- **Supervised ML**
  - Generative $\times$ Discriminative
  - Classification $\times$ Regression
  - Perceptron
  - Linear regression
  - Logistic regression
  - ML diagram (X, Y, unknown target f, dataset D, Hypothesis space, learning algorithm, final hypothesis h)
  - E_{in}, E_{out}, performance metrics
  - VC theory (Hoeffding inequality, dichotomies, growth function, VC dimension)
  - Bias-variance model
  - Overfitting
  - Regularization
  - Validation / Model Selection
  - Neural networks
  - SVM

Unsupervised ML

- Clustering
- Dimensionality reduction (PCA, SOM, auto-encoders)
- Visualization (projections)

Other types of learning

- Deep learning
- Reinforcement learning
- Active learning
- Self-learning
- Semi-supervised learning
- Graphical models (generative approach)

Revisiting

Generative $\times$ **Discriminative**

Bayesian classifier – c classes

$\mathbf{x} = (x_1, x_2, \ldots, x_d)$
$P(y_j) = P(y = j)$, $j = 1, 2, \ldots, c$

$$P(y_j \,|\, \mathbf{x}) = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{\sum_{i=1}^{c} P(y_i)\, P(\mathbf{x} \,|\, y_i)} = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{P(\mathbf{x})}$$

Bayes classification rule (Maximum a posteriori)

Choose $\quad j^* = \arg\max_j \{P(y_j|\mathbf{x})\}$

Bayesian classifier – c classes

$\mathbf{x} = (x_1, x_2, \ldots, x_d)$
$P(y_j) = P(y = j)$, $j = 1, 2, \ldots, c$

prior

$$P(y_j \,|\, \mathbf{x}) = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{\sum_{i=1}^{c} P(y_i)\, P(\mathbf{x} \,|\, y_i)} = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{P(\mathbf{x})}$$

Bayes classification rule (Maximum a posteriori)

Choose $\quad j^* = \arg\max_j \{P(y_j|\mathbf{x})\}$

Bayesian classifier – c classes

$\mathbf{x} = (x_1, x_2, \ldots, x_d)$
$P(y_j) = P(y = j)$, $j = 1, 2, \ldots, c$

$$P(y_j \,|\, \mathbf{x}) = \frac{\overbrace{P(y_j)\, P(\mathbf{x} \,|\, y_j)}^{\text{likelihood}}}{\sum_{i=1}^{c} P(y_i)\, P(\mathbf{x} \,|\, y_i)} = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{P(\mathbf{x})}$$

Bayes classification rule (Maximum a posteriori)

Choose $\quad j^* = \arg\max_j \{P(y_j|\mathbf{x})\}$

Bayesian classifier – c classes

$\mathbf{x} = (x_1, x_2, \ldots, x_d)$
$P(y_j) = P(y = j)$, $j = 1, 2, \ldots, c$

$$P(y_j \,|\, \mathbf{x}) = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{\sum_{i=1}^{c} P(y_i)\, P(\mathbf{x} \,|\, y_i)} \qquad \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{P(\mathbf{x})}$$

evidence $p(\mathbf{x})$

Bayes classification rule (Maximum a posteriori)

Choose $\quad j^* = \arg\max_j \{P(y_j|\mathbf{x})\}$

Bayesian classifier – c classes

$\mathbf{x} = (x_1, x_2, \ldots, x_d)$
$P(y_j) = P(y = j)$, $j = 1, 2, \ldots, c$

$$\underbrace{P(y_j \,|\, \mathbf{x})}_{\text{posterior}} = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{\sum_{i=1}^{c} P(y_i)\, P(\mathbf{x} \,|\, y_i)} = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{P(\mathbf{x})}$$

Bayes classification rule (Maximum a posteriori)

Choose $\quad j^* = \arg\max_j \{P(y_j|\mathbf{x})\}$

Bayesian classifier – c classes

$\mathbf{x} = (x_1, x_2, \ldots, x_d)$
$P(y_j) = P(y = j)$, $j = 1, 2, \ldots, c$

$$P(y_j \,|\, \mathbf{x}) = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{\sum_{i=1}^{c} P(y_i)\, P(\mathbf{x} \,|\, y_i)} = \frac{P(y_j)\, P(\mathbf{x} \,|\, y_j)}{P(\mathbf{x})}$$

$$posterior = \frac{prior \times likelihood}{evidence}$$

Bayes classification rule (Maximum a posteriori)

