Problems in ML & Performance Evaluation

Lương Thái Lê

Outline of the Lecture

1. **Learning Machine**
2. **Problems in ML**
   - **Emprical Risk Minimize**
   - **Feature Engineering**
   - **Over fitting**
3. **ML model Performance Evaluation**
4. **Evaluation metrics**

Learning Machine

- A learning machine capable of implementing a set of functions

$$f(x, \mathrm{w}), w \in \Omega$$

- The learning problem is to choose from the given set of functions the one which best approximates the supervisor's response.
    - The selection is based on training samples $(x_i, y_i), i = 1, \dots, l$
    - => Need to choose and optimize (depend on concrete problem)

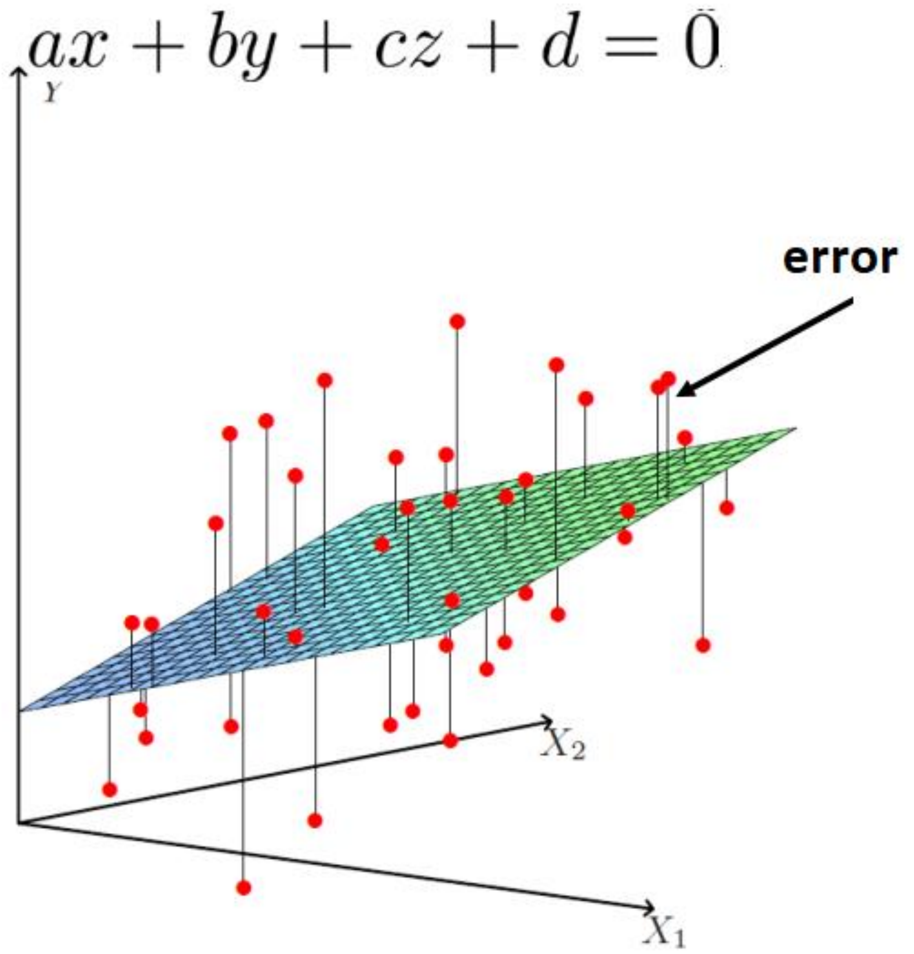
Outline of the Lecture

1. Learning Machine
2. **Problems in ML**
   - **Emprical Risk Minimize**
   - **Feature Engineering**
   - **Over fitting**
3. **ML model Performance Evaluation**
4. **Evaluation metrics**

Regression Example: Find road surface

- Given the points, estimate parameters

- Data/Feature
  - dimension (p=2) $x_i = (x_{i1}, x_{i2}, \dots x_{ip})^T$
  - # training samples $(x_1, y_1) \dots (x_l, y_l)$
  - parameters $\alpha = (\alpha_0, \alpha_1, \dots \alpha_p)^T$
- Evaluation Metric
  - How good is the fitted plane?



Loss Function

- To chose the best function, it makes sense to **minimize** a loss between the response of the supervisor and the learning machine, given an input *x*

$$L\big(y, f(x, \alpha)\big)$$

- Since we want to minimize the loss over *all* samples, we are interested in minimizing the *expected* loss

$$R(\alpha) = \int L\big(y, f(x, \alpha)\big) dF(x, y)$$

  - $R(\alpha)$ is called Risk function
  - $F(x, y)$ is the joint probability distribution function

=> Find $f(x, \alpha^*)$ that minimize $R(\alpha)$ with the only available information is contained in the training set: $(x_i, y_i),\ i = 1, \dots, l$

Empirical Risk Minimization Principle

$$R(\alpha) = \int L\big(y, f(x, \alpha)\big) dF(x, y)$$

- The risk functional is replaced by the *empirical risk functional*

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} L\big(y_i, f(x_i, \alpha)\big)$$

=> find $f(x, \alpha *)$ that minimize $R(\alpha)$ over class of function $f(x, \alpha)$

Loss function: A Probabilistic View

- $L\big(y, f(x,\alpha)\big) = \sum_{i=1}^{l}(y_i - f(x_i,\alpha)) \qquad = \sum_{i=1}^{l}(y_i - \alpha_0 - \sum_{j=1}^{p} x_{ij}\alpha_j)$
- Let the noise $||\varepsilon|| = ||L\big(y, f(x,\alpha)\big)||$
- If we model the noise as zero mean Gaussian random variable with variance δ^2, the distribution is:

$$p(\varepsilon_i) = \frac{1}{\sqrt{2\pi}\delta}\exp\left(-\frac{{\varepsilon_i}^2}{2\delta^2}\right) = \frac{1}{\sqrt{2\pi}\delta}\exp\left(-\frac{(y_i - \alpha_0 - \sum_{j=1}^{p} x_{ij}\alpha_j)^2}{2\delta^2}\right)$$

$$= \frac{1}{\sqrt{2\pi}\delta}\exp\left(-\frac{(y_i - \alpha^T x_i)^2}{2\delta^2}\right)$$

$$\Rightarrow p(\varepsilon) = \prod_1^l p(\varepsilon_i) = \frac{1}{(\sqrt{2\pi}\delta)^l}\exp -\frac{1}{2}\left(\frac{\sum_{i=1}^{l}(y_i - \alpha^T x_i)^2}{\delta^2}\right)$$

Likelihood function

$$\Rightarrow p(\varepsilon) = p(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_l) = \frac{1}{(\sqrt{2\pi}\delta)^l} \exp -\frac{1}{2}\left(\frac{\|y-\alpha^T x\|^2}{\delta^2}\right)$$

We can view this joint probability as a function of the parameters

$$\boxed{L(\alpha) = p(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_l|\alpha) = \frac{1}{(\sqrt{2\pi}\delta)^l} \exp -\frac{1}{2}\left(\frac{\|y-\alpha^T x\|}{\delta}\right)^2}$$

=> Need to maximum Likelihood function

Maximum Likelihood Function

- Maximize the likelihood over all available samples

$$\alpha^* = argmax_\alpha \, p(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_l | \alpha) = argmax_\alpha \frac{1}{(\sqrt{2\pi}\delta)^l} \exp -\frac{1}{2}\left(\frac{\|y-\alpha^T x\|}{\delta}\right)^2$$

- Since log is a monotonic function, often log-likelihood is used:

$$\alpha^* = argmax_\alpha \left(-\sum_{i=1}^{l} (y_i - \alpha^T x_i)^2 + const\right)$$

=> That is where Gradient Descent comes in

Outline of the Lecture

1. Learning Machine
2. Problems in ML
   - Emprical Risk Minimize
   - **Feature Engineering**
   - **Over fitting**
3. **ML model Performance Evaluation**
4. **Evaluation metrics**

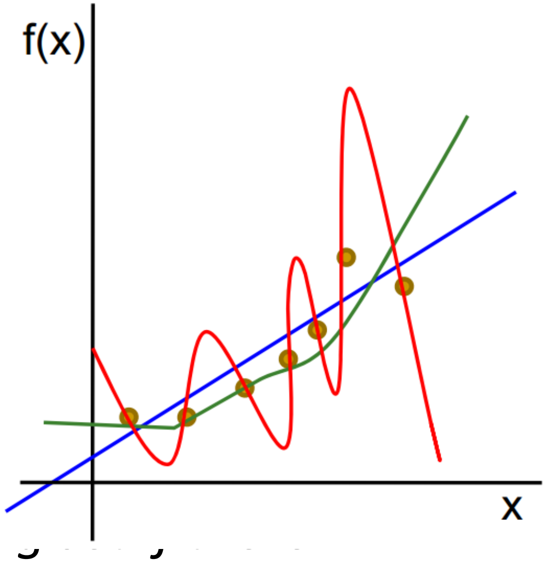
Feature Engineering

- Features are individual independent variables that act as an input in your system. In simpler feature is a column of data in your input dataset. Ex: Age, Sex, Income…
- Two types of feature:
  - Categorical: has little values, such as: Sex, Class of ticket (Economy, Premium…), Color,…
  - Numerical: has continuous/discrete values: Age, Price, Name…
- Can create a new feature from a root feature to improve learning
  - Name: Mr John May => create new feature: Tittle (Mr, Miss,…)
- Group features with few values into a generic attribute
  - Tittle with few values, Ex: Rev, Dr, Capital… => Others

Outline of the Lecture

1. Learning Machine
2. Problems in ML
   - Emprical Risk Minimize
   - Feature Engineering
   - **Over fitting**
3. **ML model Performance Evaluation**
4. **Evaluation metrics**

Over fitting



- **Definition**

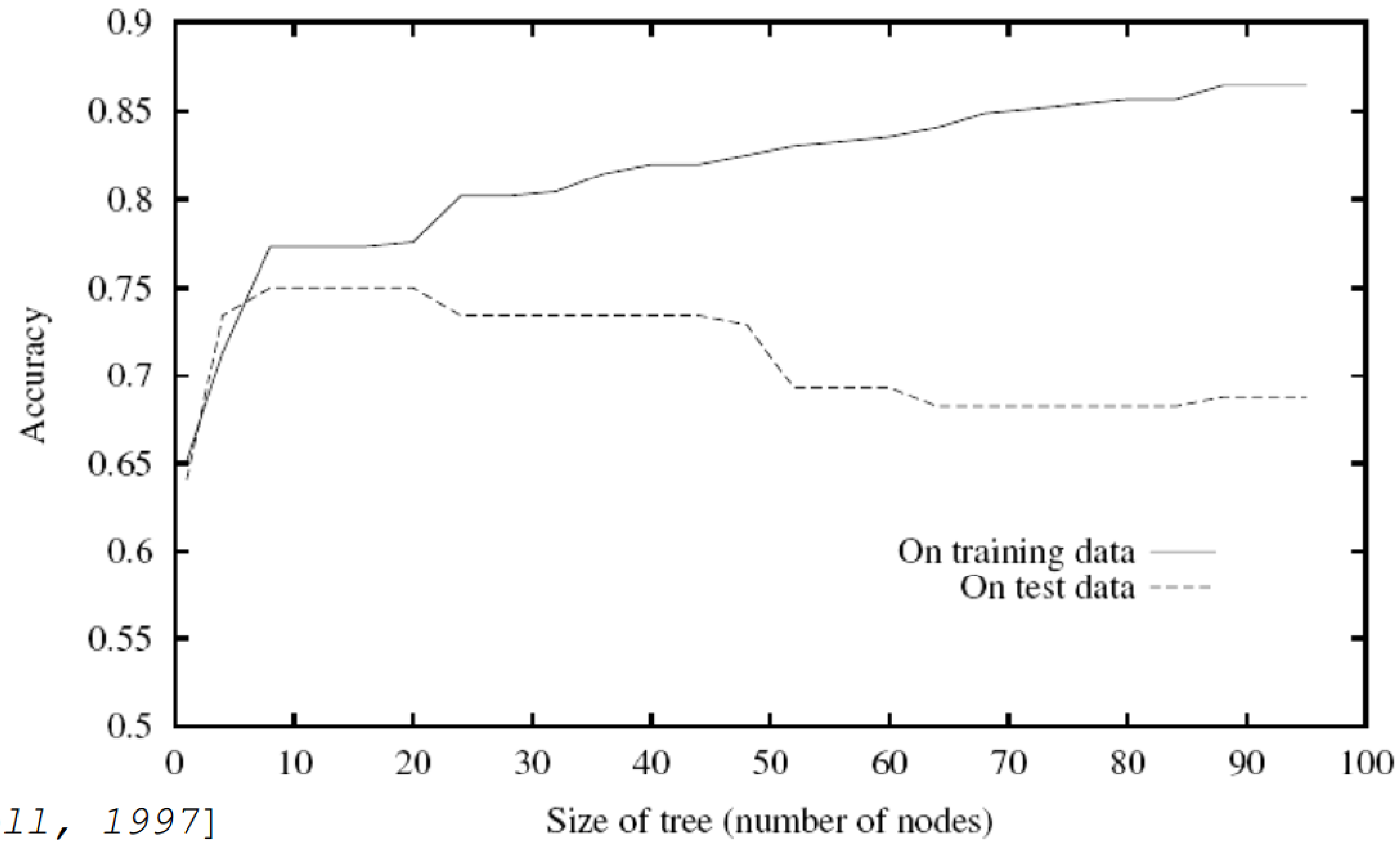
  An objective function that be learnt F will be said to overfit a learni
  exists another objective function F' such that:

  - *F' is less suitable (gain less accurate) than F for the training set, but*
  - *F' is more accurate than F for the entire dataset (including examples used in future)*

- **Reasons of Over-fitting**:
  - Error (noise) in the training set (due to the colletion/construction data process)
  - The number of learning examples is too small to represent the entire examples set of the problems

=>Preferably choose the simplest objective function that fits (nonecessarily perfect) with training examples

An Over-fitting Example



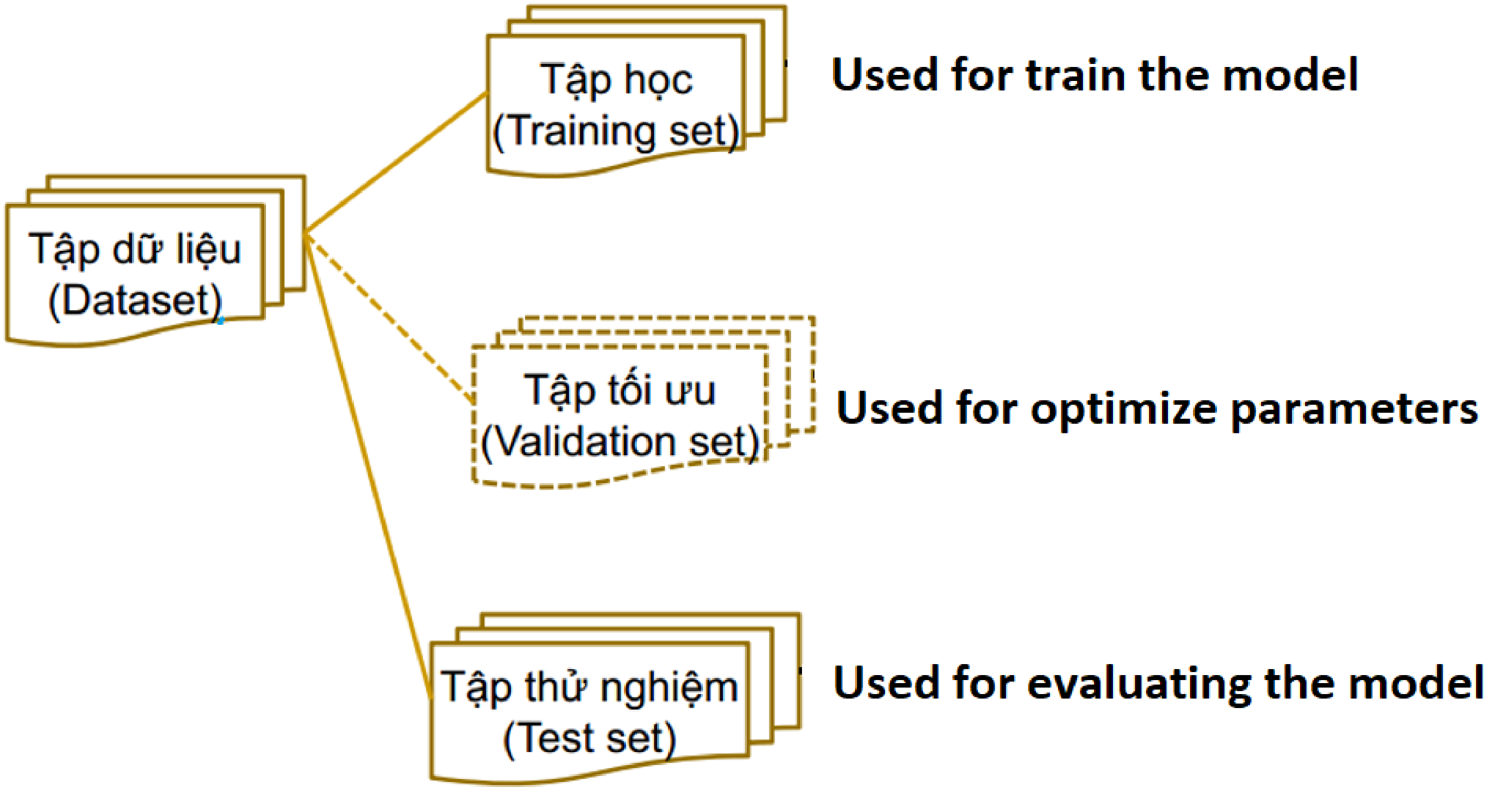
[Mitchell, 1997]

Outline of the Lecture

1. Learning Machine
2. Problems in ML
   - Emprical Risk Minimize
   - Feature Engineering
   - Over fitting
3. **ML model Performance Evaluation**
4. **Evaluation metrics**

The Model Performance evaluation (1)

- Evaluation of machine learning system performance is often perform empirically, rather than analytically.



The Model Performance evaluation (2)

- The performance of system depends not only on the machine learning algorithms are used, but also depends on:
  - Class distribution
  - Cost of misclassification
  - Size of the training set
  - Size of the test set
- How to obtain a reliable assessment of system performance?
  - The larger the training set, the better the performance of the learning system
  - The larger the test set, the more accurate the evaluation
  - Problem: It is very difficult (rarely) to obtain (very) large data sets

Evaluation Methods

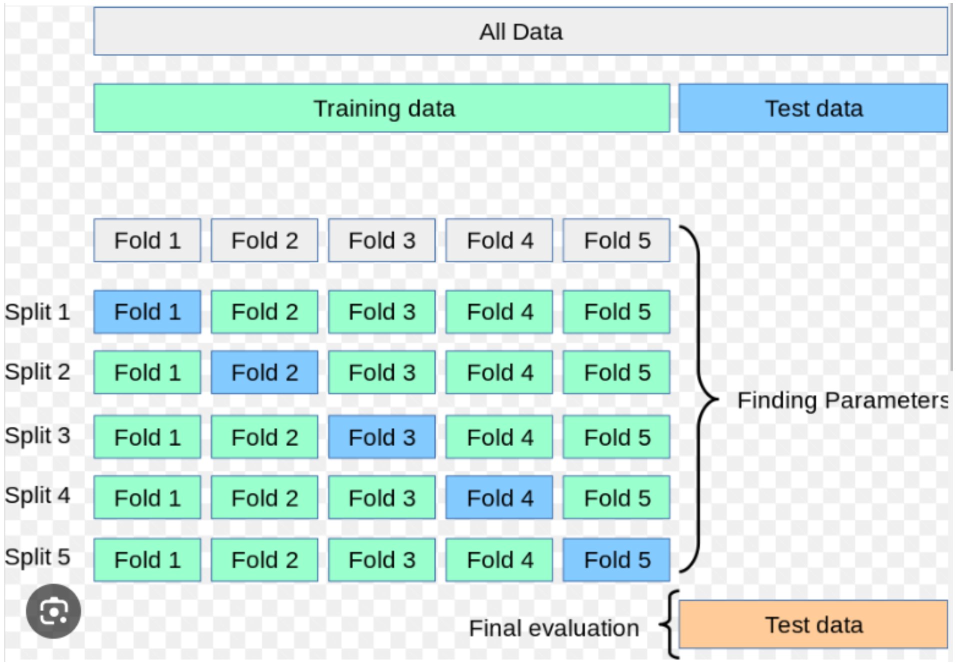
- Hold-out
- Stratified sampling
- Repeated hold-out
- Cross-validation
  - *k*-fold
  - Leave-one-out
- Bootstrap sampling

Hold-out

- Data set is devided into 2 parts:
  - Training set: D_train
  - Test set: D_test
- Requirements:
  - Any example in D_test is not used in training process
  - Any example in D_train is not used in the model evaluation process
- Popular: |D_train|=2/3.|D|; |D_test|=1/3.|D|
- Suitable for large set of examples D

Cross Validation – k fold

- The entire set of examples D is divided into k non-intersecting subsets (referred to as "fold") of approximately the same size
- Each time (of k) iterations, a subset is used as the D_test, and (k-1) the remaining subset is used as the D_train.
- k error values (each corresponding to a fold) are averaged to get the overall error value
- Popular: k= 10; or k=5



Bootstrap Sampling

- Bootstrap sampling method uses repeated sampling to create a training set
  - Suppose the whole set D consists of n examples
    - From the set D, randomly select an example x (but do not remove x from D)
    - Put example x in the training set: $D\_train = D\_train \cup x$
    - Repeat the above 2 steps n times
- Use D_train to train the model
- $D\_test = \{z \in D;\ z \notin D\_train\}$ *used for test the model*

Evaluation Criteria

- Accuracy
  - Predictability (classification) of the (trained) model with respect to test instances
- Efficiency
  - Cost of time and resources (memory) required for model training and testing
- Robustness
  - The system's ability to handle (tolerable) noise (error) or missing value
- Scalability
  - How does system performance (e.g. learning/classification rate) change with respect to the size of the data set?
- Interpretability
  - Understanding (for the user) of the system's results and operations easily
- Complexity
  - The complexity of the system model (objective function) learned

Outline of the Lecture

1. Learning Machine
2. Problems in ML
   - Emprical Risk Minimize
   - Feature Engineering
   - Over fitting
3. ML model Performance Evaluation
4. **Evaluation metrics**

Accuracy

- Show the accuracy of the model when solving the problem
- For the classification problem:

$$Accuracy = \frac{1}{|D\_test|} \sum_{x \in D\_test} id(m(x)r(x)) \qquad id(a,b) = \begin{cases} 1 & if\ a = b \\ 0 & otherwise \end{cases}$$

  - m(x) is the class that the model predict for example x
  - r(x) is the real class
- For the regression problem:

$$Error = \frac{1}{|D\_test|} \sum_{x \in D\_test} |m(x) - r(x)|$$

  - m(x) is the prediction of the model for x
  - r(x) is the real output of x

Confuse Matrix (contingency table)

- Only use for classification problem
- **TP**: Number of examples belonging to class c correctly classified into class c
- **TN**: Number of examples that do not belong to class c that be determined exactly
- **FP**: Number of examples that are not in class c isclassified in class c
- **FN**: Number of examples belong to class c but be classified in other class

For class c		**Predicted**	
		Negative (**N**) -	Positive (**P**) +
Actual	Negative -	True Negatives (**TN**)	False Positives (**FP**) **Type I error**
	Positive +	False Negatives (**FN**) **Type II error**	True Positives (**TP**)

Precision and Recall for each class c

- Often used in text classification
- Precision for class c_i:

  The total number of examples in class c_i correctly classified divided by the total number of examples classified in class c_i by the model

$$\Pr ecision(c_i) = \frac{TP_i}{TP_i + FP_i}$$

- Recall for class c_i:

  The total number of examples correctly classified by the model into class c_i divided by the total number of examples actually in class c_i

$$\operatorname{Re} call(c_i) = \frac{TP_i}{TP_i + FN_i}$$

Precision and Recall for over all classes

- Assume that the model classifies data into a set of classes $C = \{c_i\}_{i=1}^{n}$ and, we get TP_i , TN_i , FP_i , FN_i for each c_i
- Micro - averaging:

$$\mathrm{Pr}\,ecision = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FP_i)} \qquad \mathrm{Re}\,call = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FN_i)}$$

- Macro - averaging:

$$\mathrm{Pr}\,ecision = \frac{\sum_{i=1}^{|C|} \mathrm{Pr}\,ecision(c_i)}{|C|} \qquad \mathrm{Re}\,call = \frac{\sum_{i=1}^{|C|} \mathrm{Re}\,call(c_i)}{|C|}$$

F_1

- F_1 is a harmonic mean of Precision and Recall

$$F_1 = \frac{2.\mathrm{Pr}\,ecision.\mathrm{Re}\,call}{\mathrm{Pr}\,ecision + \mathrm{Re}\,call} = \frac{2}{\dfrac{1}{\mathrm{Pr}\,ecision} + \dfrac{1}{\mathrm{Re}\,call}}$$

- F_1 tends to take the closest value, whichever is the smaller of the two Precision and Recall values
- F_1 có giá trị lớn nếu cả 2 giá trị Precision và Recall đều lớn

Q&A - Thank you!