Tomáš Horváth

INTRODUCTION TO DATA MINING

Lecture 1

Supervised Learning

Institute of Computer Science, Faculty od Science

Pavol Jozef Šafárik University in Košice

Slovak Republic

Instances and labels

- **Instances** are represented by their **attributes**

$$\mathbf{x} = (x_1, \ldots, x_k) \in \mathcal{X}, \quad \mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_k$$

- An instance belongs to a **class** or have a **value**. An instance a class or a value of which is known is called **labeled**

$$(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{L}$$

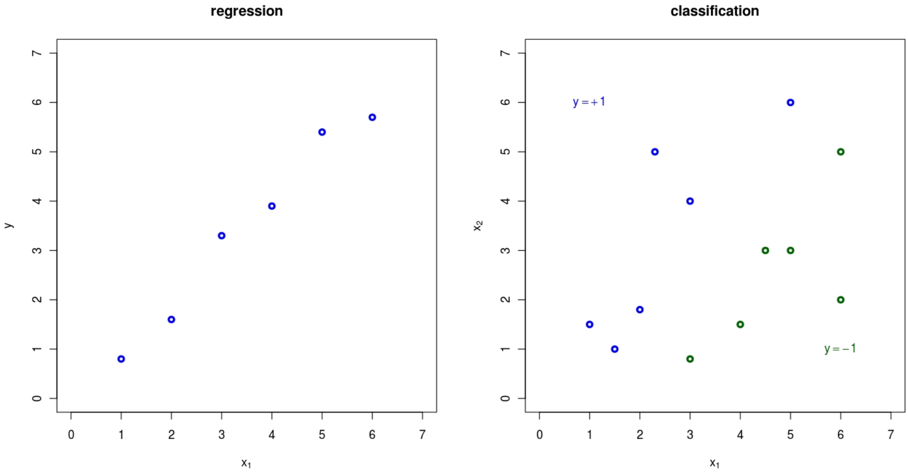
- Assume that labels are assigned according to some **unknown pattern** called labeling function

$$l : \mathcal{X} \to \mathcal{L}, \quad l(\mathbf{x}) = y$$

  - if $\mathcal{L} \subset \mathbb{Z}$[1] then l is a **classification** function (classifier)
  - if $\mathcal{L} \subset \mathbb{R}$ then l is a **regression** function (regressor)

[1] Important is, that we deal with discrete labels in case of classification.

Example



Train set and Modeling

The problem: The labeling function l is unknown.

- Good news: Even if l is not known, we have observed a sample of instances with their labels. Such a set of instances is called the **training sample**

$$\mathcal{S}^{tr} = \{(\mathbf{x}, y) | \mathbf{x} \in \mathcal{X}, y \in \mathcal{L}\}$$

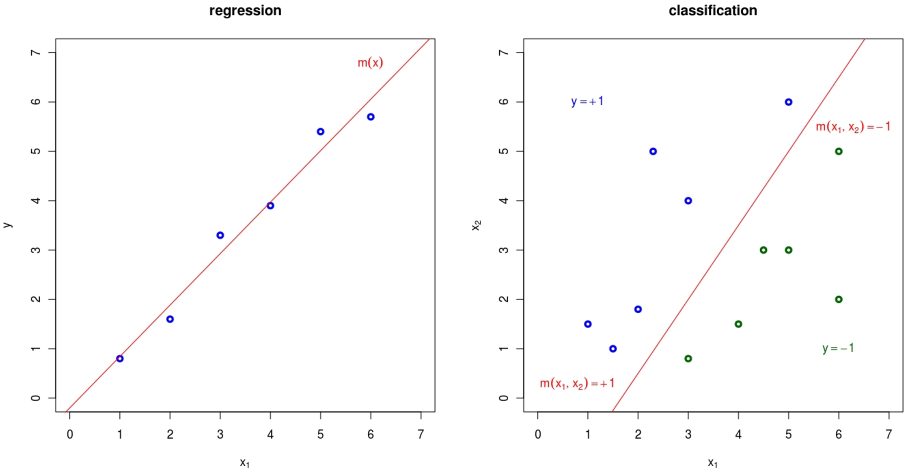
  which can be considered as an explicit definition of l.

The solution: Try somehow, using $\mathcal{S}^{tr}$, to **model** l by a mapping

$$m : \mathcal{X} \to \mathcal{L}, \quad m(\mathbf{x}) = \hat{y}$$

such that m is as close to l as possible.

Example



The quality and the parameters of m

How do we express m?

- m is given by its **type** and **parameters** $\boldsymbol{\Theta}$
  - let's focus on linear models
    - $m^{\Theta}(\mathbf{x} = (x_1, \ldots, x_k)) = \theta_0 + \theta_1 x_1 + \cdots + \theta_k x_k$
    - $\Theta = (\theta_0, \theta_1, \ldots, \theta_k)$

How to measure if m approximates l well?

- **empirical error**[1]

$$err(m^{\Theta}, \mathcal{S}^{tr}) = \sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} l_r(y, m^{\Theta}(\mathbf{x})) = \sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} (y - m^{\Theta}(\mathbf{x}))^2$$

Modeling means[2] to choose a type of m and to find its parameters $\boldsymbol{\Theta}$ such that $err(m^{\Theta}, \mathcal{S}^{tr})$ is minimal.

- **least squares estimates** (LSE)

[1] $l_r(y, m^{\Theta}(\mathbf{x}))$ is a **regression loss** function.

[2] There are also some other issues important while we are modeling, we'll explain them later.

Finding Θ analytically (1/2)

Let's $m^{\Theta}(\mathbf{x} = (x_1)) = \theta_0 + \theta_1 x_1$

- find $\Theta = (\theta_0, \theta_1)$ such that

$$err(m^{\Theta}, \mathcal{S}^{tr}) = \sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} (y - \theta_0 - \theta_1 x_1))^2$$

  is minimal.

- closed form solution

$$\theta_1 = \frac{\sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} (x_1 - \overline{x_1})(y - \overline{y})}{\sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} (x_1 - \overline{x_1})^2}$$

$$\theta_0 = \overline{y} - \theta_1 \overline{x_1}$$

  - $\overline{x_1}, \overline{y}$ denote the average values of x_1 and y over $\mathcal{S}^{tr}$, respectively

Finding Θ analytically (2/2)

Proof:

$$\frac{\partial err}{\partial \theta_0}(m^{\Theta}, \mathcal{S}^{tr}) = \sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} -2\left(y - (\theta_0 - \theta_1 x_1)\right) = 0$$

$$\Longrightarrow \left|\mathcal{S}^{tr}\right| \theta_0 = \sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} (y - \theta_1 x_1) \Longrightarrow \theta_0 = \overline{y} - \theta_1 \overline{x_1}$$

$$err(m^{\Theta}, \mathcal{S}^{tr}) = \sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} \left(y - (\overline{y} - \theta_1\overline{x_1}) - \theta_1 x_1\right)^2$$

$$\frac{\partial err}{\partial \theta_1}(m^{\Theta}, \mathcal{S}^{tr}) = \sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} -2\left(y - \overline{y} - \theta_1(x_1 - \overline{x_1})\right)(x_1 - \overline{x_1}) = 0$$

$$\Longrightarrow \theta_1 = \frac{\sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} (x_1 - \overline{x_1})(y - \overline{y})}{\sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} (x_1 - \overline{x_1})^2}$$

A generative model (1/2)

Let's $m^{\Theta}(\mathbf{x} = (x_1)) = \theta_0 + \theta_1 x_1$

- m^{Θ} approximates l with an error ϵ, i.e. $y = l(\mathbf{x}) = m^{\Theta}(\mathbf{x}) + \epsilon$
  - assume $\epsilon \sim N(0, \sigma^2)$, thus, $p(y|\mathbf{x}) \sim N(\theta_0 + \theta_1 x_1, \sigma^2)$

How is the data generated?

- assume the instances $(\mathbf{x}, y)$ are "sampled" independently
- the likelihood[1] of this sampling given some parameters $\Theta = (\theta_0, \theta_1)$ is

$$L_{\mathcal{S}^{tr}}(\Theta) = \prod_{(\mathbf{x},y)\in\mathcal{S}^{tr}} p(\mathbf{x}, y|\Theta) = \prod_{(\mathbf{x},y)\in\mathcal{S}^{tr}} p(y|\mathbf{x}, \Theta)p(\mathbf{x}, \Theta)$$

Modeling means to choose a type of m and to find its parameters Θ such that $L_{\mathcal{S}^{tr}}(\Theta)$ is maximal.

- **maximum likelihood estimates** (MLE)

[1] i.e. the probability of the data ($\mathcal{S}^{tr}$)

A generative model (2/2)

$$\prod_{(\mathbf{x},y)\in\mathcal{S}^{tr}} p(\mathbf{x},y) = \prod_{(\mathbf{x},y)\in\mathcal{S}^{tr}} p(y|\mathbf{x})p(\mathbf{x}) = \prod_{(\mathbf{x},y)\in\mathcal{S}^{tr}} p(y|\mathbf{x}) \prod_{(\mathbf{x},y)\in\mathcal{S}^{tr}} p(\mathbf{x})$$

since $p(\mathbf{x})$ doesn't depends on Θ, it's enough to maximize the **conditional likelihood**

$$L^{cond}_{\mathcal{S}^{tr}}(\Theta) = \prod_{(\mathbf{x},y)\in\mathcal{S}^{tr}} p(y|\mathbf{x}) = \prod_{(\mathbf{x},y)\in\mathcal{S}^{tr}} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-m^{\Theta}(\mathbf{x}))^2}{2\sigma^2}}$$

this is equivalent to maximize the **conditional log-likelihood**

$$ln\, L^{cond}_{\mathcal{S}^{tr}}(\Theta) = \sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} ln \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-m^{\Theta}(\mathbf{x}))^2}{2\sigma^2}} \right) \propto \sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} (y - m^{\Theta}(\mathbf{x}))^2$$

under the assumption of normality, MLE are the LSE

Gradient descent optimization

for more variables closed form solutions are bothersome

- *How to find a minimum of an "objective" function $f(\Theta)$?*
  - assume f is differentiable and convex

Gradient descent

input: $f, \alpha,$ *stopping criteria*
initialize Θ (not with zeros)
repeat
$\quad \Theta \leftarrow \Theta - \alpha \frac{\partial f}{\partial \Theta}(\Theta)$
until approximate minimum is reached
return Θ

stopping criteria

- $|\Theta^{old} - \Theta| < \epsilon$
- maximum number of iterations reached
- a combination of both

Stochastic gradient descent optimization

if f can be written as

$$f(\Theta) = \sum_{i=1}^{n} f_i(\Theta)$$

Stochastic gradient descent (SGD)

input: $f_i, \alpha,$ *stopping criteria*
initialize Θ
repeat
  for all i in random order **do**
    $\Theta \leftarrow \Theta - \alpha \frac{\partial f_i}{\partial \Theta}(\Theta)$
  end for
until approximate minimum is reached
return Θ

α is a **hyper-parameter** of the "learning" algorithm

Prediction

The aim is not to *describe* the data but rather to **predict** labels on yet unseen instances.

- **generalization error** for regression[1]

$$err(m^\Theta) = E_{(\mathbf{x},y)}\{l_r(y, m^\Theta(\mathbf{x}))\} = \int\limits_{\mathcal{X}} \int\limits_{\mathcal{L}} l_r(y, m^\Theta(\mathbf{x}))p(\mathbf{x}, y)\, \mathrm{d}y\mathrm{d}\mathbf{x}$$

- generalization error for classification[2]

$$err(m^\Theta) = E_{(\mathbf{x},y)}\{l_c(y, m^\Theta(\mathbf{x}))\} = \int\limits_{\mathcal{X}} \sum_{c \in \mathcal{L}} l_c(c, m^\Theta(\mathbf{x}))p(\mathbf{x}, y = c)\, \mathrm{d}y\mathrm{d}\mathbf{x}$$

Bayes predictor minimizes the generalization error

$$m_B = \arg\min_{m^\Theta} \; err(m^\Theta)$$

[1] $E_{(\mathbf{x},y)}\{l_r(y, m^\Theta(\mathbf{x}))\}$ is an expectation of the regression loss over $\mathcal{X} \times \mathcal{L}$.

[2] $l_c(y, m^\Theta(\mathbf{x}))$ is called **classification loss** and can be defined e.g. as $l_c(y, m^\Theta(\mathbf{x})) = 1 - \delta(y = m^\Theta(\mathbf{x}))$, with δ being a usual truth-indicator function.

Regularization

The aim is to achieve low generalization error of the model

- it means, describe the available data[1] as well as possible
- but also, *don't fit the model to the noise* in the data
- i.e. try to get a smooth model

regularized linear regression

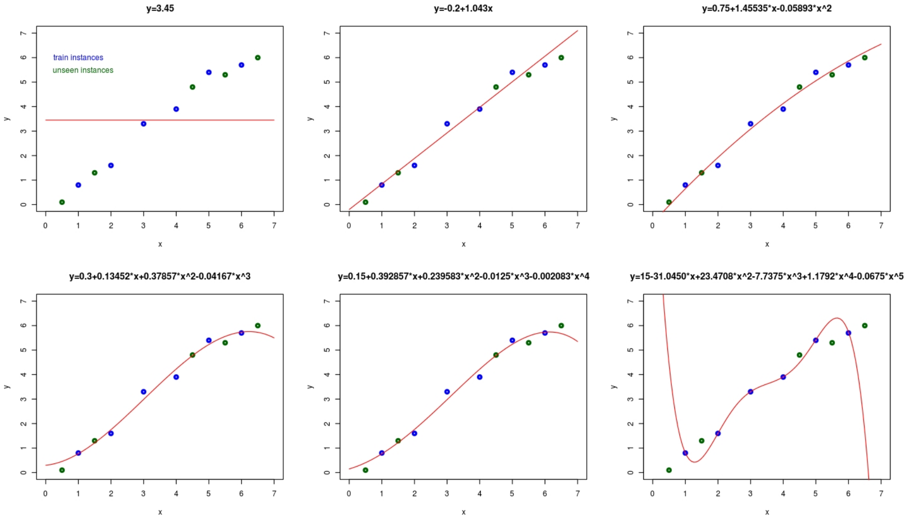
- the objective function[2] to optimize (minimize) is

$$f(\Theta) = \underbrace{\sum_{(\mathbf{x},y)\in\mathcal{S}^{tr}} (y - m^{\Theta}(\mathbf{x}))^2}_{\text{empirical error}} + \underbrace{\lambda\|\Theta\|^2}_{\text{regularization term}}$$

[1] Keep in mind that the available data is only the train set.

[2] λ is a hyper-parameter, while Θ is a parameter!

Example


y=3.45
train instances
unseen instances
y=-0.2+1.043x
y=0.75+1.45535*x-0.05893*x^2
y=0.3+0.13452*x+0.37857*x^2-0.04167*x^3
y=0.15+0.392857*x+0.239583*x^2-0.0125*x^3-0.002083*x^4
y=15-31.0450*x+23.4708*x^2-7.7375*x^3+1.1792*x^4-0.0675*x^5
x
y

The quality of a model

According to Θ, we can have many **different models** m^{Θ}.

- Which model is the best one?
- Which properties a good model should have?
  - We need some quality indicators for a model…

One model could be trained using many **different training samples**.

- What would the results be in case of using $\mathcal{S}^{tr_2}$ or any other training sample instead of $\mathcal{S}^{tr_1}$?

Bias and Variance

Bias

- measures, how $m^{\Theta,\mathcal{S}^{tr_1}}, m^{\Theta,\mathcal{S}^{tr_2}}, \dots, m^{\Theta,\mathcal{S}^{tr_m}}$ differs from l
- determines, how generic the model m^{Θ} is

Variance

- measures, how $m^{\Theta,\mathcal{S}^{tr_1}}, m^{\Theta,\mathcal{S}^{tr_2}}, \dots, m^{\Theta,\mathcal{S}^{tr_m}}$ differs frome each other
- determines, how stable the model m^{Θ} is

Underfitting vs. Overfitting

Bias

$$bias^2_{m^\Theta}(\mathbf{x}) = (\ l(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{m^{\Theta,\mathcal{S}^{tr}}(\mathbf{x})\}\ )^2$$

Variance

$$variance_{m^\Theta}(\mathbf{x}) = \mathrm{E}_{\mathcal{S}^{tr}}\{\ (\ m^{\Theta,\mathcal{S}^{tr}}(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{m^{\Theta,\mathcal{S}^{tr}}(\mathbf{x})\}\ )^2\}$$

$\mathrm{E}_{\mathcal{S}^{tr}}\{\ X\ \}$ is an **expected value** of X over all training samples.

Underfitting

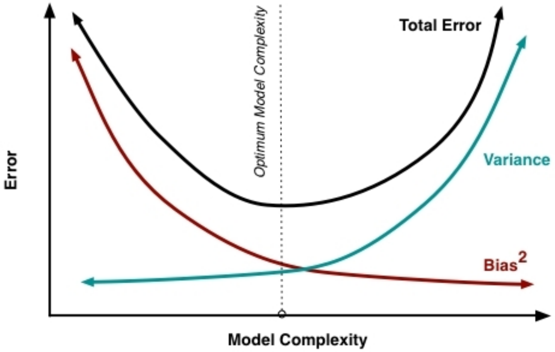
- when the model has high bias and low variance, i.e. is too general

Overfitting

- when the model has low bias and high variance, i.e. is too specific

The bias-variance tradeoff

Usually, the bias decreases with the **complexity** of the model, while variance increases with the complexity of the model. Thus, we need to find a tradeoff model, which is not too general nor too specific.



