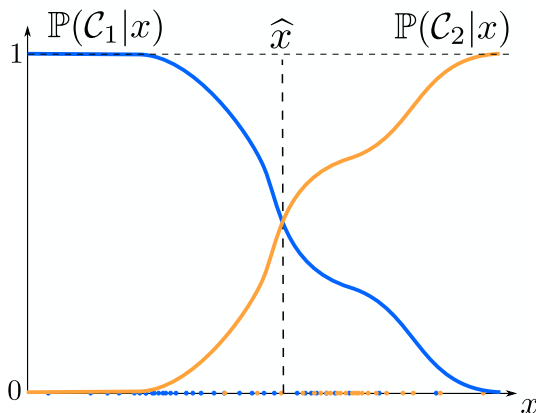
$$\mathbb{P}(\text{mistake}) = \int_{\mathcal{R}_1} p(x, \mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(x, \mathcal{C}_1) dx$$

Optimum decision rule (for misclassification rate)



$$\hat{x} = \arg \min \mathbb{P}(\text{mistake}) = \arg \min \left(\int_{\mathcal{R}_1} p(x, \mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(x, \mathcal{C}_1) dx \right)$$

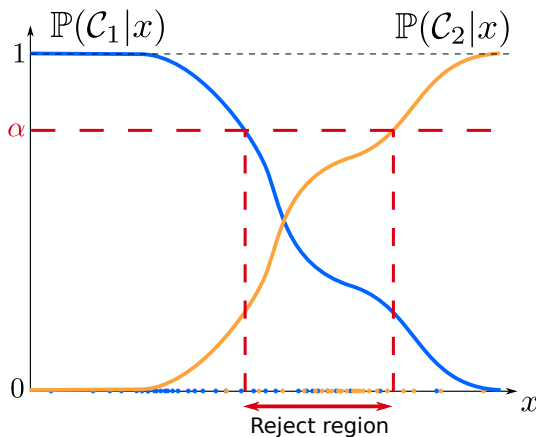
Minimizing the misclassification rate



Minimize $\mathbb{P}(\text{mistake}) \Leftrightarrow$ Assign x to class \mathcal{C}_j

- ▶ for which $p(x, \mathcal{C}_j) = \mathbb{P}(\mathcal{C}_j|x)p(x)$ is the highest
- ▶ i.e. of highest conditional probability $\mathbb{P}(\mathcal{C}_j|x)$

Reject option



- ▶ x assigned to class \mathcal{C}_1 if $\mathbb{P}(\mathcal{C}_1|x) \geq \alpha$
- ▶ x assigned to class \mathcal{C}_2 if $\mathbb{P}(\mathcal{C}_2|x) \geq \alpha$
- ▶ “I’m not sure” (reject), when both probabilities $< \alpha$

Weighing the risks

- ▶ Patient with cancer might die if not cured
- ▶ Healthy patient will suffer less inconveniences than in first case, if unnecessarily cured

Loss matrix L :

Assigned class
↓

		Cancer	Normal
True class →	Cancer	0	1000
	Normal	1	0

Loss function $\ell(\mathcal{C}_k, \mathcal{C}_j) = L_{kj}$

⚠ Beware of notation conventions (not always the same) !

Here:

- ▶ Assigned class: $\hat{y} = h(x) = \mathcal{C}_j$
- ▶ True class: $y = \mathcal{C}_k$
- ▶ Loss: $\ell(\text{true_class}, \text{assigned_class})$

How to assess/compare classification models?

Confusion matrix: Occurences of actual vs. predicted outputs

Example:

	Predicted: 0	Predicted: 1
Actual: 0	TN=115	FP= 14
Actual: 1	FN=42	TP= 15

Some usual metrics

	Predicted: 0	Predicted: 1
Actual: 0	TN	FP
Actual: 1	FN	TP

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

There are others (e.g.: Area under ROC curve, Gini index)

Classification in practice

Optimal decision boundary? \rightarrow Bayesian decision theory

Given loss ℓ (or L), we can compute the optimal decision boundary, knowing the joint distribution $p(x, C_j)$ (or equivalently $p(x|C_j)$ and $\mathbb{P}(C_j)$) for all classes

Unfortunately, we don't know the true distribution

We only have a finite dataset $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$

\Rightarrow Infer decision rule from data

This is what Machine Learning is about

Several approaches

Infer classification rule from data S ?

Direct approach Model = function h_θ

Find optimal $\hat{\theta}$ minimizing chosen error $\mathcal{E}(\theta)$ on S

E.g. perceptron algorithm (minimizing misclassification rate, using linear decision boundary, assuming data is linearly separable)

Statistical inference Model = law of parameter θ

- ▶ Conditional probability $\mathbb{P}(y \mid x, \theta)$ (discriminative model)
- ▶ Or joint density $p(x, y \mid \theta)$ (generative model)
Or alternatively $p(x \mid y)$ and $\mathbb{P}(y)$

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

- Classification
- Regression
- Bayesian decision theory

3. Principles of inference

4. Illustration with linear models

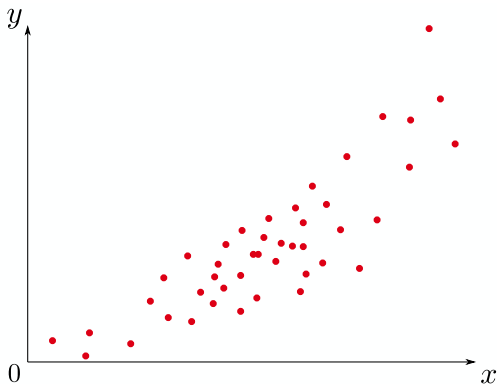
5. Methodology

6. Unsupervised learning

Types of learning tasks

- ▶ Supervised learning, using examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$
 - ▶ Classification: y_n is a label $\mathcal{C}_1, \dots, \mathcal{C}_p$
E.g. digit recognition
 - ▶ Regression: $y_n \in \mathbb{R}$ or $y_n \in \mathbb{R}^P$
E.g. curve fitting, surface fitting
- ▶ Unsupervised learning, using examples $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
 - ▶ Clustering: *cluster data according to similarity (distance)*
 - ▶ Density estimation: *What is the data distribution ?*
- ▶ Reinforcement learning *action* \rightarrow *reward*.
Objective: Maximize the reward cumulated over time

Example of a simple regression problem

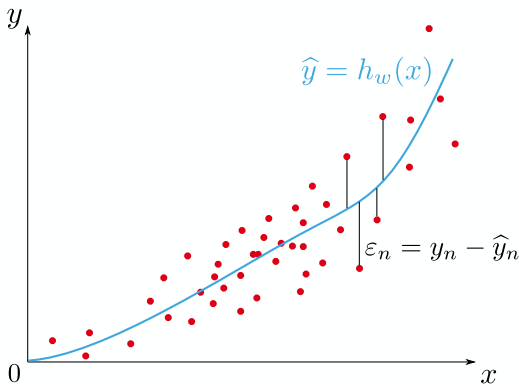


x = Number of aircraft in an airspace sector

$y \in \mathbb{R}$ = Subjective ratings of air traffic controller workload

Note: In many other regression problems $\dim(x) \gg 1$

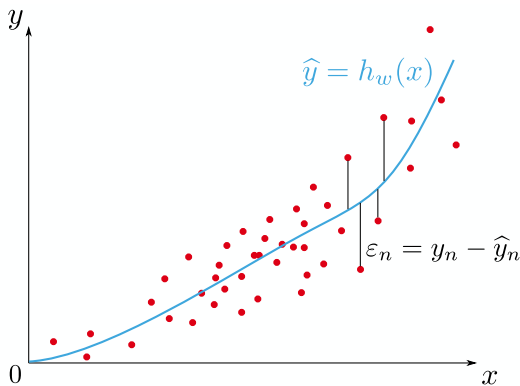
Direct approach: error minimization



Minimizing the sum-of-squares error:

$$\mathcal{E}(\mathbf{w}) = \sum_{n=1}^N (y_n - h_{\mathbf{w}}(x_n))^2 \quad \mathbf{w}^* = \arg \min_{\mathbf{w}} \mathcal{E}(\mathbf{w})$$

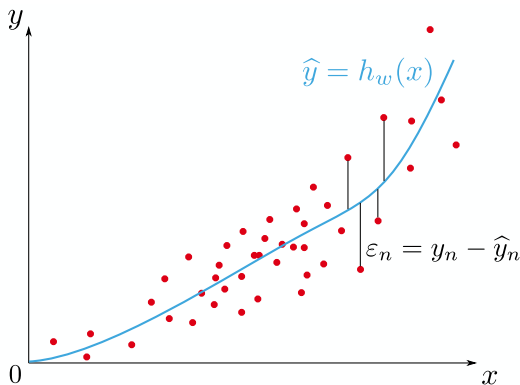
Direct approach: error minimization



Penalized least squares (an example of **regularization**):

$$\mathcal{E}_\lambda(\mathbf{w}) = \sum_{n=1}^N (y_n - h_{\mathbf{w}}(\mathbf{x}_n))^2 + \lambda \|\mathbf{w}\|^2$$

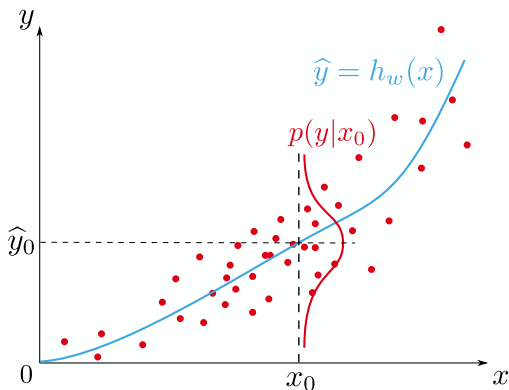
Direct approach: error minimization



There are many possible choices for

- ▶ the error function $\mathcal{E}(w)$
- ▶ the model $h_w \in \mathcal{H}$

Approach using probability densities



We can approximate a probability density instead of a function

Response \hat{y}_0 for an input x_0 is the mode of the distribution

Difference with direct approach: We obtain $h_w + \text{data distribution}$

Loss function

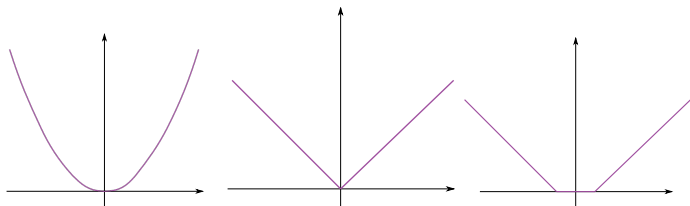
The **loss function** ℓ is the **cost of error** between :

- ▶ y , the actual response to an input x ,
- ▶ and $\hat{y} = h(x)$ computed by the model

How to choose a loss function ?

- ▶ Arbitrary choice (direct approach)
- ▶ Possibility to weigh the errors depending on the severity of the consequences
- ▶ Or based on a principle of inference
Ex. : maximum likelihood, maximum posterior

Examples of loss functions



Régression :

- ▶ $\ell(y, h(x)) = \|y - h(x)\|_2 = (y - h(x))^2$
Ordinary least squares regression
- ▶ $\ell(y, h(x)) = \|y - h(x)\|_1 = |y - h(x)|$
 L_1 norm, for robust regression
- ▶ $\ell(y, h(x)) = |y - h(x)|_\epsilon$ où $|z|_\epsilon = \begin{cases} 0 & \text{si } |z| < \epsilon \\ |z| - \epsilon & \text{sinon} \end{cases}$ ϵ -sensitive
loss

Risk minimization

Risk = loss expectation

$$\mathcal{R}(h) = \mathbb{E}_{X,Y} [\ell(Y, h(X))] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, h(x)) p(x, y) dx dy$$

Empirical risk, for a dataset $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$

$$\mathcal{R}_{emp}(h) = \frac{1}{N} \sum_{n=1}^N \ell(y_n, h(x_n))$$

Regression in practice

Optimal regression function?

Given loss ℓ , we can in theory compute an optimal regression function, knowing $p(x, y)$ (or $p(y | x)$ and $p(x)$)

Unfortunately, we don't know this distribution

We only have a finite dataset $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$

⇒ Infer from data

This is what Machine Learning is about

Several possible approaches

Direct approach Model = function h_θ

Find optimal $\hat{\theta}$ minimizing chosen $\mathcal{E}(\theta)$ on S

Statistical inference Model = density of parameter θ

- ▶ Conditional density $p(y \mid x, \theta)$ (discriminative model)
- ▶ Or joint density $p(x, y \mid \theta)$ (generative model)

Learn predictive model $p(y \mid x, S)$ from data S

For any new entry x , response $\hat{y}(x)$ is computed from $p(y \mid x, S)$ (e.g. taking adequate statistic such as the mean)

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

- Classification
- Regression
- Bayesian decision theory

3. Principles of inference

4. Illustration with linear models

5. Methodology

6. Unsupervised learning

Bayesian decision theory

Our objective in supervised learning:

Find h minimizing the risk (*i.e.* loss expectation):

$$\mathcal{R}(h) = \mathbb{E}_{X,Y} [\ell(Y, h(X))] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, h(x)) p(x, y) dx dy$$

Bayesian decision theory:

Given a loss ℓ and knowing the true joint distribution of X and Y , one can in theory find

- ▶ the optimal decision rule, in classification problems
- ▶ the optimal regression function, in regression problems

Bayesian decision theory for classification

Objective: find h minimizing the risk (*i.e.* loss expectation):

$$\mathcal{R}(h) = \mathbb{E}_{X,Y}[\ell(Y, h(X))] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, h(x)) p(x, y) dx dy$$

Classification problem with categories \mathcal{C}_k , $k \in \{1, \dots, P\}$:

$$\begin{aligned} \mathcal{R}(h) &= \int_{\mathcal{X}} \sum_k \ell(\mathcal{C}_k, h(x)) p(x, \mathcal{C}_k) dx \\ &= \int_{\mathcal{X}} \underbrace{\left[\sum_k \ell(\mathcal{C}_k, h(x)) \mathbb{P}(\mathcal{C}_k | x) \right]}_{\mathcal{R}(\hat{y}=h(x) | x)} p(x) dx \end{aligned}$$

Conditional risk (or posterior risk) in classification problems

$$\begin{aligned}\mathcal{R}(h(\mathbf{x})|\mathbf{x}) &= \mathbb{E}_{Y|X=\mathbf{x}}[\ell(Y, h(\mathbf{x}))] \\ &= \sum_k \ell(\mathcal{C}_k, h(\mathbf{x})) \mathbb{P}(\mathcal{C}_k|\mathbf{x})\end{aligned}$$

The conditional risk $\mathcal{R}(h(\mathbf{x})|\mathbf{x})$ is the risk (expected loss) of choosing category $\hat{y} = h(\mathbf{x})$, knowing \mathbf{x} .

The overall risk $\mathcal{R}(h) = \int_{\mathcal{X}} \mathcal{R}(h(\mathbf{x})|\mathbf{x})p(\mathbf{x})d\mathbf{x}$ is minimized when choosing $h(\mathbf{x}) = \mathcal{C}_j$ as follows:

$$h(\mathbf{x}) = \mathcal{C}_j = \arg \min_i (\mathcal{R}(\mathcal{C}_i|\mathbf{x})) , \quad \forall \mathbf{x} \in \mathcal{X}$$

(Bayes' decision rule)

Bayes decision rule in classification problems

Bayes' rule is the optimal decision rule:

Assign x to class C_j of lowest conditional risk $\mathcal{R}(C_j|x)$

- ▶ $\mathcal{R}(C_j|x) = \sum_k \ell(C_k, C_j) \mathbb{P}(C_k|x)$
- ▶ Bayes' theorem:

$$\mathbb{P}(C_k|x) = \frac{p(x|C_k) \mathbb{P}(C_k)}{p(x)} = \frac{p(x|C_k) \mathbb{P}(C_k)}{\sum_i p(x|C_i) \mathbb{P}(C_i)}$$

In theory, **assuming we know $p(x|C_k)$ and $\mathbb{P}(C_k)$** for each class, we can compute the optimal decision boundary (or boundaries)

Exercise 2.1

$$L = \begin{pmatrix} \ell_{11} & \ell_{12} \\ \ell_{21} & \ell_{22} \end{pmatrix}$$

Note: Here, $\ell(y, \hat{y}) = \ell_{kj}$ is the cost of assigning x to category $\hat{y} = C_j$ whereas its true class is $y = C_k$.

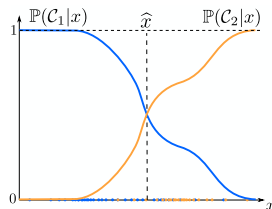
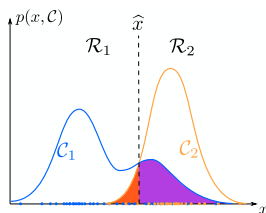
1. Assuming a binary loss $\ell_{12} = \ell_{21} = 1$, $\ell_{11} = \ell_{22} = 0$, on what condition on $\mathbb{P}(C_1|x)$ and $\mathbb{P}(C_2|x)$ do we decide to assign x to class C_1 ?
2. Same question with the following loss: $\ell_{12} = \alpha$, $\ell_{21} = 2\alpha$, $\ell_{11} = \ell_{22} = 0$, with $\alpha > 0$ a given constant.

Solution to exercise 2.1

Remark

Minimizing misclassification rate \Leftrightarrow Loss matrix $L = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

\Leftrightarrow Binary loss: $\ell(y, \hat{y}) = 0$ if $y = \hat{y}$, and 1 otherwise



$$\hat{x} = \arg \min \mathbb{P}(\text{mistake}) = \arg \min \left(\int_{\mathcal{R}_1} p(x, C_2) dx + \int_{\mathcal{R}_2} p(x, C_1) dx \right)$$

Solution to exercise 2.1 (continued)

Exercise 2.2

Knowing the true probability distribution, can we compute the optimal decision boundary? In theory, yes.

- ▶ $x \in \mathbb{R}$ (dimension 1)
- ▶ $y \in \{\mathcal{C}_1, \mathcal{C}_2\}$
- ▶ Binary loss
- ▶ Gaussian distributions for $p(x|\mathcal{C}_1)$ and $p(x|\mathcal{C}_2)$ of means μ_1 and μ_2 respectively and of equal variance σ^2
- ▶ $\mathbb{P}(\mathcal{C}_1) = \mathbb{P}(\mathcal{C}_2)$

What is the optimal decision boundary between classes \mathcal{C}_1 and \mathcal{C}_2 ?

