Classification: Model assessment and cross-validation

Rui Zhu

k nearest neighbours (kNN)

- kNN assigns the instance with features $\mathbf{x}_0$ to the class with the largest conditional probability.

$$\Pr(Y = j | X = \mathbf{x}_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

- j: class j, $j = 1, 2, \ldots, C$
- k: number of nearest neighbours
- i: index of instance
- $\mathcal{N}_0$: the nearest neighbours of $\mathbf{x}_0$
- $I(\cdot)$: indicator function

kNN

Now we know how kNN works. What's next?

- How do we know whether this kNN classifier is good or not to classify a dataset given k?
- How does k affect the classification result? How to choose k?

Model assessment: performance measure

Now we have the kNN classifier. How do we assess whether the classifier is good or not?
For regression:

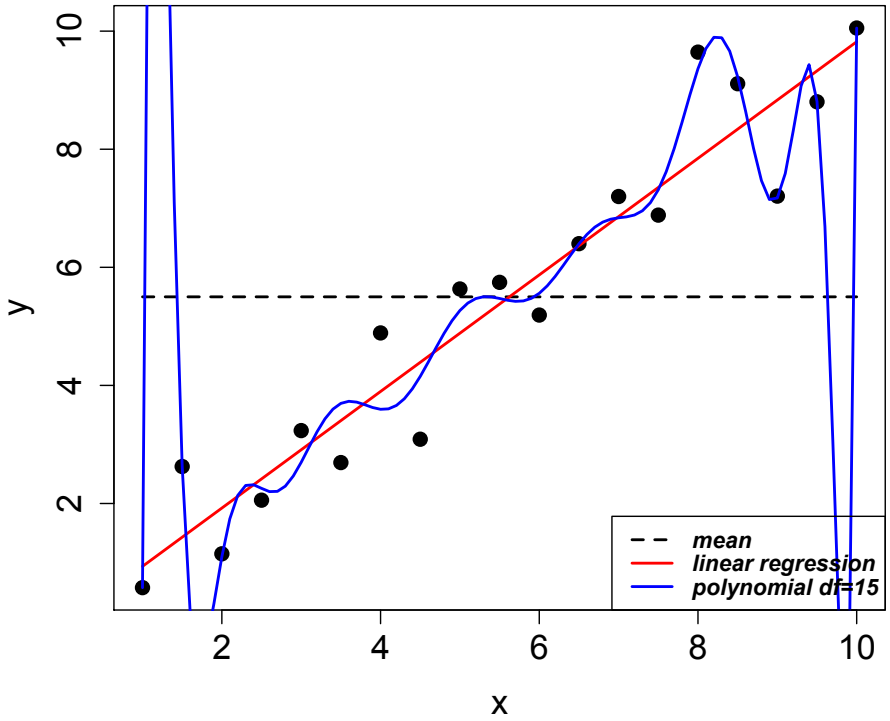
- Mean square error (MSE):

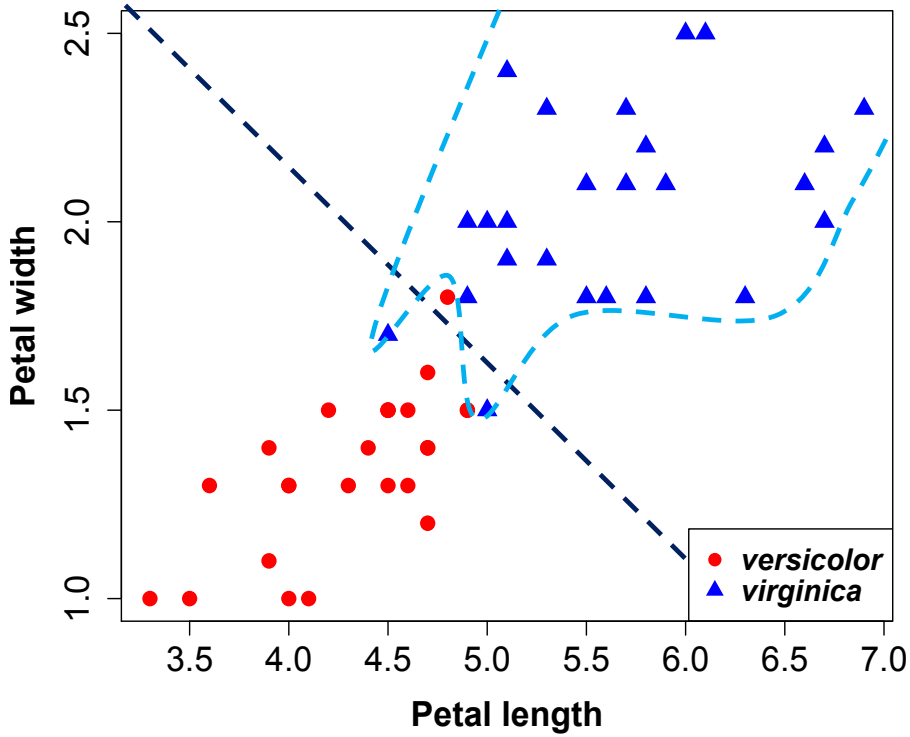
$$\text{MSE}_{Tr} = \frac{1}{N_{Tr}} \sum_{i \in Tr} (y_i - \hat{f}(\mathbf{x}_i))^2$$

$$\text{MSE}_{Te} = \frac{1}{N_{Te}} \sum_{i \in Te} (y_i - \hat{f}(\mathbf{x}_i))^2$$

- We should use MSE_{Te}.

Model assessment: performance measure





Model assessment: performance measure

For classification:

- Error rate:

$$\text{Err}_{Te} = \frac{1}{N_{Te}} \sum_{i \in Te} I(y_i \neq \hat{f}(\mathbf{x}_i))$$

- Classification accuracy:

$$\text{Acc}_{Te} = \frac{1}{N_{Te}} \sum_{i \in Te} I(y_i = \hat{f}(\mathbf{x}_i))$$

- $\text{Err}_{Te} + \text{Acc}_{Te} = 1$.
- Use either Error rate or Classification accuracy.

Model assessment: performance measure

A summary of the classification and assessment process:

- We have a dataset $\mathbf{X}$ with N instances and each of them has p features. We also know all their labels $\mathbf{y}$.
- Divide the dataset to a training set with N_{tr} instances and a test set with N_{te} instances ($N_{tr} + N_{te} = N$). No overlapping in the training set and test set.
- Train (fit) a classifier using the training set (with both labels and features).
- Predict the labels of the test set using the trained classifier and the features of the test set (pretend we don't know their labels in this step).
- Calculate the performance measure of the test set: compare the predicted labels and the true labels (ground truth).

Model assessment: performance measure

- Suppose we have a dataset with $N = 20$.
- We separate the dataset to a training set with $N_{tr} = 15$ and a test set with $N_{te} = 5$.
- We use a $3NN$ classifier to predict the labels of the test set.
- Suppose the true class labels of the test set are in a vector $\mathbf{y}_t = [0, 0, 0, 1, 1]^T$.
- We have five predicted class labels in a vector $\hat{\mathbf{y}}_t = [0, 1, 0, 0, 1]^T$.
- Then we have

$$\text{Err}_{Te} = \frac{1}{5}(0 + 1 + 0 + 1 + 0) = 0.4$$

$$\text{Acc}_{Te} = \frac{1}{5}(1 + 0 + 1 + 0 + 1) = 0.6$$

More on model assessment

Consider the Caravan data:

- 85 features that measure demographic characteristics $p = 85$
- 5,822 individuals $N = 5822$
- class label: Purchase, whether or not a given individual purchases a caravan insurance policy.

Only 6% *people purchase a caravan insurance policy. The two classes are very imbalanced!*

More on model assessment

- Given a classification method, we can obtain Err_{Te} or Acc_{Te} for a training/test split.
- Fitting a kNN model with $k = 1$, we can obtain $\text{Err}_{Te} = 0.118$

$$\text{Err}_{Te} = (50 + 68)/(873 + 50 + 68 + 9) = 0.118$$

for 1000 test instances (the true labels of the test set have 6% Purchase=Yes).