[1]

[1] image source: `http://scott.fortmann-roe.com/`

Error

What happens if we sum up the bias and the variance?[1]

$$bias^2_{m^\Theta}(\mathbf{x}) + variance_{m^\Theta}(\mathbf{x}) =$$

$$= (l(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})^2 + \mathrm{E}_{\mathcal{S}^{tr}}\{(\hat{y} - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})^2\}$$

$$= (l(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})^2 + \mathrm{E}_{\mathcal{S}^{tr}}\{(\hat{y} - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})^2\} + 2 \cdot (l(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})(\mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\} - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})$$

$$= (l(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})^2 + \mathrm{E}_{\mathcal{S}^{tr}}\{(\hat{y} - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})^2\} + 2 \cdot (l(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})\mathrm{E}_{\mathcal{S}^{tr}}\{(\mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\} - \hat{y})\}$$

$$= \mathrm{E}_{\mathcal{S}^{tr}}\{(l(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})^2\} + \mathrm{E}_{\mathcal{S}^{tr}}\{(\mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\} - \hat{y})^2\}\} + \mathrm{E}_{\mathcal{S}^{tr}}\{2 \cdot (l(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\})(\mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\} - \hat{y})\}$$

$$= \mathrm{E}_{\mathcal{S}^{tr}}\{(l(\mathbf{x}) - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\} + \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\} - \hat{y})^2\}\}$$

$$= \mathrm{E}_{\mathcal{S}^{tr}}\{(l(\mathbf{x}) - \hat{y})^2\}$$

We get the expected squared error of the model over all training samples w.r.t. the labeling.

[1] We will denote $m^{\Theta,\mathcal{S}^{tr}}(\mathbf{x})$ as $\hat{y}$ for better readability on the next slides.

Noise in sampling

The error introduced on the previous slide deals with the labeling l.

- However, the precise values of l are unknown.
  - We should consider to use the observed labels from the training sample.

As we have seen, observations are usually noisy, i.e. $y = l(\mathbf{x}) + \epsilon$ for all $(\mathbf{x}, y) \in \mathcal{S}^{tr}$, where $\mathcal{S}^{tr}$ is an arbitrary sample of instances.

- there can be more instances with same attribute values but different labels
- note, that we don't care about where the noise came from
  - non-perfect measuring devices, human factor, etc.

$$noise(\mathbf{x}) = \mathrm{E}_{(\mathbf{x},y)}\{\ (y - l(\mathbf{x}))^2\ \}$$

Noise in sampling

Usually, we assume a normally distributed sampling error $\epsilon \sim \mathcal{N}(0,1)$

- thus, $\mathrm{E}_{(\mathbf{x},y)}\{y\} = l(\mathbf{x})$

Let's rewrite the equations introduces before as

$$bias^2_{m^\Theta}(\mathbf{x}) = (\ \mathrm{E}_{(\mathbf{x},y)}\{y\} - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\}\ )^2$$

$$variance_{m^\Theta}(\mathbf{x}) = \mathrm{E}_{\mathcal{S}^{tr}}\{\ (\ \hat{y} - \mathrm{E}_{\mathcal{S}^{tr}}\{\hat{y}\}\ )^2\}$$

$$noise(\mathbf{x}) = \mathrm{E}_{(\mathbf{x},y)}\{\ (y - \mathrm{E}_{(\mathbf{x},y)}\{y\})^2\ \}$$

and sum them up

$$\underbrace{bias^2_{m^\Theta}(\mathbf{x}) + variance_{m^\Theta}(\mathbf{x})}_{\mathrm{E}_{\mathcal{S}^{tr}}\{(\mathrm{E}_{(\mathbf{x},y)}\{y\} - \hat{y})^2\}} + noise(\mathbf{x})$$

Expected squared error

$$
\begin{aligned}
&\mathrm{E}_{\mathcal{S}^{tr}}\{(\mathrm{E}_{(\mathbf{x},y)}\{y\} - \hat{y})^2\} + \mathrm{E}_{(\mathbf{x},y)}\{(y - \mathrm{E}_{(\mathbf{x},y)}\{y\})^2\} \\
&= \mathrm{E}_{(\mathbf{x},y)}\{(y - \mathrm{E}_{(\mathbf{x},y)}\{y\})^2\} + \mathrm{E}_{\mathcal{S}^{tr}}\{(\mathrm{E}_{(\mathbf{x},y)}\{y\} - \hat{y})^2\} \\
&\quad + \mathrm{E}_{\mathcal{S}^{tr}}\{2 \cdot (\mathrm{E}_{(\mathbf{x},y)}\{y\} - \mathrm{E}_{(\mathbf{x},y)}\{y\})(\mathrm{E}_{(\mathbf{x},y)}\{y\} - \hat{y})\} \\
&= \mathrm{E}_{\mathcal{S}^{tr}}\{\mathrm{E}_{(\mathbf{x},y)}\{(y - \mathrm{E}_{(\mathbf{x},y)}\{y\})^2\}\} + \mathrm{E}_{\mathcal{S}^{tr}}\{\mathrm{E}_{(\mathbf{x},y)}\{(\mathrm{E}_{(\mathbf{x},y)}\{y\} - \hat{y})^2\} \\
&\quad + \mathrm{E}_{\mathcal{S}^{tr}}\{\mathrm{E}_{(\mathbf{x},y)}\{2 \cdot (y - \mathrm{E}_{(\mathbf{x},y)}\{y\})(\mathrm{E}_{(\mathbf{x},y)}\{y\} - \hat{y})\}\} \\
&= \mathrm{E}_{\mathcal{S}^{tr}}\{\mathrm{E}_{(\mathbf{x},y)}\{(y - \mathrm{E}_{(\mathbf{x},y)}\{y\} + \mathrm{E}_{(\mathbf{x},y)}\{y\} - \hat{y})^2\}\} \\
&= \mathrm{E}_{\mathcal{S}^{tr}}\{\mathrm{E}_{(\mathbf{x},y)}\{(y - \hat{y})^2\}\}
\end{aligned}
$$

