Aprendizagem Computacional

Licenciatura em Ciências da Computação

Section 1

Metrics and Model Evaluation

Confusion Matrix

		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Classification Metrics

Sensitivity

- recall
- hit rate
- true positive rate (TPR)

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

Specificity

- selectivity
- true negative rate (TNR)

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

Precision

- positive predictive value (PPV)

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

Accuracy

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

Balanced Accuracy

$BA = \frac{TPR+TNR}{2}$

F1 score

- harmonic mean of precision and sensitivity

$F_1 = \frac{2TP}{2TP+FP+FN}$

Regression Metrics

R^2

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Mean Squared Error

$$\mathrm{MSE} = \frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}$$

Mean Absolute Error

$$\mathrm{MAE} = \frac{\sum_{i=1}^{N} |x_i - \hat{x}_i|}{N}$$

Root Mean Squared Error

$$\mathrm{RMSE} = \frac{\sqrt{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}}{N}$$

Underfitting vs Overfitting

Generalization ability to make accurate predictions with new data

Overfitting model fits training data perfectly but is unable to generalize

Underfitting model is unable of performing accurate predictions even with training data!

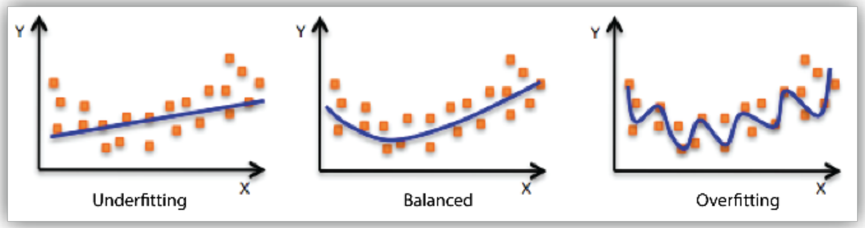


Figure 1: Underfitting vs Overfitting (fig by Amazon)

Model Evaluation

Training set	Used for training
Holdout	Only used for evaluating a model

Discussion

- Holdout is necessary for evaluating the model over data that was not using during training
- Holdout is a smaller percentage of the dataset
- Evaluation is too dependent on the Train/Holdout division
- Decisions about the hyperparameters are completely biased

Model Evaluation

Training set Used for training

Validation set Only used for providing an *unbiased* evaluation of a model on the training set while *fitting* the hyperparameters

Test set Only used for providing an *unbiased* evaluation of the *final* model fit on the training set

Train/Validation/Test

Characteristics

- Validation set must *only* used for hyperparameter fitting
- Test set must *only* be used for the *unbiased* evaluation of the *final* model fit on the training set
- Validation set cannot be part of training set
- Test set cannot be part of training or validation sets
- Preprocessing can only be *fitted* using training data
- Learning **cannot** use validation or test data in any way!

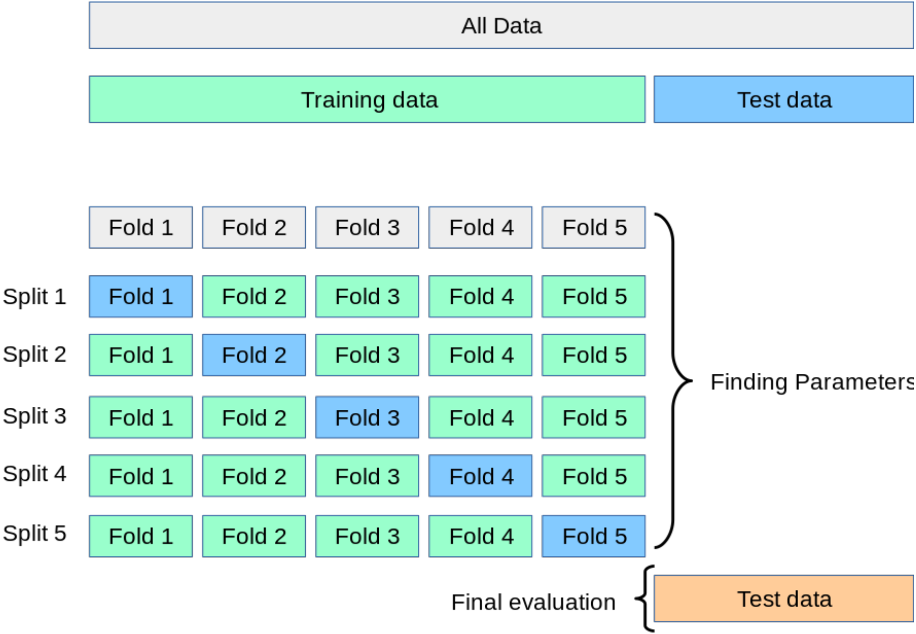
Problems

- Too dependant on the division
- May introduce unforeseen biases

Solution

- Use Cross validation

Cross validation



Cross validation

- Shuffle dataset randomly
- Divide training data into k folds
- For each f of the k folds
  - Use all folds but the f-th for training
  - Use the f-th fold for validation
  - Compute evaluation score
- Summarize the scores for all folds