- Confusion matrix:

		True label	
		No	Yes
Predicted	No	873	50
label	Yes	68	9

More on model assessment

- Something seems weird: if we don't use any model and predict all Yes as No, we only have $\text{Err}_{Te} = 0.06$.
- This is a typical problem of assessing model quality using overall error rate/classification for imbalanced data.

More on model assessment

- In this example, we may care more about the accuracy of predicting people would like to buy the insurance.

		True label	
		No	Yes
Predicted label	No	873	50
	Yes	68	9

$$9/(68+9) \approx 0.117$$

- If we visit every customer (873+50+68+9), we have 6% (59/1000) probability to meet people who want the insurance.
- If we first predict who would like to buy (68+9) and only visit these customers, we have 11.7% probability to meet people want the insurance.
- Predicting before visiting makes things more efficient.

More on model assessment

Say Purchase=Yes is Positive and Purchase=No is Negative,

- True positive (TP): number of Purchase=Yes that are predicted as Purchase=Yes, 9
- True negative (TN): number of Purchase=No that are predicted as Purchase=No, 873
- False positive (FP): number of Purchase=No that are predicted as Purchase=Yes, 68
- False negative (FN): number of Purchase=Yes that are predicted as Purchase=No, 50

		True label	
		No (N)	Yes (P)
Predicted label	No (N)	TN	FN
	Yes (P)	FP	TP

More on model assessment

Popular performance measures:

- Accuracy ACC $= (\text{TP}+\text{TF})/(\text{TP}+\text{FP}+\text{TN}+\text{FN})$
- Sensitivity/true positive rate $\text{TPR} = \text{TP}/(\text{TP}+\text{FN})$
- Specificity/true negative rate $\text{TNR} = \text{TN}/(\text{TN}+\text{FP})$
- Precision/positive predictive value $\text{PPV} = \text{TP}/(\text{TP}+\text{FP})$
- *Think about your task and data: what do you care more about? Choose measurements based on this to evaluate the model.*

More on model assessment: ROC curve

Receiver operating characteristics (ROC) curve: a graphical representation of sensitivity and specificity with different classification thresholds.

