Machine learning

Presented by : Dr. Hanaa Bayomi

Lecture 12: Feature and model selection

Model

- How many features to have in the model
- Prediction accuracy vs model interpretability

Less number of features	More number of features
Easy to interpret	Difficult to interpret
Less likely to over fit	More likely to over fit
Low prediction accuracy	High prediction accuracy

Feature selection

Performance of Machine Learning model depend on

- Choice of algorithm
- Feature selection
- Feature creation
- Model selection

Top reasons to use feature selection are:

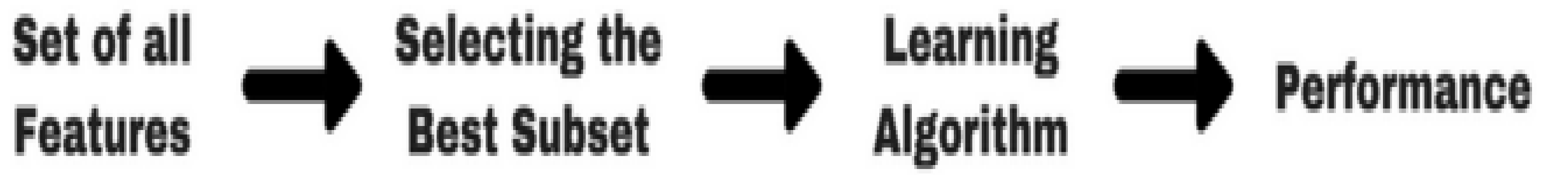
- It enables the machine learning algorithm to train faster.
- It reduces the complexity of a model and makes it easier to interpret.
- It improves the accuracy of a model if the right subset is chosen.
- It reduces overfitting.

Classification of FS methods

- **Filter (single factor analysis)**
  - Assess the relevance of features only by looking at the essential properties of the data.
  - Usually, calculate the feature relevance score and remove low-scoring features.
- **Wrapper**
  - Bundle the search for best model with the FS.
  - Generate and evaluate various subsets of features. The evaluation is obtained by training and testing a specific ML model.
- **Embedded**
  - Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are **regularization methods**.

Filter methods

- Filter methods are generally used as a *preprocessing step*. The selection of features is independent of any machine learning algorithms.
- Two steps (score-and-filter approach)
  1. assess each feature individually for its potential in discriminating among classes in the data
  2. features falling beyond threshold are eliminated



Filter methods

- **Advantages:**
  - easily scale to high-dimensional data
  - simple and fast
  - independent of the classification algorithm

- **Disadvantages:**
  - ignore the interaction with the classifier
  - most techniques are uni variate (each feature is considered separately)