Choose $\quad j^* = \arg\max_j \{P(y_j|\mathbf{x})\}$

Bayes Theorem

$$P(y_j \mid \mathbf{x}) = \frac{P(y_j)\, P(\mathbf{x} \mid y_j)}{P(\mathbf{x})}$$

Evidence $\mathbf{x}$ ($\mathbf{x}$ has been observed):

Optimal choice $\quad j^* = \arg\max_j \{P(y_j|\mathbf{x})\}$

Denominator is not important: $\quad j^* = \arg\max_j \{P(y_j)\, P(\mathbf{x}|y_j)\}$

If $P(y)$ is uniform (equiprobable classes):

Optimal choice $\quad j^* = argmax\{P(\mathbf{x}|y_j)\}$

Choice depends only on *likelihood* $P(\mathbf{x}|y_j)$

Example

Binary classification: classes y_1 and y_2

Priori:

Estimating $P(y_1)$ and $P(y_2)$ is relatively simple (relative freq.)

Likelihood: pdfs $P(\mathbf{x}|y_1)$ and $P(\mathbf{x}|y_2)$

Suppose both are univariate Gaussian (mean μ and variance σ^2):

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}}, \quad x \in \mathbb{R}$$

How to estimate the parameters?

Let the samples be:

$$x^{(1)}, x^{(2)}, \ldots, x^{(N)}$$

The likelihood function:

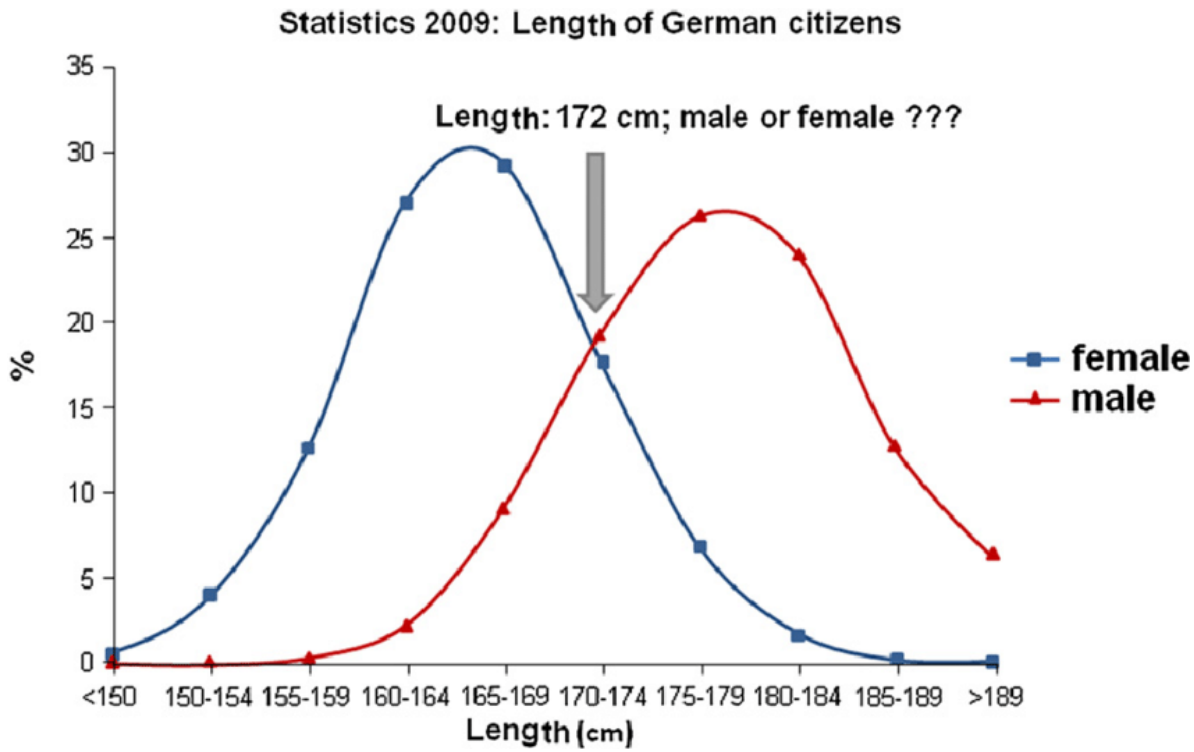
$$L(\mu, \sigma) = \sum_{i=1}^{N} \ln P(x^{(i)}|\mu, \sigma) = \sum_{i=1}^{N} \ln \frac{1}{\sqrt{2\pi}\sigma} e^{\left\{-\frac{1}{2}\left(\frac{x^{(i)}-\mu}{\sigma}\right)^2\right\}}$$

