Classification and Regression V

Krzysztof Dembczyński

Intelligent Decision Support Systems Laboratory (IDSS)
Poznań University of Technology, Poland

Software Development Technologies
Master studies, second semester
Academic year 2018/19 (winter course)

Review of the previous lectures

- Mining of massive datasets.
- Classification and regression
    - What is machine learning?
    - Supervised learning: statistical decision/learning theory, loss functions, risk.
    - Learning paradigms and principles.
    - Learning algorithms: lazy learning, decision trees, generative models, linear models.
    - Linear models for classification.
    - Feature engineering.
    - Linear models in extended feature spaces.

Outline

1 Learning Problem

2 Performance Measures in Classification

3 Summary

Outline

1 Learning Problem

2 Performance Measures in Classification

3 Summary

Learing problem

- To solve the prediction problem optimally we would need to find:

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\ell(y, h(\boldsymbol{x}))] \,.$$

  Since $P(\boldsymbol{x}, y)$ is generally unknown, we rely on a set of **training** (previously solved) examples $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$.

Learing problem

- To solve the prediction problem optimally we would need to find:

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\ell(y, h(\boldsymbol{x}))] \,.$$

  Since $P(\boldsymbol{x}, y)$ is generally unknown, we rely on a set of **training** (previously solved) examples $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$.
- The true function class $\mathcal{H}$ of h^* is also unknown, so we need to make a choice of a **proper function class**.

Learing problem

- To solve the prediction problem optimally we would need to find:

$$h^* = \operatorname*{arg\,min}_{h \in \mathcal{H}} \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\ell(y, h(\boldsymbol{x}))] \,.$$

  Since $P(\boldsymbol{x}, y)$ is generally unknown, we rely on a set of **training** (previously solved) examples $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$.
- The true function class $\mathcal{H}$ of h^* is also unknown, so we need to make a choice of a **proper function class**.
- The loss function used as the main performance measure, the so-called **task loss**, might be hard to optimize, so we often use a **surrogate loss** that is easier to cope with:

Learing problem

- To solve the prediction problem optimally we would need to find:

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\ell(y, h(\boldsymbol{x}))] \, .$$

  Since $P(\boldsymbol{x}, y)$ is generally unknown, we rely on a set of **training** (previously solved) examples $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$.

- The true function class $\mathcal{H}$ of h^* is also unknown, so we need to make a choice of a **proper function class**.
- The loss function used as the main performance measure, the so-called **task loss**, might be hard to optimize, so we often use a **surrogate loss** that is easier to cope with:
  - e.g., 0/1 loss is replaced by logistic loss.

Learing problem

- To solve the prediction problem optimally we would need to find:

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\ell(y, h(\boldsymbol{x}))] \,.$$

  Since $P(\boldsymbol{x}, y)$ is generally unknown, we rely on a set of **training** (previously solved) examples $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$.
- The true function class $\mathcal{H}$ of h^* is also unknown, so we need to make a choice of a **proper function class**.
- The loss function used as the main performance measure, the so-called **task loss**, might be hard to optimize, so we often use a **surrogate loss** that is easier to cope with:
  - e.g., 0/1 loss is replaced by logistic loss.
- **Task**: construct $h(x)$ to be the best possible approximation of $h^*(x)$.

Three sources of prediction error

- Estimation error due to finite sample size.

Three sources of prediction error

- Estimation error due to finite sample size.
- Approximation error due to the size of the function space $\mathcal{H}$.

Three sources of prediction error

- Estimation error due to finite sample size.
- Approximation error due to the size of the function space $\mathcal{H}$.
- Approximation error due to the use of a surrogate loss in place of the original task loss.

Model assessment and selection

- The generalization performance of a learning method relates to its prediction capability on independent test data.

Model assessment and selection

- The generalization performance of a learning method relates to its prediction capability on independent test data.
- Assessment of this performance is extremely important in practice, since it guides the choice of learning method or model, and gives us a measure of the quality of the ultimately chosen model.

Model assessment and selection

- Test error, also referred to as generalization error or prediction risk, is the expected prediction error over an independent test sample:

  $$\mathrm{Err}_{\mathcal{T}} = \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\ell(y, h(\boldsymbol{x})) \,|\, \mathcal{T}]$$

  where $h(x)$ is the prediction function estimated on the fixed training set $\mathcal{T}$.

