COMP309 in Week 08, 2024

measuring performance (part 2)

Marcus Frean marcus@ecs.vuw.ac.nz

this week

Concepts of Performance Metrics

- Performance metrics vs loss function

Some classification Metrics

- Accuracy, true +ve/-ve rates, etc.
- ROC curve and AUC
- “log loss” as a metric, if learner outputs probabilities

Revision of train / test / validation

Some Regression Metrics

- MSE, etc

Some Clustering Metrics

- Silhouette Score, etc

previously

this lecture

log loss

say we have a classifier, which outputs a probability for each class (ie. a sensible classifier😀), for any given input pattern.

eg: one-hot encoding of target is $t=(0,0,1)$ and classifier says $y=(0.1, 0.5, 0.4)$.
For this classifier, what's the probability of getting the correct answer?
$\Pr(\text{correct}) = 0.4$, but in general? (i.e. in terms of t and y)

It does seem odd to write it this way, but can you see it would work out correct?

$$\Pr(\text{correct}) = \prod_{\text{class } i} (y_i)^{t_i}$$

Hint: $x^0 = 1$ and $x^1 = x$, for any x

log loss – for a classifier that outputs probabilities

$$\Pr(\text{correct}) = \prod_{\text{class } i} (y_i)^{t_i}$$

and take logs

$$\log \Pr(\text{correct}) = \sum_{\text{class } j} t_j \log y_j$$

so the chance of correctly guessing
an *entire labelled dataset* is:

$$\Pr(\textit{all}\ \text{correct}) = \prod_{\text{item } i} \prod_{\text{class } j} (y_{ij})^{t_{ij}}$$

and take logs

$$\log \Pr(\textit{all}\ \text{correct}) = \sum_{\text{item } i} \sum_{\text{class } j} t_{ij} \log y_{ij}$$

This is the **log likelihood**
the classifier would
generate the right answers
(on the labelled data set)

≤ 0
zero *iff* perfectly correct

and finally:

"Log loss" = - log likelihood !!!
≥ 0, so we minimise it

log loss – for a classifier that outputs probabilities

is BOTH
- a performance metric
  (used on validation or test set); AND
- a loss function
  (used to optimise/learn a training set)

$$\text{log loss} = -\sum_{\text{item } i}\sum_{\text{class } j} t_{ij} \log y_{ij}$$

- as a performance metric:

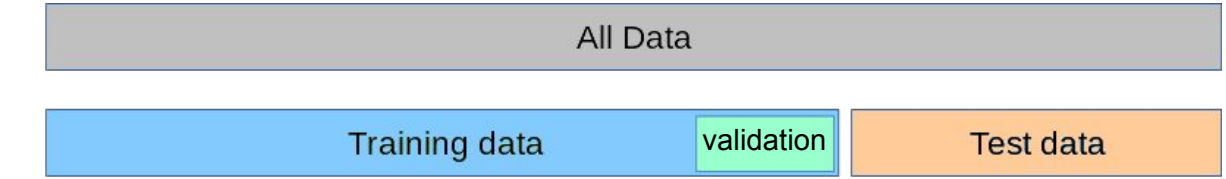
Accuracy is not always a good indicator because of its “crisp” yes or no nature.

Log loss takes into account the uncertainty of your prediction based on how much it varies from the actual label. This gives us a more nuanced view into the performance of our model. Often used as an evaluation metric in Kaggle competitions.

- as a loss function:

virtually always the basis of learning in neural nets (where we can use its gradient)

train, validation, and test



training: for the main learning process

validation is for **decisions**

e.g. between hyperparameter values, between models, between features...

test data is for **reporting**

use log loss

use log loss, or accuracy, or ?

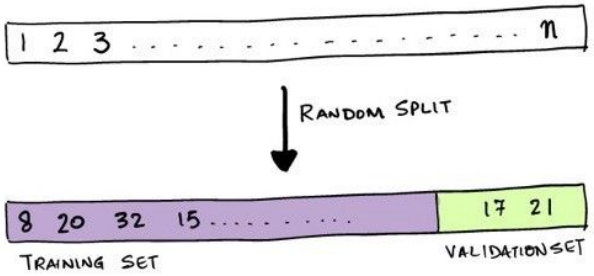
use accuracy, etc etc...