Computing the zeros of its derivative:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} \qquad \hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \hat{\mu}\right)^2$$

x: height

$y = \{\text{female}, \text{male}\}$



Multivariate Gaussian pdf: $N(\mu, \Sigma)$, μ is the mean vector and Σ is the covariance matrix

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{x}-\mu)^t \Sigma^{-1}(\mathbf{x}-\mu)\right\}, \quad \mathbf{x} \in \mathbb{R}^d$$

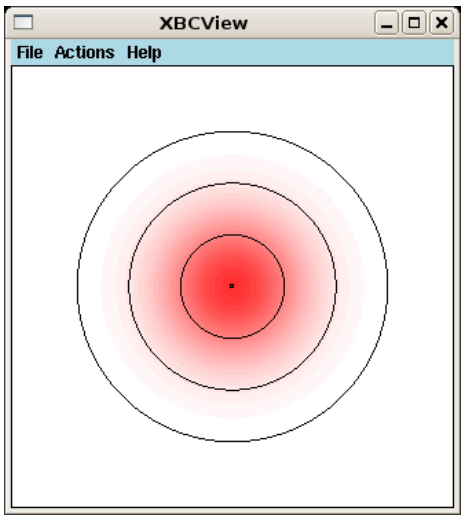
Covariance matrix

Diagonals: $\sigma_{ii} = E[(x_i - \mu_i)^2]$ (variance of component x_i)

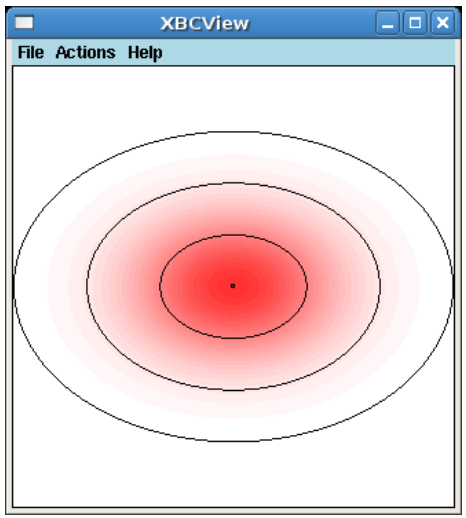
Others: $\sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)]$, $i \neq j$

This matrix is symmetric and positive.

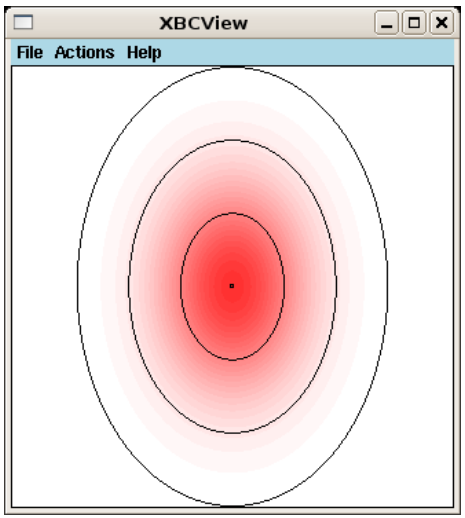
$$\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$