Reminder: Gaussian law $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2} \frac{(x - \mu)^2}{\sigma^2}\right\}$

Solution to exercise 2.2

Exercise 2.3

Binary classification problem with $(x, y) \in \mathbb{R}^2 \times \{1, 2\}$

- Dataset:

x_1	0	8	4	6	2	10
x_2	2	4	6	4	6	8
y	1	1	1	2	2	2

- Loss $\ell_{12} = \alpha$, $\ell_{21} = 2\alpha$, $\ell_{11} = \ell_{22} = 0$, with $\alpha > 0$
 Note: here, $\ell(y, \hat{y}) = \ell_{kj}$ is the cost of assigning x to category $\hat{y} = j$ whereas its true class is $y = k$
- Gaussian distributions
- $$p(x | \mathcal{C}_k) = (2\pi)^{-1} |\Sigma_k|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right\}$$
1. Give empirical estimates of $\mathbb{P}(\mathcal{C}_k)$, μ_k and Σ_k for $k \in \{1, 2\}$
 2. Give the expression of the decision boundary

Solution to exercise 2.3

Solution to exercise 2.3

Solution to exercise 2.3

Bayes estimator in regression problems

For a given loss ℓ , can we find an optimal $h \in \mathcal{H}$?

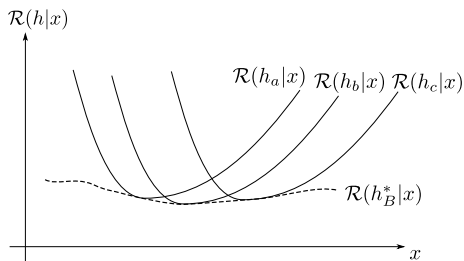
$$\begin{aligned}
 \mathcal{R}(h) &= \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, h(x)) p(x, y) dx dy \\
 &= \int_{\mathcal{X}} \int_{\mathcal{Y}} \ell(y, h(x)) p(y|x) p(x) dx dy \\
 &= \int_{\mathcal{X}} \underbrace{\left[\int_{\mathcal{Y}} \ell(y, h(x)) p(y|x) dy \right]}_{\mathbb{E}_{y|x}[\ell(y, h(x))]} p(x) dx
 \end{aligned}$$

Conditional risk (or posterior risk):

$$\mathcal{R}(h|x) = \mathbb{E}_{y|x}[\ell(y, h(x))] = \int_{\mathcal{Y}} \ell(y, h(x)) p(y|x) dy$$

We can build h_B^* by minimizing the conditional risk at every point x , *i.e.* by choosing $h_B^*(x) = \arg \min_{h \in \mathcal{H}} \mathcal{R}(h|x)$

Bayes estimator in regression problems



The function h_B^* such that $\forall x \in \mathcal{X}, \mathcal{R}(h_B^*|x) = \inf_{h \in \mathcal{H}} \mathbb{E}_{y|x}[\ell(y, h(x))]$ is called a **Bayes estimator**

Same thing as the *Bayes decision rule* in classification

h_B^* is optimal if the overall risk $\mathcal{R}(h)$ is bounded

Note : optimality “on average” on all possible draws of (x, y)

Examples of Bayes estimators (for $y \in \mathbb{R}$)

- Regression with quadratic loss $\ell(y, h(x)) = (y - h(x))^2$:

$$h_B^*(x) = \mathbb{E}(y | x) = \int p(y | x) y \, dy$$

(expectation of y knowing x)

- Robust regression with loss $\ell(y, h(x)) = |y - h(x)|$:

$$h_B^* = \text{Median}(y | x)$$

(median value of y knowing x)

To conclude on Bayes decision theory

In theory, given loss ℓ when minimizing the loss expectation (risk \mathcal{R})

- ▶ we can compute the optimal decision boundary (in classification problems)
- ▶ or the Bayes estimator (in regression problems)
- ▶ if we know the true joint distribution of x and y !

In practice, we only have a set of examples $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$

⇒ Machine Learning: Infer from data

Principles and Methodology of Machine Learning

1. Introduction
2. Supervised learning
3. Principles of inference
4. Illustration with linear models
5. Methodology
6. Unsupervised learning
7. Reinforcement learning

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

- Making assumptions
- Inference in short
- Maximum likelihood
- Maximum posterior
- Summary on inference

4. Illustration with linear models

5. Methodology

6. Machine learning in the real world

Inference, or how to learn from data

Learning bias: One cannot learn without making assumptions first!

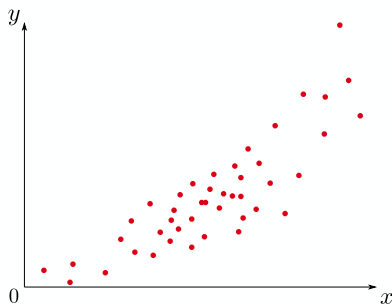
- ▶ *A priori* assumptions on the problem at hand...
- ▶ ... are refined *a posteriori* after observing the data

In practice:

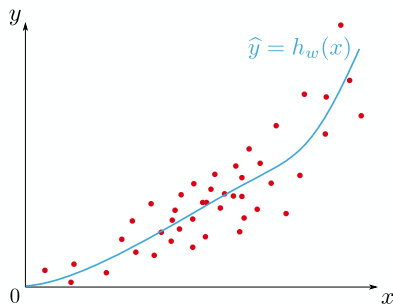
1. Choose a model = Make assumptions on the problem
2. Adjust model parameters, minimizing an error function

Be aware that when you just pick up and try a method on a problem, you are implicitly making a number of assumptions on this problem

Making assumptions: models for regression problems



Making assumptions: models for regression problems

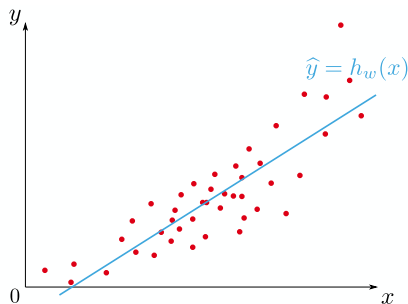


- ▶ $x \in \mathbb{R}, y \in \mathbb{R}$
- ▶ $\hat{y} = h_w(x)$ where $h_w \in \mathcal{H}$
- ▶ **Model parameters** $\theta = w$

Direct approach

E.g.: 4th degree polynomial $h_w(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

Making assumptions: models for regression problems

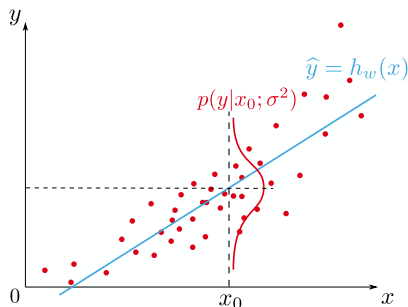


- ▶ $x \in \mathbb{R}, y \in \mathbb{R}$
- ▶ $\hat{y} = h_w(x)$ where $h_w \in \mathcal{H}$
- ▶ **Model parameters** $\theta = w$

Direct approach

E.g.: linear function $h_w(x) = w_0 + w_1 x$

Making assumptions: models for regression problems

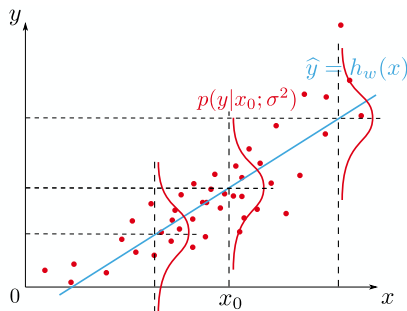


- ▶ $x \in \mathbb{R}$
- ▶ $Y = \underbrace{h_w(x)}_{\hat{y}} + \varepsilon$ random variable

Discriminative approach

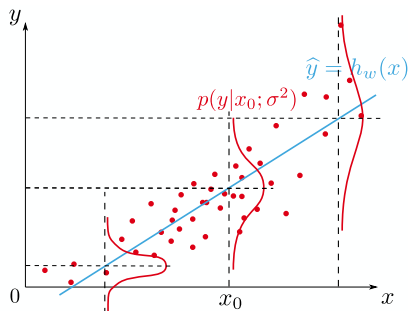
- ▶ ε r.v. of gaussian density $\mathcal{N}(0, \sigma^2)$
- ▶ Draws of Y are independants and identically distributed (i.i.d.)
- ▶ Model parameters $\theta = (w, \sigma^2)$

Making assumptions: models for regression problems



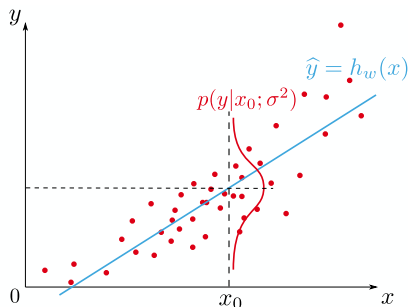
- ▶ $x \in \mathbb{R}$
- ▶ $Y = h_w(x) + \varepsilon$ random variable
- ▶ ε r.v. of gaussian density $\mathcal{N}(0, \sigma^2)$
- ▶ Homoscedasticity: variance σ^2 is constant
- ▶ Model parameters $\theta = (w, \sigma^2)$

Making assumptions: models for regression problems



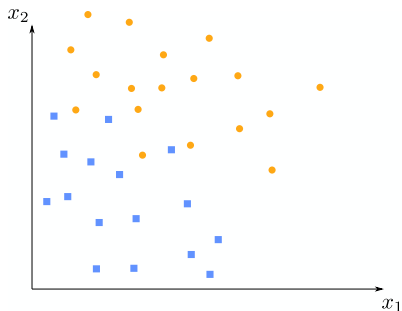
- ▶ $x \in \mathbb{R}$
- ▶ $Y = h_w(x) + \varepsilon$ random variable
- ▶ ε r.v. of gaussian density $\mathcal{N}(0, \sigma^2)$
- ▶ Variance $\sigma^2(x)$ is NOT constant
- ▶ Model parameters $\theta = (w, \text{parameters of } \sigma^2(x))$

Making assumptions: models for regression problems



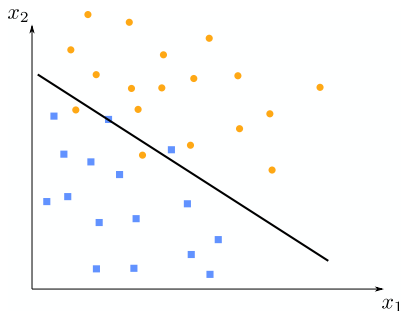
- ▶ X is a random variable (e.g. uniform $\mathcal{U}(a, b)$)
 - ▶ $Y = h_w(x) + \varepsilon$ random variable
 - ▶ ε r.v. of gaussian density $\mathcal{N}(0, \sigma^2)$
 - ▶ Model parameters $\theta = (w, \sigma^2, a, b)$
- Generative approach
modeling $p(X, Y)$

Making assumptions: models for classification problems



Binary classification problem

Making assumptions: models for classification problems



► $\mathbf{x} \in \mathbb{R}^2$, $y \in \{0, 1\}$ (or $\{-1, 1\}$)

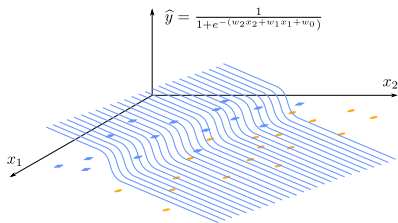
► Decision boundary $h_{\mathbf{w}} \in \mathcal{H}$

Direct approach

► Model parameters $\theta = \mathbf{w}$

E.g.: linear function $h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2$

Making assumptions: models for classification problems



► $\mathbf{x} \in \mathbb{R}^2$, 0/1 encoding for response y ($y \in \{0, 1\}$)

► Two classes \mathcal{C}_1 and \mathcal{C}_2

► $\mathbb{P}(\mathcal{C}_1|\mathbf{x}) + \mathbb{P}(\mathcal{C}_2|\mathbf{x}) = 1$

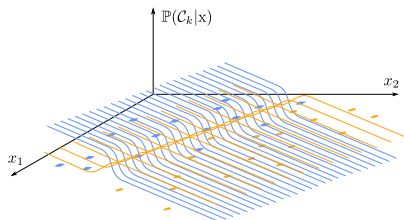
► \hat{y} interpreted as $\mathbb{P}(\mathcal{C}_1|\mathbf{x})$

E.g. $\hat{y} = h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-(w_2 x_2 + w_1 x_1 + w_0)}}$ Discriminative approach

► N independent draws $(x_1, y_1), \dots, (x_N, y_N)$

► Model parameters $\theta = \mathbf{w}$

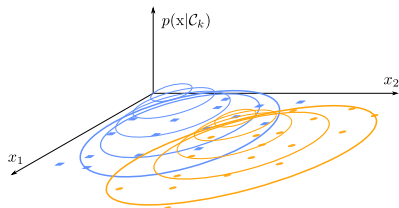
Making assumptions: models for classification problems



- ▶ $\mathbf{x} \in \mathbb{R}^2$
- ▶ $\mathbf{y} = (y_1, y_2)^T$ with $y_k \in \{0, 1\}$, $\forall k$
- ▶ Two classes \mathcal{C}_1 and \mathcal{C}_2 with $\mathbb{P}(\mathcal{C}_1|\mathbf{x}) + \mathbb{P}(\mathcal{C}_2|\mathbf{x}) = 1$
- ▶ $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2)^T$ interpreted as $(\mathbb{P}(\mathcal{C}_1|\mathbf{x}), \mathbb{P}(\mathcal{C}_2|\mathbf{x}))^T$
- ▶ $\mathbb{P}(\mathcal{C}_1|\mathbf{x})$ and $\mathbb{P}(\mathcal{C}_2|\mathbf{x})$ modeled as sigmoids
- ▶ N independent draws $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)$

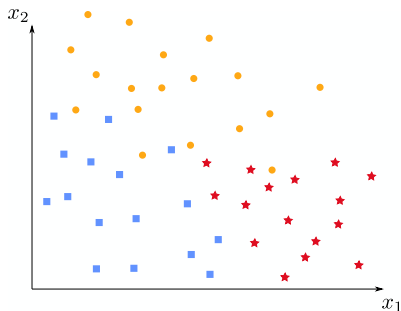
Can be generalized to more than 2 classes!

Making assumptions: models for classification problems



- ▶ $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{y} = (y_1, y_2)^T$ with $y_k \in \{0, 1\}$
- ▶ Two classes \mathcal{C}_1 and \mathcal{C}_2 with $\mathbb{P}(\mathcal{C}_1|\mathbf{x}) + \mathbb{P}(\mathcal{C}_2|\mathbf{x}) = 1$
- ▶ $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2)^T$ interpreted as $(\mathbb{P}(\mathcal{C}_1|\mathbf{x}), \mathbb{P}(\mathcal{C}_2|\mathbf{x}))^T$
- ▶ $p(\mathbf{x}|\mathcal{C}_1)$ and $p(\mathbf{x}|\mathcal{C}_2)$ gaussian densities
- ▶ $\mathbb{P}(\mathcal{C}_1) = \pi_1$, $\mathbb{P}(\mathcal{C}_2) = \pi_2$ Generative approach
- ▶ N independent draws $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)$
- ▶ Model parameters $\theta = (\pi_1, \pi_2, \text{ and parameters of the densities})$

Making assumptions: multiple-class problems



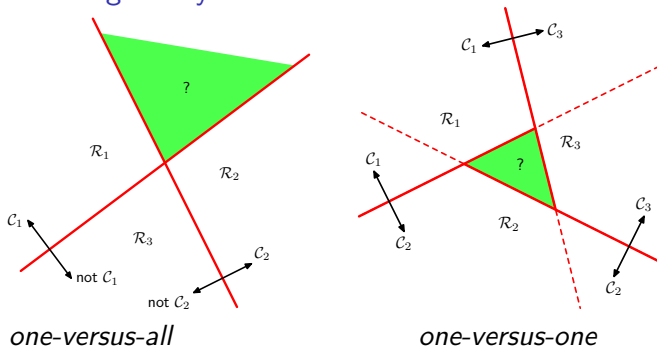
Classification problems with P classes ($P > 2$)

- ▶ Approaches using binary discriminant functions (*one-versus-all*, *one-versus-one*)
- ▶ Approaches with multiple discriminant functions

As in binary problems, one can make assumptions on $\mathbb{P}(\mathcal{C}_k | \mathbf{x})$ or $p(\mathbf{x}, \mathcal{C}_k)$ for each classe \mathcal{C}_k

Multiple-class problem (more than 2 classes)

Approaches using binary discriminant functions



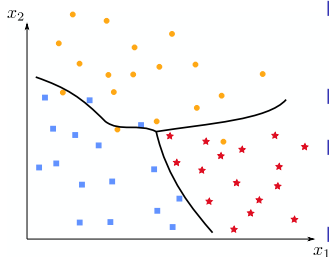
Source images : C. Bishop, *Pattern Recognition and Machine Learning*

Issue with such approaches: There are ambiguous regions

Multiple-class problems ($P \geq 2$ classes)

- ▶ $y = (y_1, \dots, y_p, \dots, y_P)^T$ has P components
- ▶ 0/1 encoding: $y_p = 1$ if x in class \mathcal{C}_p , and 0 otherwise
- ▶ Non-overlapping classes: $\sum_p y_p = 1$

Discriminant functions



- ▶ One multi-class discriminant made of P functions $\hat{y}_p = h_p(x)$
- ▶ $\hat{y} = (\hat{y}_1, \dots, \hat{y}_p, \dots, \hat{y}_P)^T$
- ▶ New x assigned to \mathcal{C}_k if $\forall p, h_k(x) > h_p(x)$
- ▶ Decision boundary between classes \mathcal{C}_i and \mathcal{C}_j : $h_i(x) = h_j(x)$

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

- Making assumptions
- Inference in short
- Maximum likelihood
- Maximum posterior
- Summary on inference

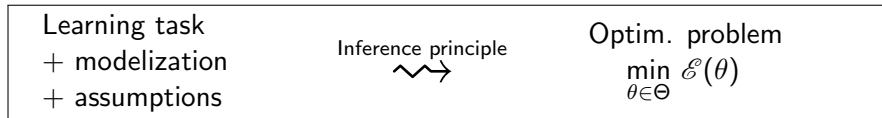
4. Illustration with linear models

5. Methodology

6. Machine learning in the real world

Inference principles

How do we adjust the model parameters θ on the examples?

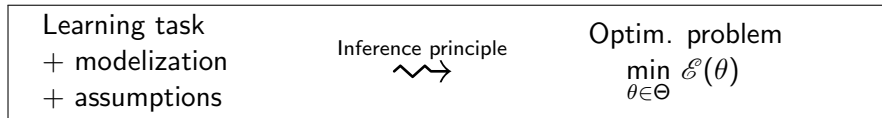


Several approaches to inference

- ▶ *Maximum likelihood*
- ▶ *Maximum posterior*
- ▶ *Minimum description length*
- ▶ *Bayesian inference*
- ▶ *etc.*

Inference principles

How do we adjust the model parameters θ on the examples?



Choice of optimization method will depend on

- ▶ Expression of model $h_{\theta} \in \mathcal{H}$ of parameter θ
- ▶ Expression of error function $\mathcal{E}(\theta)$

Frequentist vs. bayesian statistical inference

Frequentist approach: θ fixed parameter

Find optimal $\hat{\theta}_S$ according to inference principle (e.g. Max. Lik.)

Predictive model $p(y \mid x; \hat{\theta}_S)$

Bayesian inference: θ outcome of random variable θ

Predictive model $p(y \mid x, S) = \int p(y \mid x, \theta) p(\theta \mid S) d\theta$

Several principles of statistical inference

Assuming **Model** $\mathcal{M}(y|x, \theta)$ for $p(y|x)$

Frequentist approach:

- ▶ **Maximum likelihood (ML)**

$$\hat{\theta}^{ML} = \arg \max_{\theta \in \Theta} p(S|\theta) \quad p(y|x) \approx \mathcal{M}(y|x; \hat{\theta}^{ML})$$

Bayesian approach : *a priori* $p(\theta)$, *a posteriori* $p(\theta|S)$

- ▶ **Maximum posterior (MAP)**

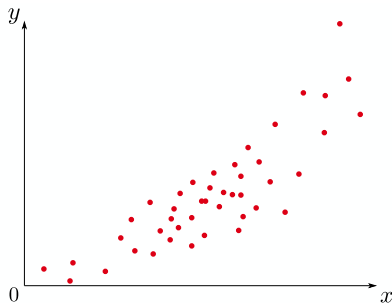
$$\hat{\theta}^{MAP} = \arg \max_{\theta \in \Theta} p(\theta|S) = \arg \max_{\theta \in \Theta} p(S|\theta)p(\theta) \quad p(y|x) \approx \mathcal{M}(y|x, \hat{\theta}^{MAP})$$

- ▶ **Bayesian inference**

$$p(y|x, S) = \int_{\Theta} \mathcal{M}(y|x, \theta) p(\theta|S) d\theta \quad p(\theta|S) = p(\theta|X, Y) = \frac{p(Y|X, \theta)p(\theta)}{p(Y)}$$

Information theory : Minimum description length (MDL)
etc.


Example 1: Simple regression with Gaussian error



Example 1: Simple regression with Gaussian error

- ▶ $y = \underbrace{h_w(x)}_{\hat{y}} + \varepsilon$
- ▶ ε Gaussian $\mathcal{N}(0, \sigma^2)$
- ▶ σ^2 constant
- ▶ Data $(x_1, y_1), \dots, (x_N, y_N)$
- ▶ N independent draws

Regression problem


 Max. Lik. Minimize
 sum-of-squares

Principle
of
inference

Optimization problem

Example 1: Simple regression with Gaussian error

When using Maximum Likelihood principle:

- ▶ Regression problem where ε is $\mathcal{N}(0, \sigma^2)$ with σ^2 constant

\rightsquigarrow sum-of-squares error

$$\mathcal{E}(\mathbf{w}) = \sum_{n=1}^N (y_n - h_{\mathbf{w}}(\mathbf{x}_n))^2$$

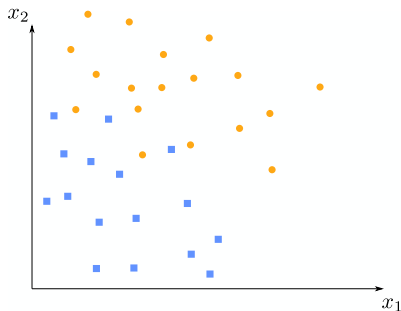
When using Maximum Posterior:

- ▶ Regression problem where ε is $\mathcal{N}(0, \sigma^2)$ with σ^2 constant
+ prior $\mathcal{N}(0_M, \beta^2 \mathbf{I}_M)$ for $p(\mathbf{w})$

\rightsquigarrow **regularized** sum-of-squares


$$\mathcal{E}_{\lambda}(\mathbf{w}) = \sum_{n=1}^N (y_n - h_{\mathbf{w}}(\mathbf{x}_n))^2 + \lambda \mathbf{w}^T \mathbf{w} \quad \text{where } \lambda = \frac{\sigma^2}{\beta^2}$$

Example 2: Binary classification with 0/1 encoding



Example 2: Binary classification with 0/1 encoding

- ▶ Two classes \mathcal{C}_1 and \mathcal{C}_2
- ▶ $\mathbb{P}(\mathcal{C}_1|x) + \mathbb{P}(\mathcal{C}_2|x) = 1$
- ▶ Binary loss $L = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- ▶ 0/1 encoding for response y
- ▶ \hat{y} interpreted as $\mathbb{P}(\mathcal{C}_1|x)$
- ▶ Data $(x_1, y_1), \dots, (x_N, y_N)$
- ▶ N independent draws


 Max. Lik. Minimize
cross-entropy

Principle
of
inference

Classification problem

Optimization problem

$$\min_{\theta \in \Theta} \mathcal{E}(\theta)$$

Example 2: Binary classification with 0/1 encoding

Maximizing the log-likelihood $\ln p(S|\theta)$ is equivalent to minimizing the following error function
(cross-entropy)

$$\mathcal{E}(\theta) = - \sum_{n=1}^N \left[y_n \ln \hat{y}_n + (1 - y_n) \ln(1 - \hat{y}_n) \right]$$

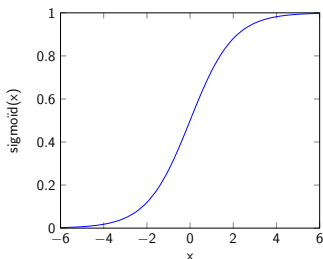
Example: Binary classification with 0/1 encoding

We made an important assumption

- \hat{y} interpreted as $\mathbb{P}(\mathcal{C}_1 | \mathbf{x}) \Rightarrow$ We must ensure $\hat{y} \in [0, 1]$

Example: logistic regression

$$\hat{y} = h_{\theta}(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + w_0)}}$$



- Parameters: $\theta = (\mathbf{w}, w_0)$
- Logistic regression is a **classification method (!)**

Logistic regression (a classification method!)

Where does it come from?

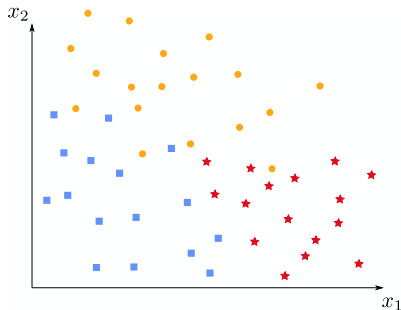
$$\mathbb{P}(\mathcal{C}_j|x) = \frac{p(x|\mathcal{C}_j) \mathbb{P}(\mathcal{C}_j)}{p(x|\mathcal{C}_1) \mathbb{P}(\mathcal{C}_1) + p(x|\mathcal{C}_2) \mathbb{P}(\mathcal{C}_2)} = \frac{1}{1 + e^{-a(x)}}$$

$$\text{where } a(x) = \ln \frac{p(x|\mathcal{C}_1) \mathbb{P}(\mathcal{C}_1)}{p(x|\mathcal{C}_2) \mathbb{P}(\mathcal{C}_2)}$$

It can be shown that if $p(x|\mathcal{C}_1)$ and $p(x|\mathcal{C}_2)$ are Gaussian with a same covariance matrix then a is linear:


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

Example 3: Classification with P classes ($P \geq 2$)



Example 3: Classification with P classes ($P \geq 2$)

- ▶ P non-overlapping classes $\mathcal{C}_1, \dots, \mathcal{C}_P$
- ▶ Binary loss
- ▶ $y = (y_1, \dots, y_P)^T$
where $y_p = 1$ if x in class \mathcal{C}_p ,
0 otherwise, $\forall p$
- ▶ $\hat{y} = (\hat{y}_1, \dots, \hat{y}_P)^T$
- ▶ \hat{y}_p interpreted as $\mathbb{P}(\mathcal{C}_p | x)$, $\forall p$
- ▶ $(x_1, y_1), \dots, (x_N, y_N)$ *i.i.d.*


 Max. Lik. Minimize
 cross-entropy

Principle
 of
 inference

Classification problem

Optimization problem

$$\min_{\theta \in \Theta} \mathcal{E}(\theta)$$

Example 3: Classification with P classes ($P \geq 2$)

$$\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_P)^T \quad \text{with } \hat{y}_p \text{ interpreted as } \mathbb{P}(\mathcal{C}_p | \mathbf{x}), \forall p$$

$$\Rightarrow \hat{y}_p \in [0, 1]$$

$$\hat{y}_p = \mathbb{P}(\mathcal{C}_p | \mathbf{x}) = \frac{p(\mathbf{x} | \mathcal{C}_p) \mathbb{P}(\mathcal{C}_p)}{\sum_k p(\mathbf{x} | \mathcal{C}_k) \mathbb{P}(\mathcal{C}_k)} = \frac{\exp\{a_p(\mathbf{x})\}}{\sum_k \exp\{a_k(\mathbf{x})\}}$$

$$\forall k \quad a_k(\mathbf{x}) = \ln p(\mathbf{x} | \mathcal{C}_k) \mathbb{P}(\mathcal{C}_k)$$

Softmax function, used in output layers of neural networks, or in multiclass logistic regression

- ▶ Multiclass logistic regression: a_k linear function of inputs \mathbf{x}
- ▶ Neural network for classification: a_k linear function of hidden layers' outputs

Error function to minimize:

$$\mathcal{E}(\theta) = - \sum_{n=1}^N \sum_{p=1}^P y_{np} \ln \hat{y}_{np}$$

(Max. Lik. principle to find $\hat{\theta}^{ML}$, considering model $\hat{\mathbf{y}} = h_{\theta}(\mathbf{x})$)

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

- Making assumptions
- Inference in short
- Maximum likelihood
- Maximum posterior
- Summary on inference

4. Illustration with linear models

5. Methodology

6. Machine learning in the real world

Maximum Likelihood (ML)

Let us consider a **model of parameter θ** and some **data \mathcal{D}** Likelihood = probability that \mathcal{D} was produced by model θ

$$\mathcal{L} : \theta \rightarrow \mathcal{L}(\theta; \mathcal{D}) = \mathbb{P}(\mathcal{D}|\theta)$$

- ▶ Frequentist point of view: the model is fixed
Notation $\mathbb{P}(\mathcal{D}; \theta)$ to express that θ is a fixed parameter
- ▶ Bayesian point of view: θ is an outcome of random variable Θ
Notation $\mathbb{P}(\mathcal{D}|\theta)$ to express a conditional probability

Principle of maximum likelihood

Choose

$$\hat{\theta}^{ML} = \arg \max_{\theta \in \Theta} \mathcal{L}(\theta; \mathcal{D})$$

Likelihood for continuous variables

If \mathcal{D} is the outcome of D having density p

- ▶ $\mathbb{P}(D = \mathcal{D} | \theta) = 0$!!
- ▶ We consider $\mathbb{P}(D \in [\mathcal{D}, \mathcal{D} + \Delta] | \theta) \underset{\Delta \text{ small}}{\approx} \Delta \times p(\mathcal{D} | \theta)$

For a continuous variable of density p , the likelihood is defined using the density:

$$\mathcal{L} : \theta \rightarrow \mathcal{L}(\theta; \mathcal{D}) = p(\mathcal{D} | \theta)$$

Note: Likelihood is defined up to a constant multiplier. This works because we compare likelihoods using ratios.

Exercise 3.1: Simple regression with Gaussian error

Dataset $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$ with N independent draws

Model:

► $y = \underbrace{h_w(x)}_{\hat{y}} + \varepsilon$ with w a vector of weights

► ε of Gaussian density $\mathcal{N}(0, \sigma^2)$: $p(\varepsilon) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2} \frac{\varepsilon^2}{\sigma^2}\right\}$

► Homoscedasticity: variance σ^2 is constant

1. What error is minimized, when applying Max. Lik. principle?
2. Give the expression of $\hat{\theta}^{ML} = (\hat{w}^{ML}, \hat{\sigma}^{2ML})$, the parameter values at maximum likelihood

Hints:

- $Y \mid x, w, \sigma^2$ has law $\mathcal{N}(h_w(x), \sigma^2)$. Write $p(y \mid x, w, \sigma^2)$
- $p(x_1, \dots, x_N, y_1, \dots, y_N; w, \sigma^2) = \prod_{n=1}^N p(x_n, y_n; w, \sigma^2)$
- Take the opposite of the logarithm

Solution to exercise 3.1

Solution to exercise 3.1

Solution to exercise 3.1


Solution to exercise 3.1

Solution to exercise 3.1

Simple regression with Gaussian error

- ▶ $y = \underbrace{h_w(x)}_{\hat{y}} + \varepsilon$
- ▶ ε Gaussian $\mathcal{N}(0, \sigma^2)$
- ▶ σ^2 constant
- ▶ Data $(x_1, y_1), \dots, (x_N, y_N)$
- ▶ N independent draws

Regression problem

 Minimize
Max. Lik. sum-of-squares

Principle
of
inference

Optimization problem

Exercise 3.2: Binary classification with 0/1 encoding

Two classes \mathcal{C}_1 and \mathcal{C}_2 . Any \mathbf{x} is either in \mathcal{C}_1 or in \mathcal{C}_2

$$\mathbb{P}(\mathcal{C}_1|\mathbf{x}) + \mathbb{P}(\mathcal{C}_2|\mathbf{x}) = 1$$

Dataset $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ where $\forall n \in 1, \dots, N$

- ▶ $y_n = 1$ if \mathbf{x} in class \mathcal{C}_1
- ▶ $y_n = 0$ if \mathbf{x} otherwise

Model h_θ of parameter θ , computing $\hat{y}_n = h_\theta(\mathbf{x}_n) \in [0, 1]$

- ▶ \hat{y}_n interpreted as $\mathbb{P}(\mathcal{C}_1|\mathbf{x}_n)$
- ▶ and $1 - \hat{y}_n$ as $\mathbb{P}(\mathcal{C}_2|\mathbf{x}_n)$

What error is minimized, when applying Max. Lik. principle?

Hint: We can write $\mathbb{P}(y_n|\mathbf{x}_n, \theta) = \hat{y}_n^{y_n} (1 - \hat{y}_n)^{1-y_n}$

Solution to exercise 3.2


Solution to exercise 3.2

Solution to exercise 3.2

Solution to exercise 3.2

Binary classification with 0/1 encoding

- ▶ Two classes \mathcal{C}_1 and \mathcal{C}_2
- ▶ $\mathbb{P}(\mathcal{C}_1|x) + \mathbb{P}(\mathcal{C}_2|x) = 1$
- ▶ Binary loss $L = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- ▶ 0/1 encoding for response y
- ▶ \hat{y} interpreted as $\mathbb{P}(\mathcal{C}_1|x)$
- ▶ Data $(x_1, y_1), \dots, (x_N, y_N)$
- ▶ N independent draws


 Max. Lik. Minimize cross-entropy

Principle
of
inference

Classification problem

Optimization problem

$$\min_{\theta \in \Theta} \mathcal{E}(\theta)$$

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

- Making assumptions
- Inference in short
- Maximum likelihood
- Maximum posterior
- Summary on inference

4. Illustration with linear models

5. Methodology

6. Machine learning in the real world

Maximum Posterior (MAP)

Let us consider a model of parameter θ and some data \mathcal{D}

- ▶ Frequentist point of view: θ is fixed (not a random variable)
- ▶ Bayesian point of view: θ is an outcome of random variable θ

Principle of maximum posterior: Choose $\hat{\theta}$ maximizing the probability $\mathbb{P}(\theta = \theta | \mathcal{D} = \mathcal{D})$ of observing θ knowing the data \mathcal{D}

$$\text{Bayes' theorem } \mathbb{P}(\theta | \mathcal{D}) = \frac{\mathbb{P}(\mathcal{D} | \theta) \mathbb{P}(\theta)}{p(\mathcal{D})}$$

Principle of maximum posterior

Choose $\hat{\theta}^{MAP} = \arg \max_{\theta \in \Theta} \mathbb{P}(\theta | \mathcal{D}) = \arg \max_{\theta \in \Theta} \mathbb{P}(\mathcal{D} | \theta) \mathbb{P}(\theta)$

Maximum Posterior for continuous variables

If θ and \mathcal{D} are continuous random variables:

$$\hat{\theta}^{MAP} = \arg \max_{\theta \in \Theta} p(\theta | \mathcal{D}) = \arg \max_{\theta \in \Theta} p(\mathcal{D} | \theta) p(\theta)$$

Exercise 3.3: Maximum posterior for a regression problem

Dataset $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$ with N independent draws

Model:

- ▶ $y = \underbrace{h_w(x)}_{\hat{y}} + \varepsilon$ with w a vector of weights
- ▶ ε of Gaussian density $\mathcal{N}(0, \sigma^2)$: $p(\varepsilon) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2} \frac{\varepsilon^2}{\sigma^2}\right\}$
- ▶ Homoscedasticity: variance σ^2 is constant
- ▶ $p(w)$ of Gaussian density $\mathcal{N}(0_M, \beta^2 I_M)$ (prior)

$$p(w) = (2\pi\beta^2)^{-\frac{P}{2}} \exp\left\{-\frac{1}{2} \frac{w^T w}{\beta^2}\right\}$$

What is the error to minimize when using the maximum posterior principle? What is the variance $\widehat{\sigma^2}^{MAP}$ at the maximum posterior?

Solution to exercise 3.3

Solution to exercise 3.3

Solution to exercise 3.3

Solution to exercise 3.3

Solution to exercise 3.3

Solution to exercise 3.3

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

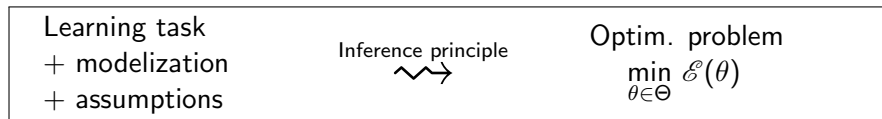
3. Principles of inference

- Making assumptions
- Inference in short
- Maximum likelihood
- Maximum posterior
- Summary on inference

4. Illustration with linear models

5. Methodology

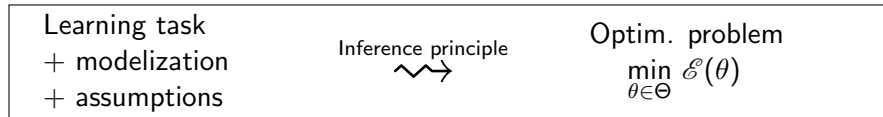
What to remember on inference?



Examples of inference principles:

- ▶ *Maximum likelihood*: choose $\hat{\theta}^{ML}$ maximizing the probability to observe data S while fixed parameter takes value θ
- ▶ *Maximum posterior*: choose $\hat{\theta}^{MAP}$ maximizing the probability that θ (random variable) takes a specific value θ , knowing the data S .
- ▶ *Bayesian inference*: θ is a random variable. Outputs are computed by averaging over all values of θ , weighed by the distribution $p(\theta|S)$. The learning phase consists in estimating $p(\theta|S)$, based on the dataset S and a *priori* distribution $p(\theta)$.

What to remember on inference?



We haven't seen how to solve the optimization problem yet

Choice of optimization method depends on

- ▶ Expression of $h_{\theta} \in \mathcal{H}$
- ▶ Expression of $\mathcal{E}(\theta)$

Useful things to remember

When using Maximum Likelihood principle:

- ▶ Regression problem where ε is $\mathcal{N}(0, \sigma^2)$ with σ^2 constant

\rightsquigarrow sum-of-squares error

$$\mathcal{E}(\mathbf{w}) = \sum_{n=1}^N (y_n - h_{\mathbf{w}}(\mathbf{x}_n))^2$$

- ▶ “Natural pairings”, for classification problems

	Output function	Error function $\mathcal{E}(\mathbf{w})$
$P = 2$	$\text{sigmoid}(a) = \frac{1}{1 + \exp\{-a\}}$	$-\sum_{n=1}^N \left[y_n \ln \hat{y}_n + (1 - y_n) \ln(1 - \hat{y}_n) \right]$
$P > 2$	$\text{softmax}_j(a) = \frac{\exp\{a_j(\mathbf{x})\}}{\sum_k \exp\{a_k(\mathbf{x})\}}$	$-\sum_{n=1}^N \sum_{k=1}^K y_{nk} \ln \hat{y}_{nk}$

“Cross-entropy”

Useful things to remember

When using Maximum Posterior:

- ▶ Regression problem where ε is $\mathcal{N}(0, \sigma^2)$ with σ^2 constant
+ prior $\mathcal{N}(0_M, \beta^2 \mathbf{I}_M)$ for $p(\mathbf{w})$
 \rightsquigarrow **regularized** sum-of-squares

$$\mathcal{E}_\lambda(\mathbf{w}) = \sum_{n=1}^N (y_n - h_{\mathbf{w}}(\mathbf{x}_n))^2 + \lambda \mathbf{w}^T \mathbf{w} \quad \text{where } \lambda = \frac{\sigma^2}{\beta^2}$$

Principles and Methodology of Machine Learning

1. Introduction
2. Supervised learning
3. Principles of inference
4. Illustration with linear models
5. Methodology
6. Unsupervised learning
7. Reinforcement learning

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

4. Illustration with linear models

- Classification with linear models
 - Rosenblatt's perceptron
 - Logistic regression
 - Linear discriminant analysis
- Linear models for regression

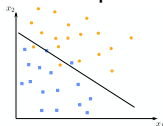
5. Methodology

6. Unsupervised learning

Linear models for classification

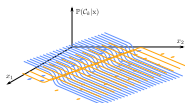
Direct approach Model = function h_θ

Find optimal $\hat{\theta}$ minimizing chosen error $\mathcal{E}(\theta)$ on S



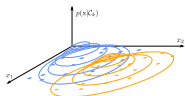
- ▶ Example: **Perceptron algorithm** (for linearly separable data)

Statistical inference Model = law of parameter θ



- ▶ Conditional probability $\mathbb{P}(y \mid x, \theta)$ (discriminative model)

Example: **Logistic regression**

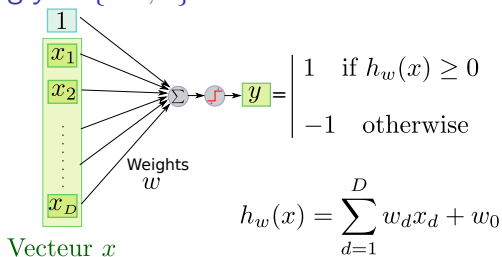


- ▶ Or joint density $p(x, y \mid \theta)$ (generative model)
Or alternatively $p(x \mid y)$ and $\mathbb{P}(y)$

Example: **Linear discriminant analysis**

Example of direct approach: Rosenblatt's perceptron

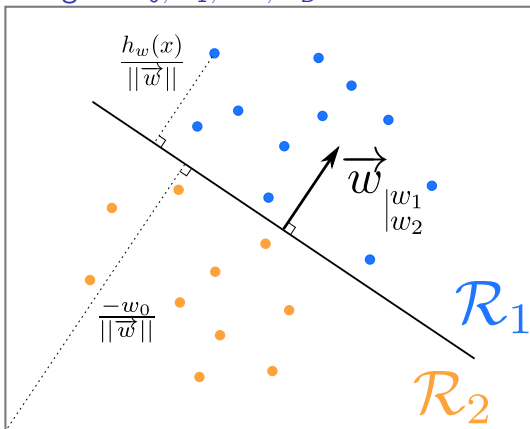
Binary encoding $y \in \{-1, 1\}$



- ▶ Invented in 1956 by Frank Rosenblatt, it is the simplest possible neural network.
- ▶ Linear classifier for a two-class problem
- ▶ Examples are considered one by one, sequentially
- ▶ For linearly separable data only

Rosenblatt's perceptron

Parameters = weights w_0, w_1, \dots, w_D



w_0 is called the *bias*

Remark

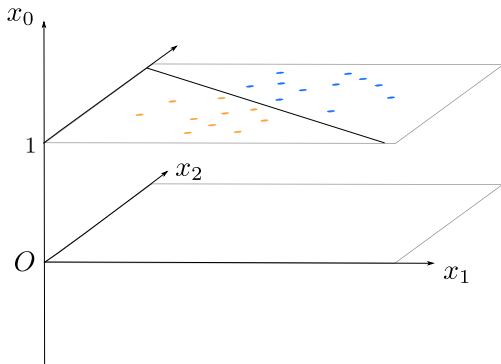
Dimension D . Hyperplane defined by $D + 1$ parameters!

\Rightarrow same hyperplane for (w_0, w_1, \dots, w_D) et $(k w_0, k w_1, \dots, k w_D)$

We can fix:

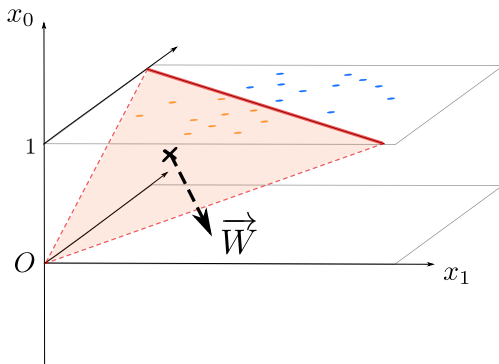
- ▶ $w_0 = 1$
- ▶ or $\|\vec{w}\| = 1$
- ▶ or any other similar constraint...

Another view, in dimension $D + 1$



All data is in hyperplane $x_0 = 1$, of dimension D

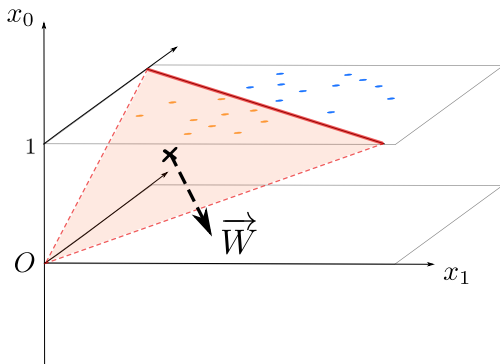
Another view, in dimension $D + 1$



Separator = hyperplane **passing through the origin**

$$\boxed{\vec{w} \cdot \vec{x} = 0}, \text{ with } x = (\mathbf{1}, x_1, \dots, x_D)^T \in \mathcal{X}_e \text{ and } w = (\mathbf{w_0}, w_1, \dots, w_D)^T$$

Another view, in dimension $D + 1$



Place hyperplane (*i.e.* find w) so as to minimize the number of misclassified points

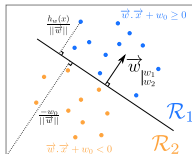
Sign of $w^T x \rightsquigarrow x$ in \mathcal{C}_1 or \mathcal{C}_2

Geometric interpretations of $\sum_{d=1}^D w_d x_d + w_0 = 0$

► Hyperplane of dimension $D - 1$

$$\vec{w} \cdot \vec{x} = -w_0, \text{ with}$$

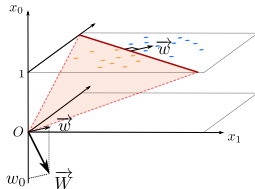
- $\vec{x} = (x_1, \dots, x_D)^T$
- $\vec{w} = (w_1, \dots, w_D)^T$



► Hyperplane of dimension D passing through the origin, in a space of dimension $D + 1$

$$\vec{w} \cdot \vec{x} = 0, \text{ with}$$

- $\vec{x} = (\mathbf{1}, x_1, \dots, x_D)^T$
- $\vec{w} = (w_0, w_1, \dots, w_D)^T$



The perceptron algorithm (first version)

Algorithm 1 The perceptron algorithm

```
1: choose  $\eta$  positive
2: initialize vector  $\mathbf{w} = (w_0, w_1, \dots, w_D)^T$ 
3: repeat
4:   for  $n = 1$  to  $N$  do
5:     if example  $\mathbf{x}_n$  is incorrectly classified then
6:       if  $y_n = \mathcal{C}_1$  then
7:          $\mathbf{w} \leftarrow \mathbf{w} + \eta \mathbf{x}_n$ 
8:       else
9:          $\mathbf{w} \leftarrow \mathbf{w} - \eta \mathbf{x}_n$ 
10:      end if
11:    end if
12:  end for
13: until all examples are correctly classified
```

A more compact version of the perceptron algorithm

Noticing that:

- ▶ $y_n = 1$ for \mathcal{C}_1 , and $y_n = -1$ for \mathcal{C}_2
- ▶ same hyperplane for w or $kw \Rightarrow$ take $\eta = 1$

Algorithm 2 The perceptron algorithm

- 1: initialize vector w
 - 2: **repeat**
 - 3: **for** $n = 1$ to N **do**
 - 4: **if** $w^T x_n y_n \leq 0$ (example x_n incorrectly classified) **then**
 - 5: $w \leftarrow w + y_n x_n$
 - 6: **end if**
 - 7: **end for**
 - 8: **until** all examples are correctly classified
-

Rosenblatt's perceptron

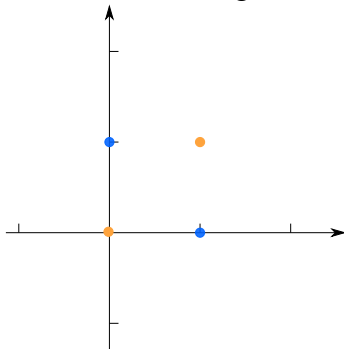
- ▶ Examples are learned sequentially
- ▶ Minimizes criterion

$$E(\mathbf{w}) = - \sum_{\mathbf{x}_n \text{ incorrectly classified}} \mathbf{w}^T \mathbf{x}_n y_n$$

- ▶ Does not converge when data is not linearly separable!
- ▶ Not used anymore

The XOR controversy

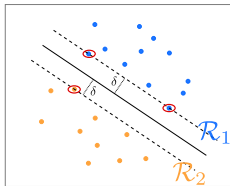
The perceptron cannot learn the following function (XOR):



In 1969, a book from Minsky and Papert literally stopped the development of the connectionist approach for more than a decade

From the perceptron to other approaches

- ▶ Replace X with $\Phi(X)$, where Φ is any transformation of the inputs (e.g. non-linear transformations)
- ▶ Multi-layer perceptrons (with non-linear activation functions)
- ▶ Support Vector Classifier

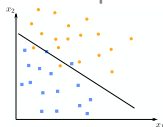


- ▶ Optimal placement of the separating hyperplane
- ▶ Extension to problems that are not linearly separable:
Support Vector Machines

Linear models for classification

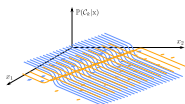
Direct approach Model = function h_θ

Find optimal $\hat{\theta}$ minimizing chosen error $\mathcal{E}(\theta)$ on S



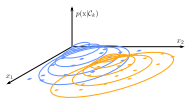
- ▶ Example: Perceptron algorithm (for linearly separable data)

Statistical inference Model = law of parameter θ



- ▶ Conditional probability $\mathbb{P}(y \mid x, \theta)$ (discriminative model)

Example: **Logistic regression**



- ▶ Or joint density $p(x, y \mid \theta)$ (generative model)
Or alternatively $p(x \mid y)$ and $\mathbb{P}(y)$


Example: Linear discriminant analysis

Logistic regression (for classification problems!)

Reminder on inference:

- ▶ Two classes \mathcal{C}_1 and \mathcal{C}_2
- ▶ $\mathbb{P}(\mathcal{C}_1|x) + \mathbb{P}(\mathcal{C}_2|x) = 1$
- ▶ Binary loss $L = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- ▶ 0/1 encoding for response y
- ▶ \hat{y} interpreted as $\mathbb{P}(\mathcal{C}_1|x)$
- ▶ Data $(x_1, y_1), \dots, (x_N, y_N)$
- ▶ N independent draws

Classification problem


 Max. Lik. Minimize cross-entropy

Optimization problem

$$\min_{\theta \in \Theta} \mathcal{E}(\theta)$$

Logistic regression (for classification problems!)

Binary encoding with $y \in \{0, 1\}$ $y = 1$ when $x \in \mathcal{C}_1$, 0 otherwise

Model:
$$\hat{y} = \mathbb{P}(y = 1 \mid x) = \frac{1}{1 + e^{-w^T x}}$$

- ▶ Output \hat{y} is the conditional probability that x belongs to \mathcal{C}_1
- ▶ $x = (\mathbf{1}, x_1, \dots, x_D)^T \in \mathcal{X}_e$
 \mathcal{X}_e : input space extended with extra dimension x_0

Model parameters: $w = (w_0, w_1, \dots, w_D)^T$

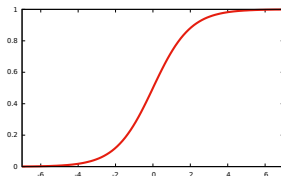
Minimized error, when applying Max. Lik. principle:

$$\mathcal{E}(w) = - \sum_{n=1}^N \left[y_n \ln \hat{y}_n + (1 - y_n) \ln(1 - \hat{y}_n) \right]$$

(cross-entropy for 2 classes)

Where is the linearity in logistic regression?

$$\hat{y} = \sigma(\mathbf{w}^T \mathbf{x}) \quad \text{where} \quad \sigma(a) = \frac{1}{1 + e^{-a}}$$



This function is not linear, but the decision boundary is linear

Exercise 4.1: Linear boundary in logistic regression

Assuming a binary loss, give the expression of the decision boundary when performing a logistic regression on a classification problem with 2 classes.

Hints:

- ▶ Reminder of Ex. 2.1: With a binary loss,
$$\mathbf{x} \in \mathcal{C}_1 \Leftrightarrow \frac{\mathbb{P}(\mathcal{C}_1 | \mathbf{x})}{\mathbb{P}(\mathcal{C}_2 | \mathbf{x})} > 1$$
- ▶ The output $\hat{y} = \mathbb{P}(y = 1 | \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$ is interpreted as $\mathbb{P}(\mathcal{C}_1 | \mathbf{x})$
- ▶ Non-overlapping classes: $\mathbb{P}(\mathcal{C}_1 | \mathbf{x}) + \mathbb{P}(\mathcal{C}_2 | \mathbf{x}) = 1$

Solution to exercise 4.1

Exercise 4.2

The previous decision boundary corresponded to a binary loss

$$L = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ (i.e. minimizing the overall misclassification rate)}$$

\Leftrightarrow Assign x to \mathcal{C}_1 if $\mathbb{P}(\mathcal{C}_1|x) \geq 0.5$

What is the decision rule if we choose a different threshold?

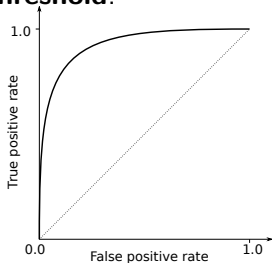
E.g. $\mathbb{P}(\mathcal{C}_1|x) \geq 0.2$

Solution to exercise 4.2

ROC curve

Changing the threshold for $\frac{\mathbb{P}(\mathcal{C}_1|x)}{\mathbb{P}(\mathcal{C}_2|x)}$ will change the true-positive and false-positive rates.

The ROC curve shows both rates for a given class (\mathcal{C}_1 for instance), **for various values of the threshold**:



The term ROC (receiver operating characteristics) comes from the communications theory

ROC curves can be used to compare classification methods

Fitting a logistic regression model

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \mathcal{E}(\mathbf{w}) \quad \text{How to compute } \hat{\mathbf{w}}?$$

Considering N examples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$,
and with $\hat{y} = \frac{1}{1+e^{-\mathbf{w}^T \mathbf{x}}} = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1+e^{\mathbf{w}^T \mathbf{x}}}$ we have

$$\begin{aligned} \mathcal{E}(\mathbf{w}) &= - \sum_{n=1}^N \left[y_n \ln \hat{y}_n + (1 - y_n) \ln(1 - \hat{y}_n) \right] \\ &= \sum_{n=1}^N \left[y_n \mathbf{w}^T \mathbf{x}_n - \ln \left(1 + e^{\mathbf{w}^T \mathbf{x}_n} \right) \right] \end{aligned}$$

This function is convex.

An iterative gradient-descent method leads to the unique minimum

Fitting a logistic regression model

$$\text{Dataset of examples} \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1D} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nD} \\ \vdots & \vdots & & \vdots \\ 1 & x_{N1} & \dots & x_{ND} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{pmatrix}$$

$$\text{Vector of weights} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$$

$$\nabla \mathcal{E}(\mathbf{w}) = \sum_{n=1}^N (\hat{y}_n - y_n) \mathbf{x}_n = \mathbf{X}^T (\hat{\mathbf{y}} - \mathbf{y}) = \mathbf{X}^T (\sigma(\mathbf{X}\mathbf{w}) - \mathbf{y})$$

Fitting a logistic regression model

No closed-form solution for $\nabla \mathcal{E}(\mathbf{w}) = 0$

We use an iterative method

Taylor developpment of the gradient:

$$\nabla \mathcal{E}(\mathbf{w}_{k+1}) \approx \nabla \mathcal{E}(\mathbf{w}_k) + \nabla^2 \mathcal{E}(\mathbf{w}_k)(\mathbf{w}_{k+1} - \mathbf{w}_k)$$

Assuming the Hessian matrix $H_k = \nabla^2 \mathcal{E}(\mathbf{w}_k)$ is invertible, and with the objective $\nabla \mathcal{E}(\mathbf{w}_{k+1}) = 0$, we obtain the following Newton-Raphson update formula:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - H_k^{-1} \nabla \mathcal{E}(\mathbf{w}_k)$$

Fitting a logistic regression model

Iterative reweighted least squares

Newton-Raphson update formula for logistic regression:

$$\begin{aligned} \mathbf{w}_{k+1} &= \mathbf{w}_k - H_k^{-1} \nabla \mathcal{E}(\mathbf{w}_k) \\ &= \left(\mathbf{X}^T \mathbf{R}_k \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{R}_k \mathbf{z} \end{aligned}$$

where \mathbf{z} is an N -dimensional vector

$$\mathbf{z} = \mathbf{X} \mathbf{w}_k - \mathbf{R}_k^{-1} (\hat{\mathbf{y}}_k - \mathbf{y})$$

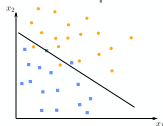
and \mathbf{R}_k is the $N \times N$ diagonal matrix such that $R_{nn} = \hat{y}_n(1 - \hat{y}_n)$, computed at each iteration k

As the Hessian is definite positive, $\mathcal{E}(\mathbf{w})$ is convex and the method converges to a unique minimum

Linear models for classification

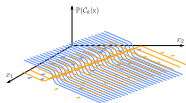
Direct approach Model = function h_θ

Find optimal $\hat{\theta}$ minimizing chosen error $\mathcal{E}(\theta)$ on S

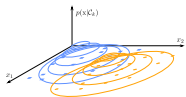


- ▶ Example: Perceptron algorithm (for linearly separable data)

Statistical inference Model = law of parameter θ



- ▶ Conditional probability $\mathbb{P}(y \mid x, \theta)$ (discriminative model)
Example: Logistic regression



- ▶ Or joint density $p(x, y \mid \theta)$ (generative model)
Or alternatively $p(x \mid y)$ and $\mathbb{P}(y)$
Example: **Linear discriminant analysis**

Linear discriminant analysis, for P classes

- ▶ Logistic regression uses a discriminative model $\mathbb{P}(\mathcal{C}_p | \mathbf{x})$,
- ▶ LDA is a generative approach, with model $p(\mathbf{x}, \mathcal{C}_p) = p(\mathbf{x} | \mathcal{C}_p) \mathbb{P}(\mathcal{C}_p)$, for all $p \in \{1, \dots, P\}$

Bayes' theorem:

$$\mathbb{P}(\mathcal{C}_p | \mathbf{x}) = \frac{p(\mathbf{x} | \mathcal{C}_p) \mathbb{P}(\mathcal{C}_p)}{p(\mathbf{x})} = \frac{p(\mathbf{x} | \mathcal{C}_p) \mathbb{P}(\mathcal{C}_p)}{\sum_k p(\mathbf{x} | \mathcal{C}_k) \mathbb{P}(\mathcal{C}_k)}$$

Ideas:

- ▶ Estimate $\mathbb{P}(\mathcal{C}_p)$ for all classes (e.g. by counting occurrences)
- ▶ Assume a model for $p(\mathbf{x} | \mathcal{C}_p)$, and estimate its parameters
Gaussian assumption, in the case of linear or quadratic discriminant analysis)

Linear discriminant analysis, for P classes

Model: $p(\mathbf{x}, \mathcal{C}_k) = p(\mathbf{x} | \mathcal{C}_k) \mathbb{P}(\mathcal{C}_k), \quad \forall k \in \{1, \dots, P\}$

$$\mathbb{P}(\mathcal{C}_k) = \pi_k$$

$$p(\mathbf{x} | \mathcal{C}_k) = (2\pi)^{-\frac{P}{2}} |\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_k)^T \Sigma^{-1} (\mathbf{x} - \mu_k) \right\}$$

Model parameters (assuming P classes):

- ▶ Shared covariance matrix Σ (homoscedasticity)
- ▶ Means μ_1, \dots, μ_P
- ▶ Class priors π_1, \dots, π_P

Linear discriminant analysis

Decision rule? Minimizing misclassification rate, select class \mathcal{C}_k for which $\mathbb{P}(\mathcal{C}_k|x)$ is the greatest

$$\mathbb{P}(\mathcal{C}_k|x) > \mathbb{P}(\mathcal{C}_l|x) \quad \Leftrightarrow \quad \ln \frac{\mathbb{P}(\mathcal{C}_k|x)}{\mathbb{P}(\mathcal{C}_l|x)} > 0$$

$$\begin{aligned} \ln \frac{\mathbb{P}(\mathcal{C}_k|x)}{\mathbb{P}(\mathcal{C}_l|x)} &= \ln \frac{p(x|\mathcal{C}_k) \pi_k}{p(x|\mathcal{C}_l) \pi_l} = \ln \frac{\pi_k}{\pi_l} + \ln \frac{p(x|\mathcal{C}_k)}{p(x|\mathcal{C}_l)} \\ &= \ln \frac{\pi_k}{\pi_l} - \frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k) + \frac{1}{2}(x - \mu_l)^T \Sigma^{-1}(x - \mu_l) \\ &= x^T \Sigma^{-1}(\mu_k - \mu_l) - \frac{1}{2}\mu_k^T \Sigma^{-1}\mu_k + \frac{1}{2}\mu_l^T \Sigma^{-1}\mu_l + \ln \frac{\pi_k}{\pi_l} \end{aligned}$$

Linear boundary $x^T w + w_0$

Linear discriminant analysis

Equivalent description of the decision rule, using linear discriminant functions:

$$\delta_k(\mathbf{x}) = \mathbf{x}^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \ln \pi_k$$

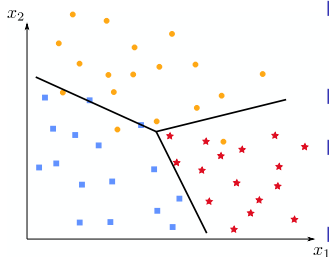
Assign \mathbf{x} to class \mathcal{C}_k for which $\delta_k(\mathbf{x})$ is the greatest

In practice, we don't know the actual values of Σ , μ_k and π_k
→ use estimates

Linear discriminant analysis ($P \geq 2$ classes)

- ▶ $y = (y_1, \dots, y_p, \dots, y_P)^T$ has P components
- ▶ 0/1 encoding: $y_p = 1$ if x in class \mathcal{C}_p , and 0 otherwise
- ▶ Non-overlapping classes: $\sum_p y_p = 1$

Discriminant functions



- ▶ One multi-class discriminant made of P functions $\hat{y}_p = \delta_p(x)$
- ▶ $\hat{y} = (\hat{y}_1, \dots, \hat{y}_p, \dots, \hat{y}_P)^T$
- ▶ New x assigned to \mathcal{C}_k if $\forall p, \delta_k(x) > \delta_p(x)$
- ▶ Decision boundary between classes \mathcal{C}_i and \mathcal{C}_j : $\delta_i(x) = \delta_j(x)$

Linear discriminant analysis

Parameters can be estimated using Maximum Likelihood principle

Maximum likelihood estimates:

$$\blacktriangleright \hat{\pi}_k = N_k / N$$

$$\blacktriangleright \hat{\mu}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

$$\blacktriangleright \hat{\Sigma} = \frac{1}{N - K} \sum_{k=1}^K \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \hat{\mu}_k)(\mathbf{x}_n - \hat{\mu}_k)^T$$

LDA, QDA, and Naive Bayes classifiers

Model: $p(\mathbf{x}, \mathcal{C}_k) = p(\mathbf{x} | \mathcal{C}_k) \mathbb{P}(\mathcal{C}_k)$

$$\mathbb{P}(\mathcal{C}_k) = \pi_k$$

$$p(\mathbf{x} | \mathcal{C}_k) = (2\pi)^{-\frac{P}{2}} |\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k) \right\}$$

Linear discriminant analysis (QDA) assumes a covariance matrix common to all classes ($\Sigma_k = \Sigma, \quad \forall k$)

Quadratic discriminant analysis (QDA) assumes a specific Σ_k for each class

The **Naïve Bayes classifier** assumes independance of the input variables (D components of \mathbf{x}): off-diagonal correlation terms of Σ (or Σ_k) are zero

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

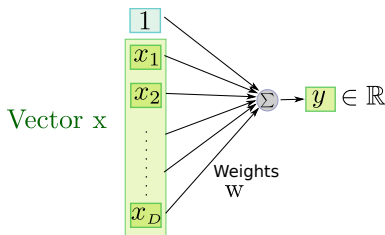
4. Illustration with linear models

- Classification with linear models
- Linear models for regression
 - Linear least squares regression
 - Regularization: ridge and lasso regression
 - Statistical inference
 - Linear models with $y \in \mathbb{R}^P$

5. Methodology

Linear models for regression

Case when $y \in \mathbb{R}$



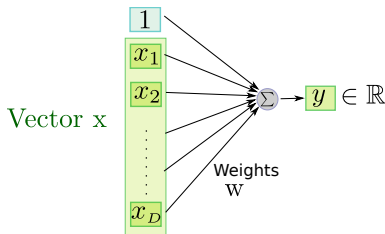
$$\mathcal{H} = \{h_{\mathbf{w}, w_0} : \mathcal{X} \rightarrow \mathbb{R}, \quad h_{\mathbf{w}, w_0}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0\}$$

$D = \dim \mathcal{X} = 1$: simple regression

$D = \dim \mathcal{X} > 1$: multiple regression

Concise notation, in dimension $D + 1$

Case when $y \in \mathbb{R}$



Let us introduce an additional dimension x_0

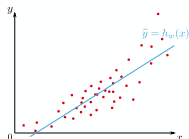
\mathcal{X}_e : space of vectors $\mathbf{x} = (\mathbf{x}_0, x_1, \dots, x_D)^T$ of dimension $D + 1$ such that $\mathbf{x}_0 = 1$ for data points

$$\mathcal{H} = \{h_{\mathbf{w}} : \mathcal{X}_e \rightarrow \mathbb{R}, \quad h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}\}$$

Inference: Direct approach or statistical model

$$y = \underbrace{\mathbf{w}^T \mathbf{x}}_{h_{\mathbf{w}}(\mathbf{x})} + \varepsilon$$

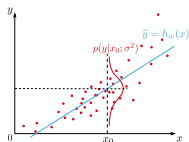
Model = function $h_{\mathbf{w}}$ (direct approach)



- Compute $\hat{\mathbf{w}} = (\hat{w}_0, \hat{w}_1, \dots, \hat{w}_D)$ minimizing error $\mathcal{E}(\mathbf{w})$ on examples $(x_1, y_1), \dots, (x_N, y_N)$

Model = probability density $p(y|x)$ (or $p(x, y)$)

(discriminative or generative approach)



- $(x), y, \varepsilon$ outcomes of random variables $(X), Y, \varepsilon$
- Statistics on $\hat{\mathbf{w}}, Y, \varepsilon$, goodness of fit (R^2)
- Confidence intervals, statistical tests, for \mathbf{w}

Generalization

$$y = \underbrace{\mathbf{w}^T \mathbf{x}}_{h_{\mathbf{w}}(\mathbf{x})} + \varepsilon$$

Once we have inferred $\hat{\mathbf{w}}$ from dataset $(x_1, y_1), \dots, (x_N, y_N)$

$\hat{y} = \hat{\mathbf{w}}^T \mathbf{x}$ used to **predict outputs** corresponding to fresh inputs \mathbf{x}

Linear least squares regression (with $y \in \mathbb{R}$)

Dataset of examples $\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1D} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nD} \\ \vdots & \vdots & & \vdots \\ 1 & x_{N1} & \dots & x_{ND} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{pmatrix}$

Vector of weights $\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$

Least squares: $\min_{\mathbf{w}} \mathcal{E}(\mathbf{w})$ with

$$\mathcal{E}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

Ordinary least squares

$$\begin{aligned}\text{Error: } \mathcal{E}(\mathbf{w}) &= (\mathbf{y} - \mathbf{X} \mathbf{w})^T (\mathbf{y} - \mathbf{X} \mathbf{w}) \\ &= \sum_{n=1}^N \left(y_n - \mathbf{w}^T \mathbf{x}_n \right)^2 \\ &= \sum_{n=1}^N \left(y_n - w_0 - \sum_{d=1}^D w_d x_{nd} \right)^2\end{aligned}$$

$$\text{Gradient: } \frac{\partial \mathcal{E}}{\partial \mathbf{w}} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X} \mathbf{w})$$

$$\text{Hessian: } \frac{\partial^2 \mathcal{E}}{\partial \mathbf{w} \partial \mathbf{w}^T} = 2\mathbf{X}^T \mathbf{X}$$

Ordinary least squares

If $\mathbf{X}^T \mathbf{X}$ is invertible (hessian matrix is definite positive),
the minimum of $\mathcal{E}(\mathbf{w})$ is reached when

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \mathbf{w}} = 0 &\Leftrightarrow \mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \\ &\Leftrightarrow \mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{X}^T \mathbf{y} \\ &\Leftrightarrow \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\end{aligned}$$

$$\hat{\mathbf{w}}^{\text{LS}} = \left(\mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{y}$$

Note: The system to solve is linear in \mathbf{w} !

You can replace \mathbf{X} with any transformation Φ of the initial inputs

Exercise 4.3

$$\mathcal{E}(\mathbf{w}) = \sum_{n=1}^N \left(y_n - w_0 - \sum_{d=1}^D w_d x_{nd} \right)^2$$

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1D} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nD} \\ \vdots & \vdots & & \vdots \\ 1 & x_{N1} & \dots & x_{ND} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{pmatrix}$$

Compute the value of the intercept w_0 by cancelling $\frac{\partial \mathcal{E}}{\partial w_0}$

Solution to exercise 4.3

Exercise 4.4

Linear combination of basis functions $h_w(\mathbf{x}) = \sum_{m=0}^M w_m \phi_m(\mathbf{x})$

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \dots & \phi_M(\mathbf{x}_1) \\ \vdots & & \vdots \\ \phi_0(\mathbf{x}_n) & \dots & \phi_M(\mathbf{x}_n) \\ \vdots & & \vdots \\ \phi_0(\mathbf{x}_N) & \dots & \phi_M(\mathbf{x}_N) \end{pmatrix} \leftarrow \text{transformed input } \phi(\mathbf{x}_n)$$

Penalized (or regularized) least squares:

$$\mathcal{E}(\mathbf{w}) = (\mathbf{y} - \Phi \mathbf{w})^T (\mathbf{y} - \Phi \mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$$

What is the optimum $\hat{\mathbf{w}}^{\text{PLS}}$?

Solution to exercise 4.4

Ridge regression

$$\mathcal{E}_\lambda(w_0, \dots, w_D) = \sum_{n=1}^N \left(y_n - w_0 - \sum_{d=1}^D w_d x_{nd} \right)^2 + \lambda \sum_{d=1}^D w_d^2$$

- ▶ Note that w_0 does not appear in $\lambda \sum_{d=1}^D w_d^2$
- ▶ Why?
 - ▶ Bias w_0 compensates the difference between $\sum_{d=1}^D w_d \bar{x}_d$ and \bar{y}
 - ▶ Adjusted model should return $y + c$ when shifting response examples y_1, \dots, y_N by a constant c
 - ▶ Optimizing with term $\lambda \sum_{d=0}^D w_d^2$ would shift w_0 by $\frac{Nc}{N-\lambda} < c$

Ridge regression

Use centered variables $x_{nd} - \bar{x}_d$ instead of x_{nd} $\bar{x}_d = \frac{1}{N} \sum_{n=1}^N x_{nd}$

$$\mathcal{E}_\lambda(w'_0, w_1, \dots, w_D) = \sum_{n=1}^N \left(y_n - w'_0 - \sum_{d=1}^D w_d (x_{nd} - \bar{x}_d) \right)^2 + \lambda \sum_{d=1}^D w_d^2$$

with $w'_0 = w_0 + \sum_{d=1}^D w_d \bar{x}_d$

Ridge regression

1. Compute optimal w'_0 by cancelling $\frac{\partial \mathcal{E}_\lambda}{\partial w'_0}$:

$$\widehat{w'_0}^{\text{ridge}} = \frac{1}{N} \sum_{n=1}^N y_n = \bar{y}$$

Consequently, we have

$$\widehat{w}_0^{\text{ridge}} = \bar{y} - \sum_{d=1}^D w_d \bar{X}_d$$

Ridge regression

2. Use centered values of the observations:

$$\mathbf{X}_c = \begin{pmatrix} x_{11} - \bar{x}_1 & \dots & x_{1D} - \bar{x}_D \\ \vdots & & \vdots \\ x_{N1} - \bar{x}_1 & \dots & x_{ND} - \bar{x}_D \end{pmatrix} \quad \mathbf{y}_c = \begin{pmatrix} y_1 - \bar{y} \\ \vdots \\ y_N - \bar{y} \end{pmatrix}$$

and optimize the following error, with $\mathbf{w}_c = (w_1, \dots, w_D)^T$

$$\mathcal{E}_\lambda(\widehat{\mathbf{w}}_0^{\text{ridge}}, \mathbf{w}_c) = (\mathbf{y}_c - \mathbf{X}_c \mathbf{w}_c)^T (\mathbf{y}_c - \mathbf{X}_c \mathbf{w}_c) + \lambda \mathbf{w}_c^T \mathbf{w}_c$$

$$\widehat{\mathbf{w}}_c^{\text{ridge}} = \left(\mathbf{X}_c^T \mathbf{X}_c + \lambda \mathbf{I}_D \right)^{-1} \mathbf{X}_c^T \mathbf{y}_c$$

Remarks

What happens to output $y = \hat{\mathbf{w}}^T \mathbf{x} = \mathbf{x}^T \hat{\mathbf{w}}$ if we scale the inputs?
i.e. replace \mathbf{X} with $\alpha \mathbf{X}$ and \mathbf{x} with $\alpha \mathbf{x}$

Least-squares

$$y = \mathbf{x}^T \underbrace{(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}}_{\hat{\mathbf{w}}^{\text{LS}}}$$

Ridge regression

$$y = \hat{w}_0 + (\mathbf{x}_1, \dots, \mathbf{x}_D)^T \underbrace{(\mathbf{X}_c^T \mathbf{X}_c + \lambda \mathbf{I}_D)^{-1} \mathbf{X}_c^T \mathbf{y}_c}_{\hat{\mathbf{w}}_c^{\text{ridge}}}$$

Ordinary least squares is scale-invariant, but ridge regression is not!
 It is usual to **normalize the inputs** when using ridge regression

Other regularization methods

$$\mathcal{E}_\lambda(w_0, \dots, w_D) = \underbrace{\sum_{n=1}^N \left(y_n - \mathbf{w}^T \phi(\mathbf{x}_n) \right)^2}_{(\mathbf{y} - \Phi \mathbf{w})^T (\mathbf{y} - \Phi \mathbf{w})} + \lambda \sum_{d=1}^D |w_d|^q$$

- ▶ $q = 1$ Lasso method
- ▶ $q = 2$ Ridge regression
- ▶ *etc.*

Reminder on constrained optimization theory

$$\min_{\mathbf{w}} \left[\sum_{n=1}^N \left(y_n - \mathbf{w}^T \phi(\mathbf{x}_n) \right)^2 + \lambda \sum_{d=1}^D |w_d|^q \right]$$

is the dual Lagrangian form of:

$$\min_{\mathbf{w}} \sum_{n=1}^N \left(y_n - \mathbf{w}^T \phi(\mathbf{x}_n) \right)^2 \quad \text{subject to} \quad \sum_{d=1}^D |w_d|^q \leq \eta \quad (1)$$

One-to-one correspondance between λ and η

Reminder on constrained optimization theory

KKT conditions:

Suppose that \mathbf{w}^* is a local solution of (1). If the objective function and constraints are continuously differentiable and the constraints are qualified, then there exists λ^* such that the following conditions hold at $(\mathbf{w}^*, \lambda^*)$:

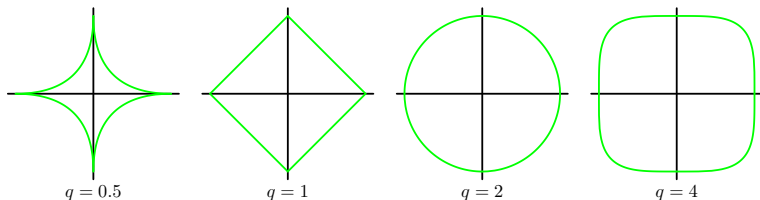
$$\nabla_{\mathbf{w}} L(\mathbf{w}^*, \lambda^*) = 0$$

$$\sum_{d=1}^D |w_d^*|^q - \eta \leq 0$$

$$\lambda^* \geq 0$$

$$\lambda^* \left(\sum_{d=1}^D |w_d^*|^q - \eta \right) = 0$$

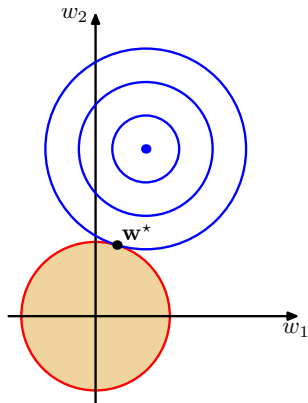
Contour of the constraint



Source images : C. Bishop, *Pattern Recognition and Machine Learning*

Ridge regression ($q=2$)

Source image : C. Bishop, *Pattern Recognition and Machine Learning*

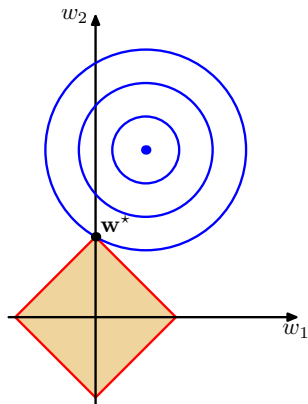


Contours of the constraint (red) and objective-function (blue)

- w_1 and w_2 have non-zero values

Lasso regression ($q=1$)

Source image : C. Bishop, *Pattern Recognition and Machine Learning*



► $w_1 = 0 \Rightarrow$ feature ϕ_1 is not used!

For λ sufficiently large, some w_i are driven to zero, and the corresponding features play no role

Sparse model

Statistical model

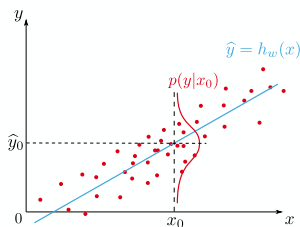
Model: $Y = \mathbf{w}^T \mathbf{X} + \varepsilon$

- ▶ Y and ε are random variables
- ▶ X is either **fixed** or **random**, depending on choice of model:
 - ▶ outcome y_n drawn from a conditional distribution $p(y|X = x_n)$
 - ▶ or outcome (x_n, y_n) drawn from a joint distribution $p(x, y)$

Linearity + assumptions on $\varepsilon \rightsquigarrow$ statistical properties of the model

Usual assumptions

$$Y = h_w(X) + \varepsilon \quad \text{with } h_w(x) = w^T x$$



1. Linearity: $\mathbb{E}(\varepsilon|X = x) = 0$, or equivalently $\mathbb{E}(Y|X = x) = h_w(x) = w^T x$
2. Independance of errors (no auto-correlation)
3. Homoscedasticity of ε (constant variance σ_ε^2)
4. X values are fixed, or, if random, independant of the errors
5. (Normality: ε has a Gaussian density)

Usual assumptions

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1D} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nD} \\ \vdots & \vdots & & \vdots \\ 1 & x_{N1} & \dots & x_{ND} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$$

$$\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{X}\mathbf{w}$$

Assumptions of linearity ($\mathbb{E}(\boldsymbol{\varepsilon}) = 0$), constant variance $\sigma_{\boldsymbol{\varepsilon}}^2$ and no auto-correlation can be expressed as:

- ▶ $\mathbb{E}[\boldsymbol{\varepsilon}] = 0$
- ▶ $\mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T] = \sigma_{\boldsymbol{\varepsilon}}^2 \mathbf{I}_N$

Expectations over different draws of data sample $S = (\mathbf{X}, \mathbf{y})$

Statistical properties of the least squares estimator

We here assume that $\mathbf{X}^T \mathbf{X}$ is invertible of rank $D + 1$

- Under assumption $\mathbb{E}[\epsilon] = 0$, the least squares estimator $\hat{\mathbf{w}}^{\text{LS}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ is **unbiased**:

$$\mathbb{E}(\hat{\mathbf{w}}^{\text{LS}}) = \mathbf{w}$$

Note: basic assumption is that the “true model” is linear $y = \mathbf{w}^T \mathbf{x}$, which is not necessarily true in practice

- Under assumptions $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[\epsilon \epsilon^T] = \sigma_\epsilon^2 \mathbf{I}_N$, the variance of the least squares estimator is:

$$\mathbb{V}(\hat{\mathbf{w}}^{\text{LS}}) = \sigma_\epsilon^2 (\mathbf{X}^T \mathbf{X})^{-1}$$

Gauss-Markov theorem: $\hat{\mathbf{w}}^{\text{LS}}$ is the best linear unbiased estimator (BLUE), i.e. the one of **minimum variance**

Statistical properties of the least squares estimator

- Under additional assumption of normality, the least squares estimator corresponds to the **maximum-likelihood** estimator

This was demonstrated in the lesson on inference

$\hat{\mathbf{w}}^{\text{LS}}$ follows a Gaussian law $\mathcal{N}\left(\mathbf{w}, \sigma_{\epsilon}^2 (\mathbf{X}^T \mathbf{X})^{-1}\right)$ (assuming \mathbf{w} is the “true”, unknown, value)

The variance $\sigma_{\epsilon}^2 (\mathbf{X}^T \mathbf{X})^{-1}$ can be used to compute confidence intervals. However, σ_{ϵ}^2 is unknown.

Unbiased estimation:

$$\widehat{\sigma_{\epsilon}^2} = \frac{\boldsymbol{\epsilon}^T \boldsymbol{\epsilon}}{N - D - 1} = \frac{1}{N - D - 1} \sum_{n=1}^N (y_n - \hat{y}_n)^2$$

Properties of ridge regression

- Biased estimation:

$$\mathbb{E}(\hat{\mathbf{w}}_c^{\text{ridge}} | \lambda) = \mathbf{w}_c - \lambda \mathbf{S}_\lambda^{-1} \mathbf{w}_c \quad \text{with } \mathbf{S}_\lambda = \mathbf{X}_c^T \mathbf{X}_c + \lambda \mathbf{I}_D$$

- Smaller variance than least-squares!

$$\mathbb{V}(\hat{\mathbf{w}}_c^{\text{ridge}} | \lambda) = \mathbf{P}^T \text{Diag} \left(\frac{\lambda_1}{(\lambda_1 + \lambda)^2}, \dots, \frac{\lambda_D}{(\lambda_D + \lambda)^2} \right) \mathbf{P} \text{ where } \mathbf{P} \text{ is}$$

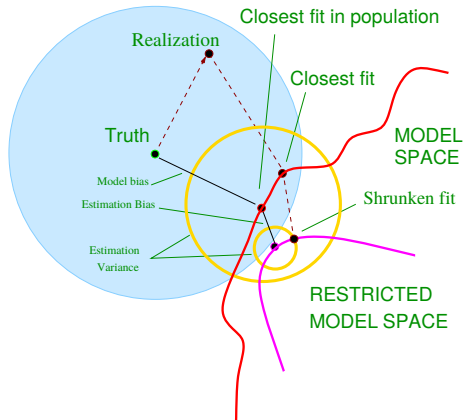
an orthogonal matrix, and $\lambda_1, \dots, \lambda_D$ are the eigenvalues of $\mathbf{X}_c^T \mathbf{X}_c$

The hyperparameter λ sets the bias-variance compromise

- Also, if $\mathbf{X}^T \mathbf{X}$ is singular, we can choose λ large enough so that \mathbf{S}_λ is not singular!

Bias-variance compromise

Elements of Statistical Learning (2nd Ed.) c Hastie, Tibshirani & Friedman 2009 Chap 7



How closely does the model fit the data?

Residual Standard Error

$$\hat{\sigma}_{\epsilon} = \sqrt{\frac{\epsilon^T \epsilon}{N - D - 1}} = \sqrt{\frac{1}{N - D - 1} \sum_{n=1}^N (y_n - \hat{y}_n)^2}$$

Absolute measure of the lack of fit, expressed in units of y

How closely does the model fit the data?

R^2 statistic = $\frac{\text{Variance explained}}{\text{Total variance}}$

$$\begin{aligned} R^2 &= \frac{(\hat{\mathbf{y}} - \bar{\mathbf{y}})^T (\hat{\mathbf{y}} - \bar{\mathbf{y}})}{(\mathbf{y} - \bar{\mathbf{y}})^T (\mathbf{y} - \bar{\mathbf{y}})} = \frac{\sum_{n=1}^N (\hat{y}_n - \bar{y})^2}{\sum_{n=1}^N (y_n - \bar{y})^2} \\ &= 1 - \frac{\sum_{n=1}^N \varepsilon_n^2}{\sum_{n=1}^N (y_n - \bar{y})^2} \end{aligned}$$

Dimensionless. The closest to 1, the better.

Geometric interpretation

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1D} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nD} \\ \vdots & \vdots & & \vdots \\ 1 & x_{N1} & \dots & x_{ND} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$$

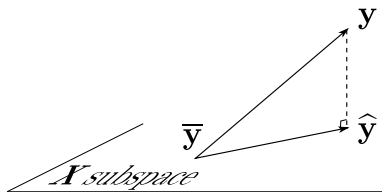
Linear least squares regression: $\mathcal{E}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$

$$\frac{\partial \mathcal{E}}{\partial \mathbf{w}} = 0 \quad \Leftrightarrow \quad \mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$$

$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}^{\text{LS}}$ is the **orthogonal projection** of \mathbf{y} onto the subspace spanned by column vectors $\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_D$ (column space)

Geometric interpretation

Both $\bar{\mathbf{y}}$ and $\hat{\mathbf{y}}$ are in the subspace spanned by $\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_D$
 $\bar{\mathbf{y}} = \bar{y}\mathbf{X}_0$ (with $\mathbf{X}_0 = \mathbf{1}$) and $\hat{\mathbf{y}} = \sum_{d=0}^D \hat{w}_d \mathbf{X}_d$



R is the cosine of the angle between $\mathbf{y} - \bar{\mathbf{y}}$ and $\hat{\mathbf{y}} - \bar{\mathbf{y}}$

Linear models with response $y \in \mathbb{R}^P$

So far, we have considered regression with a response $y \in \mathbb{R}$

$D = \dim \mathcal{X} = 1$: simple regression

$D = \dim \mathcal{X} > 1$: multiple regression

What happens if we consider models with multiple outputs, where $y = (y_1, \dots, y_P, \dots, y_P)^T$ has P components, with $P > 1$?

$$\mathcal{H} = \{h_W : \mathcal{X}_e \rightarrow \mathbb{R}^P, \quad h_W(x) = Wx\}$$

Linear models with response $y \in \mathbb{R}^P$

Can we use P separate linear models $\hat{y}_p = w_p^T x$?

Direct approach (model=function): yes

Probabilistic approach (model= density): it depends

Assume $y = \hat{y} + \varepsilon$ and noise ε follows Gaussian law $\mathcal{N}(0_P, \Sigma)$:

$$p(\varepsilon) = (2\pi)^{-\frac{P}{2}} |\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} \varepsilon^T \Sigma^{-1} \varepsilon \right\}$$

- ▶ If Σ is diagonal, we have P independent linear models
- ▶ If Σ is not diagonal, we cannot consider the P response components independently

Principles and Methodology of Machine Learning

1. Introduction
2. Supervised learning
3. Principles of inference
4. Illustration with linear models
5. Methodology
6. Unsupervised learning
7. Reinforcement learning

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

4. Illustration with linear models

5. Methodology

- Pitfalls and issues in Machine Learning
- The bias-variance tradeoff
- Model selection and evaluation
- Feature selection and extraction

6. Unsupervised learning

Pitfalls and issues in Machine Learning

- **Overfitting** issues when minimizing **empirical risk**

$$\mathcal{R}_{emp}(h, S) = \frac{1}{|S|} \sum_{(x_n, y_n) \in S} \ell(y_n, h(x_n))$$

using a finite set of examples $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$, instead of minimising the real risk (loss expectation)

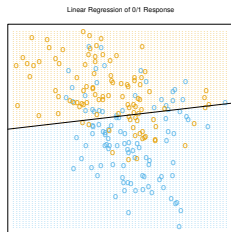
$$\mathcal{R}(h) = \mathbb{E}_{X, Y} [\ell(Y, h(X))] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, h(x)) \mathbb{P}_{X, Y}(dx, dy)$$

- **Curse of dimensionality**

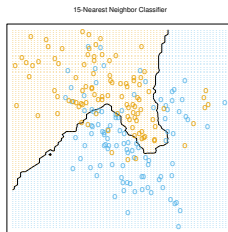
Methods performance \searrow when $\dim \mathcal{X} \nearrow$

Overfitting

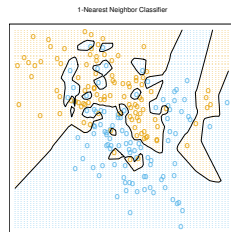
An example on a classification problem



Linear Model



15-Nearest Neighbours



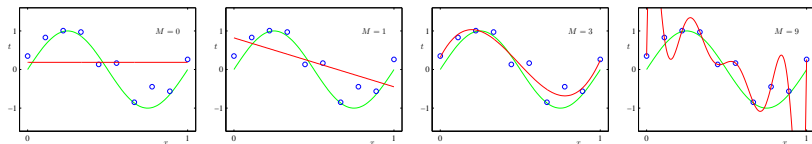
1-Nearest Neighbour

Source images : Hastie, Tibshirani & Friedman, *Elements of Statistical Learning* (2nd Ed.)

What is the best compromise?