Model assessment and selection

- Test error, also referred to as generalization error or prediction risk, is the expected prediction error over an independent test sample:

$$\mathrm{Err}_{\mathcal{T}} = \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\ell(y, h(\boldsymbol{x})) \,|\, \mathcal{T}]$$

  where $h(x)$ is the prediction function estimated on the fixed training set $\mathcal{T}$.

- A related quantity is the expected prediction error (or expected test error):

$$\mathrm{Err} = \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\ell(y, h(x))] = \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\mathrm{Err}_{\mathcal{T}}]$$

Empirical risk minimization

- The learning is performed by minimization of the empirical risk:

$$\widehat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))\,.$$

  where it is usually assumed that training examples are independent and identically distributed (i.i.d.).

Empirical risk minimization

- The learning is performed by minimization of the empirical risk:

$$\widehat{L}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))\,.$$

  where it is usually assumed that training examples are independent and identically distributed (i.i.d.).
- In general, it is not a problem to fit the training data arbitrary well, reducing the empirical risk to zero:

$$\widehat{L}(h) \simeq 0.$$

Empirical risk minimization

- The learning is performed by minimization of the empirical risk:

$$\widehat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))\,.$$

  where it is usually assumed that training examples are independent and identically distributed (i.i.d.).
- In general, it is not a problem to fit the training data arbitrary well, reducing the empirical risk to zero:

$$\widehat{L}(h) \simeq 0.$$

- This is seldom the best thing to do, because fitting the training data too well can increase prediction risk on future predictions.

Empirical risk minimization

- The learning is performed by minimization of the empirical risk:

$$\widehat{L}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))\,.$$

  where it is usually assumed that training examples are independent and identically distributed (i.i.d.).
- In general, it is not a problem to fit the training data arbitrary well, reducing the empirical risk to zero:

$$\widehat{L}(h) \simeq 0.$$

- This is seldom the best thing to do, because fitting the training data too well can increase prediction risk on future predictions.
- This is a phenomenon called **overfitting**.

Empirical risk minimization

- The learning is performed by minimization of the empirical risk:

$$\widehat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i)) \,.$$

  where it is usually assumed that training examples are independent and identically distributed (i.i.d.).
- In general, it is not a problem to fit the training data arbitrary well, reducing the empirical risk to zero:

$$\widehat{L}(h) \simeq 0.$$

- This is seldom the best thing to do, because fitting the training data too well can increase prediction risk on future predictions.
- This is a phenomenon called **overfitting**.
- We usually choose h from a restricted family of functions, but being to restrictive may result in **underfitting**.

Training vs. test error

- **Training error of function h:**

$$\widehat{L}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))$$

$\Longrightarrow$ Average error on a sample of size n.

Training vs. test error

- **Training error of function h:**

$$\widehat{L}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))$$

$\Longrightarrow$ Average error on a sample of size n.

- **Generalization error (expected prediction error) of function h:**

$$L(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}\left[\ell(y, h(\boldsymbol{x}))\right]$$

$\Longrightarrow$ Expected error h makes on a randomly drawn instance.

Training vs. test error

- **Training error of function h:**

$$\widehat{L}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))$$

$\implies$ Average error on a sample of size n.

- **Generalization error (expected prediction error) of function h:**

$$L(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}\left[\ell(y, h(\boldsymbol{x}))\right]$$

$\implies$ Expected error h makes on a randomly drawn instance.

- For fixed h, $\widehat{L}(h) \simeq L(h)$ (law of large numbers).

Training vs. test error

- **Training error of function h:**

$$\widehat{L}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))$$

$\Longrightarrow$ Average error on a sample of size n.

- **Generalization error (expected prediction error) of function h:**

$$L(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}\left[\ell(y, h(\boldsymbol{x}))\right]$$

$\Longrightarrow$ Expected error h makes on a randomly drawn instance.

- For fixed h, $\widehat{L}(h) \simeq L(h)$ (law of large numbers).
- But h is not fixed, it is **chosen** based on the training sample!