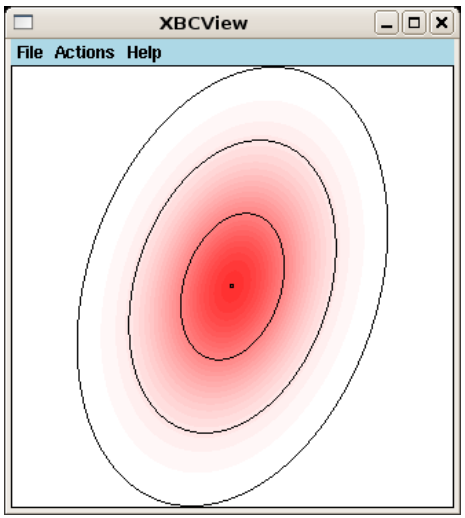
$$\Sigma = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}$$



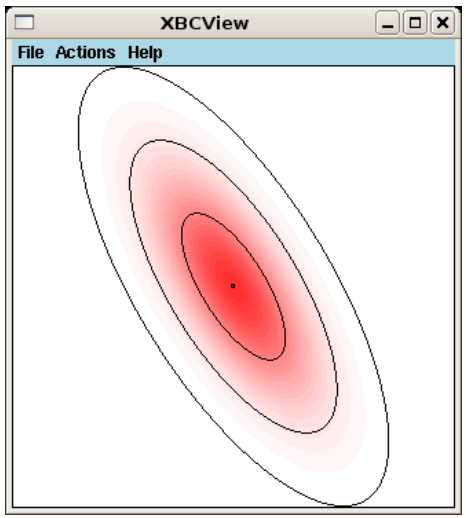
$$\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$$



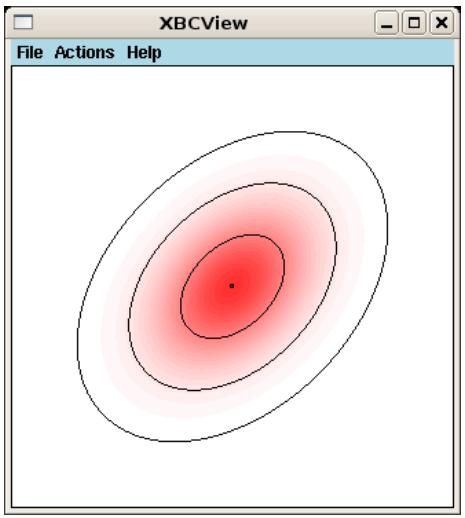
$$\Sigma = \begin{bmatrix} 2 & .75 \\ .75 & 2 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 2 & .75 \\ .75 & 4 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$$



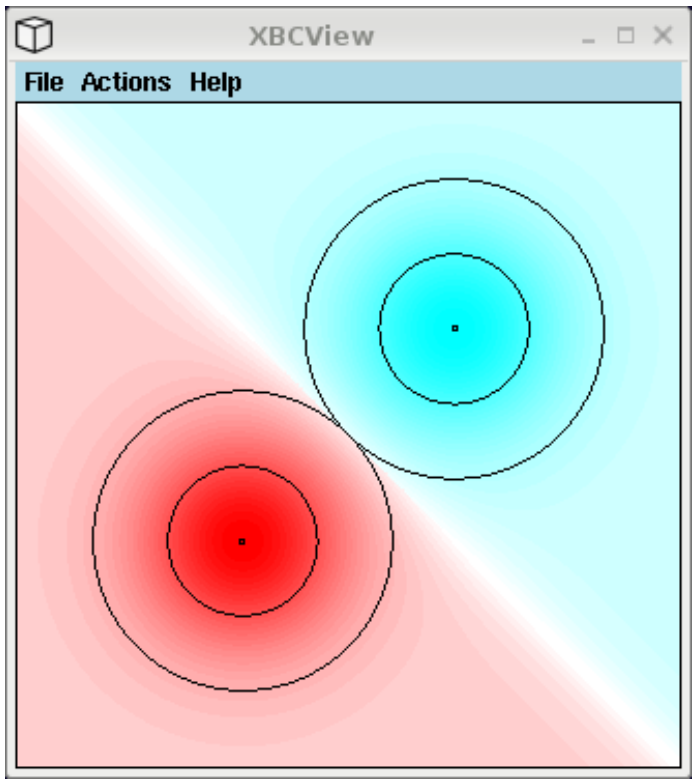
Parameter estimation for multivariate Gaussian pdf