We get the expected squared error of the model over all training samples and all instances w.r.t. the observed labeling.

- known labels for observed instances

Test set, RMSE and MAE

In practice, we train a model m^{Θ} on a train set $\mathcal{S}^{tr}$ and test its error on a so-called **test sample** $\mathcal{S}^{te}$ defined as

$$\mathcal{S}^{te} \subset \mathcal{X} \times \mathcal{L} \setminus \mathcal{S}^{tr}$$

Root mean squared error (regression)

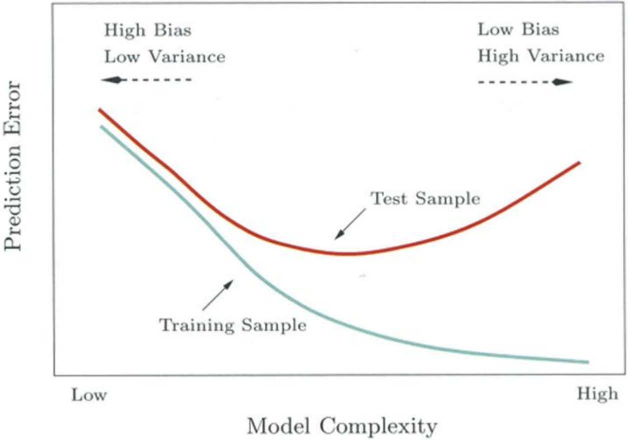
$$rmse(m^{\Theta,\mathcal{S}^{tr}}(\mathbf{x}), \mathcal{S}^{te}) = \sqrt{\frac{\sum_{(\mathbf{x},y)\in\mathcal{S}^{te}} (m^{\Theta,\mathcal{S}^{tr}}(\mathbf{x}) - y)^2}{|\mathcal{S}^{te}|}}$$

Mean absolute error (classification)

$$mae(m^{\Theta,\mathcal{S}^{tr}}(\mathbf{x}), \mathcal{S}^{te}) = \frac{\sum_{(\mathbf{x},y)\in\mathcal{S}^{te}} I(m^{\Theta,\mathcal{S}^{tr}}(\mathbf{x}) \neq y)}{|\mathcal{S}^{te}|}$$

where $I(\cdot) = 1$ if the condition $(\cdot)$ holds, otherwise $I(\cdot) = 0$.

Bias-variance, test set, train set, . . .



[1]

[1] image from Google images.

Cross-validation

A small complication: As usual, we have only one training and one test set on the input! Moreover, the labels of instances in the test set are "hidden"[1] to the model.

- Question: How can we get the model with the least expected error?
  - which means evaluating over all training samples and all instances...
- Answer: Try to simulate learning over "more" training sets and "more" instances.
  - which means creating more training sets (with lower sizes) from the original one...
  - this process is called cross-validation

[1] The test set should be usually used for the final evaluation of the model but not for tuning it (selection of a best technique or good parameters, etc.)

k-fold Cross-validation

One possible alternative[1]:

1. Split (systematically or randomly) the training sample $\mathcal{S}^{tr}$ to k parts of similar size
$$\mathcal{S}^{tr} = \bigcup_k \mathcal{S}^{tr}_k$$
2. choose those hyper-parameters Ξ such that[2]
$$\Xi = \underset{m^{\Xi}}{arg\,min} \left\{ \frac{1}{k} \sum_{i=1}^{k} err(m^{\Theta,\Xi,\bigcup_{1\leq j\leq k, j\neq i} \mathcal{S}^{tr}_j}, \mathcal{S}^{tr}_i) \right\}$$
   - $\mathcal{S}^{tr}_i$ is called **validation fold**.
3. "re-learn" the final m^{Θ} using Ξ on the whole training set $\mathcal{S}^{tr}$

[1] Ξ denotes the hyper-parameters of the model.