Question: which metric should each of these processes use?

model selection and hyperparameter tuning:

Using a validation set of “hold out” data

we need a **proxy** for out-of-sample data



✅ simple

✅ cheap

❌ high variance – the estimate depends on the (random) choice of validation set

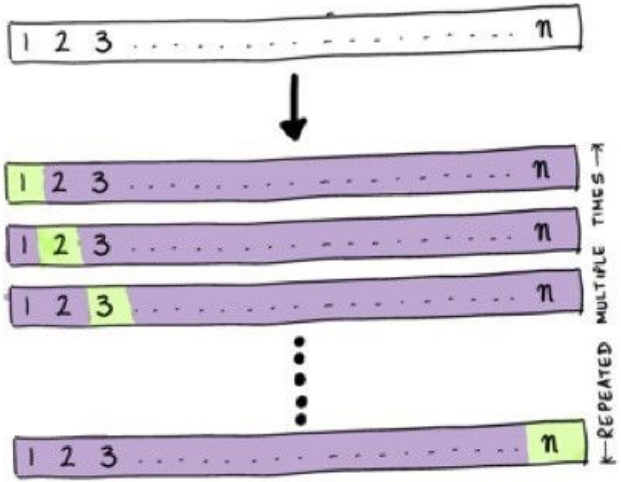
Could make validation set a larger ratio but...

❌ wastes training data

⇒ *don’t* use on a small dataset

model selection and hyperparameter tuning:

Leave-one-out Cross-validation (LOOCV)



✅ very accurate guess at what test performance will be

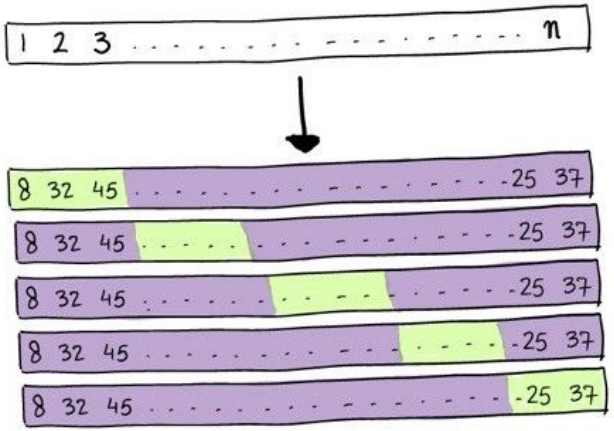
✅ uses virtually all the data for each split

❌ expensive! - we do n fits

⇒ *Don't* use if fitting is expensive (e.g. big data + slow training)

model selection and hyperparameter tuning:

k-fold cross validation



✅ quite accurate guess

✅ uses *most* data in each split

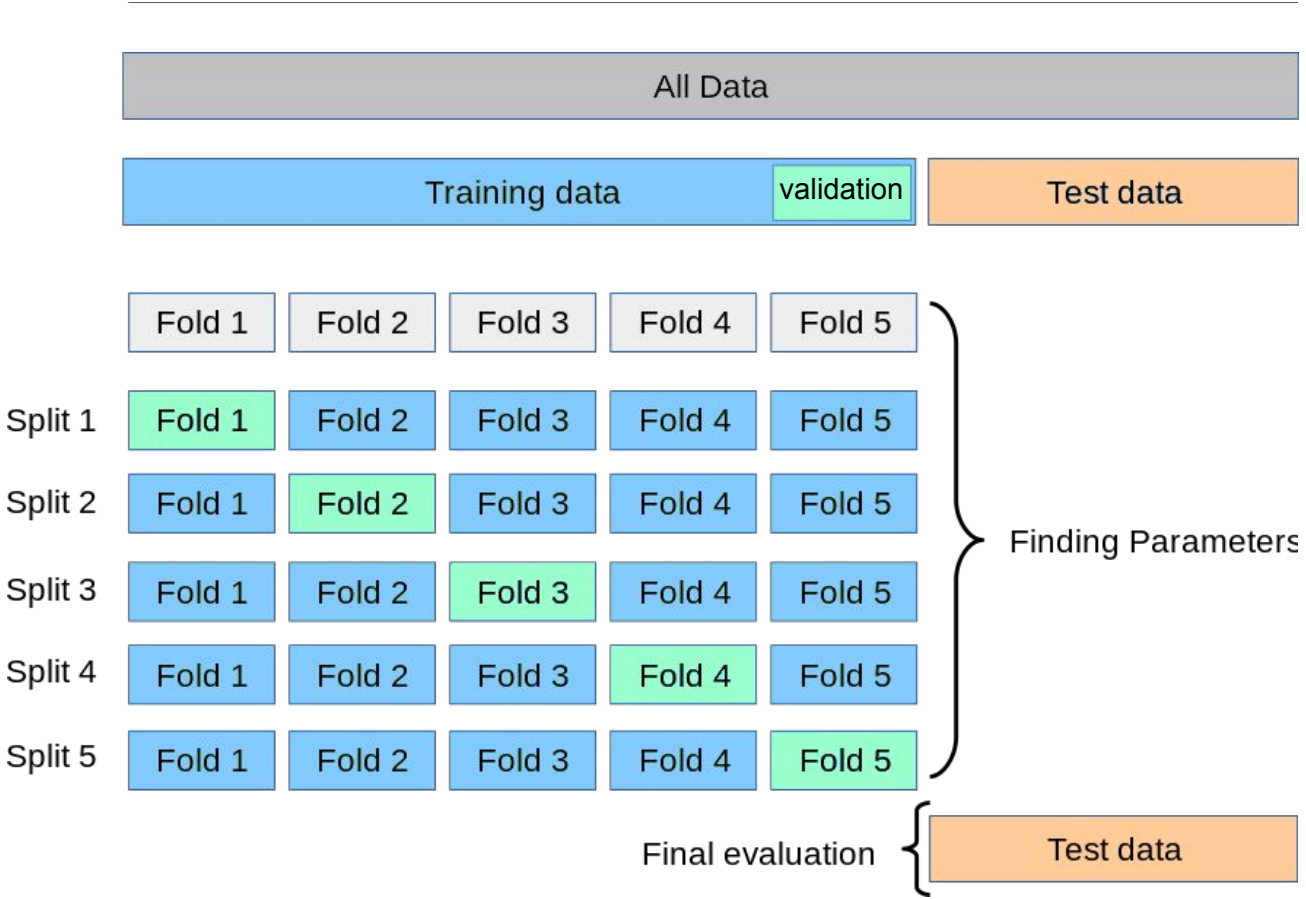
✅ not too many splits (n/k)

⇒ A good compromise when fitting is expensive

Notice that as $k \to n$ this reverts to LOOCV. Here, k=5. k=10 is common.

model selection and hyperparameter tuning:

k-fold cross validation

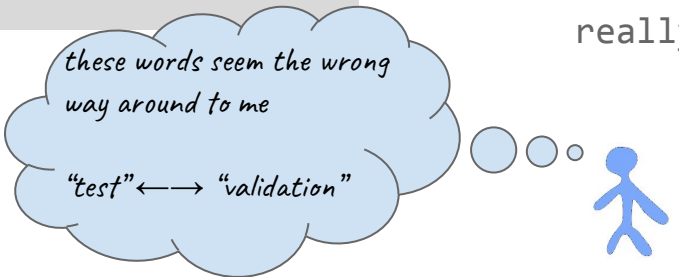


don't let validation data "leak" into evaluation

from earlier:

TWO REASONS we might want to withhold some data from a given training process:

1. to decide hyperparameter settings, and do model selection, i.e. part of learning
2. OR to evaluate the final trained model



Either on its own could use Cross-Validation

Q: Can we use Cross-Validation both to decide on hyperparameters/model AND for reporting?!
Beware you can't use the same data for both.

- In this course, your task is usually to build the best classifier, and there are no rewards for *telling us how good you think* it is. Hence I don't think you need a "Test Set".
- Ergo if you split into train and test, and use the "test" for model selection etc, you're really using it as "Validation" data.

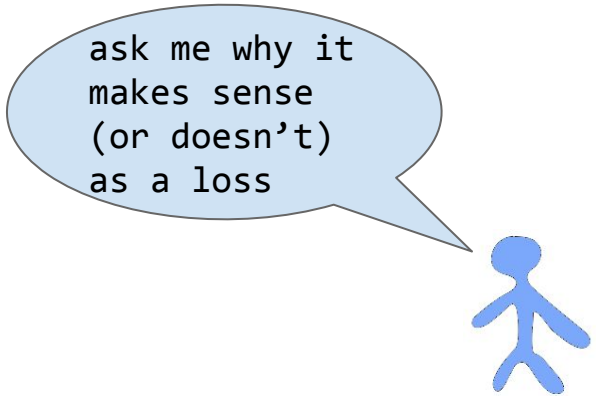
Regression Metrics - MSE ("mean squared error")

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2$$

where t is the target and y is the prediction (i.e. output of the regressor)

This is the most commonly used metric for regression

AND also the most common loss function



Regression Metrics - RMSE (*root* mean squared error)

In some ways the square root of MSE makes more sense...

In parameter space, the optima are in the same place
(i.e. $\sqrt{x}$ is monotonic in x, and $x>0$)

As a **metric**, it's in natural coordinates (the "size" of the "error")

But the shape of the "valley" is different, so

- gradients are different,
- path to convergence different.

However, as a **loss**, taking the square root rarely helps, in practice.

Regression Metrics - RSE (*relative* squared error)

$$RSE = \frac{\sum_{i=1}^{N}(t_i - y_i)^2}{\sum_{i=1}^{N}(\overline{y} - y_i)^2}$$

Imagine comparing two different regressions, using MSE.

One in cm, the other in inches.

MSE would tend to be bigger for the “cm” one, but this isn’t a useful comparison…

With RSE we *normalise* :
in effect, we divide out by the regressor’s own variability.

e.g. the RSE for the two regressions in the example would come out equal.

RSE is widely used and recommended. (as a metric)
A good regression model should have RSE < 1.

Regression Metrics - MAE (mean *absolute* error)

$$MAE = \frac{1}{N}\sum_{i} |t_i - y_i|$$

MAE is exactly the same as MSE except

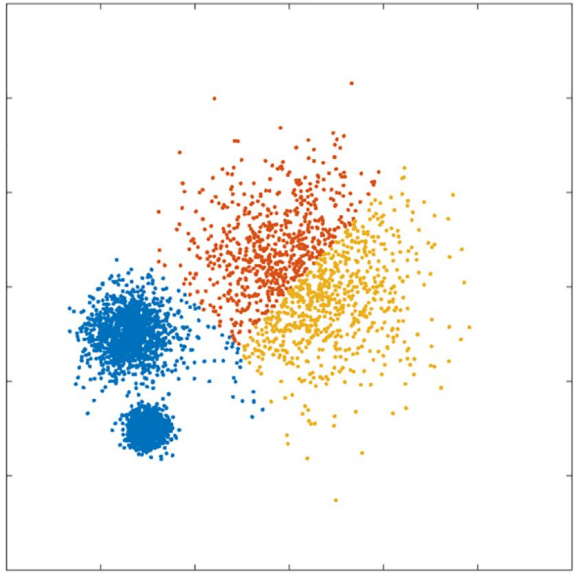
instead of taking the mean of the ~~square~~ of the error,

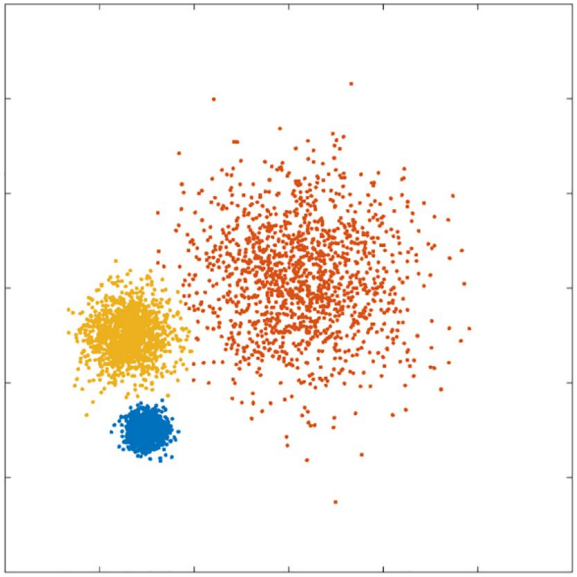
absolute value

NB: → less sensitive to outliers (since x^2 grows faster than $|x|$ does)

A metric for clustering – Silhouette score

what is it that makes that makes you prefer the second?



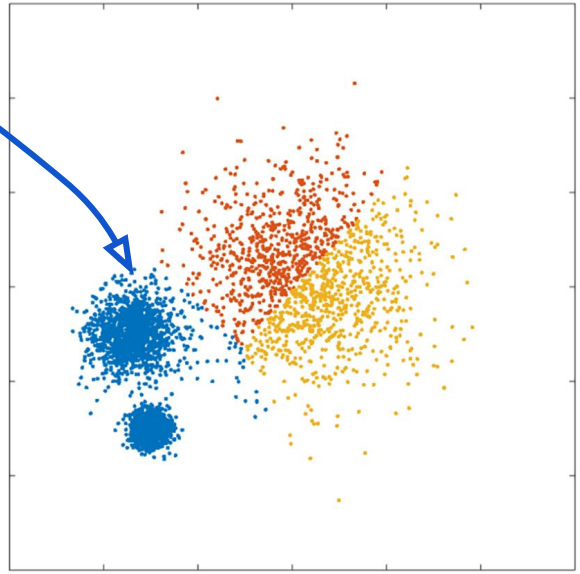


A metric for clustering – Silhouette score

For any single observation in particular

a = Mean distance between the observation and all other data points in the **same cluster** *(hope it's small)*

b = Mean distance between the observation and all other data points of the **nearest other** cluster *(hope it's big)*



Silhouette score for that point:

$$S = \frac{(b-a)}{max(a,b)}$$

S≈1: for points in a dense cluster that is distinct

S≈0: where clusters overlap, indistinct

S≈-1: clusters are wrongly assigned

Analysis using Silhouette scores

$$S = \frac{(b-a)}{max(a,b)}$$