Naive Bayes Classifier

One of the most practical ML models like decision trees and neural networks.

Successful applications. Diagnosis, classifying text documents

Limitations.

- Moderate or large training data is available
- Input attributes are conditionally independent given classification

Naive Bayes Classifier

Consider target function/concept $c : X \to T$ where each instance $\mathbf{x} \in X$ is represented by input attributes $\mathbf{x} = (x_1, \ldots, x_n)^\top$

The most probable classification of new instance $\mathbf{x}$ is

$$\begin{aligned} t_{\text{MAP}} &= \arg\max_{t \in T} P(t|x_1, \ldots, x_n) \\ &= \arg\max_{t \in T} \frac{P(x_1, \ldots, x_n|t)P(t)}{P(x_1, \ldots, x_n)} \\ &= \arg\max_{t \in T} P(x_1, \ldots, x_n|t)P(t) \end{aligned}$$

Using naive Bayes assumption $P(x_1, \ldots, x_n|t) = \prod_{i=1}^{n} P(x_i|t)$,

$$t_{\text{NB}} = \arg\max_{t \in T} P(t) \prod_{i=1}^{n} P(x_i|t) \ .$$

Naive Bayes Algorithm

NAIVE-BAYES-LEARN(D)

For each value of target output t

$\widehat{P}(t) \leftarrow$ estimate $P(t)$ using D

For each value of attribute x_i

$\widehat{P}(x_i|t) \leftarrow$ estimate $P(x_i|t)$ using D

CLASSIFY-NEW-INSTANCE($\mathbf{x}$)

$$t_{\text{NB}} = \arg\max_{t \in T} \widehat{P}(t) \prod_{i=1}^{n} \widehat{P}(x_i|t)$$

Naive Bayes for *PlayTennis*

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes for *PlayTennis*

Example. Predict target concept *PlayTennis* of new instance

(Outlook = *Sunny*, Temperature = *Cool*, Humidity = *High*, Wind = *Strong*)

$P(Yes)\ P(Sunny|Yes)\ P(Cool|Yes)\ P(High|Yes)\ P(Strong|Yes)$

=

$P(No)\ P(Sunny|No)\ P(Cool|No)\ P(High|No)\ P(Strong|No)$

=

t_{NB} =

$P(No|Sunny, Cool, High, Strong)$ =

Properties of Naive Bayes

- Conditional independence assumption is often violated:

$$P(x_1, \ldots, x_n|t) = \prod_{i=1}^{n} P(x_i|t)$$

  - But, Naive Bayes works surprisingly well in practice
  - Estimated posteriors $\widehat{P}(t|\mathbf{x})$ need not be correct; only need that

$$\arg\max_{t \in T} \widehat{P}(t) \prod_{i=1}^{n} \widehat{P}(x_i|t) = \arg\max_{t \in T} P(t) P(x_1, \ldots, x_n|t)$$

Properties of Naive Bayes

- What if none of the training instances with target output value t have attribute value x_i? Then,

  $\widehat{P}(x_i|t) = 0 \qquad \text{and} \qquad \widehat{P}(t)\prod_{i=1}^{n}\widehat{P}(x_i|t) = 0\ .$

  Solution. Use Bayesian estimate $\widehat{P}(x_i|t) \leftarrow \dfrac{|D_{tx_i}| + mp}{|D_t| + m}$ where

  - $|D_t|$ is no. of training examples with target output value t,
  - $|D_{tx_i}|$ is no. of training examples with target output value t and attribute value x_i,
  - p is prior estimate for $\widehat{P}(x_i|t)$, and
  - m is weight given to prior p (no. of "virtual" examples).

Expectation Maximization (EM)

When to use?

- Data is only partially observable
- Unsupervised clustering (target output unobservable)
- Supervised learning (some input attributes unobservable)

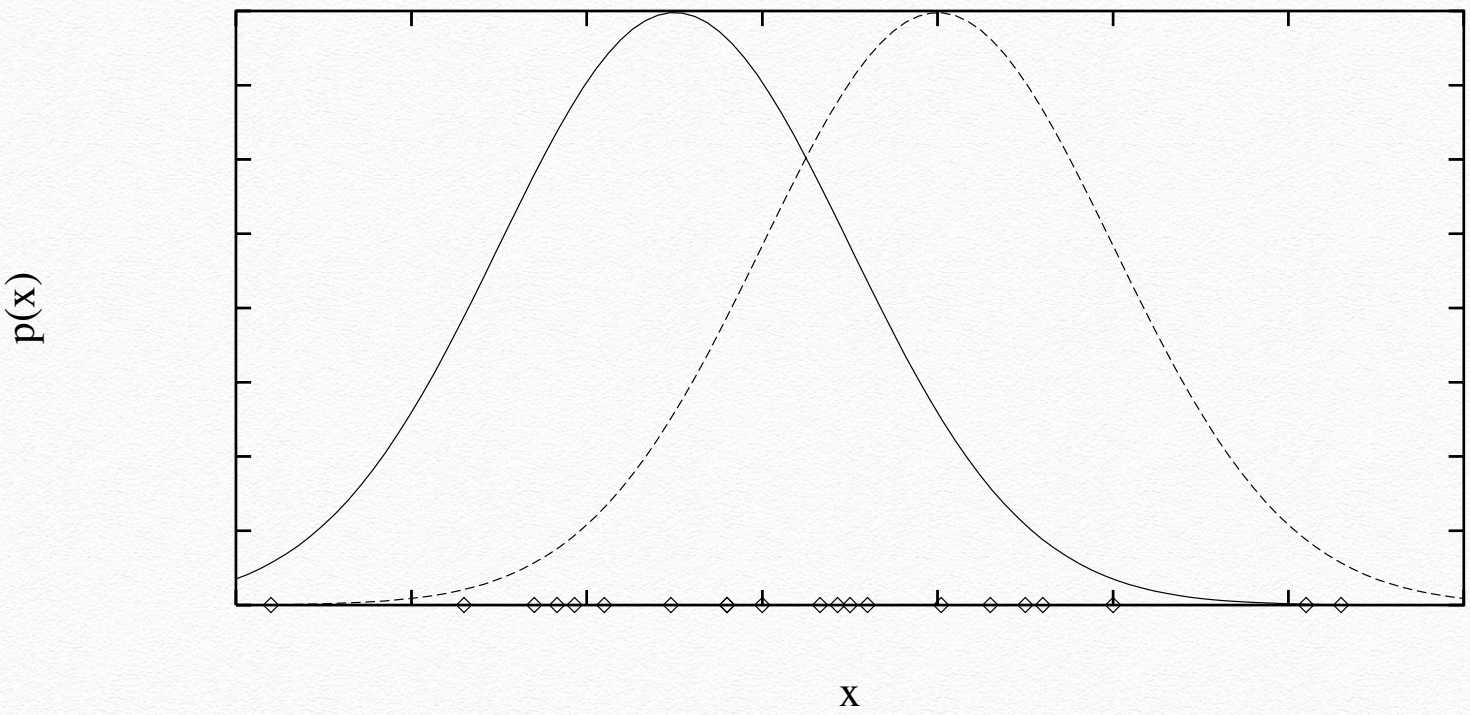
Applications.

- Training Bayesian belief networks
- Unsupervised clustering
- Learning hidden Markov models
- Inverse reinforcement learning

Generating Data from a Mixture of *M* Gaussians

Each instance x_d is generated by

- Selecting one of the *M* Gaussians with uniform probability
- Sampling an instance from the selected Gaussian



EM for Estimating M Means

Given

- Instances from X generated by mixture of M Gaussians with the same known variance σ^2
- Unknown means $\langle \mu_1, \ldots, \mu_M \rangle$ of the M Gaussians
- Don't know which instance x_d is generated by which Gaussian

Determine maximum likelihood (ML) estimates of $\langle \mu_1, \ldots, \mu_M \rangle$

Consider full description of each instance as $d = \langle x_d, z_{d1}, z_{d2} \rangle$ where

- z_{dm} is unobservable and is of value 1 if m-th Gaussian is selected to generate x_d, and 0 otherwise
- x_d is observable

EM for Estimating M Means

EM Algorithm. Pick random initial $h = \langle \mu_1, \mu_2 \rangle$. Then, iterate

- E Step. Calculate the expected value $\mathbb{E}[z_{dm}]$ of each hidden/latent variable z_{dm}, assuming the current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds.

$$\mathbb{E}[z_{dm}] = \frac{p(x_d|\mu_m)}{\sum_{\ell=1}^{2} p(x_d|\mu_\ell)} = \frac{\exp(-\frac{1}{2\sigma^2}(x_d - \mu_m)^2)}{\sum_{\ell=1}^{2} \exp(-\frac{1}{2\sigma^2}(x_d - \mu_\ell)^2)}$$

- M Step. Calculate a new ML hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming the value taken on by each latent variable z_{dm} is its expected value $\mathbb{E}[z_{dm}]$ computed above. Replace h by $h' = \langle \mu'_1, \mu'_2 \rangle$.

$$\mu'_m \leftarrow \frac{\sum_{d \in D} \mathbb{E}[z_{dm}]\ x_d}{\sum_{d \in D} \mathbb{E}[z_{dm}]}$$

EM Algorithm

Converges to local ML hypothesis *h'* and provides estimates of hidden/latent variables z_{dm}

In fact, local maximum in $\mathbb{E}[\ln p(D|h')]$

- *D* is complete (observable plus unobservable variables) data
- Expectation is wrt unobserved variables in *D*