[2] $m^{\Theta,\Xi,\bigcup_{1\leq j\leq k, j\neq i} \mathcal{S}^{tr}_j}$ denotes a model whose parameters Θ were learned using hyper-parameters Ξ on the sample $\bigcup_{1\leq j\leq k, j\neq i} \mathcal{S}^{tr}_j$.

Bayes Classifier

Let's have $C_1, \dots, C_K$ mutually exclusive and exhaustive classes

prior probability $P(C_i)$

- probability that an arbitrary instance is labeled with class C_i

likelihood $P(\mathbf{x}|C_i)$

- probability that an arbitrary instance belonging to class C_i is associated with the instance $\mathbf{x}$

evidence $P(\mathbf{x})$

- probability that the instance $\mathbf{x}$ is seen regardless of its class

posterior probability $P(C_i|\mathbf{x})$

- probability that the instance $\mathbf{x}$ is labeled with class C_i

$$P(C_i|\mathbf{x}) = \frac{P(\mathbf{x}|C_i)P(C_i)}{P(\mathbf{x})}$$

for $\mathbf{x}$ **predict** C_i for which $P(C_i|\mathbf{x})$ is maximal

Discriminant function

in case of K classes, classification can be seen as an implementation of K discriminant functions $g_1(\mathbf{x}), \ldots, g_K(\mathbf{x})$ such that

- for $\mathbf{x}$ **predict** C_i for which $g_i(\mathbf{x})$ is maximal

binary classification

- $K = 2$, i.e. labels of instances belong to $\mathcal{L} = \{0, 1\}$
- e.g. $g_1(\mathbf{x}) = P(\mathbf{x}|C_1)P(C_1)$ and $g_2(\mathbf{x}) = P(\mathbf{x}|C_2)P(C_2)$
- a single discriminant is enough

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x})$$

- for $\mathbf{x}$ predict C_1 if $g(\mathbf{x}) > 0$, and predict C_2 if $g(\mathbf{x}) < 0$

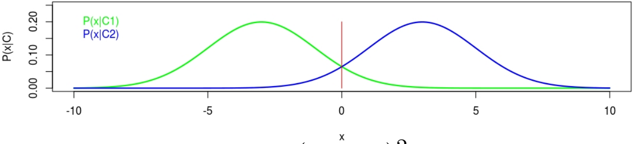
decision boundary

- separates the feature space into **decision regions**
- $g(\mathbf{x}) = 0$ for any $\mathbf{x}$ lying on the decision boundary

Example

one dimensional feature space, two classes

- assume equal priors, i.e. $P(C_1) = P(C_2)$
- assume normal likelihoods, i.e. $P(\mathbf{x}|C_i) = \mathcal{N}(\mu_i, \sigma_i^2)$
    - assume equal standard deviations, i.e. $\sigma_1^2 = \sigma_2^2$
- $g_i(\mathbf{x}) = logP(\mathbf{x}|C_i) + logP(C_i)$



$$g_i(\mathbf{x}) = \underbrace{-\frac{1}{2}log\ 2\pi}_{\text{constant}} \underbrace{-log\ \sigma_i}_{\text{equal variances}} - \frac{(x-\mu_i)^2}{\underbrace{2\sigma_i^2}_{\text{equal variances}}} \underbrace{+logP(C_i)}_{\text{equal priors}} = -(x-\mu_i)^2$$

- we use the estimates m_i for μ_i, i.e. $g_i(\mathbf{x}) = -(x - m_i)^2$
    - assign $\mathbf{x}$ to the class C_i with the nearest mean m_i
- decision boundary, where $g_1(\mathbf{x}) = g_2(\mathbf{x})$, i.e. $\mathbf{x} = \frac{m_1+m_2}{2}$

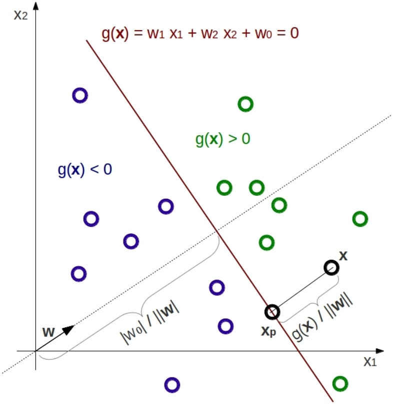
Linear classifier

linear discriminant function – a hyperplane

- $g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x}) = (\mathbf{w}_1^T\mathbf{x} + w_{10}) - (\mathbf{w}_2^T\mathbf{x} + w_{20})$

$$g(\mathbf{x}) = (\mathbf{w}_1 - \mathbf{w}_2)^T\mathbf{x} + (w_{10} - w_{20}) = \mathbf{w}^T\mathbf{x} + w_0$$

- assign $\mathbf{x}$ to the class C_1 if $g(\mathbf{x}) > 0$, and to C_2 if $g(\mathbf{x}) < 0$



Linear classifier – properties

Let $\mathbf{x}_1, \mathbf{x}_2$ be two points on the hyperplane

- $g(\mathbf{x}_1) = \mathbf{w}^T\mathbf{x}_1 + w_0 = 0 = \mathbf{w}^T\mathbf{x}_2 + w_0 = g(\mathbf{x}_2) \Longrightarrow \mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$
    - $\mathbf{w}$ is orthogonal to the hyperplane, i.e. **defines** its **direction**

Let $\mathbf{x}_p$ be the projection of $\mathbf{x}$ on the hyperplane, i.e. $g(\mathbf{x}_p) = 0$

- $\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{||\mathbf{w}||}$, where r is the distance of $\mathbf{x}$ from the hyperplane
    - $g(\mathbf{x}) = g(\mathbf{x}_p + r\frac{\mathbf{w}}{||\mathbf{w}||}) = \mathbf{w}^T(\mathbf{x}_p + r\frac{\mathbf{w}}{||\mathbf{w}||}) + w_0 = \underbrace{\mathbf{w}^T\mathbf{x}_p + w_0}_{g(\mathbf{x}_p)=0} + r\frac{\mathbf{w}^T\mathbf{w}}{||\mathbf{w}||}$
    - $g(\mathbf{x}) = r||\mathbf{w}|| \Longrightarrow r = \frac{g(\mathbf{x})}{||\mathbf{w}||}$

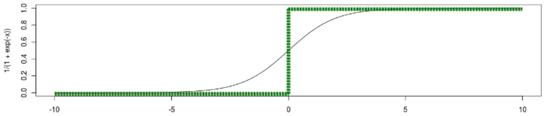
Let $\mathbf{x} = \mathbf{0}$

- $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \Longrightarrow \underbrace{\frac{g(\mathbf{x})}{||\mathbf{w}||}}_{r_\mathbf{0}} = \underbrace{\frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||}}_{0} + \frac{w_0}{||\mathbf{w}||} \Longrightarrow r_\mathbf{0} = \frac{w_0}{||\mathbf{w}||}$
    - w_0 **defines** the **distance** of the hyperplane from the origin

Logistic regression (1/2)

Can be the probability $P(C = 1|\mathbf{x})$ approximated by a linear function?

- $P(C = 0|\mathbf{x}) = 1 - P(C = 1|\mathbf{x})$ in a binary case
- find parameters $\mathbf{w}, w_0$ such that $P(C = 1|\mathbf{x}) = (\mathbf{w}^T\mathbf{x} + w_0) + \epsilon$
    - problem: a simple regression model can predict values outside the interval[1] $[0, 1]$
        - solution: use a **sigmoid logistic function** $s(t) = \frac{1}{1+e^{-t}}$



- **logistic regression model**

$$P(C = 1|\mathbf{x}) = s(\mathbf{w}^T\mathbf{x} + w_0) + \epsilon = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x} + w_0)}} + \epsilon$$

[1] Probabilities should lie between 0 and 1.

Logistic regression (2/2)

Maximum lilelihood estimate

- instances $(\mathbf{x}_i, c_i) \in \mathcal{S}^{tr}$ in a training set, where $c_i \in \{0, 1\}$
- Bernoulli distribution for binary targets
- conditional likelihood with $\Theta = (\mathbf{w}, w_o)$

$$L^{cond}_{\mathcal{S}^{tr}}(\Theta) = \prod_{(\mathbf{x}_i, c_i) \in \mathcal{S}^{tr}} p(C = c_i | \mathbf{x}_i) =$$

$$= \prod_{(\mathbf{x}_i, c_i) \in \mathcal{S}^{tr}} p(C = 1 | \mathbf{x}_i)^{c_i} \left(1 - p(C = 1 | \mathbf{x}_i)\right)^{(1 - c_i)}$$

- conditional log-likelihood $ln\, L^{cond}_{\mathcal{S}^{tr}}(\Theta)$ to maximize is

$$\sum_{(\mathbf{x}_i, c_i) \in \mathcal{S}^{tr}} \left(c_i\, ln\Big(\frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + w_0)}}\Big) + (1 - c_i)\, ln\Big(1 - \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + w_0)}}\Big) \right)$$

Thanks for your attention

Questions?

Tomas.Horvath@upjs.sk

http://www.ics.upjs.sk/~horvath