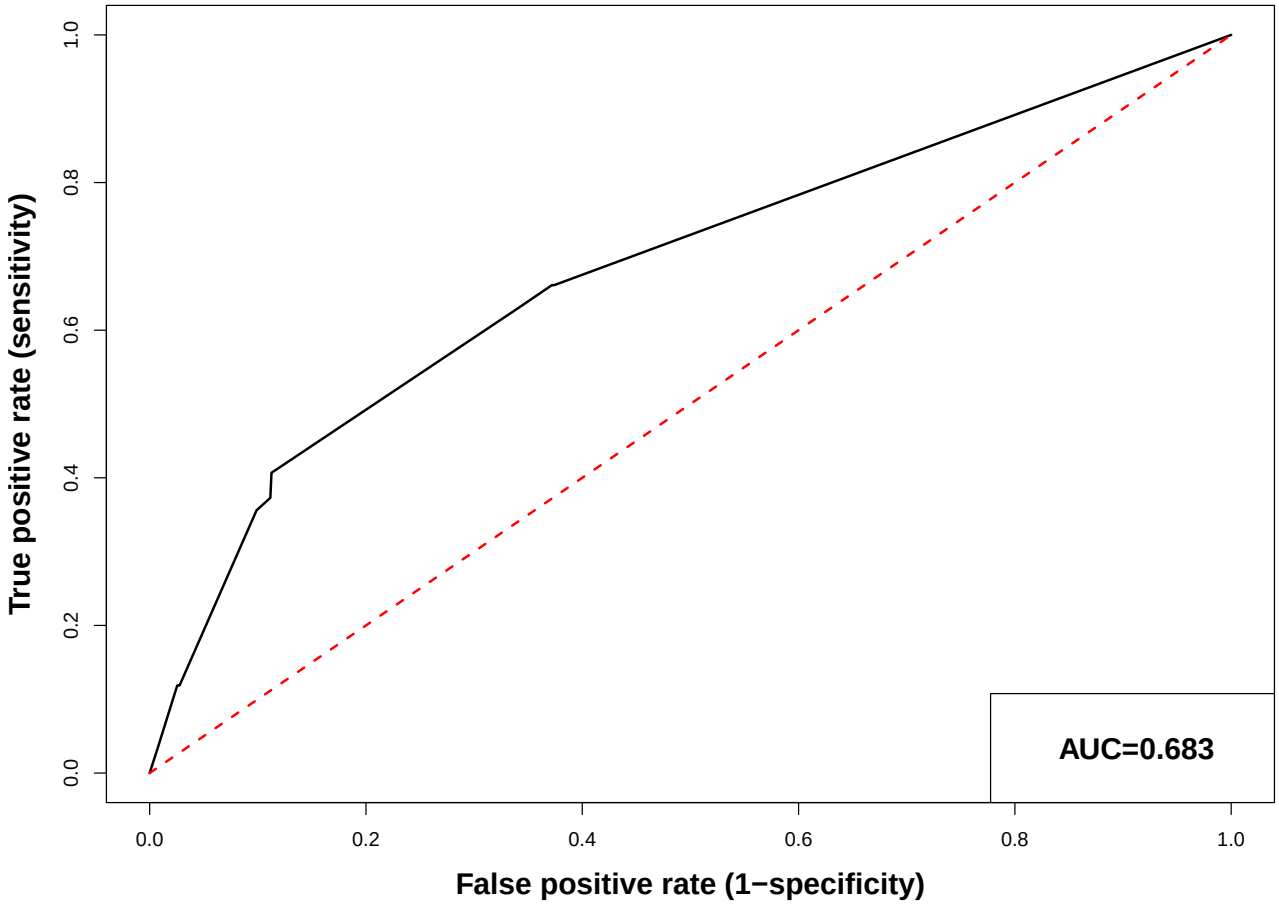
- kNN assigns the instance with features $\mathbf{x}_0$ to the class with the largest conditional probability.

$$\Pr(Y = j|X = \mathbf{x}_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

- If we have two classes, i.e. $j = 1, 2$, then we assign $\mathbf{x}_0$ to class j if $\Pr(Y = j|X = \mathbf{x}_0) > 0.5$.
- We use 0.5 as the threshold. Other thresholds are also possible, e.g. 0.2, $0.3 \ldots$

More on model assessment: ROC curve

9NN on the Caravan data:



