HYPER PARAMETER SELECTION : VARIANCE REDUCTION

LECTURE 5
SECTION 2
JUNE 9th

INSTITUTE FOR APPLIED
COMPUTATIONAL SCIENCE
AT HARVARD UNIVERSITY

UNIVERSITY of
RWANDA

HYPER PARAMETER SELECTION

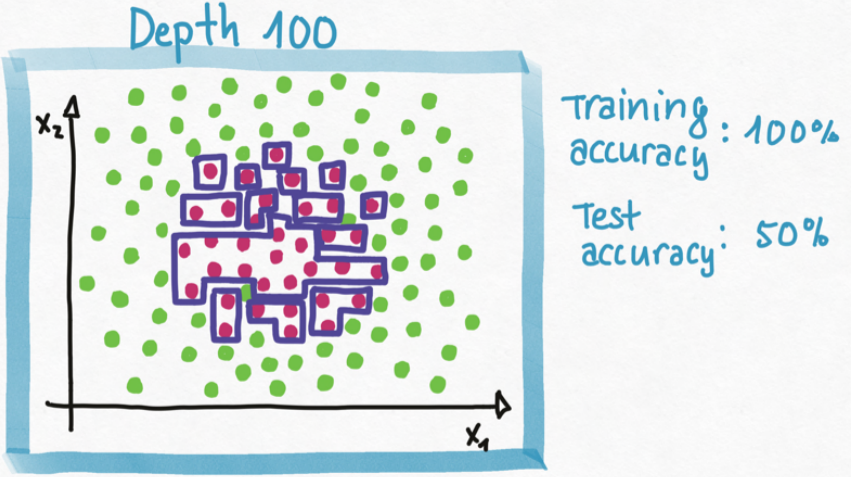
SELECTING HYPER-PARAMETERS:

The depth d of the decision tree model and the k of the KNN classifier are both hyperparameters – we need to choose them before training begins.

If we choose the depth or k that perform the best on the training set the we will likely overfit (choosing an high depth and small k).

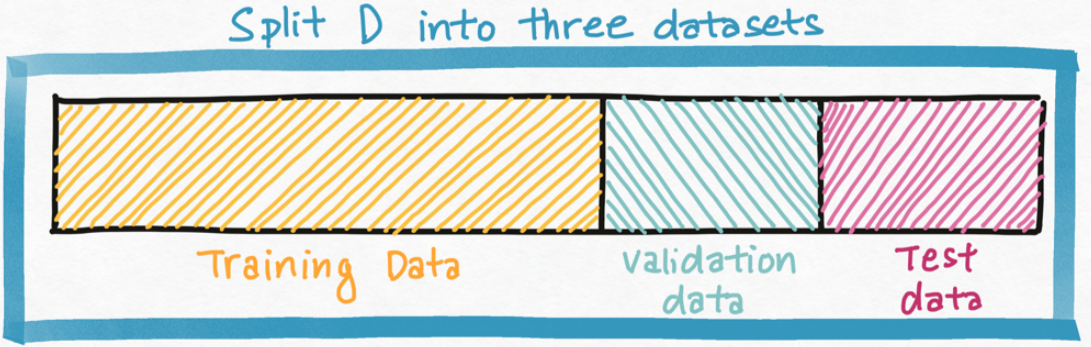
If we choose the depth or k that performs the best on the test set then we no longer have an held-out set of entirely new data to test our model

We need a new data set, different from train and test, to help choose hyper parameters!



VALIDATION DATASET:

In the same way that we hold out a portion of our given data D to create a test data set, we can hold out another part of D for hyperparameter tuning. This data set is called validation data.



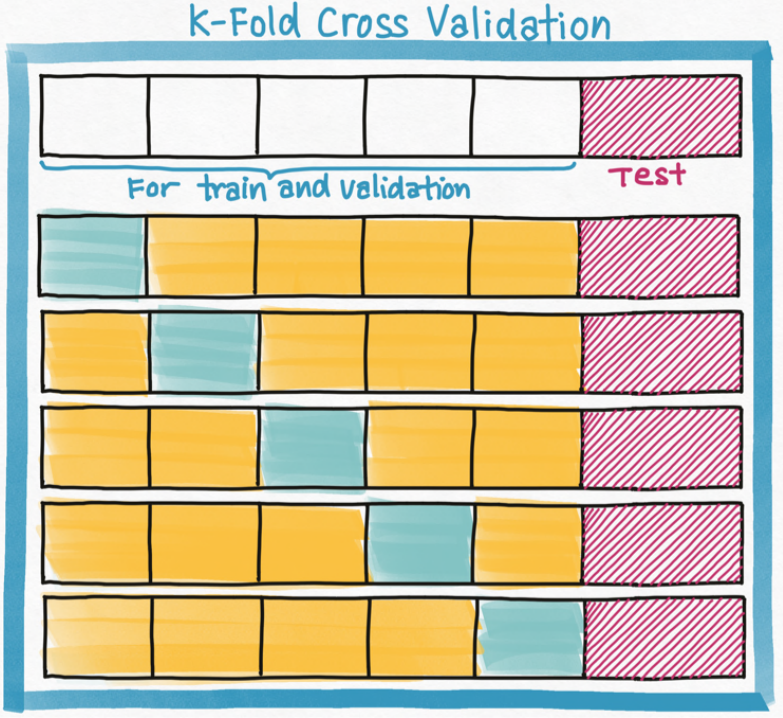
We fit M number of decision trees with depths $\{d_1, \ldots, d_M\}$ on the training set. We test each tree on the validation set. We select the depth d_m with the best performance on validation. We test the tree with depth d_m on the test data.

Same procedure to choose the best k for KNN.

K-FOLD CROSS-VALIDATION:

The danger of testing only once is that the test set may be too easy or too hard by chance. Ideally we'd want to have multiple testing or validation data sets. But where do we get all this new data?

If the total amount of data in D is small we cannot hold out enough data for multiple validation or test sets. We need to reuse data!



1. Divide D into two parts, use one for final testing. Divide the other into k-parts.
2. Repeat for i = 1, ..., k:
   A. hold out the k-th part
   B. train on the rest and evaluate on the ith part.
3. Average the validation performance for K-validation sets for each model.
4. Select the model with the best average validation performance.
5. Test the chosen model on the test set

K-FOLD CROSS VALIDATION IN SKLEARN.

```
[23] from sklearn.model_selection import GridSearchCV
     parameters = {'alpha':[1e-2, 1e-1, 1e0, 1e1, 1e2]}
     ridge = Ridge()
     tuned_ridge = GridSearchCV(ridge, parameters)
     tuned_ridge.fit(X, y)
```

Define hyperparameters you want to search over

Define your model

Define a grid search object to search over model hyper-parameters

Find the best hyperparameters

```
[29] tuned_ridge.best_params_

     {'alpha': 1.0}
```

.best_params_ is the best hyper parameter found

```
[30] tuned_ridge.best_estimator_

     Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=None,
           normalize=False, random_state=None, solver='auto', tol=0.001)
```

.best_estimator_ is the fitted model with the best hyperparameter.

```
[31] tuned_ridge.best_estimator_.coef_

     array([4.92821494, 7.78745937, 1.93451015, 7.92767123, 4.03614264,
            1.27093637])
```

.best_estimator_.coef_ is the array of parameters of the fitted model with the best hyperparameter.

VARIANCE REDUCTION

L_p REGULARIZATION:

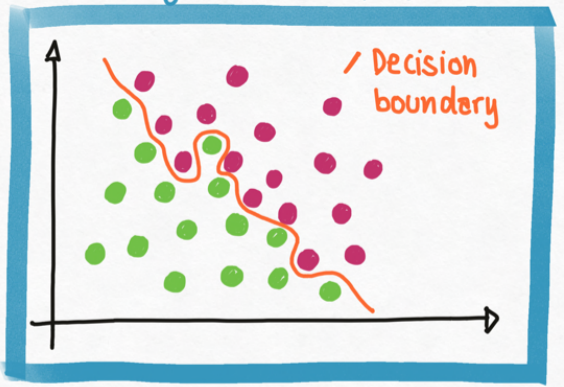
Logist regression with polynomial decision boundaries can overfit just as in the case of regression.

Just like in polynomial regression, we can penalize the parameters w of the polynomial to keep the values of w small.

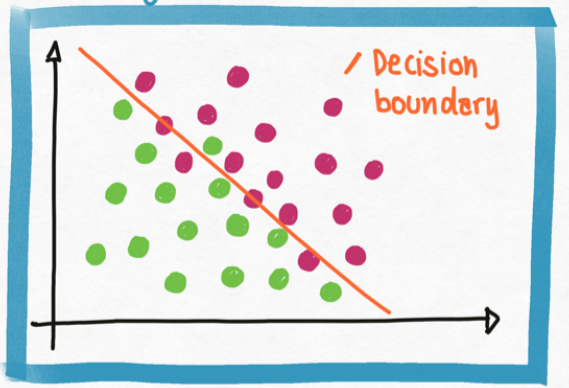
This changes our training objective to be:

$$w^* = \arg\min_w \underbrace{-\ell(w)}_{\text{negative log likelihood}} + \underbrace{\|w\|_2}_{\sum_i w_i}$$

Unregularized Classifier



Regularized Classifier



RANDOM FORESTS:

We've seen that ensembling many models reduces variance, and hence reduces generalization error. When we train many decision trees on bootstrap samples from D and we ensemble the trees, we get a random forest: