Foundations of Machine Learning
CentraleSupélec — Fall 2017

5. Bayesian decision theory

Chloé-Agathe Azencott
Centre for Computational Biology, Mines ParisTech
chloe-agathe.azencott@mines-paristech.fr

Inserm
Institut national
de la santé et de la recherche médicale

institutCurie

PSL
RESEARCH UNIVERSITY PARIS

MINES
ParisTech

Practical matters...

- **I do not grade homework that is sent as .docx**
- (Partial) solutions to Lab 2 are at the end of the slides of Chap 4.

Learning objectives

After this lecture, you should be able to

- **Apply Bayes rule** for simple inference and decision problems;
- Explain the connection between **Bayes decision rule**, **empirical risk minimization**, **maximum a priori** and **maximum likelihood;**
- Apply the **Naive Bayes** algorithm.

Let's start by tossing coins...

Probability and inference

- Result of **tossing a coin:** x in {heads, tails}
  - x = f(**z**) z: **unobserved variables**
  - Replace f(z) (maybe deterministic but unknown) with the **random variable** X in {0, 1} drawn from a **probability distribution** P(X=x).

Probability and inference

- Result of **tossing a coin:** x in {heads, tails}
  - x = f(**z**) z: **unobserved variables**
  - Replace f(z) (maybe deterministic but unknown) with the **random variable** X in {0, 1} drawn from a **probability distribution** P(X=x).

E.g: a complex physical function of the composition of the coin, the force that is applied to it, initial conditions, etc.

Probability and inference

- Result of **tossing a coin:** x in {heads, tails}
  - x = f(**z**) z: **unobserved variables**
  - Replace f(z) (maybe deterministic but unknown) with the **random variable** X in {0, 1} drawn from a **probability distribution** P(X=x).
- We need to model P

E.g: a complex physical function of the composition of the coin, the force that is applied to it, initial conditions, etc.

Probability and inference

- Result of **tossing a coin:** x in {heads, tails}
  - x = f(**z**) z: **unobserved variables**
  - Replace f(z) (maybe deterministic but unknown) with the **random variable** X in {0, 1} drawn from a **probability distribution** P(X=x).
- **Bernouilli distribution**

$$P(X = x) = {p_0}^x (1 - p_0)^{(1-x)}$$

- We do not know P but a **sample** $X = \{x^i\}_{i=1,\dots,n}$
- Goal: **approximate P** (from which X is drawn)

Probability and inference

- Result of **tossing a coin:** x in {heads, tails}
  - x = f(**z**) z: **unobserved variables**
  - Replace f(z) (maybe deterministic but unknown) with the **random variable** X in {0, 1} drawn from a **probability distribution** P(X=x).
- **Bernouilli distribution**

$$P(X = x) = {p_0}^x (1 - p_0)^{(1-x)}$$

- We do not know P but a **sample** $X = \{x^i\}_{i=1,\dots,n}$
- Goal: **approximate P** (from which X is drawn)

  p_0 = # heads / # tosses
- **Prediction** of next toss: 

Probability and inference

- Result of **tossing a coin:** x in {heads, tails}
  - x = f(**z**) z: **unobserved variables**
  - Replace f(z) (maybe deterministic but unknown) with the **random variable** X in {0, 1} drawn from a **probability distribution** P(X=x).
- **Bernouilli distribution**

$$P(X = x) = {p_0}^x (1 - p_0)^{(1-x)}$$

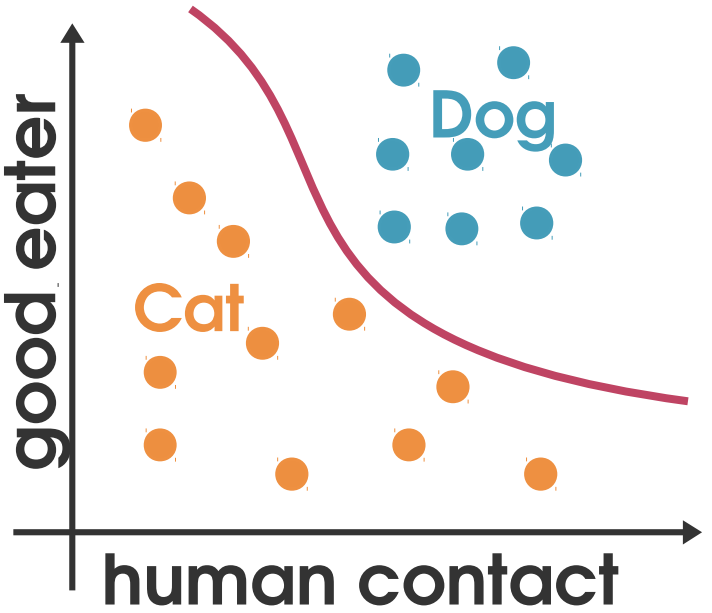
- We do not know P but a **sample** $X = \{x^i\}_{i=1,\dots,n}$
- Goal: **approximate P** (from which X is drawn)

  p_0 = # heads / # tosses
- **Prediction** of next toss:

  heads if $p_0 > 0.5$, tails otherwise

Classification

- Cat vs. dog
  - Cat = 1 (positive)
  - Dog = 0 (negative)
  - x_1 = human contact
  - x_2 = good eater
- **Prediction:**



$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x_1,x_2) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x_1,x_2) > P(y=0|x_1,x_2) \\ 0 & \text{otherwise} \end{cases}$$

Bayes rule

Reverend Thomas Bayes

170?-1761

… possibly

Bayes rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Example: rare disease testing

- test is correct 99% of the time
- disease prevalence = 1 out of 10,000

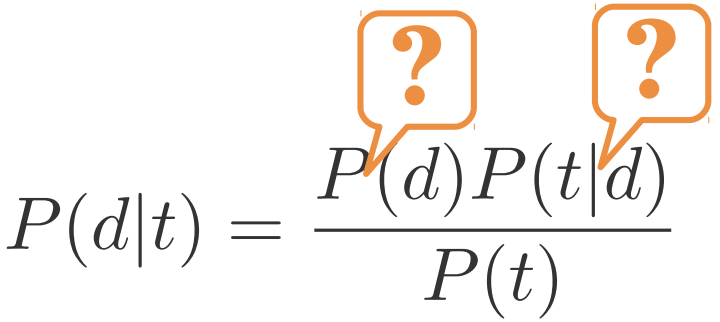
What is the probability that a patient that tested positive actually has the disease?

99% ? 90% ? 10% ? 1% ?

Example: rare disease testing

- test is correct 99% of the time
- disease prevalence = 1 out of 10,000

What is the probability that a patient that tested positive actually has the disease?



$$P(d|t) = \frac{P(d)P(t|d)}{P(t)}$$

Example: rare disease testing

- test is correct 99% of the time $P(t|d) = P(\bar{t}|\bar{d}) = 0.99$
- disease prevalence = 1 out of 10,000 $P(d) = 10^{-4}$

What is the probability that a patient that tested positive actually has the disease?

$$P(d|t) = \frac{\overset{0.0001}{P(d)}\overset{0.99}{P(t|d)}}{P(t)}$$

Example: rare disease testing

- test is correct 99% of the time $P(t|d) = P(\bar{t}|\bar{d}) = 0.99$
- disease prevalence = 1 out of 10,000 $P(d) = 10^{-4}$

What is the probability that a patient that tested positive actually has the disease?

0.0001 0.99

$$P(d|t) = \frac{P(d)P(t|d)}{P(t)}$$

$$P(t) = P(t|d)P(d) + P(t|\bar{d})P(\bar{d})$$

0.99 0.0001

Example: rare disease testing

- test is correct 99% of the time $P(t|d) = P(\bar{t}|\bar{d}) = 0.99$
- disease prevalence = 1 out of 10,000 $P(d) = 10^{-4}$

What is the probability that a patient that tested positive actually has the disease?

0.0001 0.99

$$P(d|t) = \frac{P(d)P(t|d)}{P(t)}$$

(1-0.99)

$$P(t) = P(t|d)P(d) + P(t|\bar{d})P(\bar{d})$$

0.99 0.0001 (1-0.0001)

Example: rare disease testing

- test is correct 99% of the time $P(t|d) = P(\bar{t}|\bar{d}) = 0.99$
- disease prevalence = 1 out of 10,000 $P(d) = 10^{-4}$

What is the probability that a patient that tested positive actually has the disease?

0.0001 0.99

$$P(d|t) = \frac{P(d)P(t|d)}{P(t)} \approx 0.0098.$$

(1-0.99)

$$P(t) = P(t|d)P(d) + P(t|\bar{d})P(\bar{d})$$

0.99 0.0001 (1-0.0001)

Bayes rule

$$P(y = c|\boldsymbol{x}) = \frac{P(y = c)p(\boldsymbol{x}|y = c)}{p(\boldsymbol{x})}$$

prior, likelihood, posterior, evidence

$$P(y = 0) + P(y = 1) = 1$$
$$P(y = 0|\boldsymbol{x}) + P(y = 1|\boldsymbol{x}) = 1$$

$$p(\boldsymbol{x}) = p(\boldsymbol{x}|y = 1)P(y = 1) + p(\boldsymbol{x}|y = 0)P(y = 0)$$

Bayes' decision rule:

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|\boldsymbol{x}) > P(y = 0|\boldsymbol{x}) \\ 0 & \text{otherwise.} \end{cases}$$

Maximum A Posteriori criterion

- **MAP decision rule:**
  - pick the hypothesis that is most probable
  - i.e. **maximize the posterior**

$$P(y=c|\boldsymbol{x}) = \frac{P(y=c)p(\boldsymbol{x}|y=c)}{p(\boldsymbol{x})}$$

$$\Lambda_{\mathrm{MAP}}(\boldsymbol{x}) = \frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})}$$

- **Decision rule:** 

Maximum A Posteriori criterion

- **MAP decision rule:**
  - pick the hypothesis that is most probable
  - i.e. **maximize the posterior**

$$P(y=c|\boldsymbol{x}) = \frac{P(y=c)p(\boldsymbol{x}|y=c)}{p(\boldsymbol{x})}$$

$$\Lambda_{\mathrm{MAP}}(\boldsymbol{x}) = \frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})}$$

- **Decision rule:**

  If $\Lambda_{\mathbf{MAP}}(\mathbf{x}) > 1$

  then choose y=1

  else choose y=0.

$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|\boldsymbol{x}) > P(y=0|\boldsymbol{x}) \\ 0 & \text{otherwise.} \end{cases}$$

Likelihood ratio test (LRT)

$$\Lambda_{\mathrm{MAP}}(\boldsymbol{x}) = \frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})} \qquad \Lambda_{\mathrm{MAP}}(\boldsymbol{x}) >^{?} 1 \qquad P(y=c|\boldsymbol{x}) = \frac{P(y=c)p(\boldsymbol{x}|y=c)}{p(\boldsymbol{x})}$$

$$\Lambda_{\mathrm{MAP}}(\boldsymbol{x}) = \frac{P(y=1)p(\boldsymbol{x}|y=1)p(\boldsymbol{x})}{P(y=0)p(\boldsymbol{x}|y=0)p(\boldsymbol{x})}$$

p(**x**) does not affect the decision rule.

- **Likelihood ratio test:**

  test whether the **likelihood ratio** Λ(**x**) is larger than 

$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)}$$

Likelihood ratio test (LRT)

$$\Lambda_{\mathrm{MAP}}(\boldsymbol{x}) = \frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})} \qquad \Lambda_{\mathrm{MAP}}(\boldsymbol{x}) >^{?} 1 \qquad P(y=c|\boldsymbol{x}) = \frac{P(y=c)p(\boldsymbol{x}|y=c)}{p(\boldsymbol{x})}$$

$$\Lambda_{\mathrm{MAP}}(\boldsymbol{x}) = \frac{P(y=1)p(\boldsymbol{x}|y=1)p(\boldsymbol{x})}{P(y=0)p(\boldsymbol{x}|y=0)p(\boldsymbol{x})}$$

p(**x**) does not affect the decision rule.

- **Likelihood ratio test:**

  test whether the **likelihood ratio** Λ(**x**) is larger than $\frac{P(y=0)}{P(y=1)}$

$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)}$$

decision rule:

$$\Lambda(\boldsymbol{x}) >^{?} \frac{P(y=0)}{P(y=1)}$$

Example: LRT decision rule

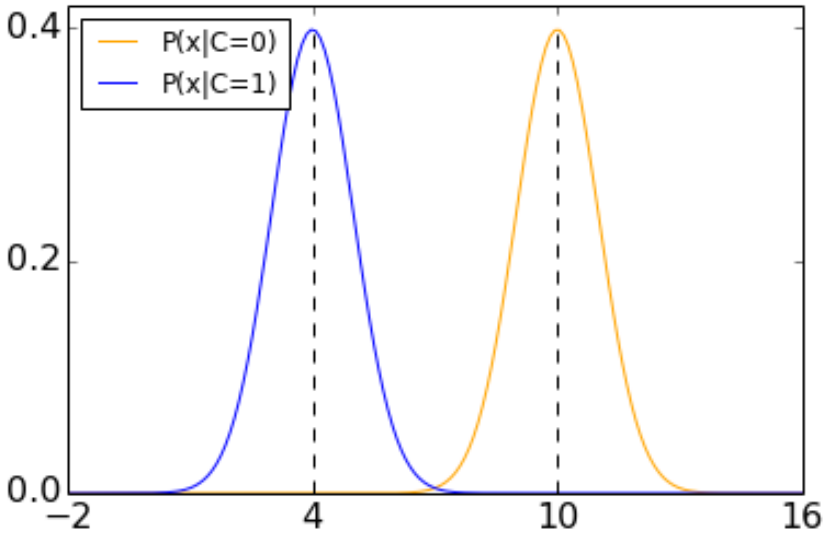
$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)} >^{?} \frac{P(y=0)}{P(y=1)}$$

Assuming the likelihoods below <u>and equal priors,</u> derive a decision rule based on the LRT.

$$p(x|y=1) \sim \mathcal{N}(4,1) \qquad p(x|y=0) \sim \mathcal{N}(10,1)$$

$Z \sim \mathcal{N}(\mu, \sigma^2)$:

$$f(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(z-\mu)^2/(2\sigma^2)}$$



- **Likelihood ratio:**

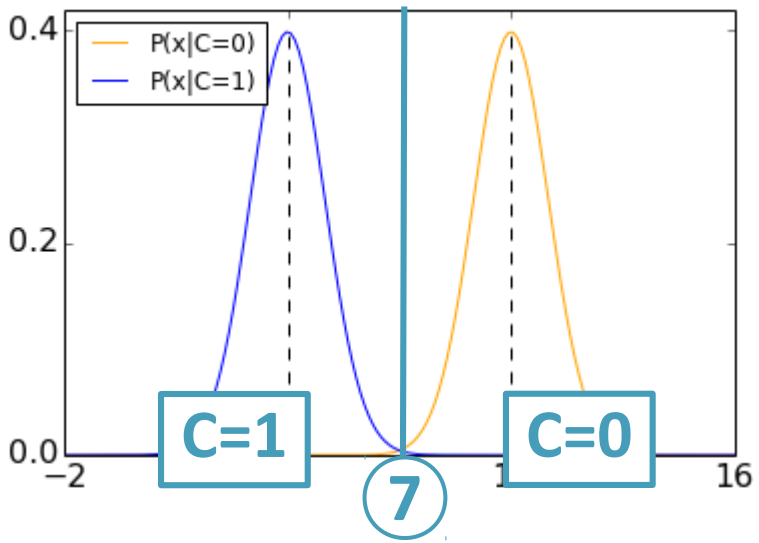
$$\Lambda(x) = \frac{(1/\sqrt{2\pi})e^{-(x-4)^2/2}}{(1/\sqrt{2\pi})e^{-(x-10)^2/2}}$$

- Simplifying the equation and taking the log:

$$\log(\Lambda(x)) = -(x-4)^2 + (x-10)^2$$

- **Equal priors** mean we're testing whether **log(LR) > 0**

  Hence: If x < 7 then assign y=1 else assign y=0



- **Likelihood ratio:**

$$\Lambda(x) = \frac{(1/\sqrt{2\pi})e^{-(x-4)^2/2}}{(1/\sqrt{2\pi})e^{-(x-10)^2/2}}$$

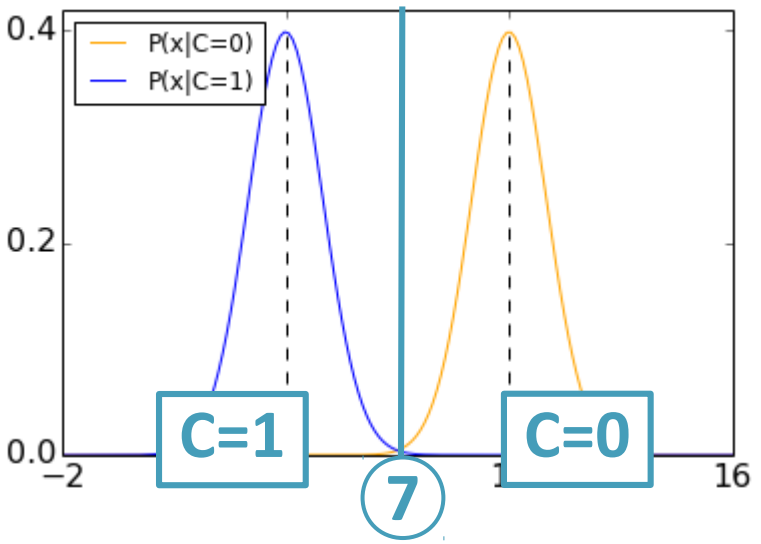
- Simplifying the equation and taking the log:

$$\log(\Lambda(x)) = -(x-4)^2 + (x-10)^2$$

- **Equal priors** mean we're testing whether **log(LR) > 0**

  Hence: If x < 7 then assign y=1 else assign y=0

Now assume P(y=1) = 2 P(y=0)



- **Likelihood ratio:**

$$\Lambda(x) = \frac{(1/\sqrt{2\pi})e^{-(x-4)^2/2}}{(1/\sqrt{2\pi})e^{-(x-10)^2/2}}$$

- Simplifying the equation and taking the log:

$$\log(\Lambda(x)) = -(x-4)^2 + (x-10)^2$$

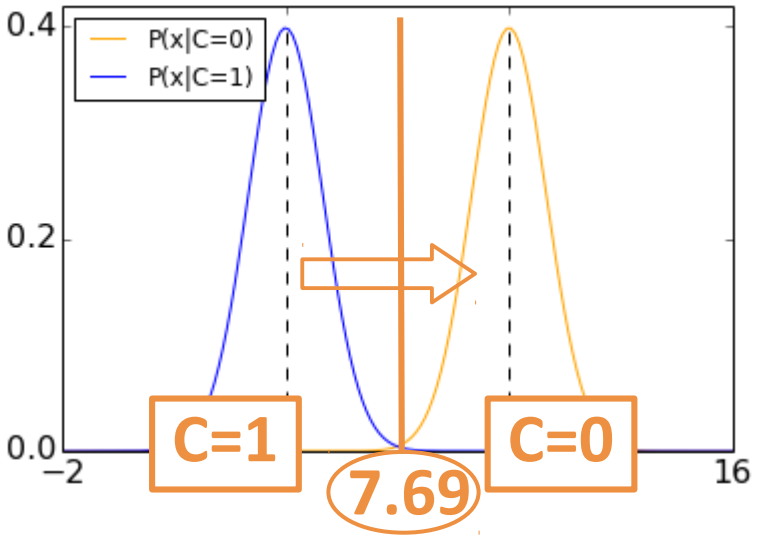
- **Equal priors** mean we're testing whether **log(LR) > 0**

  Hence: If x < 7 then assign y=1 else assign y=0

Now assume P(y=1) = 2 P(y=0)

x < 7 – log(1/2) ≈ 7.69

y=1 is more likely.



Maximum likelihood criterion

- Consider **equal priors** P(y=1) = P(y=0)

$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)} >^{?} \frac{P(y=0)}{P(y=1)} \quad \mathbf{1}$$

- Bayes decision rule seeks to maximize P(x|y=c) and is hence called the **Maximum Likelihood criterion**

$$\Lambda_{\mathrm{ML}}(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)}$$

  - **Decision rule:**

    If $\Lambda_{ML}(x) > 1$ then choose y=1 else choose y=0

Bayes rule for K > 2

- **Bayes rule:**

$$P(y = c_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = c_k)P(y = C_k)}{\sum_{l=1}^{K} p(\boldsymbol{x}|y = c_l)P(y = c_l)}$$

- $P(y = c_k)$  $P(y = c_1) + P(y = c_2) + \cdots + P(y = c_K)$ 
- **What is the decision rule?**

Bayes rule for K > 2

- **Bayes rule:**

$$P(y = c_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = c_k)P(y = C_k)}{\sum_{l=1}^{K} p(\boldsymbol{x}|y = c_l)P(y = c_l)}$$

- $P(y = c_k) \geq 0 \quad P(y = c_1) + P(y = c_2) + \cdots + P(y = c_K) = 1$
- **Decision** 

Bayes rule for K > 2

- **Bayes rule:**

$$P(y = c_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = c_k)P(y = C_k)}{\sum_{l=1}^{K} p(\boldsymbol{x}|y = c_l)P(y = c_l)}$$

- $P(y = c_k) \geq 0 \quad P(y = c_1) + P(y = c_2) + \cdots + P(y = c_K) = 1$
- **Decision**

$$\hat{y} = c_k \text{ if } P(y = c_k|\boldsymbol{x}) = \max_l P(y = c_l|\boldsymbol{x})$$

Risk minimization

Losses and risks

- So far we've assumed all errors were **equally costly.**

  But misclassfying a cancer sufferer as a healthy patient is much more problematic than the other way around.

- **Action α_k:** assigining class c_k
- **Loss:** quantify the cost λ_{kl} of taking action α_k when the true class is c_l
- **Expected risk:**

$$R(\alpha_k|\boldsymbol{x}) = \sum_{l=1}^{K} \lambda_{lk} P(y = c_l|\boldsymbol{x})$$

- **Decision (Bayes Classifier):** $\arg\min_k R(\alpha_k|\boldsymbol{x})$

Discriminant functions

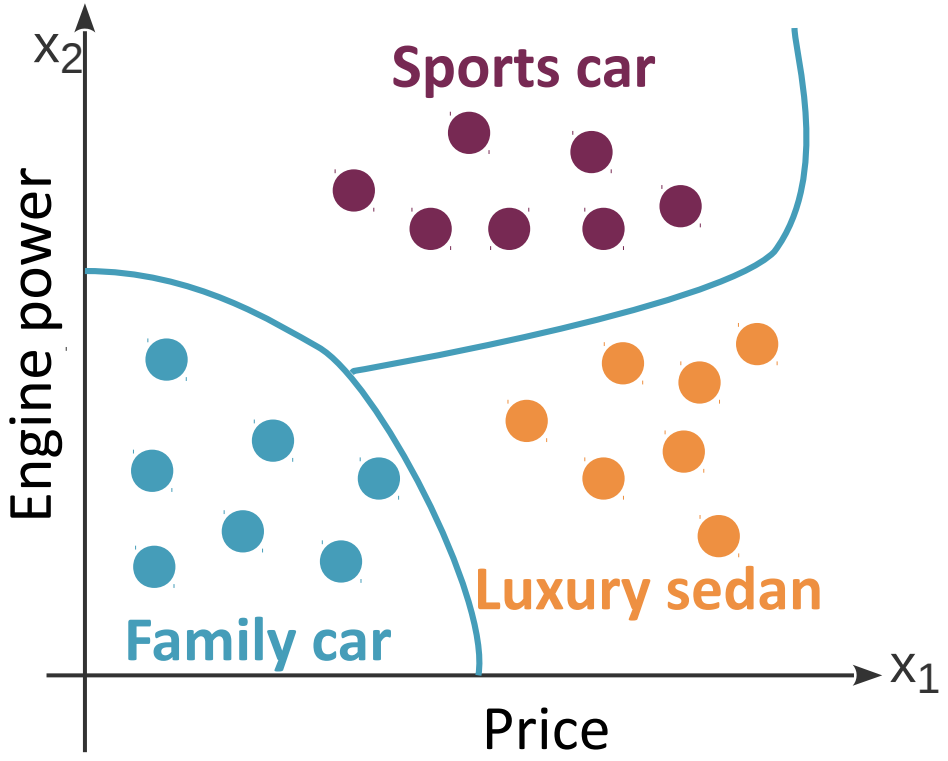
Classification = find K **discriminant functions** f_k s.t. **x** is assigned class c_k if k = argmax f_l(**x**)

- Bayes classifier: $f_k(\boldsymbol{x}) = -R(\alpha_k|\boldsymbol{x})$

Discriminant functions

Classification = find K **discriminant functions** f_k s.t. **x** is assigned class c_k if k = argmax $f_l(\mathbf{x})$

- Bayes classifier: $f_k(\boldsymbol{x}) = -R(\alpha_k|\boldsymbol{x})$
- Defines K **decision regions** $R_k = \{\boldsymbol{x} : f_k(\boldsymbol{x}) = \max_l f_l(\boldsymbol{x})\}$



Bayes risk minimization

- **Bayes risk:** overall expected risk

$$R(\boldsymbol{x}) = \sum_{k=1}^{K} \sum_{l=1}^{K} \lambda_{lk} \; p(\boldsymbol{x} \in R_k | y = c_l) P(y = c_l)$$

- **Bayes decision rule:** use the discriminant functions that **minimize the Bayes risk.**

Bayes risk minimization

- **Bayes risk:** overall expected risk

$$R(\boldsymbol{x}) = \sum_{k=1}^{K} \sum_{l=1}^{K} \lambda_{lk} \; p(\boldsymbol{x} \in R_k | y = c_l) P(y = c_l)$$

- **Bayes decision rule:** use the discriminant functions that **minimize the Bayes risk.**
- This is also a LRT.

  For 2 classes, let us show that Bayes decision rule is equivalent to:

$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)} \overset{?}{>} \frac{(\lambda_{10} - \lambda_{00}) P(y=0)}{(\lambda_{01} - \lambda_{11}) P(y=1)}$$

0/1 Loss

- All misclassifications are **equally costly.**
- $\lambda_{kl} = 0$ if k=l and 1 otherwise

$$\begin{aligned} R(\alpha_k|\boldsymbol{x}) &= \sum_{l=1}^{K} \lambda_{lk} P(y = C_l|\boldsymbol{x}) \\ &= \sum_{l \neq k} P(y = C_l|\boldsymbol{x}) \\ &= 1 - P(y = C_k|\boldsymbol{x}) \end{aligned}$$

- **Minimizing the risk:**
  - choose the most probable class (MAP)
  - this is equivalent to the Bayes decision rule.

$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)} >^{?} \frac{(\lambda_{10} - \lambda_{00})P(y=0)}{(\lambda_{01} - \lambda_{11})P(y=1)}$$

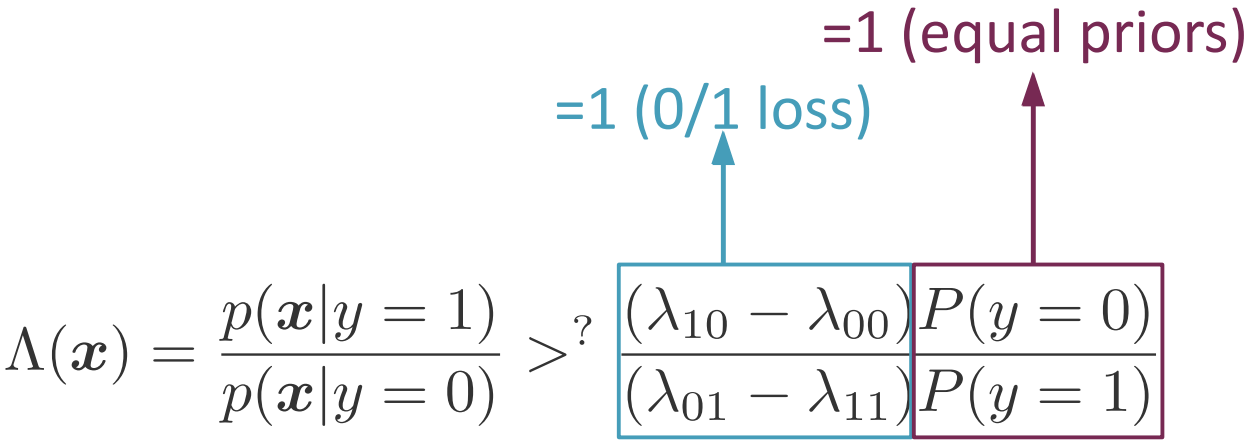
Maximum likelihood criterion

- Consider **equal priors** P(y=1) = P(y=0)
- Consider the **0/1 loss function**

$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)} \overset{?}{>} \frac{(\lambda_{10} - \lambda_{00})}{(\lambda_{01} - \lambda_{11})} \frac{P(y=0)}{P(y=1)}$$

Maximum likelihood criterion

- Consider **equal priors** P(y=1) = P(y=0)
- Consider the **0/1 loss function**



Maximum likelihood criterion

- Consider **equal priors** P(y=1) = P(y=0)
- Consider the **0/1 loss function**
- Bayes decision rule is equivalent to the **Maximum likelihood criterion**

Decision rule:

If $\Lambda_{ML}(x) > 1$ then choose y=1 else choose y=0

$$\Lambda_{\mathrm{ML}}(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)}$$

$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)} >^{?} \frac{(\lambda_{10}-\lambda_{00})}{(\lambda_{01}-\lambda_{11})} \frac{P(y=0)}{P(y=1)}$$

=1 (0/1 loss)

=1 (equal priors)

Reject

- Add an artificial "reject" class (K+1) for **refusing to take a decision.**

  E.g. Zip code detection.

- $\lambda_{kl} = \begin{cases} 0 \text{ if } k = k \\ \lambda \text{ if } k = K+1 \\ 1 \text{ otherwise} \end{cases}$

$$R(\alpha_k|\boldsymbol{x}) = \sum_{l \neq k} P(y = c_l|\boldsymbol{x}) = 1 - P(y = c_k|\boldsymbol{x})$$

$$R(\alpha_{K+1}|\boldsymbol{x}) = \sum_{l=1}^{K} \lambda P(y = c_l|\boldsymbol{x}) = \lambda$$

- **Decision:**

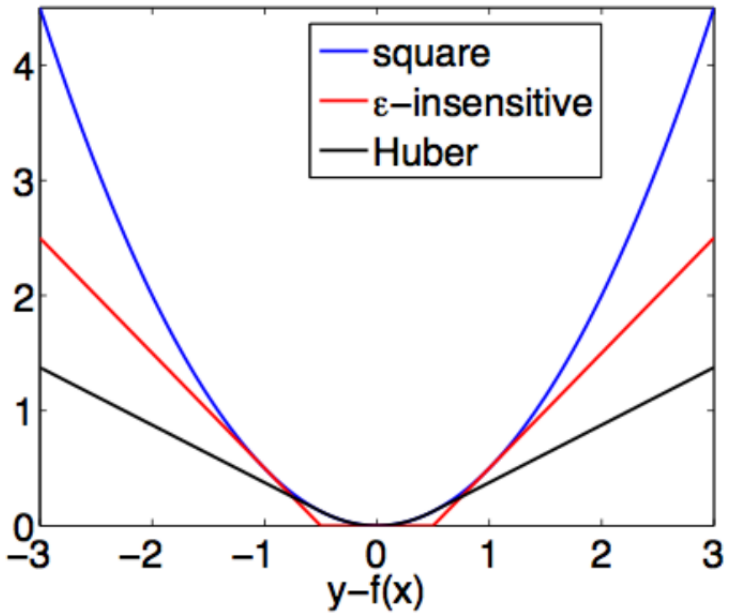
$$\hat{y} = c_k \text{ if } P(y = c_k|\boldsymbol{x}) > P(y = c_l|\boldsymbol{x}) \text{ for all } l \neq k \text{ and } P(y = c_k|\boldsymbol{x}) > 1 - \lambda$$

else reject.

Only meaningful if $0 < \lambda < 1$

Losses for regression

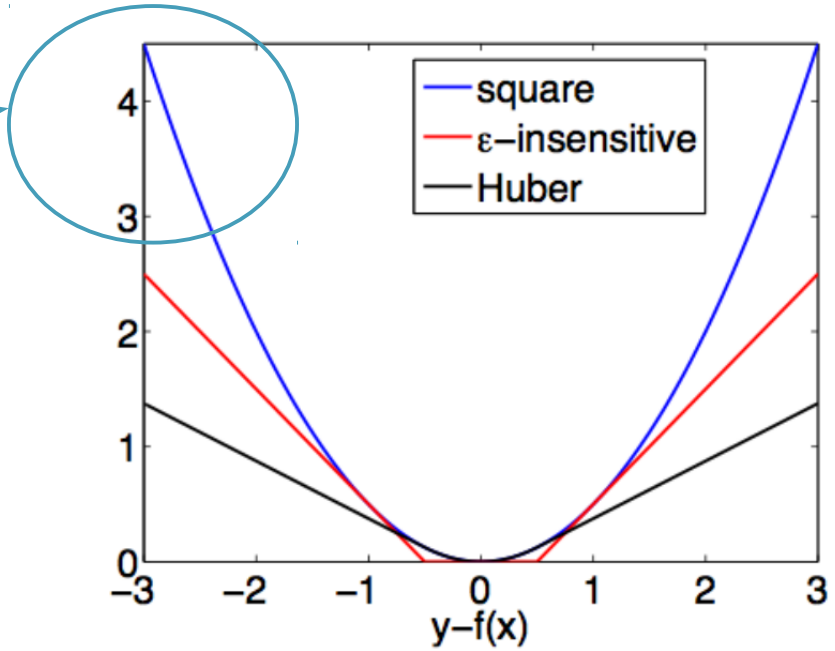
- **Square loss:** $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$



Losses for regression

- **Square loss:** $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$

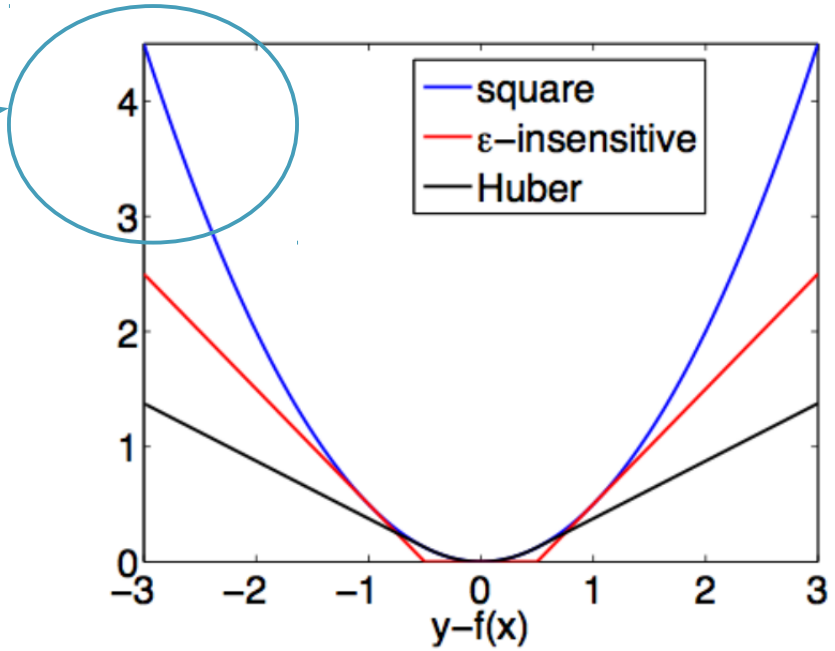
square loss:
dominated by outliers



Losses for regression

- **Square loss:** $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$
- **ε-insensitive loss:** $L(f(\mathbf{x}), y) = (|f(\mathbf{x}) - y| - \varepsilon)_+$

square loss:
dominated by outliers

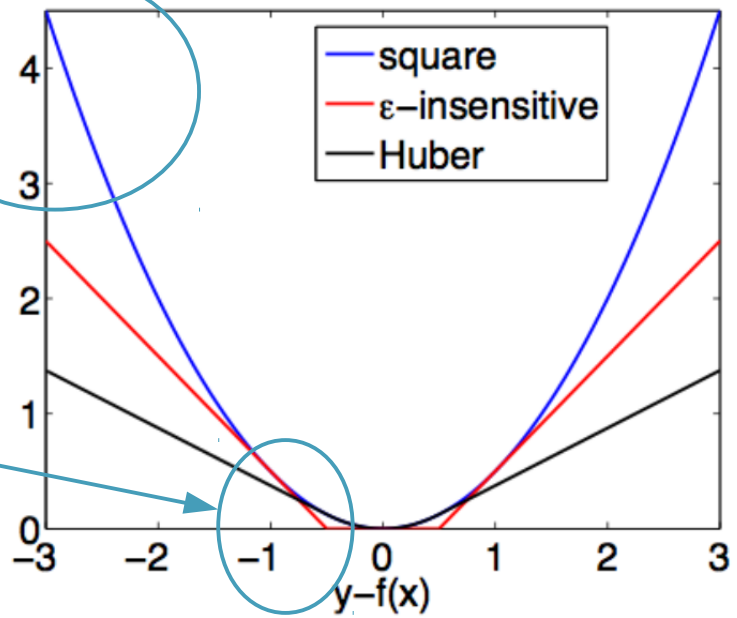


Losses for regression

- **Square loss:** $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$
- **ε-insensitive loss:** $L(f(\mathbf{x}), y) = (|f(\mathbf{x}) - y| - \varepsilon)_+$

square loss:
dominated by outliers

ε-insensitive loss:
non smooth



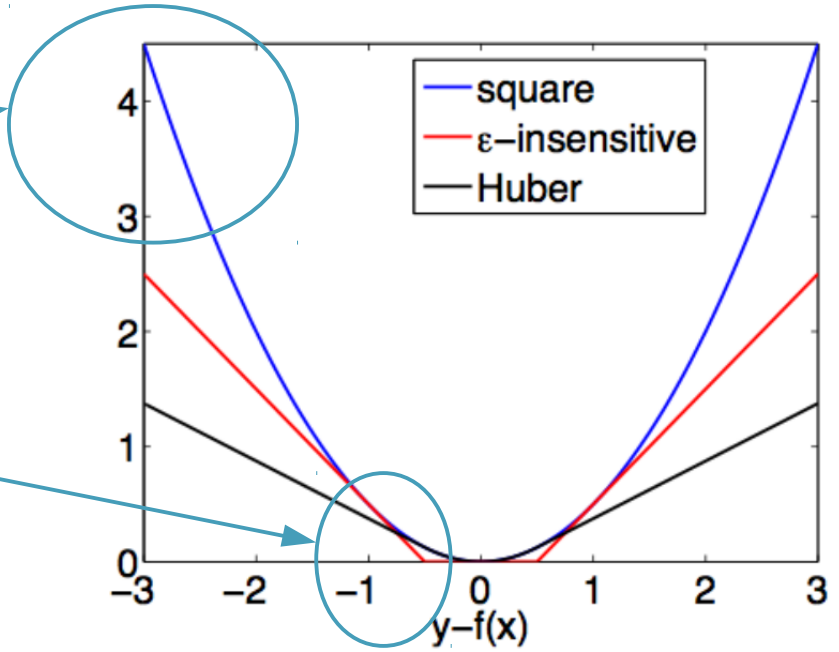
Losses for regression

- **Square loss:** $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$
- **ε-insensitive loss:** $L(f(\mathbf{x}), y) = (|f(\mathbf{x}) - y| - \varepsilon)_+$
- **Huber loss:** mix of linear and quadratic

$$L_\delta(f(\boldsymbol{x}), y) = \begin{cases} \frac{1}{2}\left(y - f(\boldsymbol{x})\right)^2 & \text{if } |y - f(\boldsymbol{x})| \leq \delta \\ \delta|y - f(\boldsymbol{x})| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

square loss: dominated by outliers

ε-insensitive loss: non smooth



Empirical risk minimization (ERM)

- **Loss:** L(f(**x**), y) small when f(**x**) predicts y well
- **Expected risk:**

$$R = \mathbb{E}[L(f(\boldsymbol{x}), y)]$$

- **Empirical risk:**

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}^i), y^i)$$

- The **ERM estimator** of the functional class F is the solution, when it exists, of:

$$\hat{f}_n = \arg\min_{f \in \mathcal{F}} R_n(f)$$

Solving ERM

- There can sometimes be an **explicit analytical solution**
- Otherwise: **convex optimization** (if the loss function is convex in f)
- **Limits of ERM:**
  - **ill-posed**
  - **not statistically consistent**

    This is particularly true in **high dimension.**

ERM is ill-posed

- **Well-posed problems** (Hadamard):

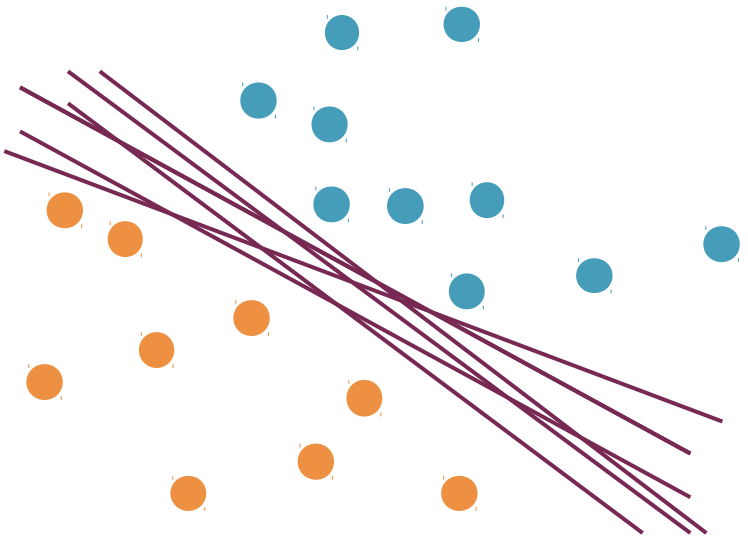
  Mathematical models of physical phenomena such that

  – a solution exists;

  – the solution is unique;

  – the solution's behavior changes continuously with the initial conditions.

- It can be that **an infinite number of solutions minimize the empirical risk** to zero.



ERM is not statistically consistent

- **Statistical consistency:** Estimator θ_N of θ that converges in probability towards θ as N increases.

$$\forall \epsilon > 0 \quad \lim_{N \to \infty} Pr(|\theta_N - \theta| \geq \epsilon) = 0$$

- From the **law of large numbers**,

$$\forall f \in \mathcal{F}, \quad R_N(f) \xrightarrow[N \to \infty]{} R(f)$$

  but this isn't enough to guarantee that minimizing $R_N(f)$ gives a good estimator of the minimizer of R(f).

- Vapnik showed that this is only true if the capacity of hypothesis space $\mathcal{F}$ is "not too large".

Multivariate classification: Naive Bayes

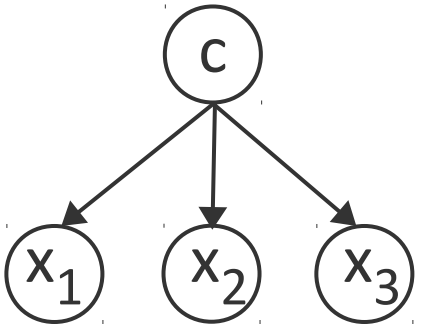
Naive Bayes

- Multivariate classification: **x** is multidimensional
- Assume the variables x_1, x_2, … x_p are **conditionally independent**: $p(x_{j_1}|y = c, x_{j_2}) = p(x_{j_1}|y = c)$

Graphical representation

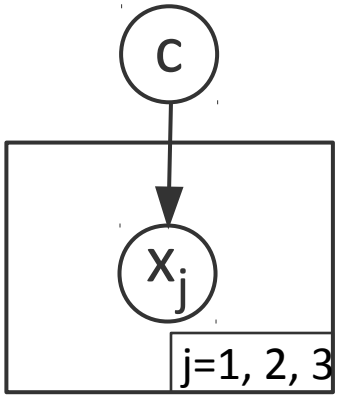
- We can use a graph to represent **conditional independence**:
  - arc from c to x_j means the distribution of X_j **depends** on c
  - no arc between X_{j1} and X_{j2} means that X_{j1} and X_{j2} are **independent given C**:

$$p(x_{j_1}|y=c, x_{j_2}) = p(x_{j_1}|y=c)$$



- A **plate** represents repeated structure:

  all X_j inside the same plate follow the same probability distribution.



Naive Bayes

- Multivariate classification: x is multidimensional
- Assume the variables x_1, x_2, … x_p are **conditionally independent**: $p(x_{j_1}|y = c, x_{j_2}) = p(x_{j_1}|y = c)$

$$P(y = c_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = c_k)P(y = c_k)}{\sum_{l=1}^{K} p(\boldsymbol{x}|y = c_l)P(y = c_l)}$$

$$p(x_1, \ldots, x_p|y = c) = p(x_1|y = c)p(x_2|y = c) \ldots p(x_p|y = c)$$

Hence:

$$P(y = c|x_1, \ldots, x_p) = \frac{1}{Z} P(y = c)p(x_1|y = c)p(x_2|y = c) \ldots p(x_p|y = c)$$

scaling factor, independent of c_k

Maximum a posteriori estimation

- **MAP decision rule:** pick the hypothesis that is most probable
- For Naive Bayes:

$$\hat{y} = \arg \max_{k=1,\ldots,K} p(y = c_k) \prod_{i=1}^{n} p(\boldsymbol{x}^i | y = c_k)$$

Naive Bayes spam filtering

- Input: email

  bag of words

  $(x_1, x_2, \ldots x_p) = (0, 1, \ldots, 0)$

  rich
  CLICK
  viagra

- Output: spam / ham
- Naive Bayes assumption: conditional independence

$$P(y = c_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = c_k)P(y = c_k)}{\sum_{l=1}^{K} p(\boldsymbol{x}|y = c_l)P(y = c_l)}$$

SPAM

Your Mail-Box has exceeded its storage Limit CLICK=HERE FILL and Click on FINISH for to get more space or you wont be able to send Mail

SPAM

Dear Dr Azencott,

We obtained your contact information from your excellent papers, and would like to know if our company could serve you. Does your current work require the generation of custom monoclonal antibodies? If so, we would be glad to perform this tedious and time-consuming task on your behalf.

NOT SPAM

Dear Dr Azencotte,

Thank you very much for your review of manuscript CHIN-D-15-00031

We greatly appreciate your assistance.

Best wishes,

Samuel Winthrop
Journal of Cheminformatics

- **$P(spam|(x_1, x_2, \ldots x_p))$**

  $= 1/Z\ p(spam)\ p(x_1|spam)\ p(x_2|spam) \ldots p(x_p|spam)$

- **$P(ham|(x_1, x_2, \ldots x_p))$**

  $= 1/Z\ p(ham)\ p(x_1|ham)\ p(x_2|ham) \ldots p(x_p|ham)$

- **Decision:**

  If $P(spam|(x_1, x_2, \ldots, x_p)) > P(ham|(x_1, x_2, \ldots, x_p))$ then spam else ham

- **Inference:** we need to determine

  $p(spam)$, $p(ham)$, $p(x_j|spam)$, $p(x_j|ham)$

- **P(spam|(x_1, x_2, … x_p))**

  = 1/Z p(spam) p(x_1|spam) p(x_2|spam) … p(x_p|spam)

- **P(ham|(x_1, x_2, … x_p))**

  = 1/Z p(ham) p(x_1|ham) p(x_2|ham) … p(x_p|ham)

- **Decision:**

  If P(spam|(x_1, x_2, …, x_p)) > P(ham|(x_1, x_2, …, x_p)) then spam else ham

- **Inference:** we need to determine

  p(spam), p(ham), p(x_j|spam), p(x_j|ham)

frequency of spam in the training data

- **P(spam|(x_1, x_2, … x_p))**

  = 1/Z p(spam) p(x_1|spam) p(x_2|spam) … p(x_p|spam)

- **P(ham|(x_1, x_2, … x_p))**

  = 1/Z p(ham) p(x_1|ham) p(x_2|ham) … p(x_p|ham)

- **Decision:**

  If P(spam|(x_1, x_2, …, x_p)) > P(ham|(x_1, x_2, …, x_p)) then spam else ham

- **Inference:** we need to determine

  p(spam), p(ham), p(x_j|spam), p(x_j|ham)

frequency of spam in the training data

- **Bernouilli Naive Bayes:**
  - Each email is the outcome of p Bernouilli trials
  - **Naive assumption:** the trials are independent

    word co-occurences in a category aren't independent

    still, independence assumptions can give good results

$$p(x_j|\text{spam}) = {p_j}^{x_j}(1-p_j)^{(1-x_j)}$$

  - **Direct estimate of p_j:** $p_j = S_j / S$
    - S = # spams in train set
    - S_j = # spams containing word j in train set
  - **What happens if a word is never seen?**

- **Bernouilli Naive Bayes:**
  - Each email is the outcome of p Bernouilli trials
  - **Naive assumption:** the trials are independent

    word co-occurences in a category aren't independent

    still, independence assumptions can give good results

$$p(x_j|\text{spam}) = {p_j}^{x_j}(1-p_j)^{(1-x_j)}$$

- S = # spams in train set
- S_j = # spams containing word j in train set

  - **Direct estimate of p_j:** $p_j = S_j / S$
  - **Laplace-smoothed estimate of p_j:** $p_j = (S_j + 1) / (S + 2)$

For a word that's not in the training set now p_j=0.5 instead of 0

- $P(spam|(x_1, x_2, \ldots x_p))$

  $= 1/Z\ p(spam)\ p(x_1|spam)\ p(x_2|spam) \ldots p(x_p|spam)$

- $P(ham|(x_1, x_2, \ldots x_p))$

  $= 1/Z\ p(ham)\ p(x_1|ham)\ p(x_2|ham) \ldots p(x_p|ham)$

- **Decision:**

  If $P(spam|(x_1, x_2, \ldots, x_p)) > P(ham|(x_1, x_2, \ldots, x_p))$ then spam else ham

- **Inference:** we need to determine

  $p(spam)$, $p(ham)$, $p(x_j|spam)$, $p(x_j|ham)$

frequency of spam in the training data

Bernouilli Naive Bayes: $p_j^{x_j}(1-p_j)^{(1-x_j)}$

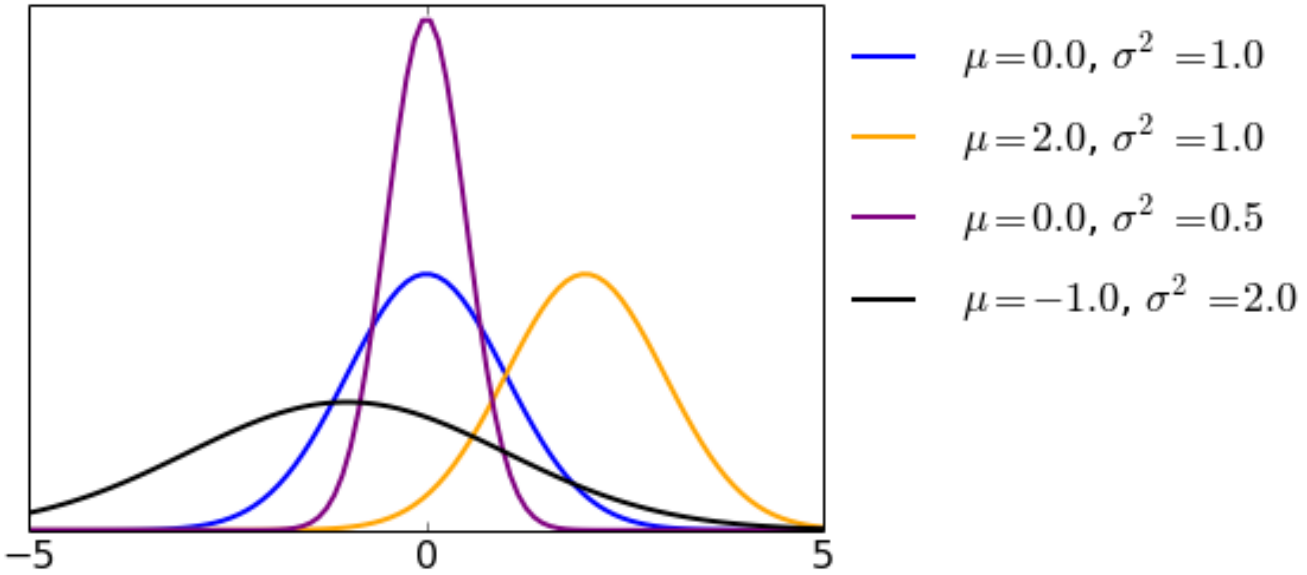
$p_j = (S_j + 1) / (S + 2)$
S = # spams in train set
S_j = # spams with word j in train set

Gaussian naive Bayes

- Assume

  $p(x_j|y=c_k)$ **univariate Gaussian**

$$p(x_j|y=c_k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x_j-\mu)^2/(2\sigma^2)}$$



Bayesian model selection

- **Priors on model:** p(model)

$$p(\text{model}|\text{data}) = \frac{p(\text{data}|\text{model})p(\text{model})}{p(\text{data})}$$

- Regularization ≡ prior that favors simpler models.
- Take the log

$$\log p(\text{model}|\text{data}) = \log p(\text{data}|\text{model}) + \log p(\text{model}) - c$$

≡ training error

≡ model complexity

- MAP similar to minimizing

E' = empirical error + **λ** model complexity

Summary

$$P(y = c|\boldsymbol{x}) = \frac{P(y = c)p(\boldsymbol{x}|y = c)}{p(\boldsymbol{x})}$$

prior, likelihood, posterior, evidence

- **Bayes decision rule ≡ likelihood ratio test**

  choose the most probable class, given evidence (data) and prior belief.

- Equivalent to **minimizing Bayes risk**

  usually achieved approximately through **empirical risk minimization** (not equivalent!!)

- For the 0/1 loss, equivalent to **maximizing the posterior.**

- For the 0/1 loss and equal priors (uniform prior), equivalent to **maximizing the likelihood.**

References

- *A Course in Machine Learning.*
  http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf
  - **Bayes classifier:** Chap 2.1
  - **LRT:** Chap 9.4
  - **Naive Bayes:** Chap 9.3
- *The Elements of Statistical Learning.*
  http://web.stanford.edu/~hastie/ElemStatLearn/
  - **Bayes classifier:** Chap 2.4
  - **Maximum Likelihood:** Chap 2.6.3, Chap 8.3
- **Probabilistic machine learning**
  https://www.repository.cam.ac.uk/bitstream/handle/1810/248538/Ghahramani%202015%20Nature
- **Spam detection:** http://www.paulgraham.com/spam.html
- **Naive Bayes:**
  https://nlp.stanford.edu/IR-book/pdf/13bayes.pdf

kaggle challenge project

How Many Shares? Challenge

`https://www.kaggle.com/c/how-many-shares`

- **Predict the number of shares on social media** for articles from the same media site
  - From article length, topics, subjectivity and much more.
  - What kind of machine learning task is this?
- **Evaluation on**
  - Insights learned
  - Prediction performance.

kaggle challenge project

- Form **teams of 2-5 students**
  - Engineer features (see Lab 4)
  - **Model selection** for several approaches
  - Predict with selected models and submit to leaderboard
  - Choose **2 final models**
- Deadline: **December 23, 2017 23:59**
  - **Report** (2 pages + figures/tables) 25 pts
  - **Leaderboard position** 5 pts
- Get started early!
- Full instructions on the course website

Kaggle leaderboard setup

- The data is divided into:
  - Training data
  - Public validation data
  - Test validation data
- You only have the labels of the training data
- You make predictions for the **whole validation set**
- The **public part** is used to rank you on the **public leaderboard** throughout the challenge
- The **private part** is used to determine your **final ranking** at the end.

Grading rubric

- **Discussion** of **feature engineering** 4pts
- **Discussion** of **cross-validated performance** 8pts
- **Discussion** of **leaderboard performance** 4pts

  (of selected models — max 5/day)
- **Discussion** of **final model** 4pts
- **Clarity** of report 5pts

  (text, tables, figures)
- Final **performance** 5pts

Lab 3: make_Kfolds

```
    # Create a list of the n_instance indices [0, 1, ..., n_instance-1]
    list_indices = np.arange(n_instance)
    # Shuffle the list with np.random.shuffle
    np.random.shuffle(list_indices)

    # Compute the number of instances per bin (i.e. in each test set)
    n_instance_per_fold = int(round(float(n_instance)/float(n_folds)))
    print(n_instance_per_fold)

    # For each of the first K-1 folds, create the list of train set and test set indices
    fold_list = []
    for ind_fold in range(n_folds-1):
        test_list = list_indices[ind_fold*n_instance_per_fold:(ind_fold+1)*n_instance_per_fold]
        train_list = [index for index in list_indices if index not in test_list]
        # TODO add the (train_list, test_list) tuple to fold_list
        fold_list.append((train_list, test_list))

    # Process the last fold separately
    test_list = list_indices[(ind_fold+1)*n_instance_per_fold:]
    train_list = [index for index in list_indices if index not in test_list]
    fold_list.append((train_list, test_list))

    return fold_list
```

```
for ix, (tr, te) in enumerate(perso_folds):
    print("Fold %d" % ix)
    print("\t %d training points" % len(tr))
    print("\t %d test points" % len(te))
    if len(np.intersect1d(tr, te))>0:
        print('some instances are both in your training and test sets')
```

```
200
Fold 0
         801 training points
         200 test points
Fold 1
         801 training points
         200 test points
Fold 2
         801 training points
         200 test points
Fold 3
         801 training points
         200 test points
Fold 4
         800 training points
         201 test points
```

- Each index (or instance) should appear once and only once in any test data.
- Each test fold containts n/K points; the last one might contain a few more or less if n is not a multiple of K.

cross_validate

```
pred = np.zeros(labels.shape)
for tr, te in cv_folds:
    classifier.fit(design_matrix[tr,:], labels[tr])
    pred[te] = (classifier.predict_proba(design_matrix[te,:]))[:,list(classifier.classes_).index(1)]
return pred
```

- predict_proba returns **two** arrays of predictions:
  - one contains the probability, for each point, to belong to class A
  - the other the probability, for each point, to belong to class B.
- To determine which of class A and class B is the positive one, you can use classifier.classes_ which contains [class A, class B].
- Note this extends to more than 2 classes

Gaussian Naive Bayes