Model Selection & Evaluation

March 25th, 2019

Learnable Theory

How to define learning in ML?

- Several questions arise when designing and analyzing algorithms that learn from data. Examples of questions include:
    1. What can be learned efficiently?
    2. What is inherently hard to learn?
    3. How many examples are needed to learn successfully?
    4. Is there a general model of learning?
- The ***Probably Approximately Correct (PAC)*** learning framework helps defines the class of learnable concepts in terms of ***number of sample points*** needed to achieve an approximate solution, ***sample complexity***, and the time and space complexity of a learning algorithm.

- Let us denote $\mathcal{X}$ the set of all possible examples, $\mathcal{Y}$ the set of all possible label or target values, and that $\mathcal{Y} = \{0, 1\}$
- A concept $c : \mathcal{X} \mapsto \mathcal{Y}$ is a mapping from $\mathcal{X}$ to $\mathcal{Y}$.
- $\mathcal{C}$ is a concept class that comprises the concepts we may wish to learn
- The ***learning problem*** can be formulated as follows:
  - The learner considers a fixed set of all possible concepts $\mathcal{H}$, called *hypothesis set*, with input sample $\mathcal{S} = (x^i, \ldots, x^p)$ draw i.i.d according to $\mathcal{D}$ as well as the labels $c(x_i), \ldots, c(x^p)$ with $c \in \mathcal{C}$
  - The task comprise in using the labeled sample $\mathcal{S}$ to select a hypothesis $h_s \in \mathcal{X}$ that as a small ***generalization error*** with respect to c.
- The generalization error of a hypothesis $h \in \mathcal{H}$ is also known as the ***risk*** or ***true error***.

Generalization error

Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and an underlying distribution $\mathcal{D}$, the generalization error or risk of h is defined by

$$R(h) = \underset{x\sim\mathcal{D}}{\mathbb{P}}[h(x) \neq c(x)] = \underset{x\sim\mathcal{D}}{\mathbb{E}}[1_{h(x)\neq c(x)}]$$

- Since both the distribution $\mathcal{D}$ and the target concept c is unknown, a learner cannot direct access the generalization error. It can only measure the ***empirical error*** of a $h \in \mathcal{H}$ on the labeled sample $\mathcal{S}$

Empirical error

Empirical error

- Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and a sample $S = (x^i, \ldots, x^m)$, the empirical error or empirical risk of h is defined by

$$\hat{R}_s(h) = \frac{1}{m} \sum_{i=1}^{m} 1_{h(x_i) \neq c(x_i)}$$

- The ***empirical error*** of $h \in \mathcal{H}$ is its average error over the sample S, while the ***generalization error*** is its expected error based on the distribution $\mathcal{D}$

PAC-Learning

PAC-learning

- A concept class $\mathcal{C}$ is said to be PAC-learnable if there exists an algorithm $\mathcal{A}$ and a polynomial function $poly(.,.,.,.)$ such that for any $\epsilon > 0$ and $\delta > 0$, for all distributions $\mathcal{D}$ on $\mathcal{X}$ and for any target concept $c \in \mathcal{C}$, the following holds for any sample size $m \geq poly(1/\epsilon, 1/\delta, n, size(c))$

$$\underset{S \sim \mathcal{D}^m}{\mathbb{P}}[R(h_S) \leq \epsilon] \geq 1 - \delta$$

- if $\mathcal{A}$ further runs in $poly(1/\epsilon, 1/\delta, n, size(c))$, the $\mathcal{C}$ is considered to be efficiently PAC-learnable.
- When such $\mathcal{A}$ exists, it is called a *PAC-learning algorithm* for $\mathcal{C}$
- The parameter $\delta > 0$ defines the confidence interval $1 - \epsilon$ and $\epsilon > 0$ the accuracy $1 - \epsilon$.

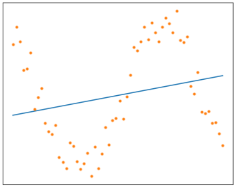
- Machine learning is fundamentally about ***generalization***
- The problem comprises in selecting a function out of a *hypothesis set*, that is a subset of the family of all functions.
- The selected function is subsequently used to label all instances, including unseen examples
- How should a hypothesis set be chosen?
  - With a rich or complex hypothesis set, the learner may choose a predictor that is consistent with the training set
  - With a less complex one, it may have unavoidable errors on the training set
- Which one will lead to a better ***generalization***?
- How should we define the complexity of a hypothesis set?

What is generalization?

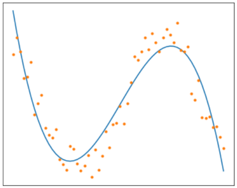
- *It is the ability of a model to adapt properly to* ***unseen data*** *drawn from the same distribution as the one used to create the model*
- Data are noisy, for different reasons
  1. errors during the acquisition phase
  2. errors in labeling the data points (i.e., teaching error)
  3. hidden or latent features
- We learn f by minimizing some variant of empirical risk, what can you say about the true risk?
- Two factors determine generalization ability:
  1. Model complexity
  2. Training set size

Model Complexity & Generalization

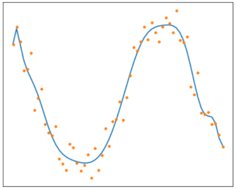
Understanding overfitting & underfitting



$\beta_0 + \beta_1 x$



$\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$



$\beta_0 + \beta_1 x + \cdots + \beta_{15} x^{15}$

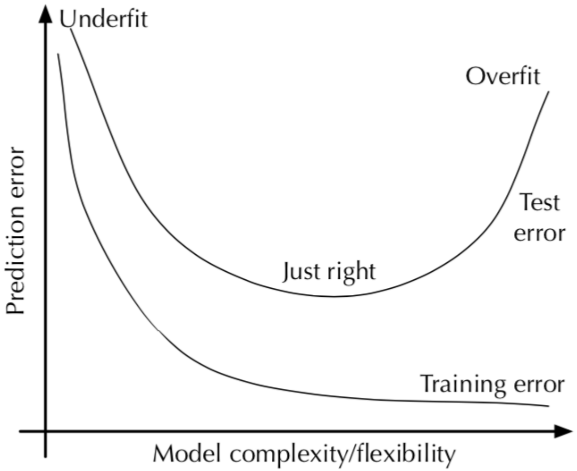
Underfitting $\mapsto$ “**high bias**”

“**Just right**”

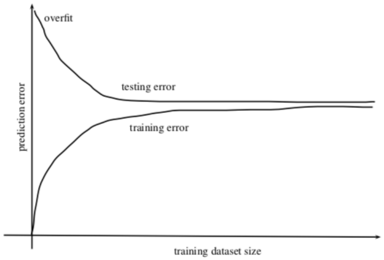
“**overfitting**” $\mapsto$ “**high variance**”

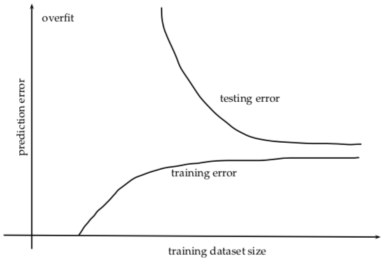
- In the **Overfitting** scenario, the learned hypothesis may fit the training set very well, but fail, but fail to ***generalize*** to new examples.
  - It is usually caused by complicated function that creates various unnecessary curves and angles unrelated to the data
  - It has a large estimation error
- **Underfitting or high bias** occurs when the hypothesis function maps poorly to the trend of the data.
  - It is usually caused by a function that is very simple or that uses only few features
  - It has a large approximate error

Generalization error vs. model complexity trade-off



Fixed model complexity vs. dataset size





Bias vs. variance trade-off

- ***Bias*** is the difference between the expected value of the estimator and the real value predicted by the estimator

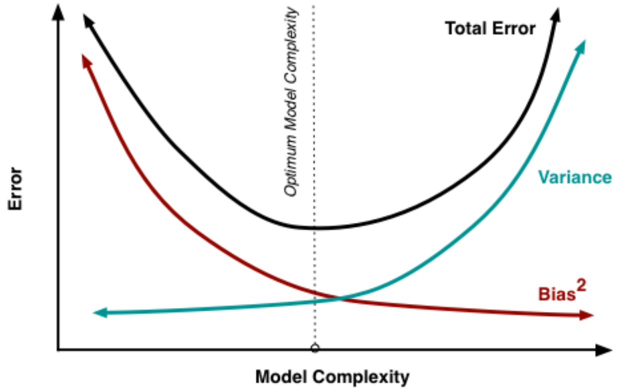
$$Bias(f(x)) = \mathbb{E}[f(x) - y]$$

  - A simple model has a high bias
  - High bias can lead to ***underfitting***
- ***Variance*** is the deviation from the expected value of the estimates

$$Var(f(x)) = \mathbb{E}[(f(x) - \mathbb{E}(f(x)))^2]$$

  - A complex model has a high variance
  - High variance usually leads to ***overfitting***

Bias vs. variance trade-off



Addressing overfitting

1. Reduce the number of features
   - Manually select which feature to keep
   - Model selection algorithm
2. Regularization
   - Keeps all the features, but reduce the magnitude of the parameters
   - It works when there are many features contributing to predict y

Supervised learning assumption

- ***Training set*** $\mathcal{D} = \{x^i, y^i\}_{i=1..n}$
- ***Regression*** $y^i \in \mathbb{R}$
- ***Classification*** $y^i \in \{0, 1\}$
- ***Goal***: find a function f on the training set such that $f(x^i) \approx y^i$
- ***Empirical error*** of f on the training set, given a loss function $\mathcal{L}$

$$\mathbb{E}(f|\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y^i, f(x^i))$$

  - ***Regression***

$$\mathcal{L}(y^i, f(x^i)) = (y^i - f(x^i))^2$$

  - ***Classification***

$$\mathcal{L}(y^i, f(x^i)) = 1_{y^i \neq f(x^i)}$$

Empirical error

- On the training set, it is a poor estimate of the ***generalization error***
- If the model is overfitting, the generalization error can be arbitrarily large
- Our goal is to estimate the generalization error on unseen data, which we might not have

Validation & Cross-Validation

Empirical vs. true risk

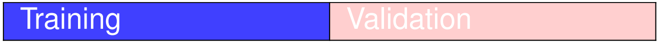
- In general, it is defined by

$$R(f) = R^{emp} + \text{overfit penalty}$$

- Overfit penalty depends on the complexity of the model
- ***Regularization*** approximates the overfit penalty. When the complexity of the model increases, we set up a larger overfit penalty
- ***Cross-validation*** tries to estimate $R(f)$ directly

Holdout method

- ***Holdout method*** is a popular approach for estimating the generalization performance of machine learning models
- Using ***holdout method***, we split the initial dataset into training and test sets



- We want to choose a model that performs best on a ***validation set*** independent of the ***training set***
- Since we have not used the validation data during the training phase, the validation set can be considered ***unseen data***
- In this case, the error on the validation set is an estimation of the generalization error
- What is another issue in this approach?

Model selection is a classification problem

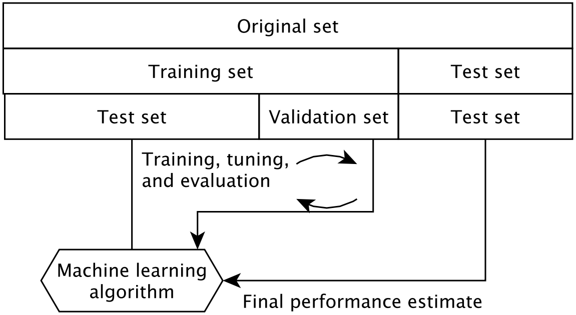
- We are interested in tuning and comparing different parameter settings to further improve the performance, for making prediction on unseen data
- This process is called ***model selection***
- Model selection refers to a given classification problem for which we want to select the optimal values of tuning parameters
- Therefore, if we reuse the same dataset over and over again during model selection, it will become part of our training data and thus the model will be more likely to overfit