Common values for k

10-fold cross validation

- found through experimentation
- low bias
- good variance

Leave One Out

- Extreme value for k
- k is the number of observations

Variations of Cross Validation

Stratified

- each fold has the same **proportion** of observations with a given categorical value
- ensures a smoother evaluation

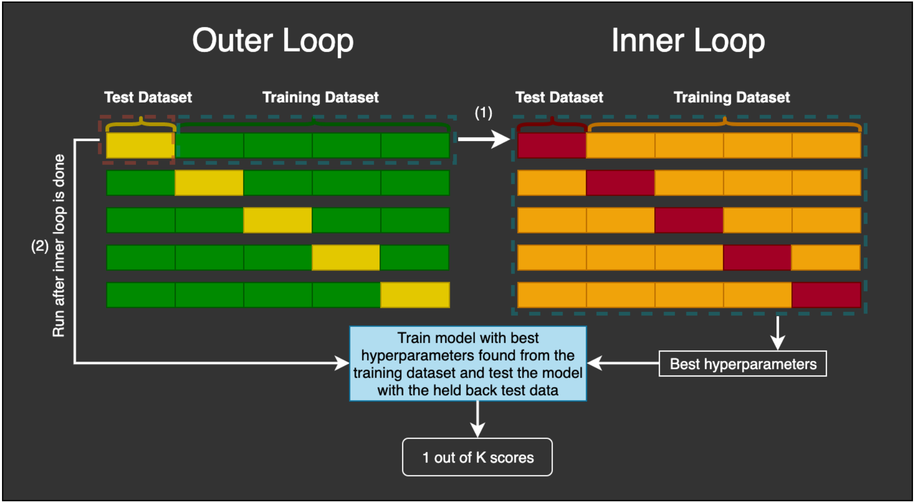
Repeated

- k-fold cross-validation procedure is repeated *n* times,
- Data is shuffled prior to **each** repetition

Nested

- k-fold cross-validation is performed within each fold of cross-validation
- hyperparameter tuning is performed **during** model evalation

Nested Cross Validation



Section 2

Algorithms

k Nearest Neighbors

- Lazy learner
- Doesn't train a model
- Uses k closest neighbors
  - Uses most frequent value for classification
  - Uses mean value for regression
- Uses distances in order to find closest neighbors

Distances

Euclidean

$$\sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^{N} |x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^{N} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

Discussion

- No training needed
- Simple to implement
- Works better with numeric values
- It is important to *scale* the attributes
- Ordinal values work better than purely categorical ones
- Preprocessing is very important
- Algorithm is difficult to use on very large datasets

Naive Bayes

$$\begin{aligned}
P(A \wedge B) &= P(A \mid B)\, P(B) = P(B \mid A)\, P(A) \\
P(A \vee B) &= P(A) + P(B) - P(A \wedge B) \\
P(h \mid D) &= \frac{P(D \mid h)\, P(h)}{P(D)} \\
P(B) &= \sum_{i=1}^{n} P(B \mid A_i)\, P(A_i) \ \mathrm{se} \sum_{i=1}^{n} P(A_i) = 1
\end{aligned}$$

Naive Bayes Classifier

Attributes $a_1, \ldots, a_n$

Classes v_i

Goal Find v_i that maximizes $P(v_i \mid a_1, \ldots, a_n)$

Formula

$$P(v_i \mid a_1, \ldots, a_n) = \frac{P(a_1, \ldots, a_n \mid v_i)\, P(v_i)}{P(a_1, \ldots, a_n)}$$

Naive Bayes Classifier

$$
\begin{aligned}
\arg\max_{v_i \in V} P(v_i \mid a_1, \ldots, a_n) &= \arg\max_{v_i \in V} \frac{P(a_1, \ldots, a_n \mid v_i)\, P(v_i)}{P(a_1, \ldots, a_n)} \\
&= \arg\max_{v_i \in V} P(a_1, \ldots, a_n \mid v_i)\, P(v_i)
\end{aligned}
$$