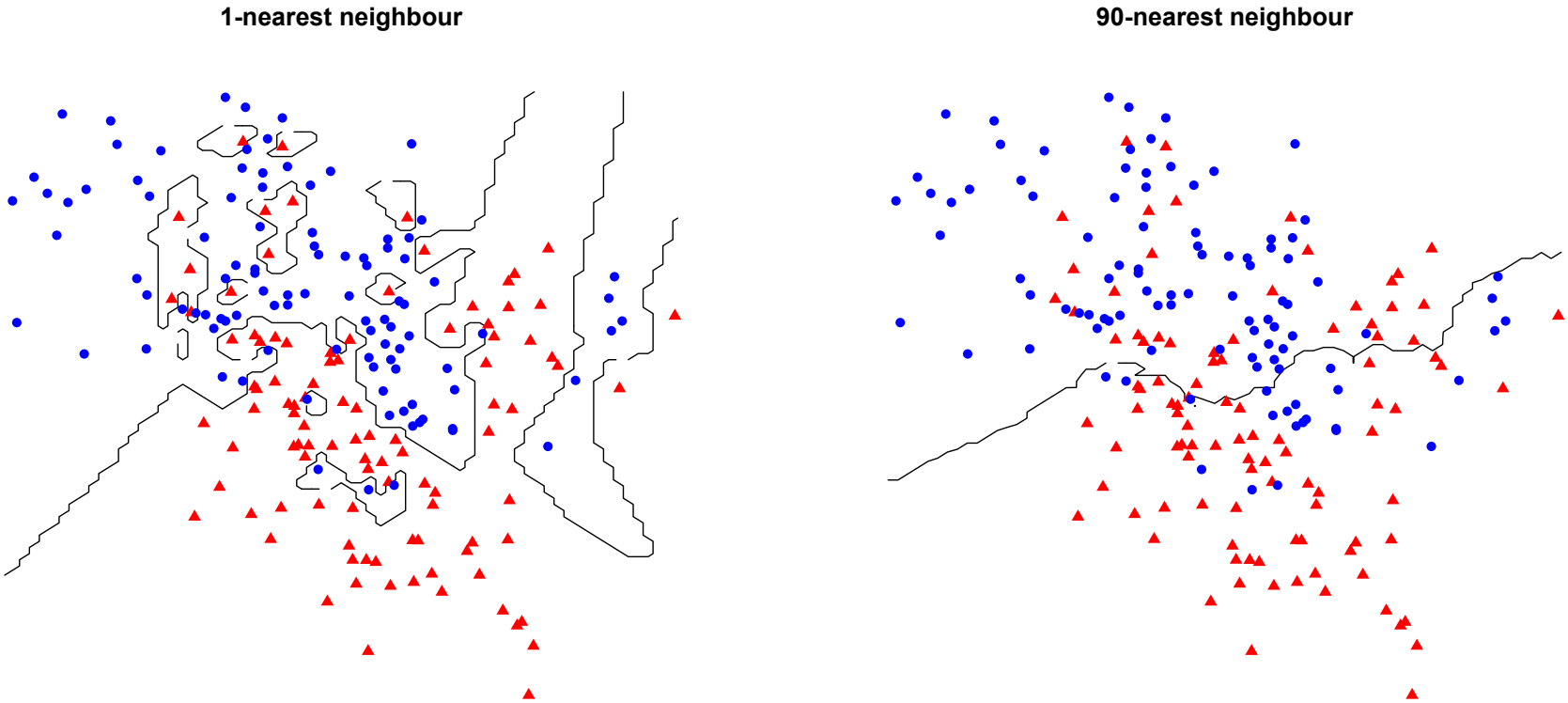
More on model assessment: ROC curve

Area under the curve (AUC): the overall performance of a classifier, summarised over all possible thresholds.

The larger the AUC, the better the classifier.

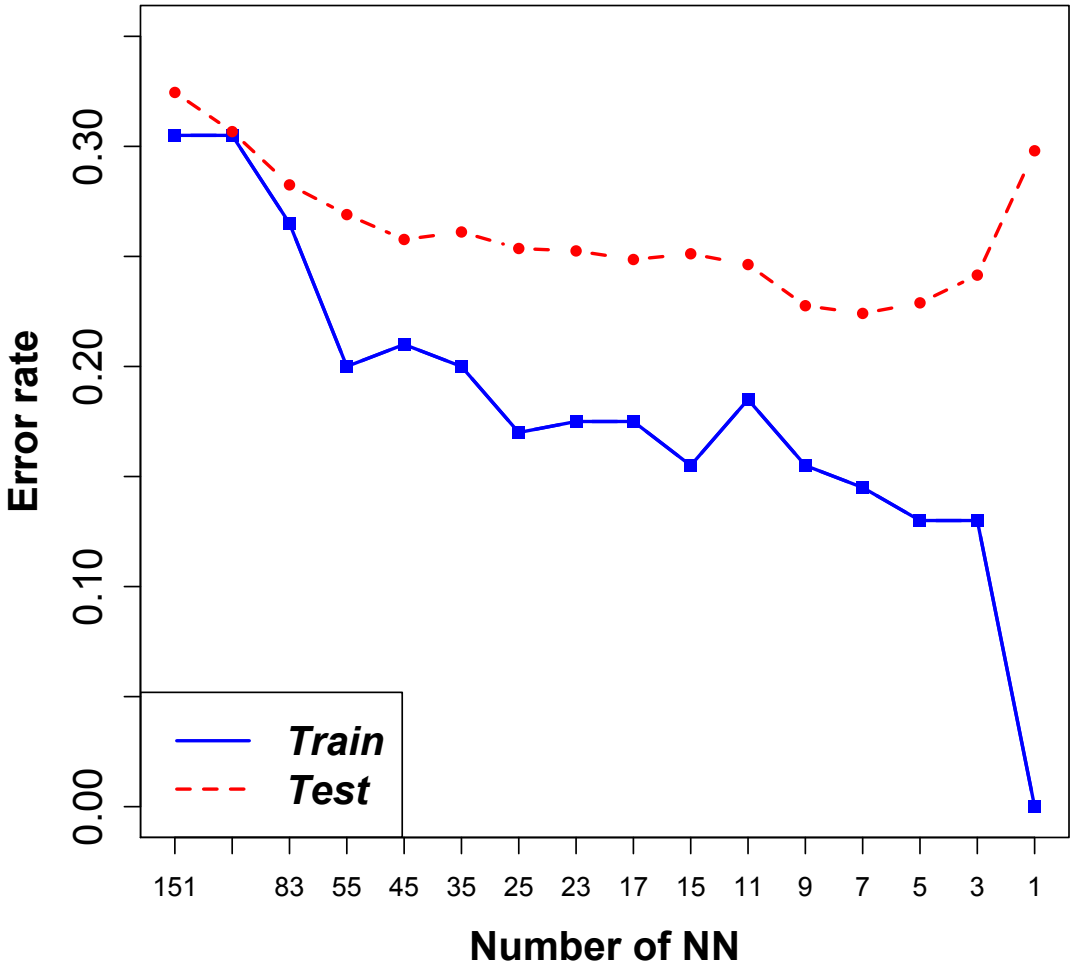
kNN: the effect of k

As k increases, the classification model becomes less flexible.



Which k is good?

Which k is good?



How to choose/tune k?

- k is the tuning parameter of kNN classifier.
- It is important to use proper tuning parameter.
- We can't use the test set to tune k: that's cheating. We have to make the test set independent from the whole training process.
- We can't use the training error.
- What to do?

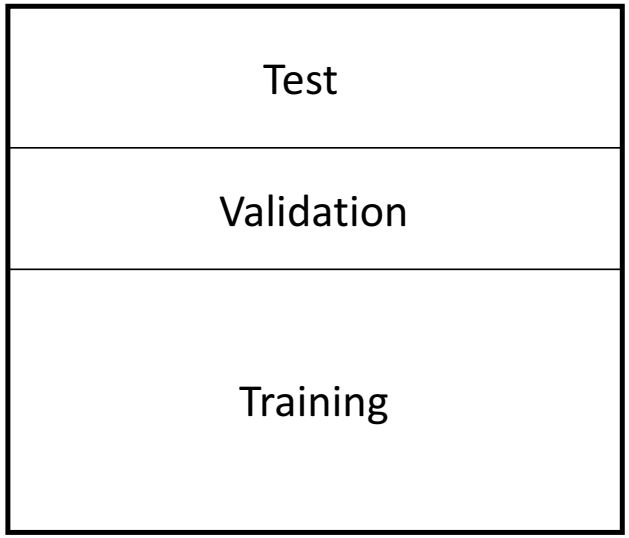
How to choose/tune k?

- We further divide the training data to training and validation set: we separate some instances to validate the classifier trained based on the (training data - validation data). Calculate the error rate of the validation set.
- We can repeat the above step several times and obtain an average error rate.
- Select k with the minimum error rate.

How to choose/tune k?

Training+Validation: tune *k*

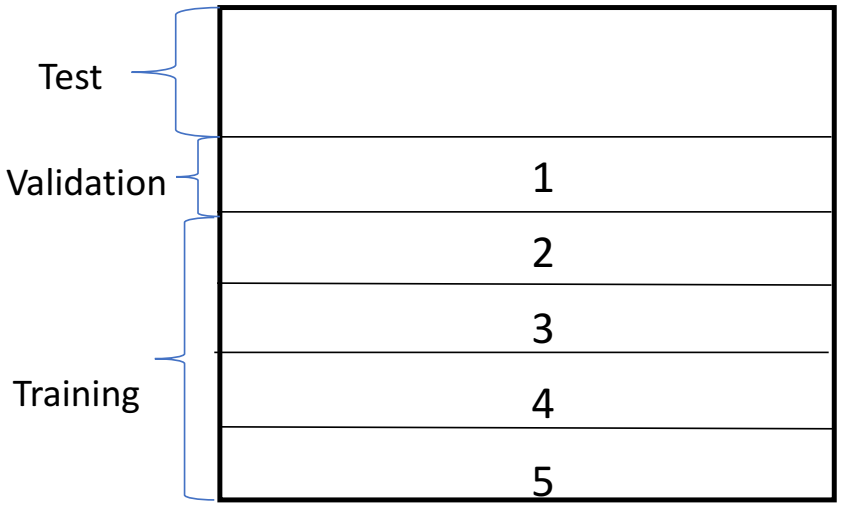
Test: model assessment



How to choose/tune k?

- K-fold cross-validation on the training data and select k with the minimum error rate.
- Note that this K is different from that in kNN.
- Divide the training data to K equal parts.

5-fold cross-validation:



How to choose/tune k: cross-validation

- For the kth part, we train the classifier on the other $K-1$ parts of the data. Calculate the performance measure of this classifier by predicting labels of the kth part of the data, e.g. Err_k. We repeat this procedure K times.
- $$\text{CV}_{(K)} = \frac{1}{K}\sum_{k=1}^{K}\text{Err}_k$$
- Leave-one-out cross-validation (LOOCV): $K = N$.
- $$\text{CV}_{(N)} = \frac{1}{N}\sum_{i=1}^{N}\text{Err}_i$$

How to choose/tune k: cross-validation

- Divide the data to training and test set
- For a specific value of k, e.g. $k = 3$, we do K-fold cross-validation (e.g. $K = 10$) on the training data.
- Obtain 10 error rates for $k = 3$. Calculate average error rate $\text{CV}^3_{(10)}$.
- For another value of k, e.g. $k = 5$, we can have $\text{CV}^5_{(10)}$.
- If we want to determine k from a set $\{1, 3, 5, 7, 9\}$: calculate $\text{CV}^1_{(10)}$, $\text{CV}^3_{(10)}$, $\text{CV}^5_{(10)}$, $\text{CV}^7_{(10)}$ and $\text{CV}^9_{(10)}$.
- Select the minimum $\text{CV}^{k^*}_{(10)}$ and have that k^*.
  - $\text{CV}^1_{(10)} = 0.4$, $\text{CV}^3_{(10)} = 0.3$, $\text{CV}^5_{(10)} = 0.2$, $\text{CV}^7_{(10)} = 0.3$ and $\text{CV}^9_{(10)} = 0.4$: $k^* = 5$.
- Use k^* in kNN to classify the test set.
- Calculate the classification measure of the test set.

How to choose/tune k: cross-validation

- Divide the data to training and test set
- Put the values of k that we aim to select from in a vector, e.g. $\mathbf{k} = (1, 3, 5, 7, 9)^T$
- for(i in $1 : length(\mathbf{k})$) $\{$
  - Do K-fold cross-validation (e.g. $K = 10$) on the training data, with a specific value $\mathbf{k}[i]$
  - Calculate average error rate $\text{CV}_{(10)}^{\mathbf{k}[i]}$.
  - Store this average error rate as the ith value of a vector **err**: $\mathbf{err}[i] = \text{CV}_{(10)}^{\mathbf{k}[i]}$.

  $\}$
- Find the smallest average error in **err** and use the corresponding k as k^*, i.e. the one we tuned from the training set.
- Use k^* in kNN to classify the test set.
- Calculate the classification measure of the test set.