Overfitting

An example on a regression problem, using polynomial models

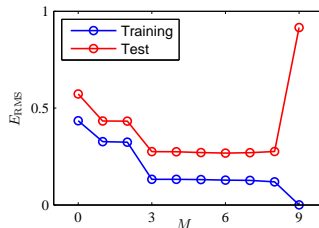


Source images : C. Bishop, *Pattern Recognition and Machine Learning*

Influence of model complexity

Simple models

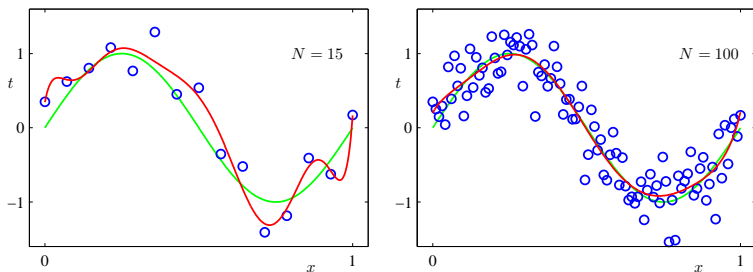
High bias
Low variance



Complex models

Low bias
High variance

Influence of dataset size



Source image : C. Bishop, *Pattern Recognition and Machine Learning*

- ▶ N small : h is “flexible enough” to adjust to points, but does not generalize well
- ▶ N big : h not enough “flexible” to pass through all points, but closer to f

“Flexibility” (complexity of h) $\approx M$ (number of parameters to adjust)

Overfitting increases with $\dim \mathcal{X}$

Example:

Linear model with input $x = (x_1, \dots, x_D)^T$ of dimension D

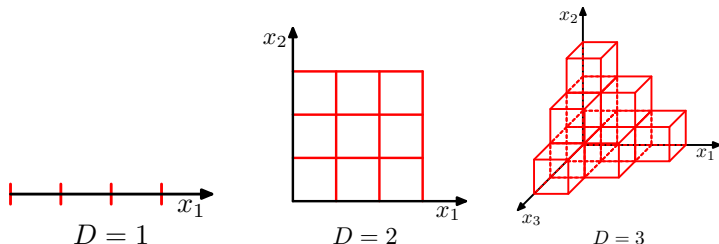
$$y = \beta_0 + \sum_{d=1}^D \beta_d x_d$$

Increasing D (*i.e.* using more explanatory variables) will increase the number of parameters β_d to adjust

\Rightarrow risk of overfitting \nearrow

Curse of dimensionality

It's more difficult to sample \mathcal{X} when $D = \dim \mathcal{X} \nearrow$



Source images : C. Bishop, *Pattern Recognition and Machine Learning*

Geometric volumes behave weirdly when $\dim \mathcal{X} \nearrow$

E.g. most of the volume of a sphere is near the surface, when D high

\Rightarrow Many methods (e.g. polynomial fitting) do not scale well

What to remember on these pitfalls ?

Observed phenomenons:

- ▶ Overfitting leads to poor generalization performances
- ▶ Overfitting ↗ when N (dataset size) ↘
- ▶ Overfitting ↗ when M (model complexity) ↗
- ▶ Overfitting ↗ when D (dimension of input) ↗
- ▶ Some methods do not scale well when D increases
- ▶ High bias, low variance for simple models
- ▶ Low bias, high variance for complex models

Other potential issues

► **Reminder:**

A model can only be as good as the data used to build it

- Build your training set with data that is representative of what you want to learn
 - Avoid undue extrapolation. Do not use a predictive model in a context that is different from the training data
- **Data leakage:** When assessing the generalization error on a separate dataset (called test set), different from the training set, one must prevent information leaks between the two sets.

Beware of data leakage!

Data leakage can lead to optimistically biased models

Training data must only contain information about the underlying phenomenon that produced the data. It should have no knowledge of the test data!

- ▶ Avoid duplications and dependencies
- ▶ Do not use outside information that would not be present at the time of prediction (*i.e.* data that could be used to guess the values of the target y in the test set but not in other unseen data)
- ▶ Pre-processing (variable selection/extraction, normalization) should be done within the training set, and then applied also to the test set. It should NOT be done before splitting the data
- ▶ Make a final sanity check of your trained model on a hold-back validation set

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

4. Illustration with linear models

5. Methodology

- Pitfalls and issues in Machine Learning
- The bias-variance tradeoff
- Model selection and evaluation
- Feature selection and extraction

6. Unsupervised learning

Bias and variance

Dataset S considered as an **outcome of a random variable D**

Decomposition of the error, assuming a quadratic loss:

$$\begin{aligned}\ell(y, h(x)) &= (y - h(x))^2 = (y - \mathbb{E} h(x) + \mathbb{E} h(x) - h(x))^2 \\ &= (y - \mathbb{E} h(x))^2 + (\mathbb{E} h(x) - h(x))^2 \\ &\quad + 2(y - \mathbb{E} h(x))(\mathbb{E} h(x) - h(x))\end{aligned}$$

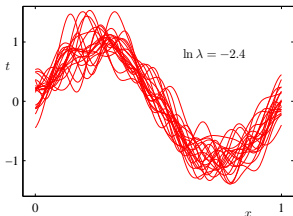
Noticing that $\mathbb{E} [\mathbb{E} h(x) - h(x)] = \mathbb{E} h(x) - \mathbb{E} h(x) = 0$, we have

$$\begin{aligned}\text{Expected error}(x) &= \mathbb{E}_D [h(x) - y]^2 \\ &= \underbrace{\mathbb{E}_D [h(x) - \mathbb{E}_D h(x)]^2}_{\substack{\text{Variance} \\ \text{Estimation error}}} + \underbrace{\mathbb{E}_D [\mathbb{E}_D h(x) - y]^2}_{\substack{\mathbb{E}[\text{Bias}^2] \\ \text{Approximation error}}}\end{aligned}$$

Bias and variance

Dataset S considered as an outcome of a random variable D

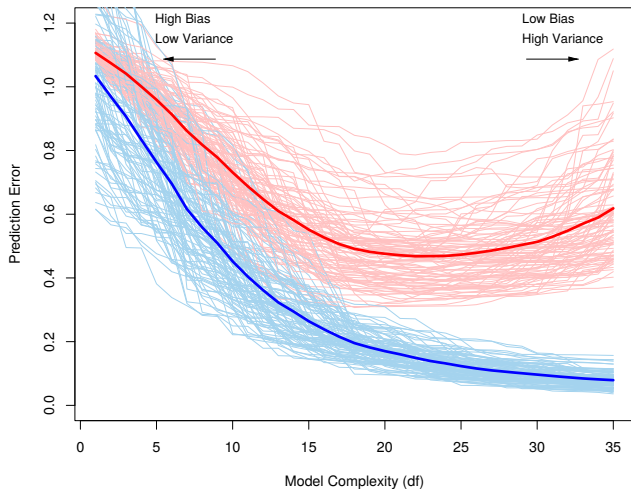
- **Bias:** Difference between average value $\mathbb{E}_D h(x)$ and $y = f(x)$
- **Variance:** $\mathbb{E}_D [h(x) - \mathbb{E}_D h(x)]^2$
Dispersion of $h(x)$ around average $\mathbb{E}_D h(x)$, considering all possible datasets S (draws of D)



Source image : C. Bishop, *Pattern Recognition and Machine Learning*

Bias-variance tradeoff

Elements of Statistical Learning (2nd Ed.) c Hastie, Tibshirani & Friedman 2009 Chap 7



Training and test errors

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

4. Illustration with linear models

5. Methodology

- Pitfalls and issues in Machine Learning
- The bias-variance tradeoff
- Model selection and evaluation
 - Cross-validation
 - Bootstrap
- Feature selection and extraction

Model selection

The term “model” is ambiguous. It can refer to

- ▶ a specific hypothesis $h \in \mathcal{H}$ (predictive model)

E.g. linear model $h_{\mathbf{w}}$ with adjusted weights $\mathbf{w} = \hat{\mathbf{w}}$

- ▶ or a subspace $\mathcal{H}_c \subset \mathcal{H}$

E.g. linear model with a given subset of features as input but with unadjusted weights, or multilayer perceptron with a given number of hidden layers and units.

Model selection: we are in the second case

Considering sub-spaces $\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_C$ of increasing complexity, we want to select the “best” model \mathcal{H}_c

Model selection

Problem statement

$\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_C$ of different complexities

Example: Neural network with c hidden units, $c \in \{1, \dots, C\}$, on a single hidden layer

We want to select model \mathcal{H}_c with the best generalization error

Problem: we only have S to estimate the generalization error

Model selection \neq Adjusting a prediction model

Model selection: select \mathcal{H}_c in $\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_C$

Adjusting a prediction model: find $\hat{\theta}_c = \hat{\theta}_c(S)$ such that $h_{\hat{\theta}_c}$ is the best hypothesis in subspace \mathcal{H}_c

Hold-out validation

Model selection:



For each candidate model:

- Adjust the model on S_T
- Assess error on S_V

Select the model
minimizing error on S_V

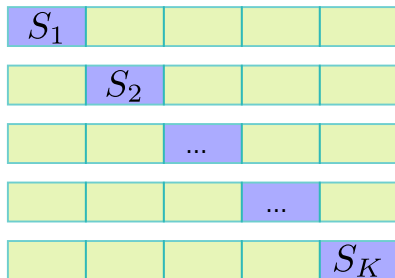
Prediction:



Apply adjusted model
on new entries

K-fold cross-validation

Model selection:



For each candidate model:

For each fold k :

- Adjust model using $S_{-k} = \bigcup_{i \neq k} S_i$
- Assess error using S_k

Compute the cumulated error for the K folds

Select model minimizing the error on the K validation subsets S_k

Prediction:



Adjust the best model found

Apply to fresh entries

Cross-validation error

Denoting $\hat{h}_c^{S_{-k}} = \mathcal{A}(S_{-k}) \in \mathcal{H}_c$ the model tuned on S_{-k} using a machine learning algorithm \mathcal{A} , the error on the validation set S_k is:

$$Err_k = \frac{1}{|S_k|} \sum_{n=1}^{|S_k|} \ell(y_n, \hat{h}^{S_{-k}}(x_n))$$

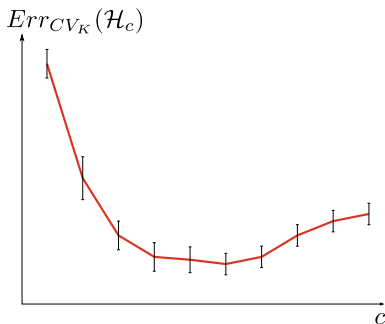
The cross-validation error for model \mathcal{H}_c is:

$$\begin{aligned} Err_{CV_K}(\mathcal{H}_c) &= \frac{1}{|S|} \sum_{k=1}^K |S_k| Err_k \\ &= \frac{1}{N} \sum_{n=1}^N \ell(y_n, \hat{h}_c^{S_{-k(n)}}(x_n)) \end{aligned}$$

where $k(n)$ is the fold k to which example n belongs

Model selection using K -fold cross-validation

- ▶ Select model \mathcal{H}_c in $\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_C$ for which $Err_{CV_K}(\mathcal{H}_c)$ is the smallest
- ▶ Or “one-standard error” rule: Select the most parsimonious model whose error is no more than one standard error above the one of the best model.



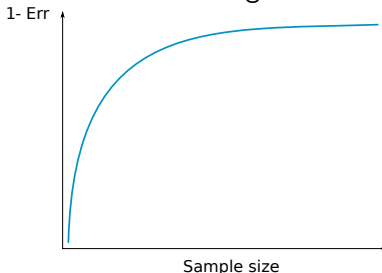
What value for K ?

$K = N$ Leave-one-out validation (LOOV)

For each of the N folds, error is assessed on 1 left-out sample.

$K < N$ Usual choice for K -fold cross validation is $K = 5$ to 10

Learning curve (where Err is the actual generalization error):



K too small \rightarrow upward bias of the error

K too large \rightarrow variance of the error \nearrow

K -fold cross-validation

Denoting \mathcal{A} a machine learning algorithm, K -fold cross-validation can be formalized as follows:

$$CV_K(\mathcal{A}, \mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_C) : \bigcup_{N=0}^{\infty} (\mathcal{X} \times \mathcal{Y})^N \rightarrow \mathcal{H} = \bigcup_{c=1}^C \mathcal{H}_c$$
$$S = \{(x_n, y_n)_{1 \leq n \leq N}\} \rightarrow h_S^*$$

CV_K is itself a machine learning algorithm

Nested cross-validation

We can use an “inner cross-validation” $CV_{K'}^{inner}$ as input algorithm \mathcal{A} of the “outer cross-validation” CV_K

Example: inner CV to select hyperparameter λ of a ridge regression, outer CV to select relevant subset of input variables

$$CV_K(CV_{K'}^{inner}(\mathcal{A}, \mathcal{H}_{\lambda_1}, \mathcal{H}_{\lambda_2}, \dots, \mathcal{H}_{\lambda_m}), \mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_C)$$

Exercise 5.1

During an internship, a student is assigned the following task: Learn a model predicting the arrival time of airborne flights, using a dataset of past trajectories. The study is limited to a single airport.

- ▶ Target variable: number of seconds before touchdown
- ▶ Explanatory variables: aircraft type, aircraft position (latitude, longitude), ground speed, track angle, vertical speed, remaining distance, time of day, weekday, month, wind (direction and magnitude) measured at the current position, wind predicted at 5 points on the path to go.

Exercise 5.1 (continued)

In a preliminary approach, the student collects 10 trajectories. Each trajectory is sampled with 10 points per trajectory and the target and explanatory variables are computed for each point.

The resulting dataset of 100 points is randomly split as follows: 70 points in a training set, 30 points in a test set.

The student then trains a neural network with 10 hidden layers and 10 units per hidden layer on the training set.

Questions:

What is the main methodological issue with this approach? Would it change anything to sample more points per trajectory?

Solution to exercise 5.1

Exercise 5.2

Same context as exercise 5.1

The student now collects 30 million points from 20000 flights. A training set is built by sampling 25 million points randomly. The remaining 5 million points are kept as a test set.

The neural network with 10 hidden layers and 10 units per layer now gives excellent results on the test set.

It is decided to test the prediction model on the airport.

Questions:

The results are disappointing. Why? What are the remaining issues in the student's approach?

What solution do you propose?

Solution to exercise 5.2

Exercise 5.3

Same context as exercise 5.1 and 5.2

The student now wishes to select among several network topologies (e.g. with increasing numbers of hidden layers) which one is the best.

Question:

How would you proceed to select the best model, and to assess its results?

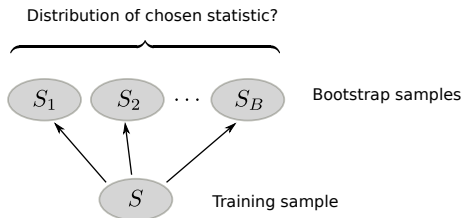
Solution to exercise 5.3

Bootstrap

Cross-validation vs. bootstrap

- ▶ Cross-validation: Assess generalization on **separate samples**
- ▶ Bootstrap: **Resample** from the same dataset

Bootstrapping to estimate the distribution of a statistic



1. Draw with replacement several datasets S_1, \dots, S_B of size N , from the initial dataset S
2. Compute chosen statistic on each S_b (e.g standard error, confidence interval)
3. Estimate any aspect of the distribution (mean, variance, etc) of this statistic from the values obtained from S_1, \dots, S_B

Bootstrapping to estimate the generalization error

Idea:

- ▶ Tune a model \hat{h}^{S_b} on each bootstrap sample S_b
- ▶ Assess prediction error on initial dataset S

Problem: $\frac{1}{B} \frac{1}{N} \sum_{b=1}^B \sum_{n=1}^N \ell(y_n, \hat{h}^{S_b}(x_n))$ is a biased estimate of the generalisation error because S and S_b are overlapping

Idea 2: consider only observations that are not in the bootstrap sample

Bootstrap estimates of generalization error

$$\widehat{Err}^{(1)} = \frac{1}{N} \sum_{n=1}^N \frac{1}{|C^{-n}|} \sum_{b \in C^{-n}} \ell(y_n, \hat{h}^{S_b}(x_n))$$

where C^{-n} is the set of indices of the bootstrap samples S_b that do not contain observation n

$$\widehat{Err}^{(.632)} = 0.368 \overline{err} + 0.632 \widehat{Err}^{(1)}$$

where \overline{err} is the average training error

$\widehat{Err}^{(.632)}$ alleviates the upward bias of $\widehat{Err}^{(1)}$ due to $|C^{-n}| < |S|$
 $\mathbb{P}(\text{obs. } n \in S_b) = 1 - (1 - \frac{1}{N})^N \rightarrow 1 - e^{-1} \approx 0.632$

Principles and Methodology of Machine Learning

1. Introduction

2. Supervised learning

3. Principles of inference

4. Illustration with linear models

5. Methodology

- Pitfalls and issues in Machine Learning
- The bias-variance tradeoff
- Model selection and evaluation
- Feature selection and extraction
 - Feature selection
 - Feature extraction

Feature selection and feature extraction

Workarounds for the curse of dimension ?

Reduce dimensionality of inputs

- ▶ Selection of relevant explanatory variables (features)
Scoring, forward or backward selection, etc.
- ▶ Extraction of new variables
Principal component analysis, autoencoding neural networks, etc.

