Model Evaluation

Type	Lecture Notes
Reviewed	☐
Available Summary?	Done
# Week	3

Outline

- Parameters vs hyperparameters
- Data splitting
- Data leakage

▼ How do we get different models?

- **K-Nearest Neighbors:** Different choices of K
- **Decision trees:** Different choices of the number of levels/leaves
- **Polynomial Regression:** Polynomials with different degrees
- **Kernel methods:** Different choices of kernels
- **Regularized Models:** Different choices of regularization hyperparameter
- **SVM:** Different choices of kernel and kernel parameters

▼ Parameters vs. Hyperparameter

▼ Parameters:

- Control the model
- inferred from training data

▼ Hyperparameters:

- control the parameter
- special type of parameter
- not inferred from training data
- give explicitly

▼ Generalization

What we actually care about is how well the model will predict the new **unseen** data.

▼ Definition:

Model does a good job of correctly predicting class labels of **previously unseen** samples.

▼ Evaluating generalization requires:

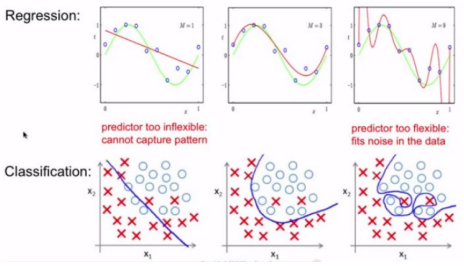
- unseen dataset
- correct labels for test set are known
- a quantitative measure (metric) of tendenct for model to predict correct labels.

▼ Model Complexity

Most learning algorithms have knobs that adjusts the model complexity:

- **Regression:** order of the polynomial
- **Naive Bayes:** number of attributes
- **Decision Trees:** Number of nodes in the tree
- **kNN:** number of nearest neighbors
- **SVM:** kernel type, cost parameter
- **Regularized Models:** Regularization hyperparameter

▼ Overfitting & Underfitting



▼ Overfitting:

- Predictor too complex
- Predictor overfits the training data → low training error, high test error

▼ Fix:

- getting more training examples might help
- try reducing features
- try less complex model (increase k in kNN, decrease polynomial degree in regression, etc.)

▼ Underfitting:

- Predictor too simplistic
- Not powerful enough to capture patterns in the data

▼ Fix:

- getting more training examples will not help
- get more features
- try hyperparameters that lead to more complex model (lower k in kNN, increase polynomial degree in regression, etc.)

▼ Model Comparing & Evaluation

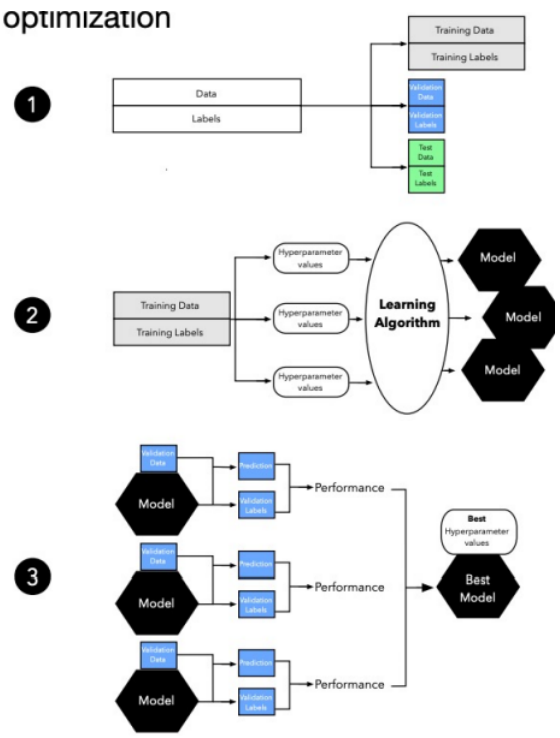
▼ Hold out method:

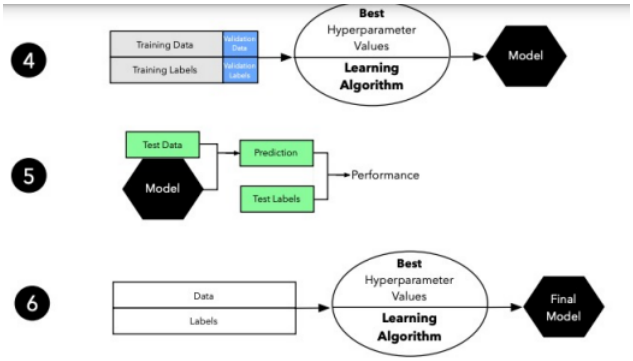
Split the data into three parts (training, validation, test) randomly.

▼ 3-way hold-out

- instead of **regular** holdout to avoid data leakage during hyperparameter optimization

▼ example scheme





▼ training set

- used to drive model building and parameter optimization

▼ validation set

- used to gauge status of generalization error
- results can be used to guide decision during training process
- other names: held-out data, development data

▼ test set

- used only for final assessment of model quality, after training + validation completely finished

▼ typical approach

1. learn a model using the training set
2. estimate the performance using the validation data
3. among the models, choose the one with best performance on the validation data
4. once model is decided on retrain the model on train + validation data
5. report final performance on test data

▼ stratified splitting

- some classification problems don't have a balanced number of examples for each class label
- it is desirable to split dataset into train and test sets in a way that preserves the same proportions of examples in each class as observed in the original dataset
- you might also consider stratification based on features

▼ Advantages:

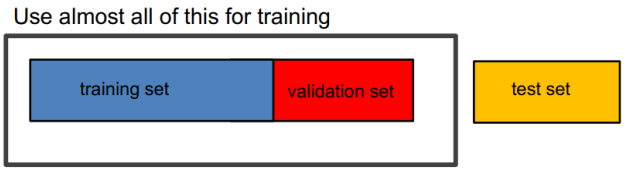
- very simple
- can then simply choose the method with the best validation-set score

▼ Disadvantages:

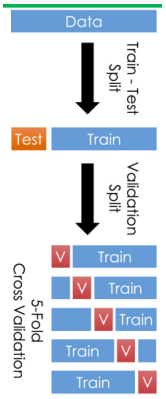
- wastes data:
  - in general, the more training data we have the better are the estimated parameters
  - the validation/test split may not be large enough to be representative. We might just get a lucky/unlucky split

▼ Cross Validation

- recycle the data



▼ cross validation scheme:



▼ Leave one out cross validation (LOOCV)

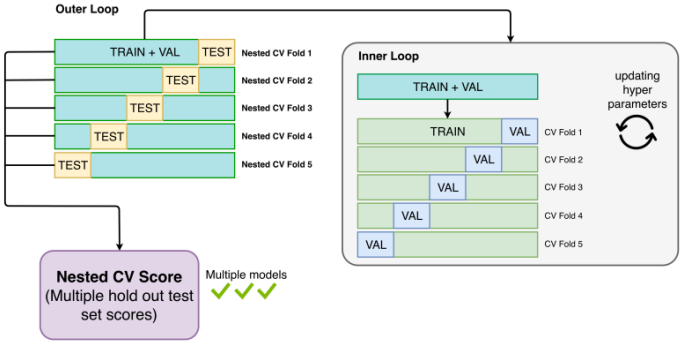
1. Let (xk,yk) be the kth example
2. Temporarily remove (xk,yk) from the dataset
3. Train on the remaining N-1 data points
4. Note your error on (xk,yk)
5. There are N data points. Do this N times. When you've done all points, report the mean error.

▼ Repeated cross-validation:

- ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

▼ Nested cross-validation:

- useful when you cannot afford to set aside test data



▼ Outer loop:

1. train each split with optimal parameters
2. average each split's test error

▼ Inner loop:

1. tune hyperparameters

▼ Final model:

- build the final model on the original data (train + val) with the hyperparameters, features and algorithms decided using the validation set
- you may also bag the different models learned in cross validation loops to form an ensemble model
- report the error on test set

▼ Model Selection

▼ Extra-sample error estimates

- Train/test split
- Cross-validation
- Bootstrap

▼ In-sample error estimates

- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)
- Minimum Description Length Principle (MDL)

▼ Baseline models

▼ **classification**

- *majority class baseline:*

  every data point is classified as the class that is most frequently represented in the training data

▼ **regression**

- mean or the median of the data used as the predicted target value

▼ Data Leakage (in model evaluation)

- introducing information about the target during that would not be legitimately available during the actual use
- if your model is too good to be true, it is probably due to giveaway features

> 💡 ***giveaway features***
>
> Even identifiers of the examples might leak information if they are not created randomly.
>
> - Imagine in data collection one class is collected before the other one. The example ids are given in the order of data collection. If you use the id in your classifier, the classifier will be able to pick it up

▼ obvious cases:

- including the label to be predicted as a feature
- including test data in the training data

▼ hidden confounding variables:

▼ image classification

- one class is taken in certain light conditions while other in another, or with different phones

▼ skin cancer classification

- the algorithm appeared more likely to interpret images with rulers as malignant

▼ however, data leakage can happen in many subtle ways:

- when information about the future that would not legitimately be available in actual use, is included in the training data

▼ sneaky data leakage

- An online store that sells movies, electronics and jewelry. Building a model to predict, based on past purchases, whether a customer is likely to buy jewelry. You gather a dataset containing purchase histories for every single customer who has ever shopped at the store, along with their total spend on each of the three product categories that the store sells. Any half decent model will learn this and make perfect predictions about the customers with 0 spend on movies and electronics

▼ leaking test information

- Performing pre-processing on the entire dataset whose results influence what is seen during training.
  - Computing parameters for normalizing and rescaling
  - Removing outliers and using the distribution of a variable across the entire dataset to estimate missing values in the training set
  - Feature selection on the entire dataset

▼ How to avoid data leakage:

▼ Exploratory data analysis:

- look for features very highly correlated with the target label or value

▼ Look for supring predictive results (looking for suprisingly good results):

- if your model evaluation results are substantially higher than the same or similar problems
- look closely at the instances or features that have most influence on the model

▼ Reliable but expensive:

- limited real world deployment of the trained model
- check if there's a big difference between the estimated performance and the actual results