Introduction to

Machine Learning and Data Mining

(Học máy và Khai phá dữ liệu)

Khoat Than

School of Information and Communication Technology

Hanoi University of Science and Technology

2020

Contents

1. Assessing performance (1)

- *How can we make a reliable assessment on the performance of an ML method?*
  - Note that performance of a method often improves as more data are available.
  - An assessment is more reliable as more data are used to test prediction.
- *How to choose a good value for a parameter in an ML method?*
- The performance of a method depends on many factors:
  - Class distribution
  - Training size
  - Representativeness of training data over the whole space,…

Assessing performance (2)

- *Theoretical evaluation:* study some theoretical properties of a method/model with some explicit mathematical proofs.
  - Learning rate?
  - How many training instances are enough?
  - What is the expected accuracy of prediction?
  - Noise-resistance? …
- *Experimental evaluation:* observe the performance of a method in practical situations, using some datasets and a performance measure. Then make a summary from those experiments.
- We will discuss experimental evaluation in this lecture.

Assessing performance (3)

- ***Model assessment:*** *we need to evaluate the performance of a method/model, only based on a given observed dataset D.*
- Evaluation:
  - Should be done automatically,
  - Does not need any help from users.
- Evaluation strategies:
  - To obtain a reliable assessment on performance.
- Evaluation measures:
  - To measure performance quantitatively.

2. Some evaluation techniques

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

Hold-out (random splitting)

- The observed dataset D is randomly splitted into 2 non-overlapping subsets:
  - D_{train}: used for training
  - D_{test}: used to test performance

- Note that:
  - No instance of D_{test} is used in the training phase.
  - No instance of D_{train} is used in the test phase.
- Popular split: $|D_{train}| = (2/3).|D|$, $|D_{test}| = (1/3).|D|$
- This technique is suitable when D is of large size.

Stratified sampling

- For small or imbalanced datasets, random splitting might result in a training dataset which are not representative.
  - A class in D_{train} might be empty or have few instances.
- *We should split D so that the class distribution in D_{train} is similar with that in D.*
- Stratified sampling fulfills this need:
  - *We randomly split each class of D into 2 parts*: one is for D_{train}, and the other is for D_{test}.
  - for each class: D_{train} D_{test}
- Note that this technique cannot be applied to regression and unsupervised learning.

Repeated hold-out

- We can do hold-out many times, and then take the average result.
  - Repeat hold-out n times. The i^{th} time will give a performance result p_i. The training data for each hold-out should be different from each other.
  - Take the average $p = mean(p_1,\ldots, p_n)$ as the final quality.
- Advantages?
- Limitations?

Cross-validation

- In repeated hold-out: there are overlapping between two training/testing datasets. It might be redundant.
- *K-fold cross-validation:*
  - *Split D into K equal parts which are non-overlapping.*
  - *Do K runs (folds): at each run, one part is used for testing and the remaining parts are used for training.*
  - *Take the average as the final quality from K individual runs.*

- Popular choices of K: 10 or 5
- It is useful to combine this technique with stratified sampling.
- This technique is suitable for small/average datasets.

Leave-one-out cross-validation

- It is K-fold cross-validation when K = |D|.
  - Each testing set consists of only one instance from D.
  - The remaining is for training.
- So all observed instances are exploited as much as possible.
- No randomness appears.
- But it is expensive, and hence is suitable with small datasets.

Bootstrap sampling

- Previous methods do not allow repetitions of an instance in any training part.
- Bootstrap sampling:
  - Assume D having n instances.
  - Build D_{train} by randomly sampling (with replacement/repetition) n instances from D.
  - D_{train} is used for the training phase.
  - $D_{test} = D \backslash D_{train}$ is used for testing quality.
  - Note that $D_{test} = \{z \in D: z \notin D_{train}\}$
- It can be shown that D_{train} contains nearly 63.2% different instances of D. 36.8% of D are used for testing.
- This technique is suitable for small datasets.

3. Model selection

- An ML method often has a set of hyperparameters that require us to select suitable values a priori.
  - Ridge regression: λ
  - Linear SVM: C
- How to choose a good value?
- ***Model selection:*** *given a dataset D, we need to choose a good setting of the hyperparameters in method (model) A such that the function learned by A generalizes well.*
- A validation set T_{valid} is often used to find a good setting.
  - It is a subset of D.
  - A good setting should help the learned function predicts well on T_{valid}.

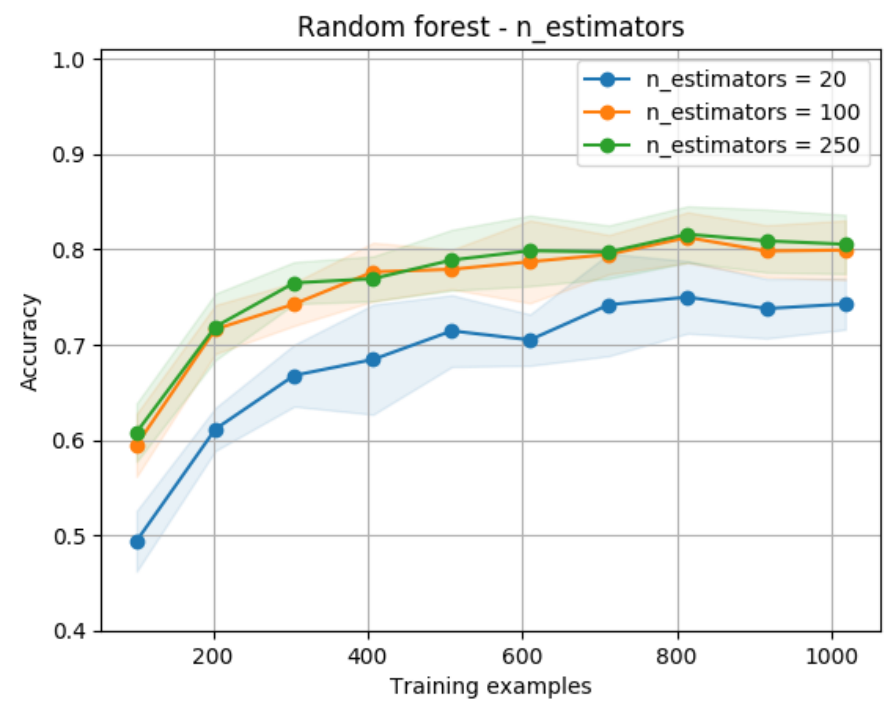
Model selection: using hold-out

- Given an observed dataset D, we can **select** a good value for hyperparamerter λ as follows:
  - *Select a finite set S which contains all potential values of λ.*
  - *Select a performance measure P.*
  - *Randomly split D into 2 non-overlapping subsets: D_{train} and T_{valid}*
  - *For each $\lambda \in S$: train the system given D_{train} and λ. Measure the quality on T_{valid} to get P_λ.*
  - *Select the best λ* which corresponds to the best P_λ.*
- It is often beneficial to learn again from D given λ* to get a better function.
- Hold-out can be replaced with other techniques e.g., sampling, cross-validation.

Example: select parameters

- Random forest for news classification
  - Parameter: n_estimates (number of trees)
- Dataset: *1135 news, 10 classes, vocabulary of 25199 terms*
- 10-fold cross-validation is used

Độc giả
Đời sống - Xã hội
Giải trí
Khoa học - Công nghệ
Kinh tế
Pháp luật
Sức khỏe
Thể thao
Thời sự
Tin khác



4. Model assessment and selection

- Given an observed dataset D, we need to **select** a good value for hyperparamerter λ and **evaluate** the overall performance of a method A:
  - *Select a finite set S which contains all potential values of λ.*
  - *Select a performance measure P.*
  - *Split D into 3 non-overlapping subsets: D_{train}, T_{valid} and T_{test}*
  - *For each $\lambda \in S$: train the system given D_{train} and λ. Measure the quality on T_{valid} to get P_λ.*
  - *Select the best λ* which corresponds to the best P_λ.*
  - Train the system again from $D_{train} \cup T_{valid}$ given λ*.
  - Test performance of the system on T_{test}.
- Hold-out can be replaced with other techniques.

5. Performance measures

- Accuracy:
  - Percentage of correct predictions on testing data.
- Efficiency:
  - The cost in time and storage when learning/prediction.
- Robustness:
  - The ability to reduce possible affects by noises/errors/missings.
- Scalability:
  - The relation between the performance and training size.
- Complexity:
  - The comlexity of the learned function.
- …

Accuracy

- Classification:

$$Accuracy = \frac{number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

- Regression: (MAE – mean absolute error)

$$MAE = \frac{1}{|D_{test}|} \sum_{x \in D_{test}} |o(x) - y(x)|$$

  - o(x) is the prediction for an instance x.
  - y(x) is the true value.

Precision and Recall (1)

- These two measures are often used for classification
- **Precision** for class c_i:
  - Percentage of correct instances, among all that are assigned to c_i.

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

- **Recall** for class c_i:
  - Percentage of instances in c_i that are correctly assigned to c_i.

$$Recall(c_i) = \frac{TP_i}{TP_i + FN_i}$$

- *TP_i*: the number of instances that are assigned correctly to class c_i.
- *FP_i*: the number of instances that are assigned incorrectly to class c_i.
- *FN_i*: the number of instances inside c_i that are assigned incorrectly to another class.
- *TN_i*: the number of instances outside c_i that are not assigned to class c_i.

Precision and Recall (1)

- These two measures are often used in information retrieval and classification
- **Precision** for class c_i:
  - Percentage of correct instances, among all that are assigned to c_i.

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

- **Recall** for class c_i:
  - Percentage of instances in c_i that are correctly assigned to c_i.

$$Recall(c_i) = \frac{TP_i}{TP_i + FN_i}$$

Precision and Recall (2)

- To give an overall summary, we can take an average from individual classes.
- Micro-averaging:

$$Precision = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FP_i)} \qquad Recall = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FN_i)}$$

- Macro-averaging:

$$Precision = \frac{\sum_{i=1}^{|C|} Precision(c_i)}{|C|} \qquad Recall = \frac{\sum_{i=1}^{|C|} Recall(c_i)}{|C|}$$

F_1

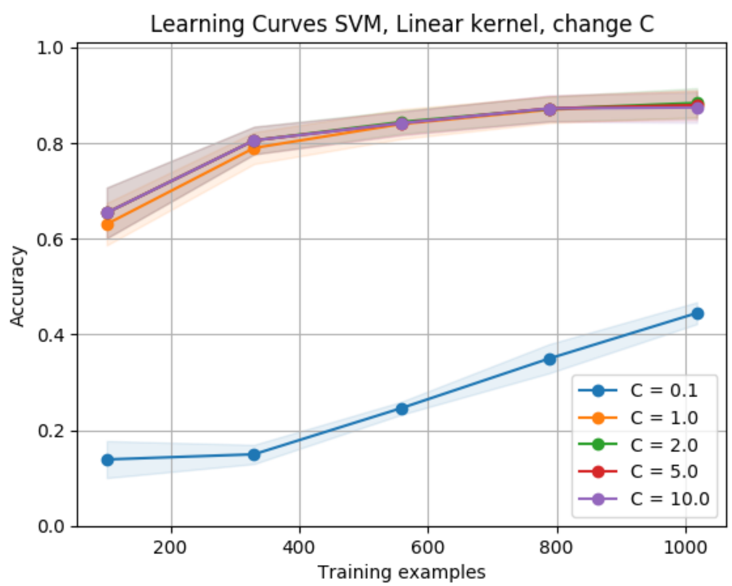
- Precision and recall provide us different views on the performance of a classifier.
- F_1 can provide us a unified view.
- F_1 is the *harmonic mean* of precision and recall, and is computed as:

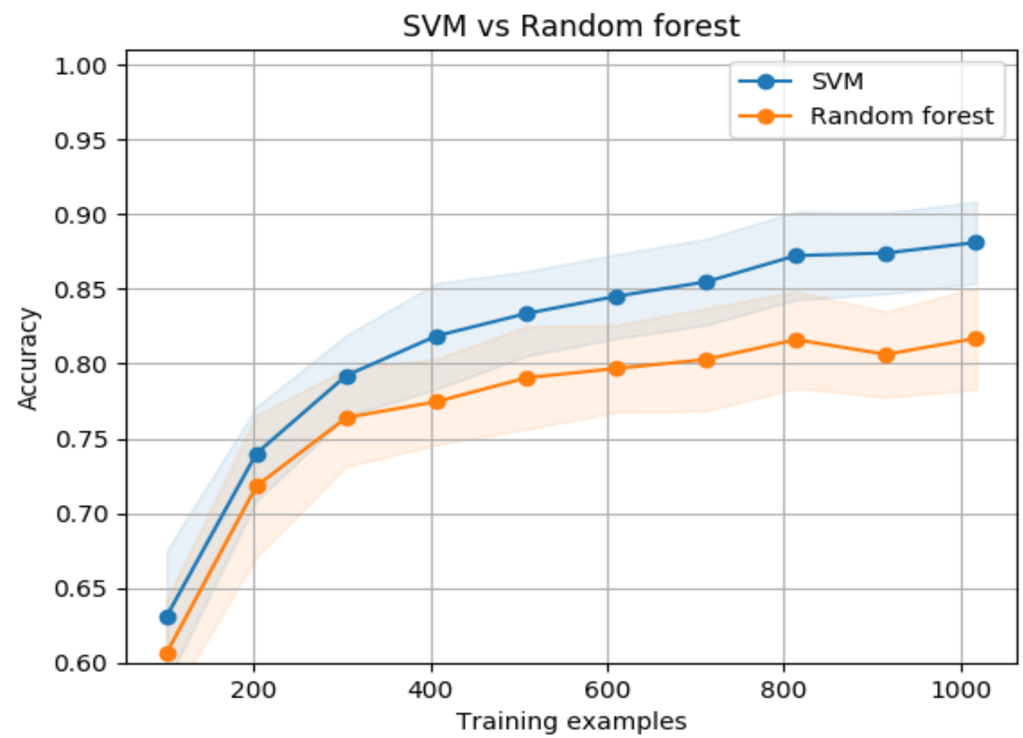
$$F_1 = \frac{2.Precision.Recall}{Precision + Recall} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

  - F_1 tends to be close to the smaller value from {precision, recall}
  - Large F_1 implies that both precision and recall are large.

Example: compare 2 methods

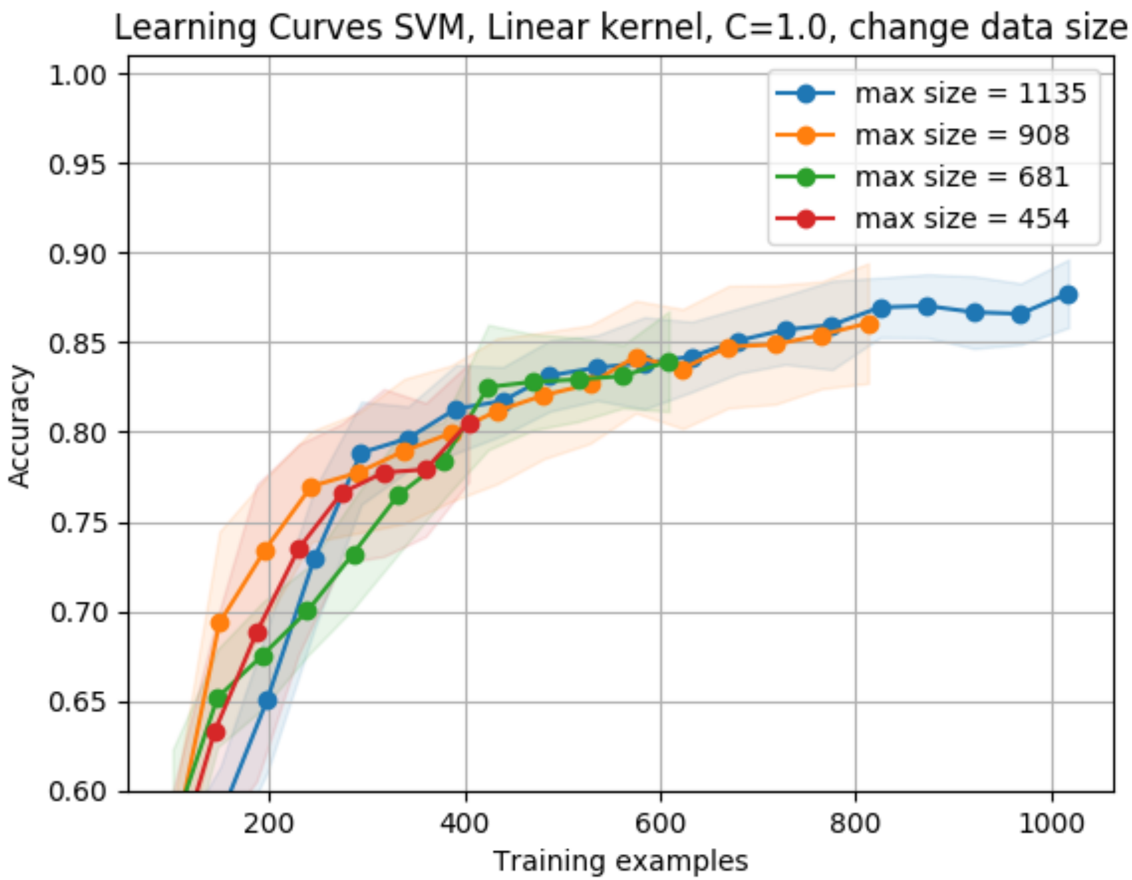
- Methods: **Random forest** vs **Support vector machines** (SVM)
- Parameter selection: 10-fold cross-validation
  - Random forest: n_estimate = 250
  - SVM: regularization constant C = 1





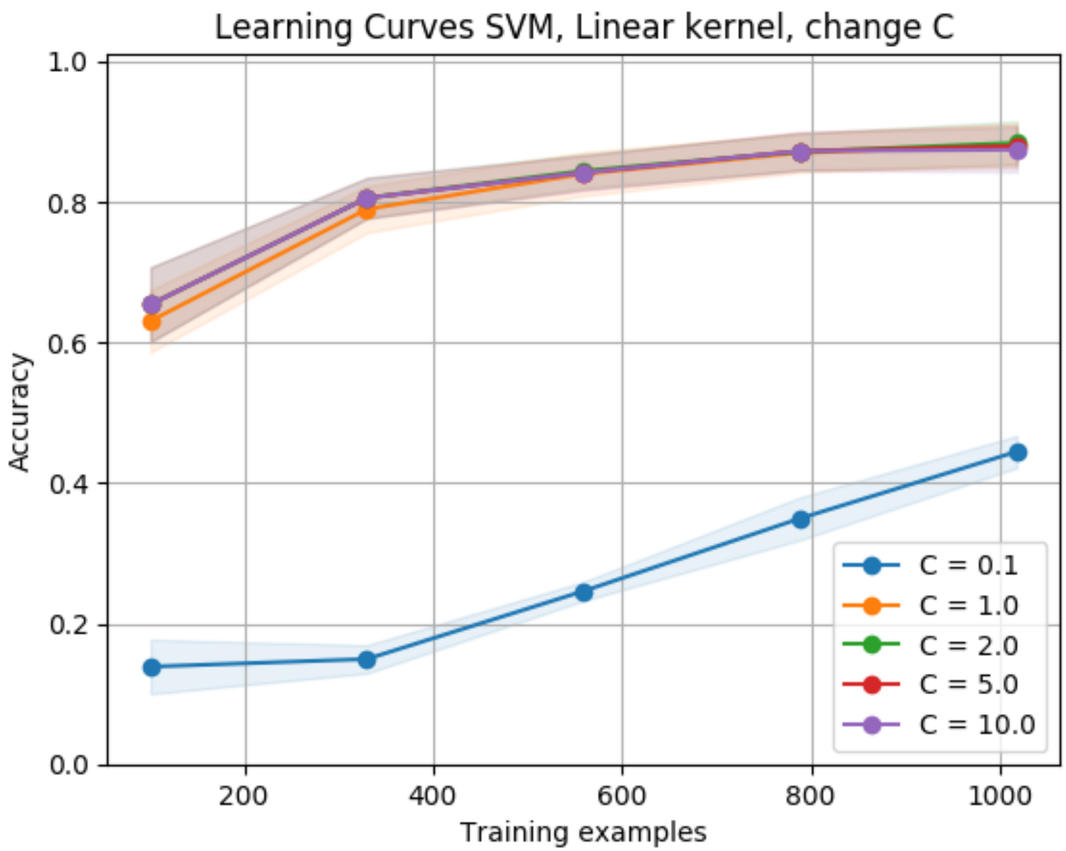
Example: effect of data size

- SVM
  - Parameter: size of training data
- Dataset: *1135 news, 10 classes, vocabulary of 25199 terms*
- 10-fold cross-validation is used



Example: effect of parameters

- SVM for news classification
  - Parameter C changes
- Dataset: *1135 news, 10 classes, vocabulary of 25199 terms*
- 10-fold cross-validation is used



References

- Trevor Hastie, Robert Tibshirani, Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.
- Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, *34*(1), 1-47.