Overfitting

- **Can you think of a “learning” algorithm which always have zero training error and poor test error?**

Overfitting

- **Can you think of a "learning" algorithm which always have zero training error and poor test error?**
- **"Memorization"**: given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, choose:

$$h(\boldsymbol{x}) = \begin{cases} y_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \text{ for some } i = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

Overfitting

- **Can you think of a “learning” algorithm which always have zero training error and poor test error?**
- **“Memorization”**: given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, choose:

$$h(\boldsymbol{x}) = \begin{cases} y_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \text{ for some } i = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

- It is **very easy** to drop training error down to 0 if our function class is flexible enough.

Overfitting

- **Can you think of a "learning" algorithm which always have zero training error and poor test error?**
- **"Memorization"**: given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, choose:

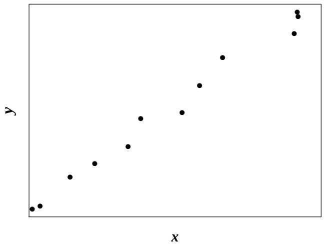
$$h(\boldsymbol{x}) = \begin{cases} y_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \text{ for some } i = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

- It is **very easy** to drop training error down to 0 if our function class is flexible enough.
- This phenomenon is called **overfitting** to the data.

Overfitting with ERM

- A simple artificial data set generator:

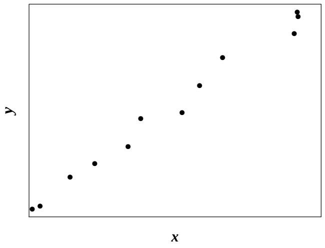
$$x \sim \text{uniform}(0, 1),$$
$$y = x + \epsilon, \qquad \epsilon \sim N(0,\ 0.05).$$



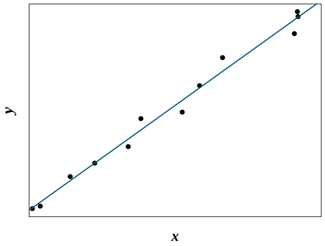
Overfitting with ERM

- A simple artificial data set generator:

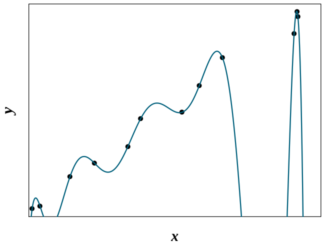
$$x \sim \text{uniform}(0, 1),$$
$$y = x + \epsilon, \qquad \epsilon \sim N(0,\ 0.05).$$



Fitting linear function:



Fitting polynomial of degree n:



Overfitting

- A more complex/flexible function class is not always a good choice.
- Too complex class $\Longrightarrow$ overfitting.
- Too simple class $\Longrightarrow$ learning too constrained model.
- Can we control how much do we overfit?

Overfitting

- A more complex/flexible function class is not always a good choice.
- Too complex class $\Longrightarrow$ overfitting.
- Too simple class $\Longrightarrow$ learning too constrained model.
- Can we control how much do we overfit?

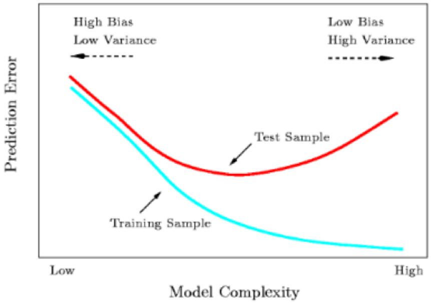
$\Longrightarrow$ **Statistical learning theory**

Bias, variance and model complexity

- As the model becomes more and more complex, it is able to adapt to more complicated underlying structures (a decrease in bias), but the estimation error increases (an increase in variance).

Bias, variance and model complexity

- As the model becomes more and more complex, it is able to adapt to more complicated underlying structures (a decrease in bias), but the estimation error increases (an increase in variance).
- In between there is an optimal model complexity that gives minimum test error.



Regularization

- The complexity of the prediction function can be controlled by **regularization** (or penalization).

Regularization

- The complexity of the prediction function can be controlled by **regularization** (or penalization).
- A general class of regularization problems has a form:

$$\min_{h \in \mathcal{H}} \left[\sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i)) + \lambda J(h) \right] ,$$

  where $J(h)$ is a penalty functional, and $\mathcal{H}$ is a family of functions on which $J(h)$ is defined, and λ controls the degree of regularization.

Regularization

- Consider a family of linear functions:

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} \,.$$

Regularization

- Consider a family of linear functions:

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} \, .$$

- In the regularization methods, parameters of the linear function are estimated through:

$$\widehat{\boldsymbol{w}}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \ell(y_i, \boldsymbol{w} \cdot \boldsymbol{x}) + \lambda R(\{w_j\}_1^m),$$

where the first term measures the loss on the training sample, and the second term is a penalty on the values of the coefficients $\{w_j\}_1^m$.

Regularization

- There are two commonly employed penalty functions $R(\{w_j\}_1^m)$:

$$R_1(\{w_j\}_1^m) = \sum_{j=1}^m |w_j| \qquad R_2(\{w_j\}_1^m) = \sum_{j=1}^m |w_j|^2$$

Regularization

- There are two commonly employed penalty functions $R(\{w_j\}_1^m)$:

$$R_1(\{w_j\}_1^m) = \sum_{j=1}^{m} |w_j| \qquad R_2(\{w_j\}_1^m) = \sum_{j=1}^{m} |w_j|^2$$

- **Lasso** penalty R_1: shrinks the absolute values of the coefficients $\{w_j\}_1^m$ from that of the unpenalized solution $\lambda = 0$, but it is indifferent to their dispersion. It tends to produce solutions with relatively few large absolute valued coefficients and many with zero value.

Regularization

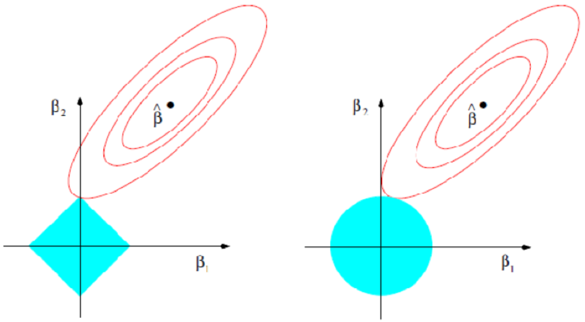
- There are two commonly employed penalty functions $R(\{w_j\}_1^m)$:

$$R_1(\{w_j\}_1^m) = \sum_{j=1}^{m} |w_j| \qquad R_2(\{w_j\}_1^m) = \sum_{j=1}^{m} |w_j|^2$$

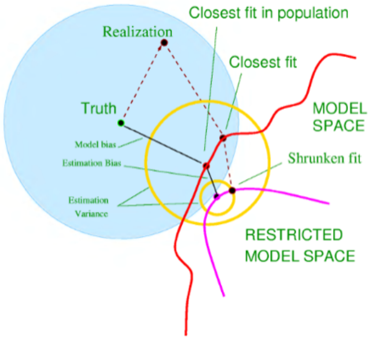
- **Lasso** penalty R_1: shrinks the absolute values of the coefficients $\{w_j\}_1^m$ from that of the unpenalized solution $\lambda = 0$, but it is indifferent to their dispersion. It tends to produce solutions with relatively few large absolute valued coefficients and many with zero value.
- **Ridge** penalty R_2: shrinks also the absolute values of the coefficients $\{w_j\}_1^m$, while discouraging dispersion among those absolute values. That is, it prefers solutions in which all the variables have similar influence on the resulting linear model.

Regularization

- Lasso vs. ridge penalty:

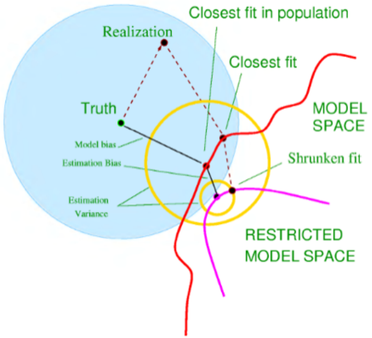


Bias, variance and model complexity



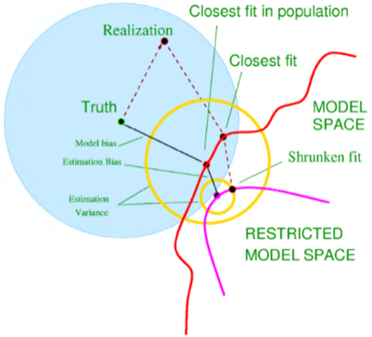
- The model space is the set of all possible predictions from the model, with the **closest fit** labeled with a black dot.

Bias, variance and model complexity



- The model space is the set of all possible predictions from the model, with the **closest fit** labeled with a black dot.
- The model bias from the truth is shown, along with the variance, indicated by the yellow circle centered at the black dot labeled **closest fit in population**.

Bias, variance and model complexity



- The model space is the set of all possible predictions from the model, with the **closest fit** labeled with a black dot.
- The model bias from the truth is shown, along with the variance, indicated by the yellow circle centered at the black dot labeled **closest fit in population**.
- A regularized fit is also shown, having additional estimation bias, but smaller prediction error due to its decreased variance.

Model assessment and selection

- Typically our model will have a tuning parameter or parameters α, i.e., $f_\alpha(\boldsymbol{x})$.
- The tuning parameter varies the complexity of our model, and we wish to find the value of α that minimizes error (produces the minimum of the test error curve).
- It is important to note that there are in fact two separate goals that we might have in mind:
    - **Model selection**: estimating the performance of different models in order to choose the (approximate) best one.
    - **Model assessment**: having chosen a final model, estimating its prediction error (generalization error) on new data.

Model assessment and selection

- In a data-rich situation, the best approach is to randomly divide the dataset into three parts:
    - **training set** – used to fit the models,
    - **validation set** – used to estimate prediction error for model selection,
    - **test set** – used for assessment of the generalization error of the final chosen model.
- Ideally, the test set should be kept in a "vault", and be brought out only at the end of the data analysis.
- Suppose instead that we use the test set repeatedly, choosing the model with smallest test set error. Then the test set error of the final chosen model will underestimate the true test error, sometimes substantially.

Model assessment and selection

- It is difficult to give a general rule on how to choose the number of observations in each of the three parts.
- A typical split might be 50% for training, and 25% each for validation and testing:

Train	Validation	Test

Separate test set

- We are given a training set $\mathcal{T} = \{y_i, \boldsymbol{x}_i\}_1^n$ and a separate testing set T.
- The error estimate is obtained from:

$$\widehat{\mathrm{Err}}_{\mathcal{T}} = \frac{1}{|T|} \sum_{(y,\boldsymbol{x})} \ell(y, f(\boldsymbol{x}))$$

- $\widehat{\mathrm{Err}}_{\mathcal{T}}$ is unbiased for Err and $\mathrm{Err}_{\mathcal{T}}$.

Repeated hold-out

- Repeat K times, $k = 1, \ldots, K$:
  - Randomly divide training set $\mathcal{T}$ into $\mathcal{T}_k$ and T_k in a fixed proportion (usually T_k contains $1/3$ of the data).
  - Train classifier f_k on $\mathcal{T}_k$.
  - Estimate the error on the k-th testing set:

$$\widehat{\mathrm{Err}}_k = \frac{1}{T_k} \sum_{(y,\boldsymbol{x}) \in T_k} \ell(y, f_k(\boldsymbol{x}))$$

  - The final error estimate is:

$$\widehat{\mathrm{Err}} = \frac{1}{K} \sum_{k=1}^{K} \widehat{\mathrm{Err}}_k$$

K-fold cross validation

$\mathcal{T}_1$	$\mathcal{T}_1$	$\mathcal{T}_1$	$\mathcal{T}_1$	T_1

- The same as repeated hold-out with the exception that K testing sets T_k are of size $\frac{n}{K}$ and are disjoint: $T_k \cup T_{k'} = \emptyset$, for $k = k'$.
- When $K = N$, this procedure is known as **leave-one-out**.
- For small K, the bias is high and variance is low.
- For large K the bias is low but the variance is high (the dataset are similar to each other).
- Usually $K = 5, 10$ or n.

K-fold cross validation

$\mathcal{T}_2$	$\mathcal{T}_2$	$\mathcal{T}_2$	T_2	$\mathcal{T}_2$

- The same as repeated hold-out with the exception that K testing sets T_k are of size $\frac{n}{K}$ and are disjoint: $T_k \cup T_{k'} = \emptyset$, for $k = k'$.
- When $K = N$, this procedure is known as **leave-one-out**.
- For small K, the bias is high and variance is low.
- For large K the bias is low but the variance is high (the dataset are similar to each other).
- Usually $K = 5, 10$ or n.

K-fold cross validation

$\mathcal{T}_3$	$\mathcal{T}_3$	T_3	$\mathcal{T}_3$	$\mathcal{T}_3$

- The same as repeated hold-out with the exception that K testing sets T_k are of size $\frac{n}{K}$ and are disjoint: $T_k \cup T_{k'} = \emptyset$, for $k = k'$.
- When $K = N$, this procedure is known as **leave-one-out**.
- For small K, the bias is high and variance is low.
- For large K the bias is low but the variance is high (the dataset are similar to each other).
- Usually $K = 5, 10$ or n.

K-fold cross validation

$\mathcal{T}_4$	T_4	$\mathcal{T}_4$	$\mathcal{T}_4$	$\mathcal{T}_4$

- The same as repeated hold-out with the exception that K testing sets T_k are of size $\frac{n}{K}$ and are disjoint: $T_k \cup T_{k'} = \emptyset$, for $k = k'$.
- When $K = N$, this procedure is known as **leave-one-out**.
- For small K, the bias is high and variance is low.
- For large K the bias is low but the variance is high (the dataset are similar to each other).
- Usually $K = 5, 10$ or n.

K-fold cross validation

T_5	$\mathcal{T}_5$	$\mathcal{T}_5$	$\mathcal{T}_5$	$\mathcal{T}_5$

- The same as repeated hold-out with the exception that K testing sets T_k are of size $\frac{n}{K}$ and are disjoint: $T_k \cup T_{k'} = \emptyset$, for $k = k'$.
- When $K = N$, this procedure is known as **leave-one-out**.
- For small K, the bias is high and variance is low.
- For large K the bias is low but the variance is high (the dataset are similar to each other).
- Usually $K = 5, 10$ or n.

K-fold cross validation

T_5	$\mathcal{T}_5$	$\mathcal{T}_5$	$\mathcal{T}_5$	$\mathcal{T}_5$

- The same as repeated hold-out with the exception that K testing sets T_k are of size $\frac{n}{K}$ and are disjoint: $T_k \cup T_{k'} = \emptyset$, for $k = k'$.
- When $K = N$, this procedure is known as **leave-one-out**.
- For small K, the bias is high and variance is low.
- For large K the bias is low but the variance is high (the dataset are similar to each other).
- Usually $K = 5, 10$ or n.
- Let us assume that there is 50 positive and 50 negative training examples: What is the best classifier in the leave-one-out procedure?

Comparison

- The repeated hold-out and CV estimates are unbiased for the Err of an classifier trained on $n - |T|$ examples.
- They are biased (downward) for Err of an classifier trained on n examples.
- Repeating CV (and large number of runs in holdout) improves the **replicability** of the experiment.
- It was experimentally shown that repeated CV works slightly better than repeated hold-out.

Outline

1 Learning Problem

2 Performance Measures in Classification

3 Summary

Performance measures for classification

- Besides **0/1 loss** (**misclassification error**), there is a **multitude** of **performance measures** used in assessment and selection of the models.
- These performance measures are sometimes referred to as **task losses**.

Confusion matrix

- Many performance measures are defined based on **confusion matrix**.
- Let $y \in \{0, 1\}$ and $h(\boldsymbol{x}) \in \{0, 1\}$ be a classifier, and $T = \{y_i, \boldsymbol{x}_i\}_1^n$ be a test set, then the confusion matrix is defined as:

Classifier	y	
$h(\boldsymbol{x})$	1	0
1	TP	FP
0	FN	TN

Confusion matrix

- True positives:

$$\mathsf{TP}(h) = \sum_{i=1}^{n} y_i h(\boldsymbol{x}_i)$$

  $\mathsf{TP}(h)/n$ estimates $P(h(\boldsymbol{x}) = 1, y = 1)$

- True negatives:

$$\mathsf{TN}(h) = \sum_{i=1}^{n} (1 - y_i)(1 - h(\boldsymbol{x}_i))$$

  $\mathsf{TN}(h)/n$ estimates $P(h(\boldsymbol{x}) = 0, y = 0)$

Confusion matrix

- False positives:

$$\mathsf{FP}(h) = \sum_{i=1}^{n} (1 - y_i) h(\boldsymbol{x}_i)$$

  $\mathsf{FP}(h)/n$ estimates $P(h(\boldsymbol{x}) = 1, y = 0)$

- False negatives:

$$\mathsf{FN}(h) = \sum_{i=1}^{n} y_i (1 - h(\boldsymbol{x}_i))$$

  $\mathsf{FN}(h)/n$ estimates $P(h(\boldsymbol{x}) = 0, y = 1)$

Accuracy

- Accuracy:

$$\mathsf{Acc}(h) = \frac{1}{n}(\mathsf{TP} + \mathsf{TN}) = \frac{1}{n}\sum_{i=1}^{n} [\![y_i = h(\boldsymbol{x}_i)]\!]$$

Accuracy estimates $P(y = h(\boldsymbol{x})) = 1 - L_{0/1}(h)$.

Classifier	y	
$h(\boldsymbol{x})$	1	0
1	TP	FP
0	FN	TN

Recall

- Recall or true positive rate:

$$\text{Recl}(h) = \text{TPR}(h) = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\sum_{i=1}^{n} y_i h(\boldsymbol{x}_i)}{\sum_{i=1}^{n} y_i}$$

Recall estimates $P(h(\boldsymbol{x}) = 1 \mid y = 1)$.

Classifier	y	
$h(\boldsymbol{x})$	1	0
1	TP	FP
0	FN	TN

Precision

- Precision:

$$\text{Prec}(h) = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\sum_{i=1}^{n} y_i h(\boldsymbol{x}_i)}{\sum_{i=1}^{n} h(\boldsymbol{x}_i)}$$

  Precision estimates $P(y = 1 \mid h(\boldsymbol{x}) = 1)$.

Classifier	y	
$h(\boldsymbol{x})$	1	0
1	TP	FP
0	FN	TN

False positive rate

- False positive rate:

$$\mathsf{FPR}(h) = \frac{\mathsf{FP}}{\mathsf{FP} + \mathsf{TN}} = \frac{\sum_{i=1}^{n} y_i h(\boldsymbol{x}_i)}{\sum_{i=1}^{n}(1 - y_i)}$$

False positive rate estimates $P(h(\boldsymbol{x})) = 1 \,|\, y = 0)$.

Classifier	y	
$h(\boldsymbol{x})$	1	0
1	TP	FP
0	FN	TN

The F-measure

- The F_β-measure is a weighted harmonic mean of precision and recall:

$$\begin{aligned} F_\beta(h) &= \frac{1+\beta^2}{\beta^2/\mathrm{Prec}(h) + 1/\mathrm{Recl}(h)} \\ &= \frac{(1+\beta^2)\sum_{i=1}^n y_i h(\boldsymbol{x}_i)}{\beta^2 \sum_{i=1}^n y_i + \sum_{i=1}^n h(\boldsymbol{x}_i)} \end{aligned}$$

The F_β-measure estimates $\frac{(1+\beta^2)P(y=1,h(\boldsymbol{x})=1)}{\beta^2 P(y=1)+P(h(\boldsymbol{x}_i)=1)}$.

Classifier	y	
$h(\boldsymbol{x})$	1	0
1	TP	FP
0	FN	TN

The F-measure

- The F-measure is better suited to imbalanced data than accuracy.
- **Example**:
  - Let $P(y = 1) = 0.1$ and $P(y = 0) = 0.9$.
  - Majority classifier $h(\boldsymbol{x})$ predicting always 0 will perform quite well in terms of accuracy, i.e., $Pr(y = h(\boldsymbol{x})) = 0.9$,
  - But the F-measure will be 0 in this case.

Exercise

- In two-class problem, we have 500 testing examples.
- We know that the probability of class -1 is 80%, and the probability of class 1 is 20%.
- Compute the expected confusion matrix for three simple classifiers: random, random taking into account the class distribution, and majority classifier.

Classifier	y	
$h(\boldsymbol{x})$	1	0
1	TP	FP
0	FN	TN

Outline

1 Learning Problem

2 Performance Measures in Classification

3 Summary

Summary

- Learning problem: how to learn and verify a predictive model?
- Is learning possible?
- Overfitting and underfitting.
- Regularization.
- Testing: separate test set, repeated hold-out, cross-validation.
- A multitude of performance measures.

Bibliography

- T. Hastie, R. Tibshirani, and J. Friedman. *Elements of Statistical Learning: Second Edition*.
  Springer, 2009
  `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*.
  Springer-Verlag, 2006
- David Barber. *Bayesian Reasoning and Machine Learning*.
  Cambridge University Press, 2012
  `http://www.cs.ucl.ac.uk/staff/d.barber/brml/`
- Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*.
  AMLBook, 2012