Classification process

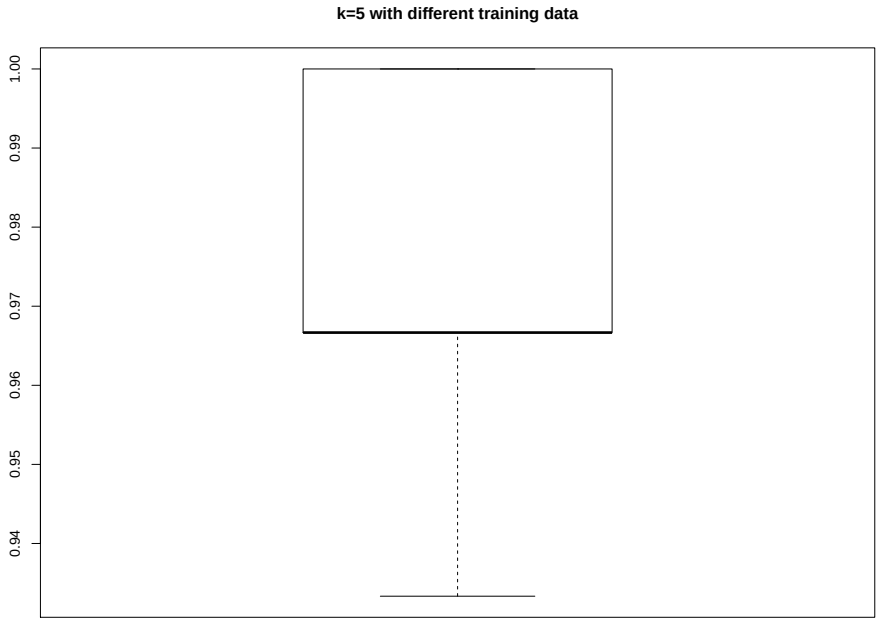
The classification process involving parameter tuning:

- We have a dataset $\mathbf{X}$ with N instances and each of them has p features. We also know all their labels $\mathbf{y}$.
- Divide the dataset to a training set with N_{tr} instances and a test set with N_{te} instances ($N_{tr} + N_{te} = N$). No overlapping in the training set and test set.
- Train (fit) a classifier using the training set (with both labels and features). Tune the parameters using K-fold cross-validation.
- Predict the labels of the test set using the trained classifier and the features of the test set (pretend we don't know their labels in this step).
- Calculate the performance measure of the test set: compare the predicted labels and the true labels (ground truth).

The effect of using different training data: iris example

We randomly select 10 instances for each class as the test set. In the rest data, we randomly select 25 instances for each class as the training set and get the classification accuracy. We repeat the second step 50 times and get 50 classification accuracies, as shown in the following boxplot.

Fixed test set, different training set with $k = 5$:



How to get a more reliable assessment of a classifier?

With different training set to train the same classifier, we can get different classification results on the same test set.

Classification accuracy based on only one training/test split is not a very reliable assessment of the classifier.

We can get an average peformance measure by repeating the classification process A times with different training and test sets.

Usually, there are two ways to do this:

- Repeat the radom split of training/test set A times.
- Cross-validation.

We can get N classification accuracies and analyse the properties of these classification accuracies: sample mean (location), sample standard deviaion (variation), etc.

How to get a more reliable assessment of a classifier?

- We have a dataset $\mathbf{X}$ with N instances and each of them has p features. We also know all their labels $\mathbf{y}$.

Repeat the following steps A times:

- Divide the dataset to a training set with N_{tr} instances and a test set with N_{te} instances ($N_{tr} + N_{te} = N$). No overlapping in the training set and test set.
- Train (fit) a classifier using the training set (with both labels and features). Tune the parameters using K-fold cross-validation.
- Predict the labels of the test set using the trained classifier and the features of the test set (pretend we don't know their labels in this step).
- Calculate the performance measure of the test set: compare the predicted labels and the true labels (ground truth).

How to get a more reliable assessment of a classifier?

A boxplot is a good graphical summary of the N classification accuracies: