Introduction to Machine Learning
NPFL 054

http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

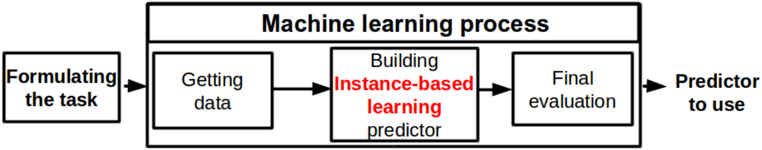
Martin Holub
holub@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Outline

- Instance-based learning
- Naïve Bayes algorithm
- Bayesian networks
- Maximum likelihood estimation

Instance-based learning



Instance-based learning
Key idea

- IBL methods initially store the training data, they are *lazy* methods
- For a new instance, prediction is based on local similarity,
  i.e. a set of similar instances are retrieved and used for prediction
- IBL methods can construct a different approximation of a target function for each distinct test instance
- Both classification and regression

Instance-based learning
Key points

1. A distance metric
2. How many nearby neighbours look at?
3. A weighting function
4. How to fit with local points?

Instance-based learning
Distance metric

Recall dissimilarity metrics for clustering. The most common ones

- **Euclidean distance**

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\sum_{r=1}^{m} (x_{i_r} - x_{j_r})^2}$$

- **Manhattan distance**

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sum_{r=1}^{m} |x_{i_r} - x_{j_r}|$$

Instance-based learning
k-Nearest Neighbour algorithm

1. **A distance metric**: Euclidian (most widely used)
2. **How many nearby neighbours look at?** k
3. **A weighting function**: unused
4. **How to fit with local points?**

- **k-NN classification**

$$f(\mathbf{x}) = \mathrm{argmax}_{v \in Y} \sum_{i=1}^{k} \delta(v, y_i), \tag{1}$$

  where $\delta(a, b) = 1$ if $a = b$, otherwise 0

- **k-NN regression**

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{k} y_i}{k} \tag{2}$$

Instance-based learning
Distance-weighted k-NN algorithm

1. **A distance metric**: Euclidian (most widely used)
2. **How many nearby neighbours look at?** k
3. **A weighting function**: greater weight of closer neighbours

$$w_i(\mathbf{x}) \equiv \frac{1}{d(\mathbf{x}, \mathbf{x_i})^2}$$

4. **How to fit with local points?**

- **Classification**

$$f(\mathbf{x}) = \mathrm{argmax}_{v \in Y} \sum_{i=1}^{k} w_i(\mathbf{x}) \delta(v, y_i) \tag{3}$$

- **Regression**

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{k} w_i(\mathbf{x}) y_i}{\sum_{i=1}^{k} w_i(\mathbf{x})} \tag{4}$$

Instance-based learning
Distance-weighted k-NN algorithm

Shepard's method

- **Classification**

$$f(\mathbf{x}) = \mathrm{argmax}_{v \in Y} \sum_{i=1}^{n} w_i(\mathbf{x}) \delta(v, y_i) \tag{5}$$

- **Regression**

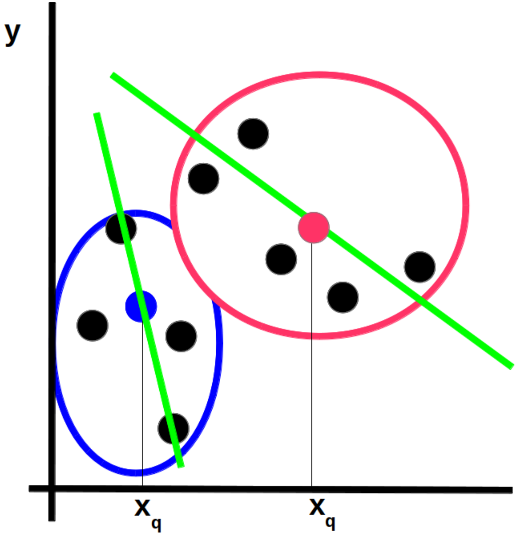
$$f(\mathbf{x}) = \frac{\sum_{i=1}^{n} w_i(\mathbf{x}) y_i}{\sum_{i=1}^{n} w_i(\mathbf{x})} \tag{6}$$

Instance-based learning
Locally weighted linear regression

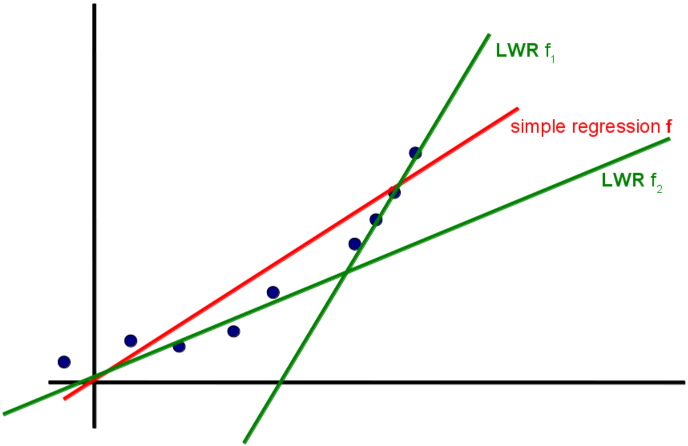
1. **A distance metric**: Euclidian (most widely used)
2. **How many nearby neighbours look at?** k
3. **A weighting function**: $w_i(\mathbf{x})$
4. **How to fit with local points?**

$$\mathbf{\Theta}^\star = \mathrm{argmin}_{\mathbf{\Theta}} \sum_{i=1}^{k} w_i(\mathbf{x})(\Theta^T \mathbf{x}_i - y_i)^2 \qquad (7)$$

Instance-based learning
Locally weighted linear regression



Instance-based learning
LW linear regression vs. simple regression



Naïve Bayes classifier
Bayes theorem

$$\text{posterior probability} = \frac{\text{prior probability} \times \text{likelihood}}{\text{marginal likelihood}} \tag{8}$$

$$\Pr(Y \mid A_1, \dots, A_m) = \frac{\Pr(Y) \times \Pr(A_1, \dots, A_m \mid Y)}{\Pr(A_1, \dots, A_m)}$$

Naïve Bayes classifier
Conditional independence

Let X, Y and Z be three descrete random variables. We say that X is *conditionally independent* of Y given Z if

$\forall x_i, y_j, z_k, x_i \in Values(X), y_j \in Values(Y), z_k \in Values(Z):$

$$\Pr(X = x_i | Y = y_j, Z = z_k) = \Pr(X = x_i | Z = z_k) \tag{9}$$

I.e., $\mathrm{P}(X|Y, Z) = \mathrm{P}(X|Z)$.

Thunder & Rain & Lighting

Assume three variables: `Thunder`, `Rain`, and `Lighting`.
`Thunder` is conditionally independent of `Rain` given `Lighting`:

$$Pr(\mathrm{Thunder}|\mathrm{Rain}, \mathrm{Lighting}) = \Pr(\mathrm{Thunder}|\mathrm{Lighting})$$

Naïve Bayes classifier
Conditional independence

If we work with two features A_1, A_2 and we assume that they are conditionally independent given the target class Y, then

$$\Pr(A_1, A_2|Y) \overset{\text{product rule}}{=} \Pr(A_1|A_2, Y) * \Pr(A_2|Y) \overset{\text{c. i. a.}}{=} \Pr(A_1|Y) * \Pr(A_2|Y)$$

Note: Product rule (a.k.a. Chain rule)

$$\Pr(A_m, \dots, A_1) = \Pr(A_m|A_{m-1}, \dots, A_1) \cdot \Pr(A_{m-1}, \dots, A_1)$$

Naïve Bayes classifier

Assume conditional independence of features $A_1, \ldots, A_m$ given Y.

Therefore

$$\Pr(\mathbf{x}|y) = \Pr(x_1, x_2, \ldots, x_m|y) \stackrel{\text{product rule}}{=} \prod_{j=1}^{m} \Pr(x_j|x_1, x_2, \ldots, x_{j-1}, y) \stackrel{\text{c. i. a.}}{=}$$

$$= \prod_{j=1}^{m} \Pr(x_j|y)$$

Naïve Bayes classifier

$$\hat{y}^\star = argmax_{y_k \in Y} \Pr(y_k) \prod_{j=1}^{m} \Pr(x_j|y_k) \tag{10}$$

Number of parameters: $2 \cdot (2^m - 1) \to 2 \cdot m$

Naïve Bayes classifier
Discriminative vs. generative classifiers

Computing $\Pr(y|\mathbf{x})$

- **discriminative classifier** does not care about how the data was generated. It directly discriminates the value of y for any $\mathbf{x}$.
- **generative classifier** models how the data was generated in order to classify an example.

Naïve Bayes classifier
Discriminative vs. generative classifiers

- Logistic regression classifier is a **discriminative classifier**

$$f(\mathbf{x};\Theta) = p(y = 1|\mathbf{x},\Theta)$$

- Naïve Bayes classifier is a **generative classifier**

1. Learn $\Pr(\mathbf{x}|y)$ and $\Pr(y)$
2. Apply Bayes rule to get

$$\Pr(y|\mathbf{x}) = \frac{\Pr(\mathbf{x}|y)\Pr(y)}{\Pr(\mathbf{x})} \sim \Pr(\mathbf{x}|y)\Pr(y)$$

3. Classify $\mathbf{x}$

$$\hat{y} = \mathrm{argmax}_y \Pr(y|\mathbf{x}) = \mathrm{argmax}_y \Pr(\mathbf{x}|y)\Pr(y)$$

Naïve Bayes classifier

Naive assumption of feature conditional independence given a target class is rarely true in real world applications. Nevertheless, Naïve Bayes classifier surprisingly often shows good performance in classification.

Naïve Bayes Classifier is a linear classifier

NB classifier gives a method for predicting rather than for building an explicit classifier.

We focus on **binary classification** $Y = \{0, 1\}$ with binary features $A_1, \ldots, A_m$.

We predict 1 iff

$$\frac{\Pr(y=1)\prod_{j=1}^{m}\Pr(x_j|y=1)}{\Pr(y=0)\prod_{j=1}^{m}\Pr(x_j|y=0)} > 1$$

Naïve Bayes Classifier is a linear classifier

Denote $p_j = \Pr(x_j = 1|y = 1)$, $q_j = \Pr(x_j = 1|y = 0)$

Then

$$\frac{\Pr(y = 1) \prod_{j=1}^{m} p_j^{x_j} (1 - p_j)^{1-x_j}}{\Pr(y = 0) \prod_{j=1}^{m} q_j^{x_j} (1 - q_j)^{1-x_j}} > 1$$

$$\frac{\Pr(y = 1) \prod_{j=1}^{m} (1 - p_j) (\frac{p_j}{1-p_j})^{x_j}}{\Pr(y = 0) \prod_{j=1}^{m} (1 - q_j) (\frac{q_j}{1-q_j})^{x_j}} > 1$$

Naïve Bayes Classifier is a linear classifier

Take logarithm

$$\log \frac{\Pr(y=1)}{\Pr(y=0)} + \sum_{j=1}^{m} \log \frac{1-p_j}{1-q_j} + \sum_{j=1}^{m} (\log \frac{p_j}{1-p_j} - \log \frac{q_j}{1-q_j}) x_j > 0$$

NB classifier as a linear classifier where

$$\Theta_0 = \log \frac{\Pr(y=1)}{\Pr(y=0)} + \sum_{j=1}^{m} \log \frac{1-p_j}{1-q_j}$$

$$\Theta_j = \log \frac{p_j}{1-p_j} - \log \frac{q_j}{1-q_j}, \quad j = 1, \dots, m$$

Bayesian belief networks
Motivation

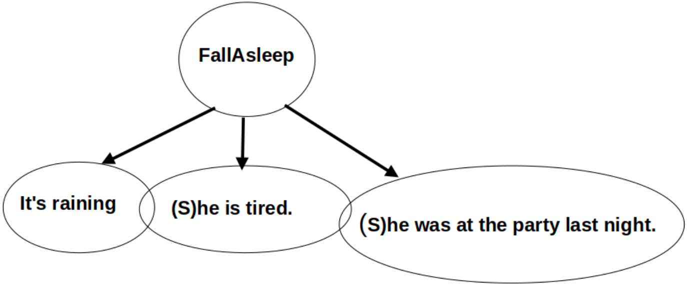
Task: Will a student fall asleep during the lecture?

Binary features
`It's raining`, `(S)he is tired`, `(S)he was at the party last night`

Target binary attribute
$Y = \texttt{FallAsleep}$

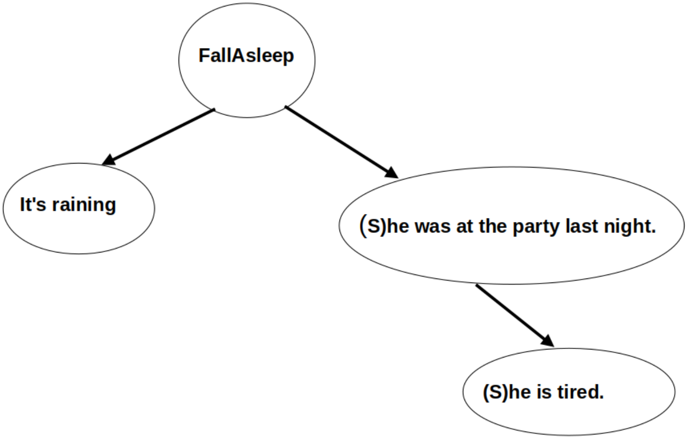
Bayesian belief networks
Motivation

Visualization of Naïve Bayes assumption



Bayesian belief networks
Motivation

... but



Bayesian belief networks
Motivation

- Naïve Bayes classifier assumes that ALL features are conditionally independent given the value of the target class.
- A Bayesian network is a probabilistic graphical model that encodes probabilistic relationships among attributes of interest.
- BBNs allow stating conditional independence assumptions that apply to subsets of the attributes.
- Dependencies are modeled as graph where nodes correspond to attributes and edges go from cause to effect.
- BBNs combine prior knowledge with observed data.

Bayesian belief networks
Settings

Consider an arbitrary set of random variables $X_1, X_2, ..., X_m$. Each variable X_i can take on the set of possible values $Values(X_i)$.

We define the **joint space** of the variables $X_1, X_2, ..., X_m$ to be the cross product $Values(X_1) \times Values(X_2) \times Values(X_3) \times ... \times Values(X_m)$.

The probability distribution over the joint space is called the **joint probability distribution** $\Pr(x_1, x_2, ..., x_m)$ where
$x_1 \in Values(X_1), x_2 \in Values(X_2), ..., x_n \in Values(X_m)$.

BBN describes the joint probability distribution for a set of variables by specifying a set of conditional independence assumptions together with sets of local conditional probabilities.

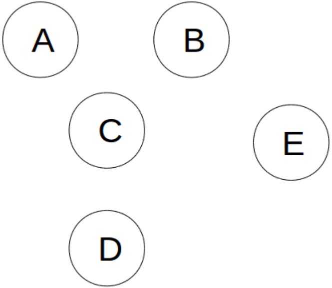
Bayesian belief networks

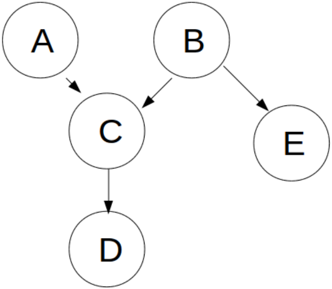
Representation

1. A directed acyclic graph $G = (V, E)$
   - nodes are random variables
   - arcs between nodes represent probabilistic dependencies
   - arcs are drawn from cause to effect
   - Y is a *descendant* of X if there is a directed path from X to Y.
2. The network arcs represent the assertion that the variable X is conditionally independent of its nondescendants given its immediate predecessors $Parents(X)$; $\Pr(X|Parents(X))$
3. A set of tables for each node in the graph - a conditional probability table is given for each variable; it describes the probability distribution for that variable given the values of its immediate predecessors.

Building a Bayes net

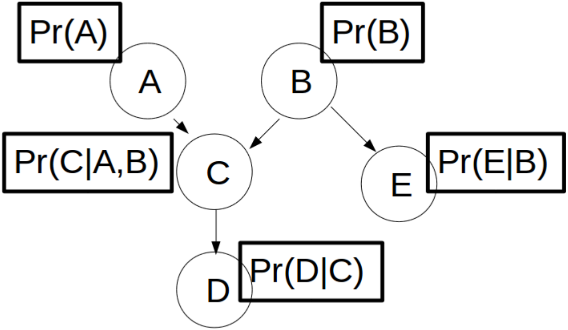
1. Choose the variables to be included in the net: A, B, C, D, E
2. Add the links





Building a Bayes net

3. Add a probability table for each root node $\Pr(X)$ and nonroot node $\Pr(X|Parents(X))$



Once the net is built ...

The join probability of any assignment of values $x_1, x_2, ..., x_m$ to the tuple of network variables $X_1, X_2, ..., X_m$ can be computed by the formula

$$\Pr(x_1, x_2, ..., x_m) = \Pr(X_1 = x_1 \wedge X_2 = x_2 \wedge \cdots \wedge X_m = x_m) = \prod_{i=1}^{m} \Pr(x_i | Parents(X_i)) \tag{11}$$

Bayesian belief networks

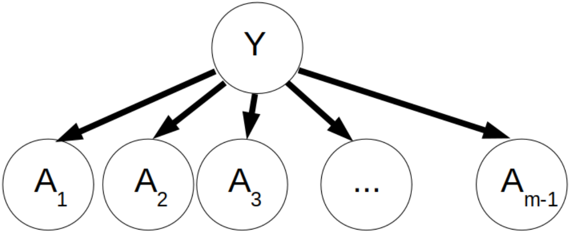
Two components

1. A function for evaluating a given network based on the data.
2. A method for searching through the space of possible networks.

Learning the network structure

- searching through the space of possible sets of edges
- estimating the conditional probability tables for each set
- computing the quality of the network

Bayesian belief networks
Naïve Bayes Classifier



K2 algorithm

This 'search and score' algorithm heuristically searches for the most probable belief-network structure given a training data.

It starts by assuming that a node has no parents, after which, in every step it adds incrementally the parent whose addition mostly increase the probability of the resulting structure. K2 stops adding parents to the nodes when the addition of a single parent cannot increase the probability of the network given the data.

Maximum likelihood estimation
Motivation

The binomial distribution is the discrete probability distribution of the number of successes in a sequence of n independent yes/no experiments, each of which yields success with probability p, $X \sim Bin(n, p)$.

Probabilistic mass function $\Pr(X = k) = f(k; n, p) = \frac{n!}{k!(n-k)!} p^k (1 - p)^{(n-k)}$

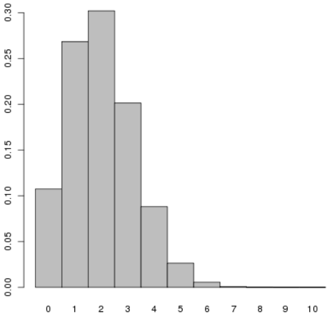
Coin tossing

Let $n = 10$, x represents the number of successes in 10 trials and probability of head on one trial is $p = 0.2$. Then

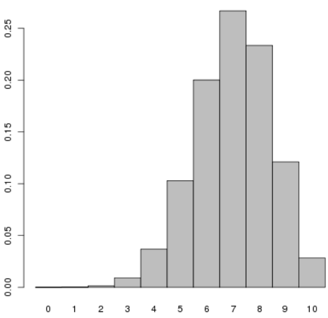
$$f(x; 10, 0.2) = \frac{10!}{x!(10 - x)!} 0.2^x (0.8)^{(10-x)}$$

Maximum likelihood estimation
Motivation

$p = 0.2$



$p = 0.7$



Maximum likelihood estimation

- data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$

Assumption: $\mathbf{x}_1, \dots, \mathbf{x}_n$ are independent and identically distributed with an unknown probability density function $f(\mathbf{X}; \Theta)$

- Θ is a vector of parameters of the probability distribution $\Theta = \langle \theta_1, \dots, \theta_m \rangle$
- joint density function $f(\mathbf{x}_1, \dots, \mathbf{x}_n; \Theta) \stackrel{i.i.d.}{=} \prod_{i=1}^{n} f(\mathbf{x}_i; \Theta)$

We determine what value of Θ would make the data $\mathbf{X}$ most likely.

Maximum likelihood estimation

MLE is a method for estimating parameters from data.

Goal: identify the population that is most likely to have generated the sample.

Likelihood function

$$\mathcal{L}(\Theta|\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{df}{=} \prod_{i=1}^{n} f(\mathbf{x}_i; \Theta) \tag{12}$$

Log-likelihood function

$$\log \mathcal{L}(\Theta|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} \log f(\mathbf{x}_i; \Theta) \tag{13}$$

Maximum likelihood estimate of Θ

$$\Theta^{\star}_{MLE} = \mathrm{argmax}_{\Theta} \log \mathcal{L}(\Theta|\mathbf{x}_1, \ldots, \mathbf{x}_n) \tag{14}$$

Maximum likelihood estimation

MLE analytically

- Likelihood equation: $\frac{\partial \log \mathcal{L}(\Theta|X)}{\partial \theta_i} = 0$ at θ_i for all $i = 1, \ldots, m$
- Maximum, not minimum: $\frac{\partial^2 \mathcal{L}(\Theta|\mathbf{x})}{\partial \theta_i^2} < 0$

Numerically

- Use an optimization algorithm (for ex. Gradient Descent)

Maximum likelihood estimation
Binomial distribution

Estimate the probability p that a coin lands head using the result of n coin tosses, k of which resulted in heads. $\Theta = \langle p \rangle$

- $f(k; n, p) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$
- $\mathcal{L}(p|n,k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$
- $\log \mathcal{L}(p|n,k) = \log \frac{n!}{k!(n-k)!} + k \log p + (n-k) log(1-p)$
- $\frac{\partial \log \mathcal{L}(p|n,k)}{\partial p} = \frac{k}{p} - \frac{n-k}{1-p} = 0$
- $\hat{p}_{MLE} = \frac{k}{n}$

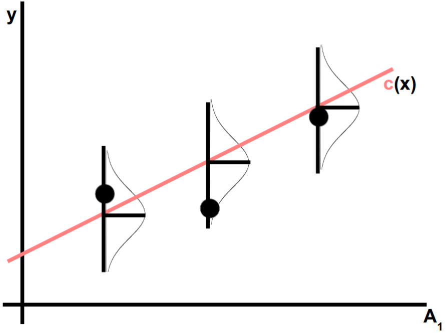
Maximum likelihood estimation
Least squares

Linear regression $y = \Theta^\top \mathbf{x}$

Learn $\hat{\Theta}^\star$ from $Data = \{\langle \mathbf{x}_i, y_i \rangle, y_i \in \mathcal{R}, i = 1, ..., n\}$ and use MLE.

Assumption: At each value of A_1, the output value y is subject to random error ϵ that is normally distributed $N(0, \sigma^2)$

$y_i = \Theta^\top \mathbf{x}_i + \epsilon_i$



Maximum likelihood estimation
Least squares

- probability density function of the Normal distribution

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

- $\epsilon_i = y_i - \Theta^\top \mathbf{x}_i \sim N(0,\sigma^2)$

$$\mathcal{L}(\mu,\sigma|\epsilon) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(\epsilon_i-\mu)^2}{2\sigma^2}}$$

$$\mathcal{L}(\Theta,\sigma|\mathbf{X},\mathbf{y}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(y_i-\Theta^\top \mathbf{x}_i)^2}{2\sigma^2}}$$

Maximum likelihood estimation
Least squares

$$\log \mathcal{L}(\Theta, \sigma | \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{(y_i - \Theta^\top \mathbf{x}_i)^2}{2\sigma^2}$$

$$\mathrm{argmax}_\Theta \log \mathcal{L}(\Theta, \sigma | \mathbf{X}, \mathbf{y}) = \mathrm{argmax}_\Theta \sum_{i=1}^{n} -\frac{1}{2\sigma^2}(y_i - \Theta^\top \mathbf{x}_i)^2$$

$$\mathrm{argmin}_{\boldsymbol{\Theta}} \log \mathcal{L}(\Theta, \sigma | \mathbf{X}, \mathbf{y}) = \mathrm{argmin}_\Theta \sum_{i=1}^{n} (y_i - \Theta^\top \mathbf{x}_i)^2$$

The maximum least square estimates are equivalent to the maximum likelihood estimates under the assumption that Y is generated by adding random noise to the true target values characterized by the Normal distribution $N(0, \sigma^2)$.

Maximum likelihood estimation
Logistic regression

Logistic regression models conditional probability using sigmoid function.

$$f(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{\Theta}^T \mathbf{x}}} = \Pr(y = 1|\mathbf{x})$$

Learn $\hat{\Theta}^\star$ from $Data = \{\langle \mathbf{x}_i, y_i \rangle, y_i \in \{0, 1\}, i = 1, ..., n\}$ and use MLE.

Maximum likelihood estimation
Logistic regression

$$f(\mathbf{x};\Theta) = \Pr(y = 1|\mathbf{x})$$

$$\prod_{i=1}^{n} \Pr(y = y_i|\mathbf{x}_i) = \prod_{i=1}^{n} f(\mathbf{x}_i;\Theta)^{y_i}(1 - f(\mathbf{x}_i;\Theta))^{1-y_i}$$

$$\mathcal{L}(\Theta|\mathbf{X},\mathbf{y}) = \prod_{i=1}^{n} f(\mathbf{x}_i;\Theta)^{y_i}(1 - f(\mathbf{x}_i;\Theta))^{1-y_i}$$

$$\log\mathcal{L}(\Theta|\mathbf{X},\mathbf{y}) = \sum_{i=1}^{n} y_i \log f(\mathbf{x}_i;\Theta) + (1 - y_i)\log(1 - f(\mathbf{x}_i;\Theta))$$

$$\hat{\Theta}_{MLE} = \mathrm{argmax}_{\Theta} \sum_{i=1}^{n} y_i \log f(\mathbf{x}_i;\Theta) + (1 - y_i)\log(1 - f(\mathbf{x}_i;\Theta))$$

Maximum likelihood estimation
Naïve Bayes classifier

$$\hat{y}^\star = argmax_{y_k \in Y} \Pr(y_k) \prod_{j=1}^{m} \Pr(x_j|y_k)$$

Maximum likelihood estimation
Naïve Bayes classifier

Categorical feature $\mathtt{A}_j$

Theorem

The Maximum likelihood estimates for NB take the form

- $\Pr(y) = \frac{c_y}{n}$ *where* $c_y = \sum_{i=1}^{n} \delta(y_i, y)$
- $\Pr(x|y) = \frac{c_{j_{x|y}}}{c_y}$ *where* $c_{j_{x|y}} = \sum_{i=1}^{n} \delta(y_i, y)\delta(\mathbf{x}_{i_j}, x)$

Maximum likelihood estimation
Naïve Bayes classifier

Continuous feature $\mathtt{A}_j$

Typical assumption, each continuous feature has a Gaussian distribution.

Theorem

The ML estimates for NB take the form

- $\overline{\mu_k} = \frac{\sum_{i=1}^{n} x_i^j \delta(y_i = y_k)}{\sum_{j=1}^{n} \delta(Y^j = y_k)}$
- $\overline{\sigma_k}^2 = \frac{\sum_{i=1}^{j} (x_i^j - \overline{\mu_k})^2 \delta(y_i = y_k)}{\sum_j \delta(Y^j = y_k)}$

$$\Pr(x|y_k) = \frac{1}{\sqrt{2\pi\overline{\sigma_k}^2}} e^{\frac{-(x-\overline{\mu_k})^2}{2\overline{\sigma_k}^2}}$$

Summary of Examination Requirements

- Instance-based learning
- (weighted) k-NN algorithm
- Locally weighted linear regression
- Discriminative and generative classifiers
- Naïve Bayes Classifier – conditional independence, linear decision boundary
- Bayesian networks – structure, conditional probabilities
- Maximum likelihood estimations – likelihood function, loss function for logistic regression, MLE and least square method