Dealing with multiple models

- What should we do if we want to choose among *k* different models?
  1. We have to train each model on the training set
  2. Then, compute the prediction error of each model on the validation set
  3. Finally, select the model with the smallest prediction error on the validation set
- In that case, what will be the generalization error?
  - It is hard to say
  - Validation data was used to select the model
  - Actually, as we have looked at the validation data, it is not anymore a good proxy for unseen data

Holdout cross-validation

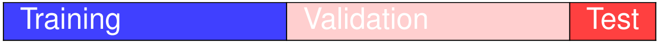
- A better way of using the holdout method for method selection comprises in splitting the dataset into three parts: a training set, a validation set, and a test set


Original set
Training set
Test set
Test set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Final performance estimate

- Therefore, the estimation error is sensitive to how we partition the training and the validation sets

Handling the problem of validation set

- We have to set aside a test set that remains untouched during the training and the validation phases
- With the test set, we can use it to estimate the generalization error



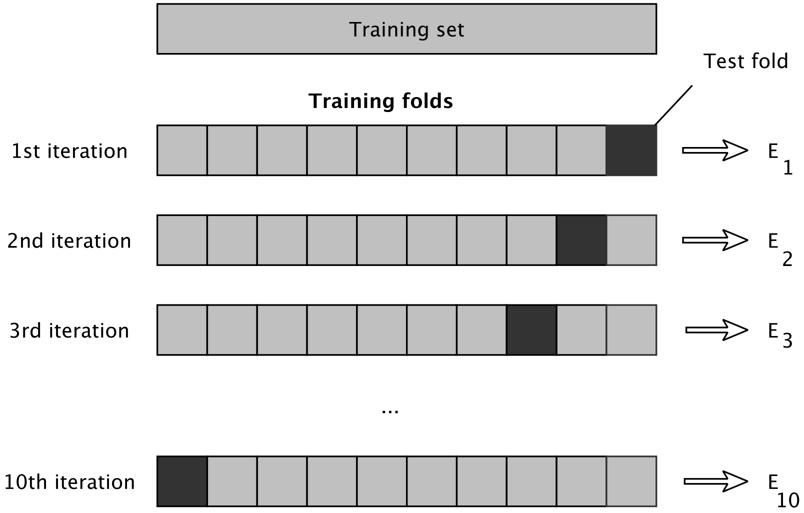
- How we decide the size of the training, validation, and test sets?
- How do we know that we have enough data to evaluate the prediction and the generalization errors?
- In ***model selection***, we aim to pick the best model
- Whereas, in ***model assessment***, we want to estimate the prediction errors on unseen data

We can use ***cross-validation*** and ***bootstrap***techniques to empirically evaluate our model

K-fold cross-validation

- ***k-fold cross-validation*** is a technique designed to give an accurate estimate of the true error without “wasting” too much data
- In the k-fold cross-validation, the original training set is partitioned into k folds without replacement
- $k - 1$ folds are used for the model training and one fold is used for testing
- For each fold, the model is estimated on the union of the other folds and then, the error of its output is estimated using the fold
- The average of all the errors is the estimate of the true error
- Once the best parameter is chosen, the model is retrained using the parameters of the entire training set

K-fold cross-validation



K-fold cross-validation algorithm

```
**Input:**
  training set $S = (x^1, y^i), \ldots, (x^p, y^p)$
  set of parameter values $\Theta$
  learning algorithm $\mathcal{A}$
  $k$ (number of folds)
split $S$ into $S_1, S_2, \ldots, S_k$
**foreach** $\theta \in \Theta$ **do**
    **for** $i = 1..k$ **do**
        $h_{i,\theta} = \mathcal{A}(S \setminus S_i; \theta)$
        $error(\theta) = \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}_{S_i}(h_{i,\theta})$
**Output:**
  $\theta^* = \arg\min_{\theta} [error(\theta)]$
  $h_{\theta^*} = \mathcal{A}(S; \theta^*)$
```

Cross-validation performance

- Estimating the prediction error

$$\begin{aligned} CV(f) &= \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y^i, f_{k(i)}(x^i)) \\ &= \frac{1}{k}\sum_{l=1}^{k} \mathbb{E}(f|D_l) \end{aligned}$$

  - where, $f_{k(i)}$ is the k_i-th part of the data removed
  - k_i is the fold in which i is
  - D_l is the fold l
- Estimating the expected prediction error

$$Error = \mathbb{E}[L(Y, f(X))]$$

Cross-validation issues

- The training set becomes $(k-1)*n/k$
  - small training set may lead to biased estimator of the error
- A special case of the k-fold cross-validation is the ***leave-one-out (LDO)***; i.e., $k = n$
  - approximately unbiased of the expected prediction error
  - potential high variance, since the training sets are similar to each other
  - computation can be very difficult
- In practice, k is set up to 5 or 10.

Bootstrap

- Randomly draws datasets with replacement from the training set
- Repeats ***B*** times (often, $B = 100$), which leads to ***B*** models
- Leave-one-out bootstrap error
  - for each training point i, predicts with the $b_i < B$ models that did not have i in their training set
  - computes the average prediction errors
- This leads for training set that has $0.632 * n$ distinct examples. Why?

$$\begin{aligned} \mathbb{P}(i \in x_k) &= 1 - (1 - \tfrac{1}{n})^n \\ &\approx 1 - e^{-1} \\ &= 0.632 \end{aligned}$$

- It has a high computational cost

Assessing Model Performance

Model evaluation metrics

- ***Precision***, ***recall***, and ***F1-score*** are performance metrics that can be used to measure a model's relevance
- The performance of a model can be summarized by means of a ***confusion matrix***

		Predicted class	
		+	−
Actual class	+	True Positives (TP)	False Negatives (FN)
	−	False Positives (FP)	True Negatives (TN)

- Each row refers to actual classes recorded in the test set, and each column to classes as predicted by the predictor
- ***False positives*** represent ***false alarms***, which are also known as ***type I errors***
- ***False negatives*** represent ***misses classifications***, which are called ***type II errors***

Computing precision, recall, and F1-score

- Prediction ***error*** (***ERR***) and ***accuracy (ACC)*** provide general information about how many samples are misclassified
- ***Error (ERR)***

$$ERR = \frac{FP + FN}{FP + FN + TP + TN}$$

- ***Accuracy (ACC)***

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

- ***Sensitivity = Recall = True Positive Rate (TPR)***

$$TPR = \frac{TP}{FN + TP}$$

Computing precision, recall, and F1-score

- ***False Positive Rate (FPR)***

$$FPR = \frac{FP}{FP + TN}$$

- ***Specificity = True Negative Rate (TNR)***

$$TNR = \frac{TN}{FP + TN}$$

- ***Precision = Positive Predictive Value (PPV)***

$$Precision = \frac{TP}{TP + FP}$$

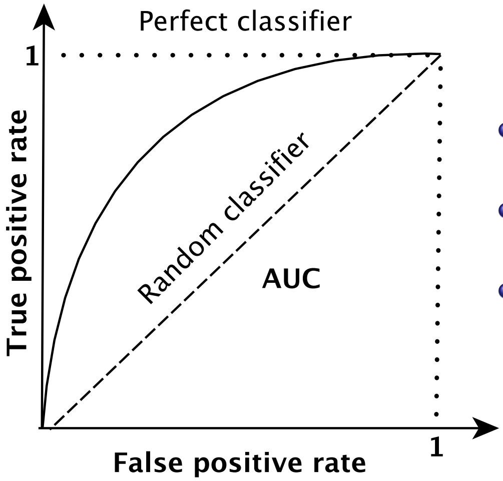
- ***F1-score*** represents the harmonic mean of precision and sensitivity

$$F1 = \frac{2TP}{2TP + FP + FN}$$

ROC curves

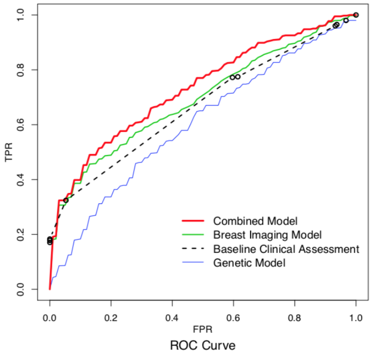
- ***Receiver operator characteristic (ROC)*** is a tool for selecting models for classification based on their performance with respect to the ***false positive*** and ***true positive*** rates
- The diagonal of an ROC plot can be interpreted as a random guessing
- It is summarized by the ***area under the curve (AUC)***, which characterize the performance of a classification model

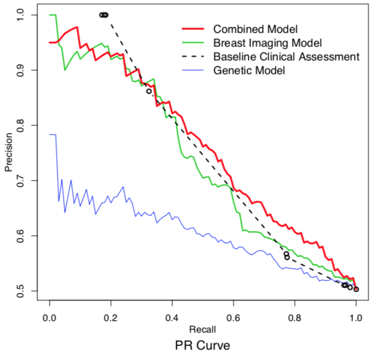
ROC curves



- ***Perfect classifier***:
  AUC = 1.0
- ***Random classifier***:
  AUC = 0.5
- ***Our classifier***:
  $0.5 < AUC < 1.0$

Example: Breast Cancer Risk Prediction on Mammograms





Predicting breast cancer risk based on mammography images. **Source**: Liu et al. (2013)[1]

- ***High recall*** means less chances to miss a case
- ***High precision*** means substantially more true diagnoses than false alarms

[1] Jie Liu et al. "Genetic variants improve breast cancer risk prediction on mammograms". In: *Annual Symposium Proceedings*. Vol. 2013. 2013, p. 876.

Assessing regression model performance

- ***Residual sum of squares (RSS)***

$$RSS = \sum_{i=1}^{n} (y_i - f(x^i))^2$$

- ***Root-mean squared error (RMSE)***

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - f(x^i))^2}{n}}$$

Assessing regression model performance

- ***Relative squared error (RSE)***

$$RSE = \frac{\sum_{i=1}^{n} (y_i - f(x^i))^2}{\sum_{i=1}^{n} (y^i - \bar{y})^2}$$

- ***Coefficient of determination***

$$R^2 = 1 - RSE$$

References

- Marianthi Markatou et al. “Analysis of Variance of Cross-Validation Estimators of the Generalization Error”. In: *Journal of Machine Learning Research* 6 (2005), pp. 1127–1168
- Bradley Efron and Robert Tibshirani. “Improvements on cross-validation: the 632+ bootstrap method”. In: *Journal of the American Statistical Association* 92.438 (1997), pp. 548–560
- L. G. Valiant. “A Theory of the Learnable”. In: *Communication of the ACM* 27.11 (1984), pp. 1134–1142
- Hal Daume III. *A Course in Machine Learning*. 2nd. Self-published, 2017. URL: `http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf`

1. **Noise**: session 2.3
2. **Overfitting**: session 2.4
3. **Bias-variance trade-off**: session 5.9
4. **Holdout method**: session 2.5
5. **Cross-validation**: session 5.6

References

6. **Assessing model performance**: session 5.5

- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. Springer, 2016. URL: https://web.stanford.edu/~hastie/Papers/ESLII.pdf
  1. **Overfitting**: session 7.1
  2. **Bias-variance trade-off**: sessions 2.9, 7.2, and 7.3
  3. **Cross-validation**: session 7.10
  4. **Bootstrap**: session 7.11