Feature selection

- ▶ Scoring individual variables
- ▶ Scoring subsets of variables
- ▶ Integrated methods

Scoring individual variables

Example: $y \in \mathbb{R}$ and $(x_1, \dots, x_D) \in \mathbb{R}^D$

$$Score = \frac{Cov(x_d, y)}{\sigma_{x_d} \sigma_y}$$

Easy to compute, in $\mathcal{O}(D)$

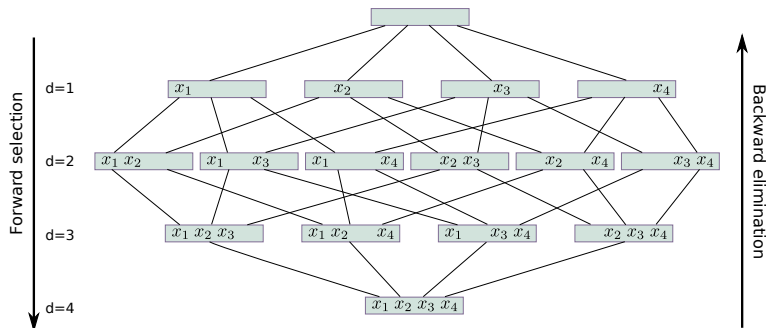
Issues:

- ▶ Redundant variables with high scores will all be selected
- ▶ Individual variables with low scores will be discarded, whereas they might be highly relevant when combined

Scoring subsets of variables

Idea: apply a same ML algorithm to different subsets of variables

Issue: highly combinatorial



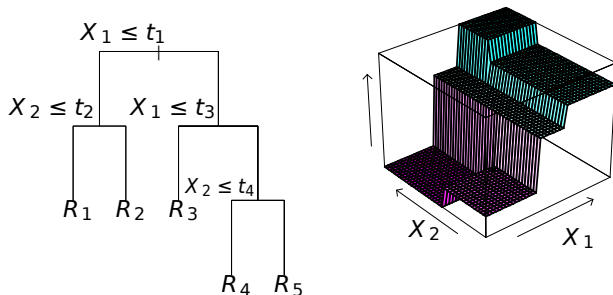
For a given dimension $d \leq D$, score each subset using a criterion J (e.g. cross-validation error, or information criterion), and select subset of highest score

Integrated variable selection

In some methods, variable selection is part of the algorithm

Example: Recursive binary splitting in Classification and Regression Trees (CART) split regions of \mathcal{X} , selecting the most pertinent variable at each step

Elements of Statistical Learning (2nd Ed.) c Hastie, Tibshirani & Friedman 2009 Chap 9



Feature extraction

Some explanatory variables may be redundant

Some combinations of variables might be more relevant than the individual variables

Idea: Transform the initial variables x_1, \dots, x_D so as to

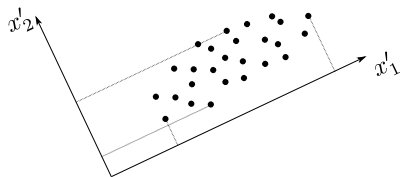
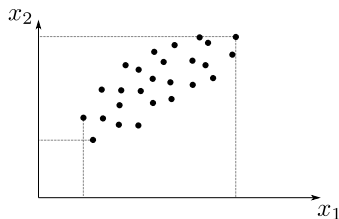
- ▶ Identify latent variables that are relevant to the learning problem at hand
- ▶ Determine the intrinsic dimensionality of the input space

Example: Principal component analysis

Finds a basis of (orthogonal) eigenvectors

Principal component analysis (PCA)

Also known as Karhunen-Loève transform



- ▶ Left: Dispersion of data is almost identical along x_1 and x_2
- ▶ Right: Much larger variance along x'_1 . If variance of data projected onto x'_2 is close to 0, we can discard x'_2

Principal component analysis (PCA)

Considering \mathcal{X} of dimension D and a dataset $\mathbf{x}_1, \dots, \mathbf{x}_N$ where each \mathbf{x}_n is a vector $\mathbf{x}_n = (x_{n1}, \dots, x_{nD})^T$

Variance of data projected onto a direction \mathbf{u} :

$$\text{Var}_{\mathbf{u}} = \frac{1}{N} \sum_{n=1}^N \left(\mathbf{u}^T \mathbf{x}_n - \mathbf{u}^T \bar{\mathbf{x}} \right)^2 = \mathbf{u}^T \Sigma \mathbf{u}$$

where $\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$ is the mean value of $\mathbf{x}_1, \dots, \mathbf{x}_N$

and Σ is the covariance matrix $\Sigma = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$

Principal component analysis (PCA)

Maximizing variance along direction u is an optimization problem:

$$\max_{u \in \mathcal{X}^D} u^T \Sigma u \quad \text{under constraints} \quad u^T u = 1$$

Dual Lagrangian form:

$$\max_{u \in \mathcal{X}^D} \left[u^T \Sigma u + \lambda(1 - u^T u) \right]$$

Cancelling the derivative with respect to u , we obtain:

$$\Sigma u = \lambda u$$

i.e. u must be an eigenvector, with corresponding eigenvalue λ

Principal component analysis (PCA)

Projecting $\Sigma \mathbf{u} = \lambda \mathbf{u}$ onto \mathbf{u} , and knowing that $\mathbf{u}^T \mathbf{u} = 1$, we obtain:

$$\text{Var}_{\mathbf{u}} = \mathbf{u}^T \Sigma \mathbf{u} = \lambda$$

Finding a direction \mathbf{u} maximizing the variance $\mathbf{u}^T \Sigma \mathbf{u}$

\rightsquigarrow Select eigenvector \mathbf{u}_1 of highest eigenvalue λ_1

The process can be repeated M times ($M \leq D$)

Principal component analysis (PCA)

$M = D$ (rotation)

Coordinates x_1, \dots, x_N of vector \mathbf{x} are replaced by x'_1, \dots, x'_N where the $x'_d = \mathbf{x}^T \mathbf{u}_d$ are the latent variables best explaining the variance of the data

One can then apply a variable selection procedure to the rotated variables

$$M < D \text{ (compression)} \quad \tilde{\mathbf{x}}_n = \sum_{d=1}^M \underbrace{(\mathbf{x}_n^T \mathbf{u}_d)}_{x'_{nd}} \mathbf{u}_d + \sum_{d=M+1}^D b_d \mathbf{u}_d$$

Optimum choice is $b_d = \bar{\mathbf{x}}^T \mathbf{u}_d$ when minimizing compression error

$$J = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2. \text{ Then } J = \sum_{d=M+1}^D \lambda_d.$$

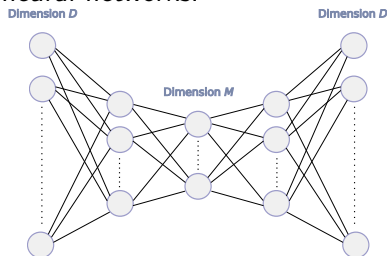
We neglect the $D - M$ terms of lowest variance.

Other transformations

PCA: linear combinations of variables

Not always the best choice. There are other approaches

E.g. auto-encoding neural networks:



Network is first tuned, using only the input variables as examples

Left part of the tuned network is used to compute new features of dimension $M < D$

“Integrated feature extraction”

In deep neural networks, the hidden layers extract and combine features in a hierarchic way: low-level features for layers close to the input, and higher-level features towards the output

In that sense, we can say that deep networks automatically extract the relevant features

Principles and Methodology of Machine Learning

1. Introduction
2. Supervised learning
3. Principles of inference
4. Illustration with linear models
5. Methodology
6. Unsupervised learning
7. Reinforcement learning

Types of learning tasks

- ▶ Supervised learning, using examples $\{(x_1, y_1), \dots, (x_N, y_N)\}$
 - ▶ Classification : y_n is a label $\mathcal{C}_1, \dots, \mathcal{C}_p$
E.g. digit recognition
 - ▶ Regression : $y_n \in \mathbb{R}$ or $y_n \in \mathbb{R}^P$
E.g. curve fitting, surface fitting
- ▶ **Unsupervised learning**, using examples $\{x_1, \dots, x_N\}$
 - ▶ Clustering: *cluster data according to similarity (distance)*
 - ▶ Density estimation: *What is the data distribution ?*
- ▶ Reinforcement learning *action* \rightarrow *reward*.
Objective: Maximize the reward cumulated over time

Unsupervised learning vs. supervised learning

Supervised learning

- ▶ uses labelled examples $\{(x_1, y_1), \dots, (x_N, y_N)\}$
- ▶ to approximate $p(y|x)$ (or $p(x, y)$)
- ▶ or directly estimate y for any new entry x

Unsupervised learning

- ▶ uses unlabelled examples $\{x_1, \dots, x_N\}$
- ▶ to approximate $p(x)$
- ▶ or identify characteristic features of the distribution of x

Unsupervised learning

Why are we interested in the distribution of x ?

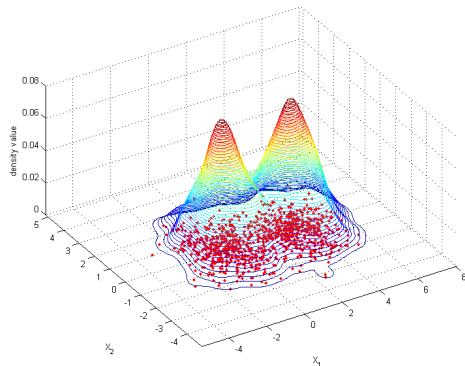
- ▶ Density estimation
- ▶ Data compression
- ▶ Dimensionality reduction (feature extraction)
- ▶ Clustering
- ▶ Discover association rules
- ▶ Smooth a signal
- ▶ *etc.*

Unsupervised learning

Techniques:

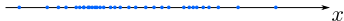
- ▶ Kernel density estimation
- ▶ Gaussian Mixtures
- ▶ Vector quantization, self-organizing maps
- ▶ Expectation-maximization (EM)
- ▶ Principal component analysis (PCA)
- ▶ Singular-value decomposition (SVD)
- ▶ Autoencoders
- ▶ K-means, DBSCAN, spectral clustering,
- ▶ Hierarchical clustering
- ▶ *etc.*

Density estimation



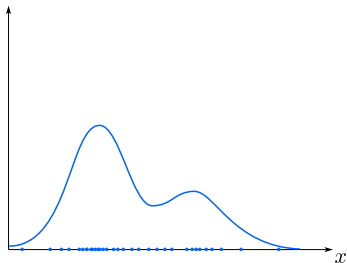
Density estimation

Example in dimension 1 :



Density estimation

What is the probability density of x ?



Density estimation

How do we proceed?

1. Make an assumption on the density
2. Estimate the model parameters, using the data

Exercise:

Assume a Gaussian density:

$$p_X(x) = \mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Given N i.i.d. observations of X , estimate μ and σ^2 , using the maximum likelihood principle.

Exercise 6.1

$$p_X(x) = \mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Find the maximum likelihood estimates of μ and σ^2

Indications:

- ▶ Maximum Likelihood: $(\hat{\mu}^{\text{ML}}, \hat{\sigma}^2)^{\text{ML}} = \arg \max_{\mu, \sigma^2} p(x_1, \dots, x_N | \mu, \sigma^2)$
- ▶ *i.i.d.*: $p(x_1, \dots, x_N | \mu, \sigma^2) = p(x_1 | \mu, \sigma^2) \times \dots \times p(x_N | \mu, \sigma^2)$
- ▶ Take the opposite of the logarithm to find the function $E(\mu, \sigma^2)$ to minimize.
- ▶ Find the estimate $\hat{\mu}$ minimizing $E(\mu, \sigma^2)$ with σ^2 fixed.
- ▶ Find the estimate $\hat{\sigma}^2$ by minimizing $E(\hat{\mu}, \sigma^2)$.

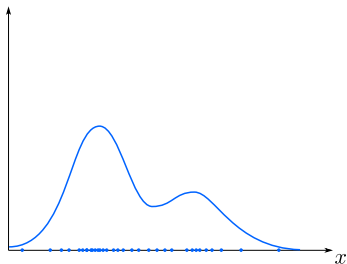
Solution to exercise 6.1

Solution to exercise 6.1

Exercise 6.2

Fitted model: $\widehat{p}_X(x) = \mathcal{N}(x; \widehat{\mu}^{\text{ML}}, \widehat{\sigma}^2)^{\text{ML}}$

Is this choice of model well adapted to this data?



Any idea for a better choice?

Kernel density estimation (KDE)

(x of dimension 1)

True density $p(x)$ unknown

$$\text{Estimated density } \hat{p}(x) = \frac{1}{N} \sum_{n=1}^N K_h(x - x_n) = \frac{1}{Nh} \sum_{n=1}^N K\left(\frac{x - x_n}{h}\right)$$

where

- ▶ K is a “kernel” function
- ▶ of bandwidth h

E.g. $K(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}$ and h std dev. of a Gaussian distribution

Multivariate Kernel Density Estimation

(x of dimension $D > 1$)

True density $p(x)$ unknown

Estimated density $\hat{p}(x) = \frac{1}{N} \sum_{n=1}^N K_H(x - x_n)$

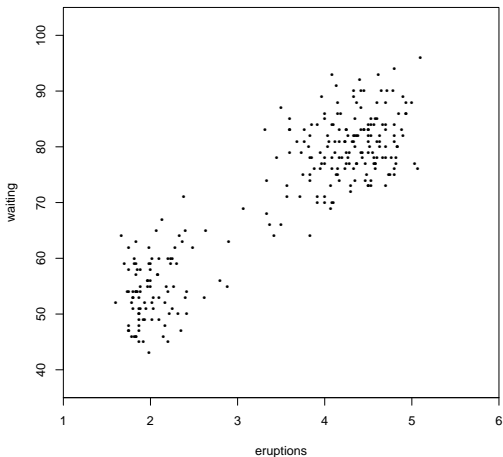
where

- ▶ K_H is a “kernel”
- ▶ of bandwidth H (a matrix)

E.g. $K_H(z) = (2\pi)^{-\frac{D}{2}} |H|^{-\frac{1}{2}} e^{z^T H^{-1} z}$ where H is a covariance matrix

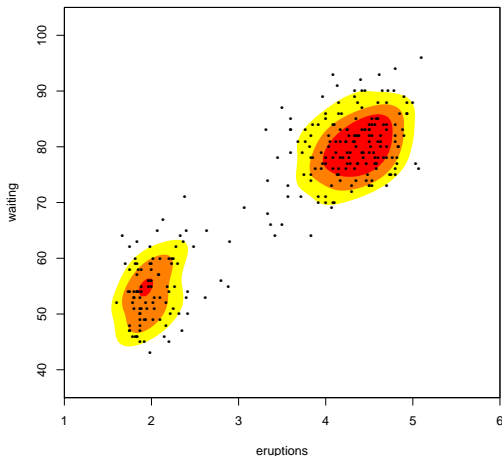
Kernel Density Estimation

Old Faithful eruptions:



Kernel Density Estimation

Old Faithful eruptions (KDE with plugin bandwidth selection):



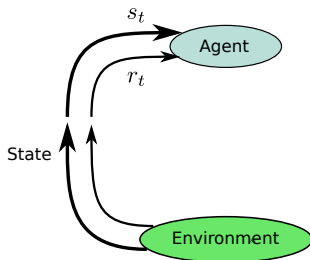
Principles and Methodology of Machine Learning

1. Introduction
2. Supervised learning
3. Principles of inference
4. Illustration with linear models
5. Methodology
6. Unsupervised learning
7. Reinforcement learning

Types of learning tasks

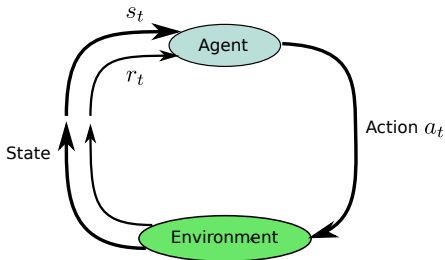
- ▶ Supervised learning, using examples $\{(x_1, y_1), \dots, (x_N, y_N)\}$
 - ▶ Classification : y_n is a label $\mathcal{C}_1, \dots, \mathcal{C}_p$
E.g. digit recognition
 - ▶ Regression : $y_n \in \mathbb{R}$ or $y_n \in \mathbb{R}^P$
E.g. curve fitting, surface fitting
- ▶ Unsupervised learning, using examples $\{x_1, \dots, x_N\}$
 - ▶ Clustering: *cluster data according to similarity (distance)*
 - ▶ Density estimation: *What is the data distribution ?*
- ▶ Reinforcement learning *action* \rightarrow *reward*.
Objective: Maximize the reward cumulated over time

Markov Decision Process



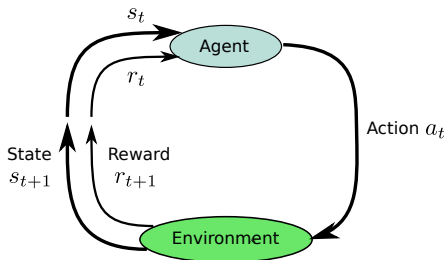
- An agent perceives state $s_t \in \mathcal{S}$ from the environment at t

Markov Decision Process



- It takes action $a_t \in \mathcal{A}$

Markov Decision Process



- ▶ As a result of this action, the agent
 - ▶ transits from state s_t to state s_{t+1} at time $t + 1$ with a probability $\mathbb{P}(s_{t+1}|s_t, a_t) = T(s_{t+1}, s_t, a_t)$
 - ▶ and receives a reward r_{t+1}

Rewards and return

Rewards r_t

Discounted return, when in state s_t at time t :

$$\begin{aligned} G_t &= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \\ &= r_{t+1} + \gamma G_{t+1} \end{aligned}$$

where γ is the discount factor ($0 \leq \gamma \leq 1$)

Markov Decision Process

The objective of the agent is to maximize the cumulated expected discounted reward:

$$\sum_{t=0}^{\infty} \gamma^t r_t \quad (0 \leq \gamma \leq 1)$$

A solution to a MDP is a **policy** $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that defines which action a to execute at given state s .

Reinforcement learning

The objective, in reinforcement learning, is to find an optimal policy π^* , while the reward function R is unknown.

- ▶ Model-based methods: $\mathbb{P}(s_{t+1}|s_t, a_t)$ is known
- ▶ Model-free methods: $\mathbb{P}(s_{t+1}|s_t, a_t)$ is unknown

Reinforcement learning explores successive states and actions, updating utility values for the states or actions, in order to evaluate/improve a policy π

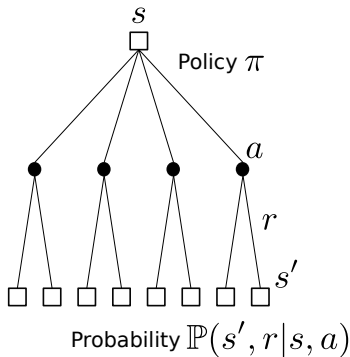
State value function

Expected return of state s under policy π .

$$v_{\pi} : \mathcal{S} \rightarrow \mathbb{R}$$

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s \right]$$

Bellman equation for state values



Bellman equation for the state values under an optimal policy π^* :

$$v_*(s) = \max_a \sum_{s', r} \mathbb{P}(s', r | s, a) [r + \gamma v_*(s')]$$

State-action value function

$$\begin{aligned}q_{\pi} &: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R} \\q_{\pi}(s, a) &= \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \\&= \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a \right]\end{aligned}$$

Expected utility of taking action a in state s under policy π

Bellman equation for the state-action values under an optimal policy π^* :

$$q_*(s, a) = \sum_{s', r} \mathbb{P}(s', r | s, a) [r + \max_{a'} q_*(s', a')]$$

Q-learning

In Q-learning, the objective is to learn an approximation Q of the optimal state-action value function q_* by exploring possible sequences of actions and updating Q -values at each step.

Update rule in TD (Temporal Difference) Q-learning, when transitioning from s to s' by taking action a :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

Deep Q-learning:

When $\mathcal{S} \times \mathcal{A}$ is too large, Q can be learned by a deep neural network