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^{(i)}$$

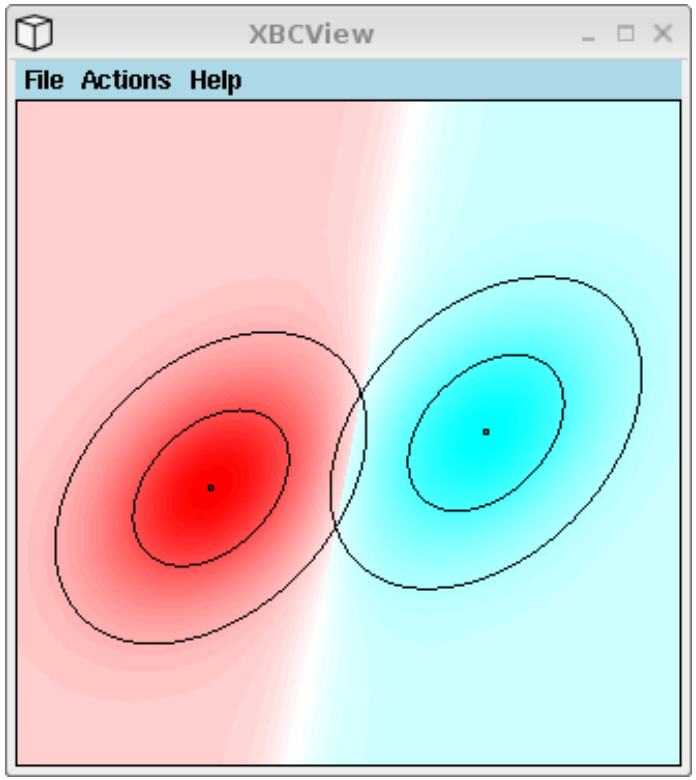
$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}^{(i)} - \hat{\mu})(\mathbf{x}^{(i)} - \hat{\mu})^t$$

Decision surfaces

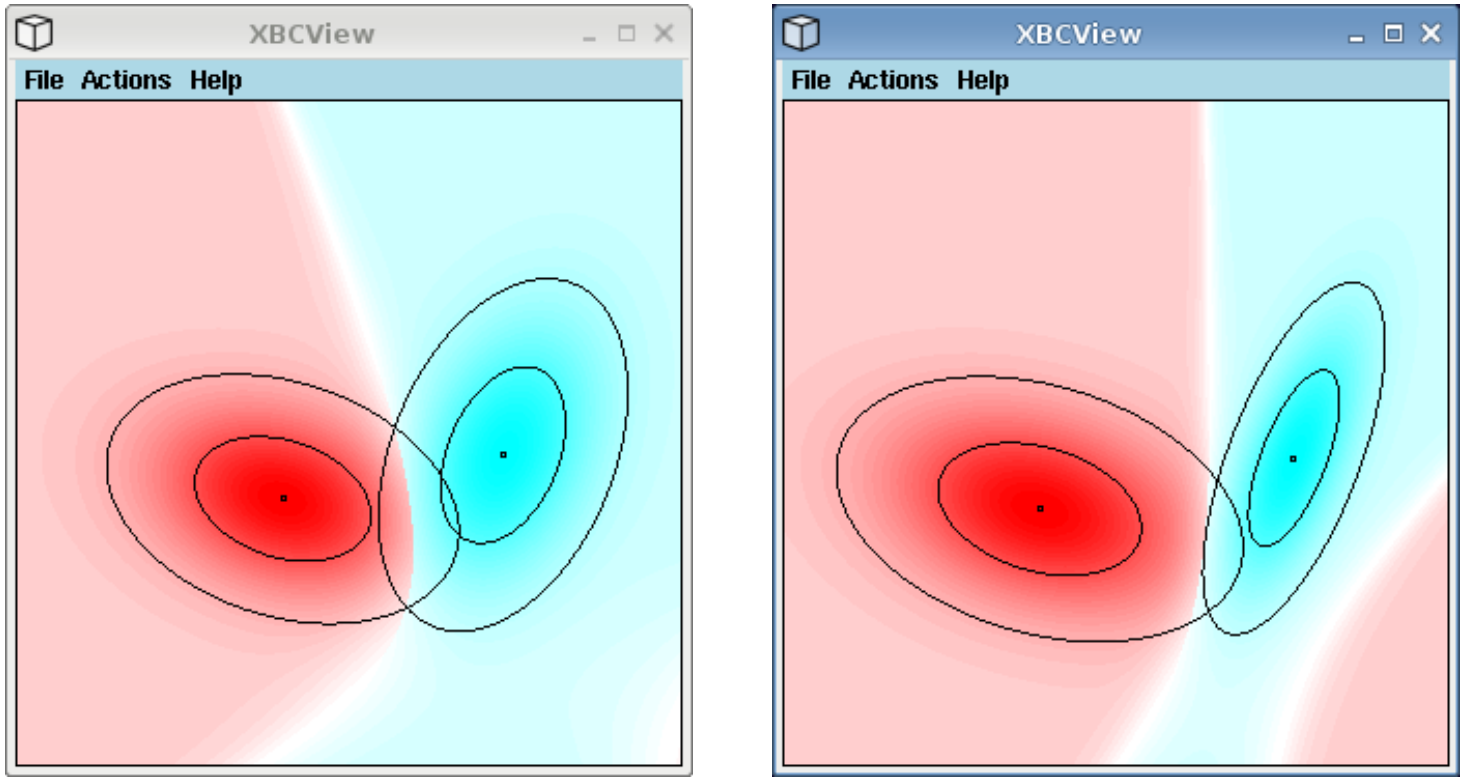
$$j^* = argmax\{P(\mathbf{x}|y_j)\}$$



$\Sigma_j = \sigma^2 I$: Two classes with a same distribution



$\Sigma_j = \Sigma$: Two classes with a same distribution



Arbitrary Σ_j: Two classes with disctinct distributions

$\Sigma_j = \Sigma \Longrightarrow$ linear decision surface

distinct $\Sigma_j \Longrightarrow$ more complex decision surface

Other methods for pdf estimation:

- histogram
- kernel density estimation (Parzen windows)
- mixture of Gaussians
- Graphical models
- ...

Naïve Bayes Classifier

Bayes classifier requires computation of

$$P(y_j|\mathbf{x}) \sim P(y_j)P(\mathbf{x}|y_j), \quad \mathbf{x} = (x_1, x_2, \ldots, x_d)$$

$p(\mathbf{x}|y_j)$ is a multivariate distribution !!

Naïve Bayes Classifier: Assumes variables x_i are independent

$$\begin{aligned} p(\mathbf{x} \mid y_j) &= P(x_1, x_2, \ldots, x_d \mid y_j) \quad \text{(multivariate)} \\ &= P(x_1 \mid y_j)P(x_2 \mid y_j) \ldots P(x_d \mid y_j) \\ &= \prod_{i=1}^{d} P(x_i \mid y_j) \quad \text{(univariate)} \end{aligned}$$

Example of naïve Bayes classifier application: spam detection

This is a two-class problem: $y = +1$ (spam) and $y = -1$ (non-spam)

Define a vocabulary set: $V = \{v_1, v_2, \ldots, v_K\}$

Given training email texts, estimate $P(y = +1)$, $P(y = -1)$, as well as $P(v_i|y = +1)$ and $P(v_i|y = -1)$

To avoid $P(v_i \,|\, y) = 0$ we use $P(v_i \,|\, y) = \frac{\#v_i + \alpha}{\#v + \alpha K}$, where $\#v_i$ and $\#v$ are, respectively, the number of occurrences of word v_i and words $v \in V$ in documents of class y

Now, given a document $\mathbf{D}$, we can compute:

$$P(y = +1|\mathbf{D}) = P(y = +1) \prod_{v_i \in \mathbf{D}} P(v_i \,|\, y = +1)$$
$$P(y = -1|\mathbf{D}) = P(y = -1) \prod_{v_i \in \mathbf{D}} P(v_i \,|\, y = -1)$$

ML evolution (a very rough account)

1970: Bayes classifier

1990: SVM, boosting, feature

2000: feature engineering / ensemble learning

2005 $\sim$ **2010**: representation learning, auto-encoders

2010 on: deep learning

Traditional machine learning

(before deep learning)

Traditional machine learning: Feature computation

Features that describe the object

- concentration
- asymmetry
- smoothness
- entropy
- spirality, etc

$(x_1, x_2, \ldots, x_n) \Rightarrow$ generic model $y = f_w(x_1, x_2, \ldots, x_n) \Rightarrow y = \text{spiral}$

Traditional $\times$ Deep learning

Traditional ML

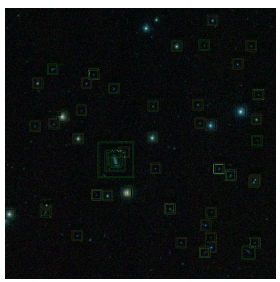
$\Rightarrow$ feature extraction/ selection $\Rightarrow$ $(x_1, x_2, \ldots, x_n)$ $\Rightarrow$ Traditional ML algoritm $\Rightarrow$ y

Traditional ML

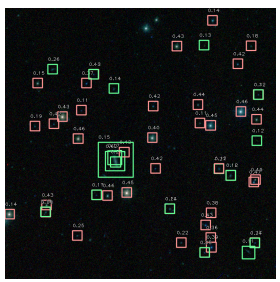
⇒ feature extraction/ selection ⇒ $(x_1, x_2, \ldots, x_n)$ ⇒ Traditional ML algoritm ⇒ y

Deep learning: end-to-end processing

⇒ Deep learning algorithm ⇒ y



⇒ Deep learning algorithm ⇒



Where deep learning is making a difference:

- Computer Vision
- Speech recognition
- Natural language processing (NLP)
- Audio recognition
- etc

Non-structured data; feature extraction is hard; complex processing pipelines

Machine learning: evolution through time

no data

rules

```
if age > 40:
    if is_home_owner:
        print("give a credit")
    else:
        if income > 5000:
            print("give a credit")
        else:
            print("to refuse")
else:
    if education == "university":
        print("...")
    else:
        print("...")
```

Machine learning: evolution through time

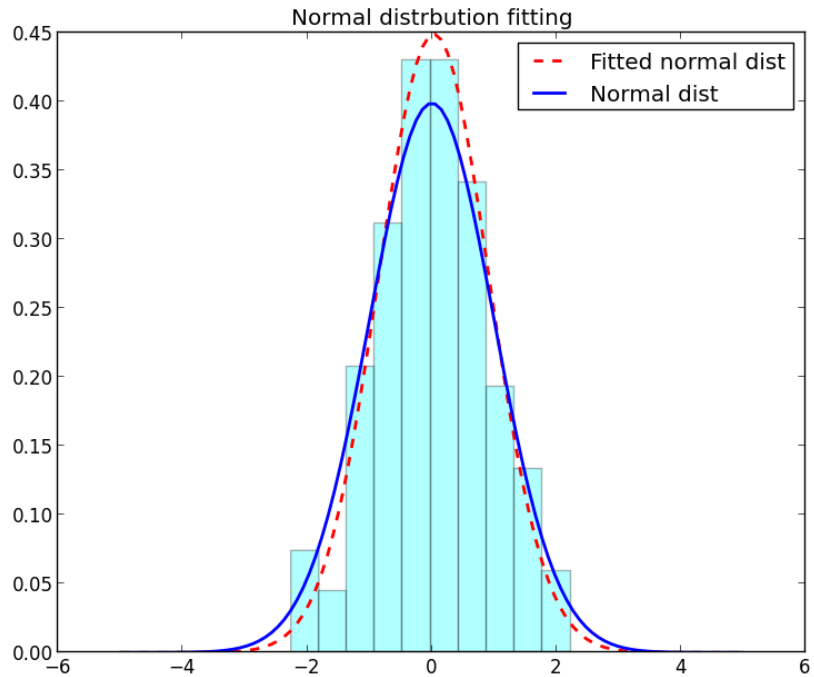
no data

rules

some data

parametric estimation

model-based



Machine learning: evolution through time

no data

rules

some data

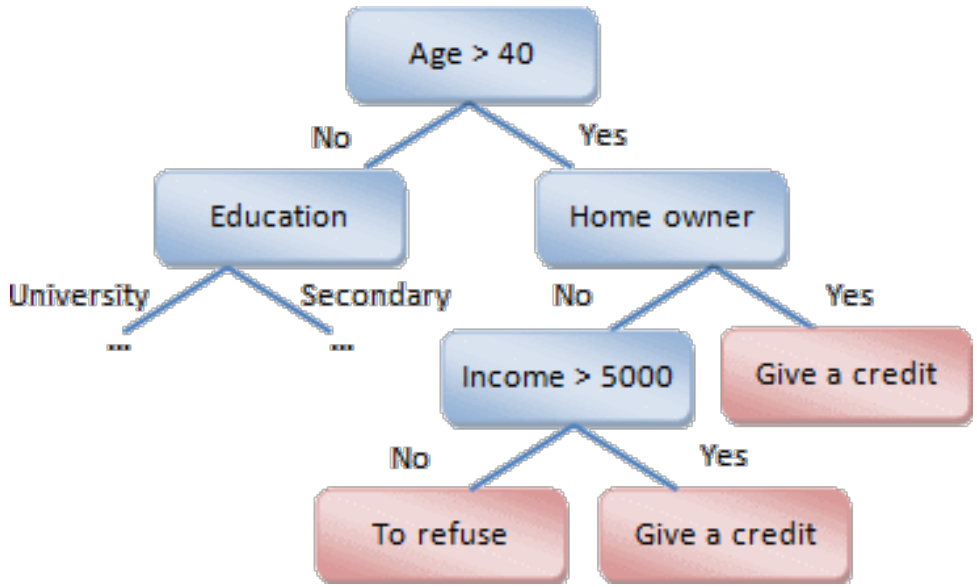
parametric estimation

model-based

more data

model induction

(ML algorithms)



Machine learning: evolution through time

no data

rules

some data

parametric estimation

model-based

more data

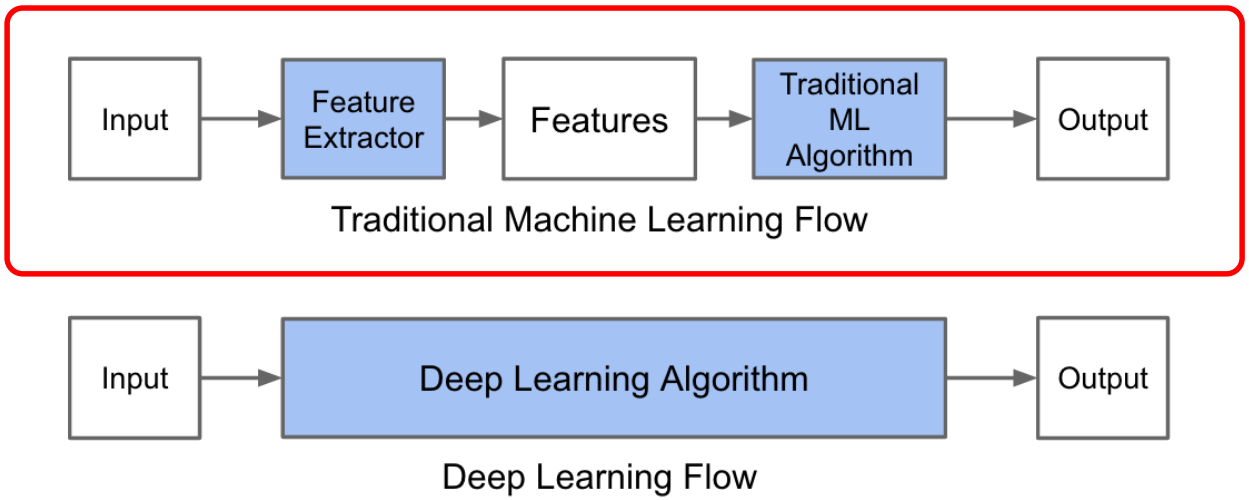
model induction

(ML algorithms)

much more data

feature engineering

ensembles



Machine learning: evolution through time

no data

rules

some data

parametric estimation

model-based

more data

model induction

(ML algorithms)

much more data

feature engineering

ensembles

Big Data

representation learning

(deep learning, end-to-end)

Traditional Machine Learning Flow

Deep Learning Flow

Machine learning: evolution through time

no data

rules

some data

parametric estimation

model-based

more data

model induction

(ML algorithms)

much more data

feature engineering

ensembles

Big Data

representation learning

(deep learning, end-to-end)

Data-driven