Problem

- Cost of estimating $P(a_1, \ldots, a_n \mid v_i)$ is very high
- We would need a lot of data!
- Therefore, assume the attribute values are *independent*

Assumption

$$P(a_1, \ldots, a_n \mid v_i) \approx \prod_{k=1}^{n} P(a_k \mid v_i)$$

Naive Bayes Classifier

$$v_{\mathrm{NB}} = \arg\max_{v_i \in V} P(v_i) \prod_{k=1}^{n} P(a_k \mid v_i)$$

Example

Outlook	Temperature	Humidity	Wind	Play Tennis?
Overcast	Hot	Normal	Weak	Yes
Overcast	Mild	High	Strong	Yes
Sunny	Mild	Normal	Strong	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Cool	Normal	Weak	Yes
Overcast	Cool	Normal	Strong	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Mild	High	Weak	Yes
Overcast	Hot	High	Weak	Yes
Rain	Cool	Normal	Strong	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	No
Rain	Mild	High	Strong	No
Sunny	Mild	High	Weak	No

Example: What is the class for

Outlook	**Temperature**	**Humidity**	**Wind**
Sunny	Cool	High	Strong

$$v_{\mathrm{NB}} = \arg\max_{v_i \in \{Yes, No\}} \begin{aligned} & P(v_i) \times \\ & P(Outlook = Sunny \mid v_i) \times \\ & P(Temperature = Cool \mid v_i) \times \\ & P(Humidity = High \mid v_i) \times \\ & P(Wind = Strong \mid v_i) \end{aligned}$$

Example: Compute frequencies

Outlook	Yes	No	Temperature	Yes	No	Humidity	Yes	No	Wind	Yes	No
Sunny	2	3	Hot	2	2	High	3	4	Strong	3	3
Overcast	4	0	Mild	4	2	Normal	6	1	Weak	6	2
Rain	3	2	Cool	3	1						

Table 1: Absolute frequencies

Outlook	Yes	No	Temperature	Yes	No	Humidity	Yes	No	Wind	Yes	No
Sunny	$\frac{2}{9}$	$\frac{3}{5}$	Hot	$\frac{2}{9}$	$\frac{2}{5}$	High	$\frac{3}{9}$	$\frac{4}{5}$	Strong	$\frac{3}{9}$	$\frac{3}{5}$
Overcast	$\frac{4}{9}$	$\frac{0}{5}$	Mild	$\frac{4}{9}$	$\frac{2}{5}$	Normal	$\frac{6}{9}$	$\frac{1}{5}$	Weak	$\frac{6}{9}$	$\frac{2}{5}$
Rain	$\frac{3}{9}$	$\frac{2}{5}$	Cool	$\frac{3}{9}$	$\frac{1}{5}$						

Table 2: Relative frequencies

Example: Compute likelihoods

$$
\begin{aligned}
P(PlayTennis = Yes) \quad \times \quad & P(Outlook = Sunny \mid PlayTennis = Yes) \times \\
& P(Temperature = Cool \mid PlayTennis = Yes) \times \\
& P(Humidity = High \mid PlayTennis = Yes) \times \\
& P(Wind = Strong \mid PlayTennis = Yes) = \\
= \quad & \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} = 0.00529
\end{aligned}
$$

$$
\begin{aligned}
P(PlayTennis = No) \quad \times \quad & P(Outlook = Sunny \mid PlayTennis = No) \times \\
& P(Temperature = Cool \mid PlayTennis = No) \times \\
& P(Humidity = High \mid PlayTennis = No) \times \\
& P(Wind = Strong \mid PlayTennis = No) = \\
= \quad & \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} = 0.02057
\end{aligned}
$$

Example

Class value

No

Probability

$$\frac{0.02057}{0.02057 + 0.00529} = 79.54\%$$

Probability estimates

- In some cases, the frequencies can be zero
- The solution is to use *probability estimates*

Variable values

n_c number of occurrences
n total number of examples
p apriori probability assuming uniform probability
m equivalent sample size

Probability estimates

$$\frac{n_c + m \times p}{n + m}$$

Example: Probability estimates when $m = 4$

$$P(Outlook = Sunny \mid PlayTennis = No) = \frac{3 + 4 \times \frac{1}{3}}{5 + 4}$$