Scores in filter methods

- **Distance measures**
  - Euclidean distance

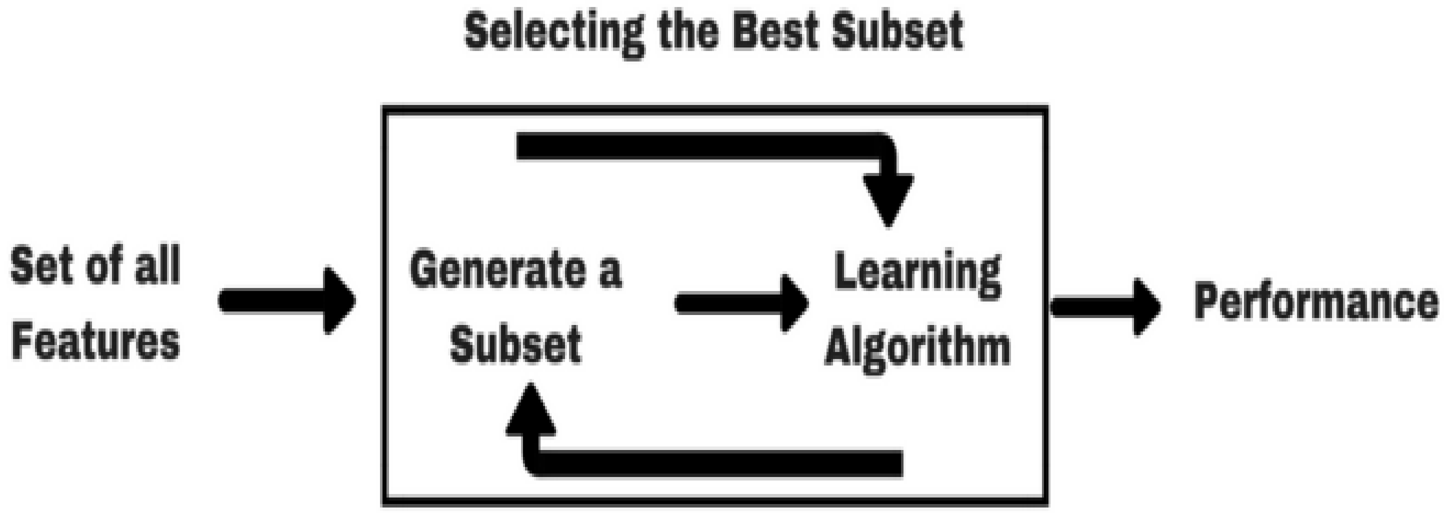
- **Dependence measures**

  - Pearson correlation coefficient
  - Chi-Square: It is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.

- **Information measures**
  - information gain
  - mutual information

Wrappers

- Search for the best feature subset in combination with a fixed classification method.
- The goodness of a feature subset is determined using cross-validation (*k*-fold, LOOCV) Leave-one-out cross-validation



Wrappers

- Advantages:
  - interaction between feature subset and model selection
  - take into account feature dependencies
  - generally more accurate

- Disadvantages:
  - higher risk of overfitting than filter methods
  - very computationally intensive

Wrappers Methods Examples

- **Subset Selection:** fit model with each possible combination of ***P*** features

  - Total number of models 2^P -1
  - if P=2 (Lets say X1,X2)

    Y=B0+B1X1

    Y=C0+C1X2

    Y=D0+D1X1+D2X2

  - If P>20 it is almost impossible to use subset selection method

Wrappers Methods Examples

- **Forward Selection:** Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.

- **Backward Elimination:** In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.

WRAPPERS METHODS EXAMPLES

Forward selection

```
Initialize s={}
Do:
       Add feature to s
       which improves K(s) most
While K(s) can be improved
```

Backward elimination

```
Initialize s={1,2,…,n}
Do:
       remove feature from s
       which improves K(s) most
While K(s) can be improved
```

	Backward	forward
	X1 **X2** X3 X4 X5	**X1**
	X1 X3 **X4** X5	**X1** X2
	X1 X3 **X5**	**X1 X2 X4**
	X1 X3 X5	**X1 X2 X4** X5
	X1 X3 X5	**X1 X2 X4 X5**

WRAPPERS METHODS EXAMPLES

Forward selection

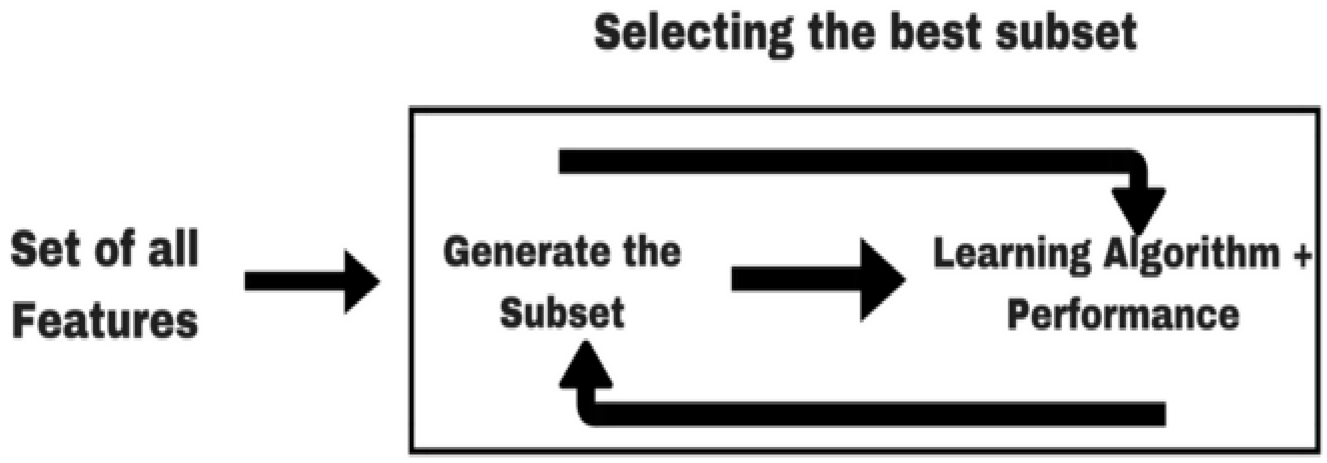
```
Initialize s={}
Do:
       Add feature to s
       which improves K(s) most
While K(s) can be improved
```

Backward elimination

```
Initialize s={1,2,…,n}
Do:
       remove feature from s
       which improves K(s) most
While K(s) can be improved
```

- Backward elimination tends to find better models
  - Better at finding models with interacting features
  - But it is frequently too expensive to fit the large models at the beginning of search

EMBEDDED



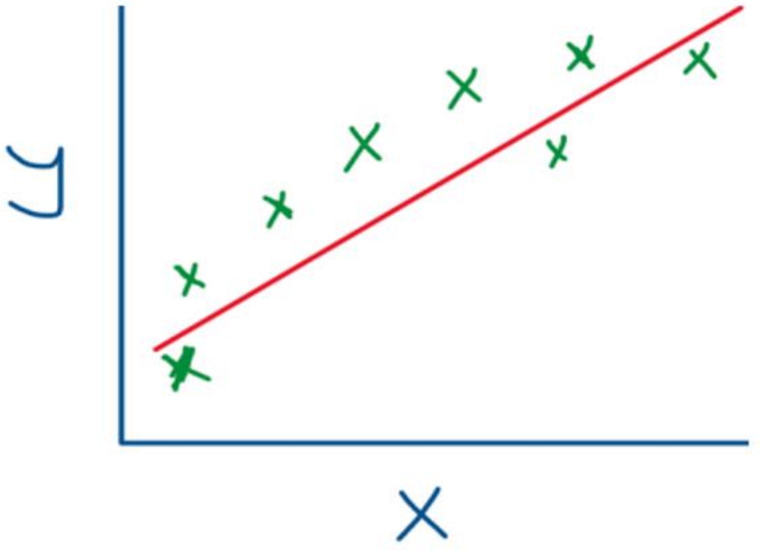
Some of the most popular examples of these methods are LASSO and RIDGE regression which have inbuilt penalization functions to reduce overfitting.

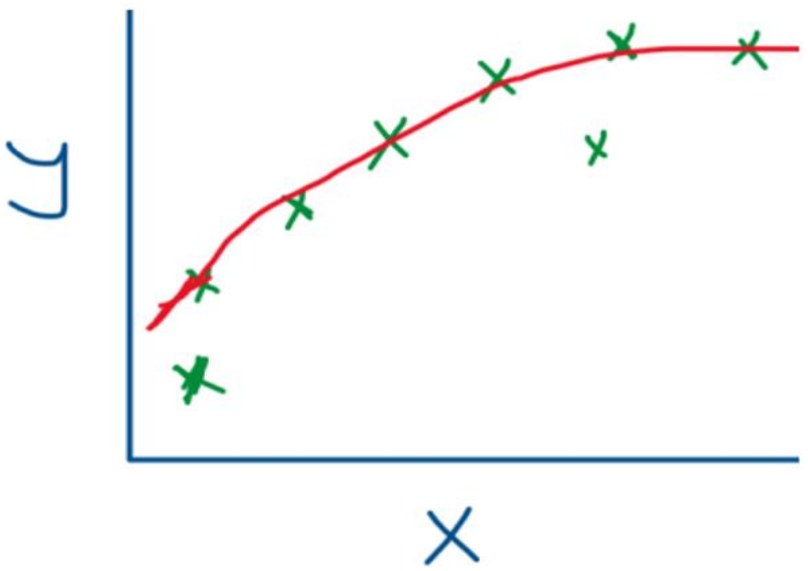
Lasso regression performs L1 regularization which adds penalty equivalent to absolute value of the magnitude of coefficients.

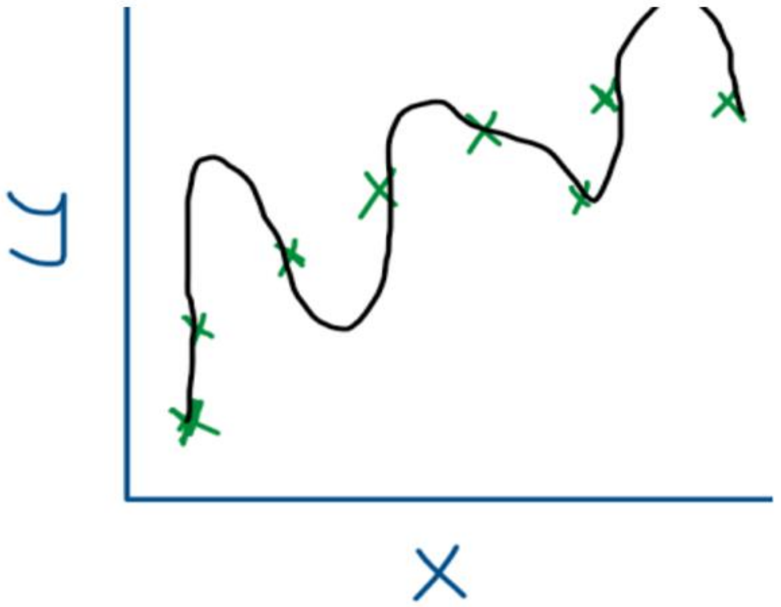
Ridge regression performs L2 regularization which adds penalty equivalent to square of the magnitude of coefficients.

REGULARIZATION

INTRODUCTION







- the linear regression is not a great model
  - This is underfitting
  - Known as high bias
  - Bias: we have a strong preconception that there should be a linear fit

$$y = \theta_0 + \theta_1 x$$

- Quadratic function
  - Works well
  - Just right

$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

- High order polynomial
  - Perform perfect on training data
  - high variance
  - Not able to generalize on unseen data
  - If we have too many features

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

- How can you tell that your model is overfitting or underfitting the data?

1. **Plotting hypothesis:** is one way to decide – you may look for "curvy – waggle"
   - But does not always work
   - Often have lots of features – harder to plot the data and visualize what feature to keep and which one to throw out
2. **cross-validation metrics:** If a model performs well on the training data but generalizes poorly according to the **cross-validation metrics**, then your model is overfitting.
   - If it performs poorly on both, then it is underfitting.
3. *Learning curves:* Another way is to look at the *learning curves*:
   - these are plots of the model's performance on the training set and the validation set as a function of the training set size
   - To generate the plots, simply train the model several times on different sized subsets of the training set

Addressing overfitting

- If you have lots of features and little data – overfitting can be a problem
- simple way to regularize a polynomial model is to reduce the number of polynomial degrees
- Reduce number of features:
  - Manually select which features to keep
  - Feature engineering – reduce numbers of features
  - But, we lose information
  - why do not we just stop adding features when we got an acceptable model!!!
  - You do not know which features to drop, and even worse if it turns out that every feature is fairly informative, which means that dropping some feature will likely ruin the algorithm performance – the answer is regularization
- **Regularization** : keep all features, but reduce magnitude of parameters θ

Regularization

- Regularization is the process of regularizing the parameters that constrain, regularizes, or shrinks the coefficient estimates towards zero
- Higher coefficient of polynomial terms lead to overfitting
  - Having large coefficients can be seen as evidence of memorizing data rather learning from them
  - For example, if you have some noises in training dataset, those noises will cause our model to put more weight into the coefficient of higher degree, and this lead to overfitting
- with an increasing number of features, a machine learning model can fit your data well – but if you add too many features we will be subjected to Overfitting

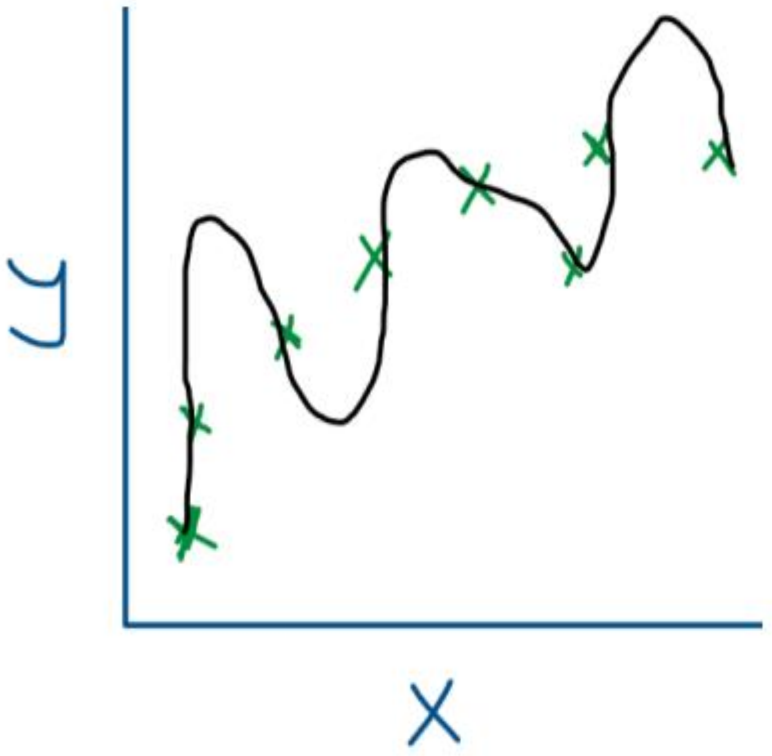
Recall: cost function

$$J = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(X^{(i)}) - y^{(i)}\right)^2$$

- You parameter can be updated in any way, just to lower the MSE value, and take care…
  - The larger your parameters become, the higher the chance your model overfit the data

How to regularize?

- Penalize and make some of the θ parameters really small
  - Like θ_3, θ_4
  - $J = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(X^{(i)}) - y^{(i)}\right)^2 + 1000\,\theta_3{}^2 + 1000\,\theta_4{}^2$
  - The only way to minimize this function is to make θ_3, θ_4 very small
  - So here we end up with θ_3 and θ_4 being close to zero
  - Now we approximately have a quadratic equation
  - $y = \theta_0 + \theta_1 x + \theta_2 x^2$



$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

How to regularize?

- Smaller values for parameters corresponds to a simpler hypothesis
  - You get rid of some of the terms
- A simpler hypothesis is less prone to overfitting
- But, we do not know what are the high order terms
  - How do we choose the ones to shrink?
- It is better to shrink all parameters

Main Goal

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(X^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

- Minimize the cost function and restrict the parameters not to become too large – thanks to the term of regularization
  - λ is a constant the control the weight and importance of regularization term
    - Control the trade off between the two goals
      - Fit the training model very well
      - Keep parameters small
  - n is the numbers of features
  - By convention you do not penalize θ_0- minimization is from θ_1 onwards
  - Minimizing the cost function consists of reducing MSE term and regularization term

- When parameter is updated to minimize MSE, and if it is becoming large, it will increase the value of cost function by regularization term, and as a result it will be penalized and updated to a small value

Regularization

- If λ is very large we end up penalizing all the parameters, so all the parameters end up being close to zero “Except θ_0”
  - Like we get rid of all the terms – underfitting
  - Too biased $\boldsymbol{h(\theta)} = \boldsymbol{\theta_0}$
- λ should be chosen carefully – not too big

Regularization with Gradient Descent

- $J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(X^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$
- θ_0 is not regularized
  - $\theta_0 = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right){x_0}^{(i)}$
- $\theta_j = \theta_j - \alpha\left[\frac{1}{m}\ \sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right){x_j}^{(i)} + \frac{\lambda}{m}\theta_j\right] \quad j = 1,2,3,\ldots,n$
- $\theta_j = \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right){x_j}^{(i)} \quad j = 1,2,3,\ldots,n$
- The term $\theta_j\left(1 - \alpha\frac{\lambda}{m}\right)$ is going to be less than 1
  - Usually learning rate is small and m is large
  - This term is often around 0.99 to 0.95
  - i.e. shrink θ_j - the term on right is the same as before "Gradient descent

LASSO Regression

Since we wish to discourage extreme values in model parameter, we need to choose a regularization term that penalizes parameter magnitudes. For our loss function, we will again use MSE.

Together our regularized loss function is:

$$L_{LASS}(\boldsymbol{\theta}^T) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{n} |\boldsymbol{\theta}_j|$$

Note that $\sum_{j=1}^{n} |\boldsymbol{\theta}_j|$ is the ***l*₁** norm of the vector $\boldsymbol{\theta}$

$$\sum_{j=1}^{n} |\boldsymbol{\theta}_j| = \| \boldsymbol{\theta} \|_1$$

Ridge Regression

Alternatively, we can choose a regularization term that penalizes the squares of the parameter magnitudes. Then, our regularized loss function is:

$$L_{Ridge}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \boldsymbol{\theta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{n} \boldsymbol{\theta}_j^2.$$

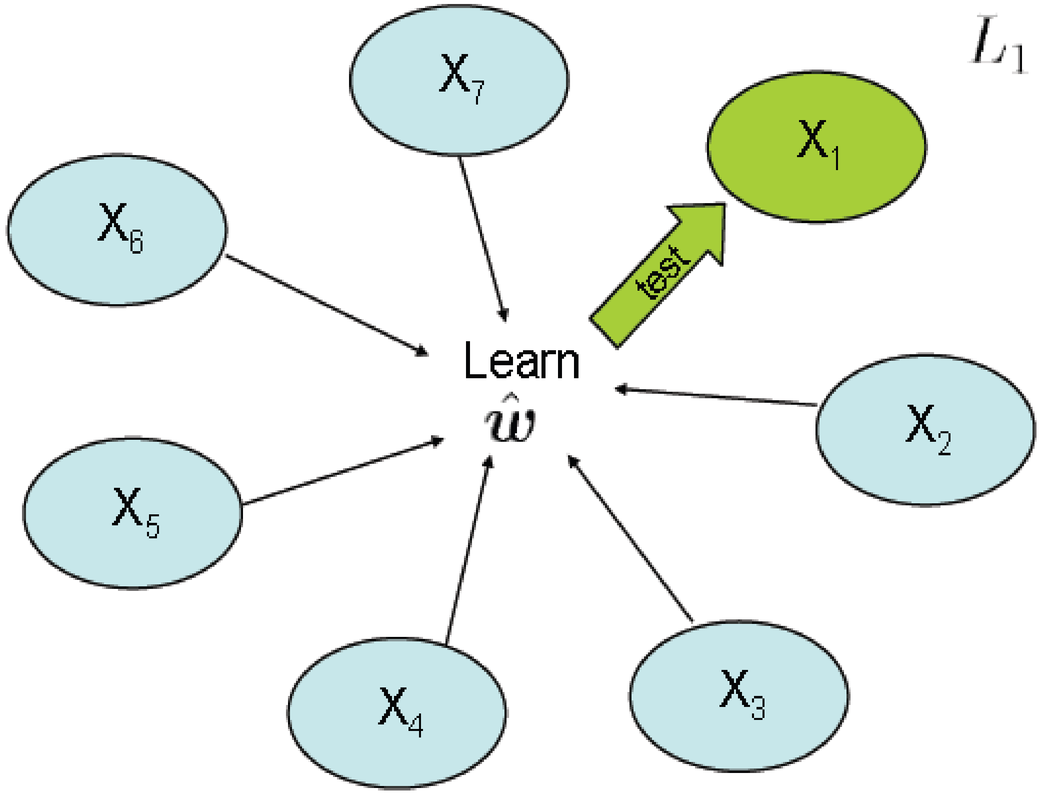
Note that $\sum_{j=1}^{n} |\boldsymbol{\theta}_j|^2$ is the square of the l_2 norm of the vector β

$$\sum_{j=1}^{n} \boldsymbol{\theta}_j^2 = \|\boldsymbol{\theta}\|_2^2$$

CHOOSING THE BEST MODEL

K-fold cross validation

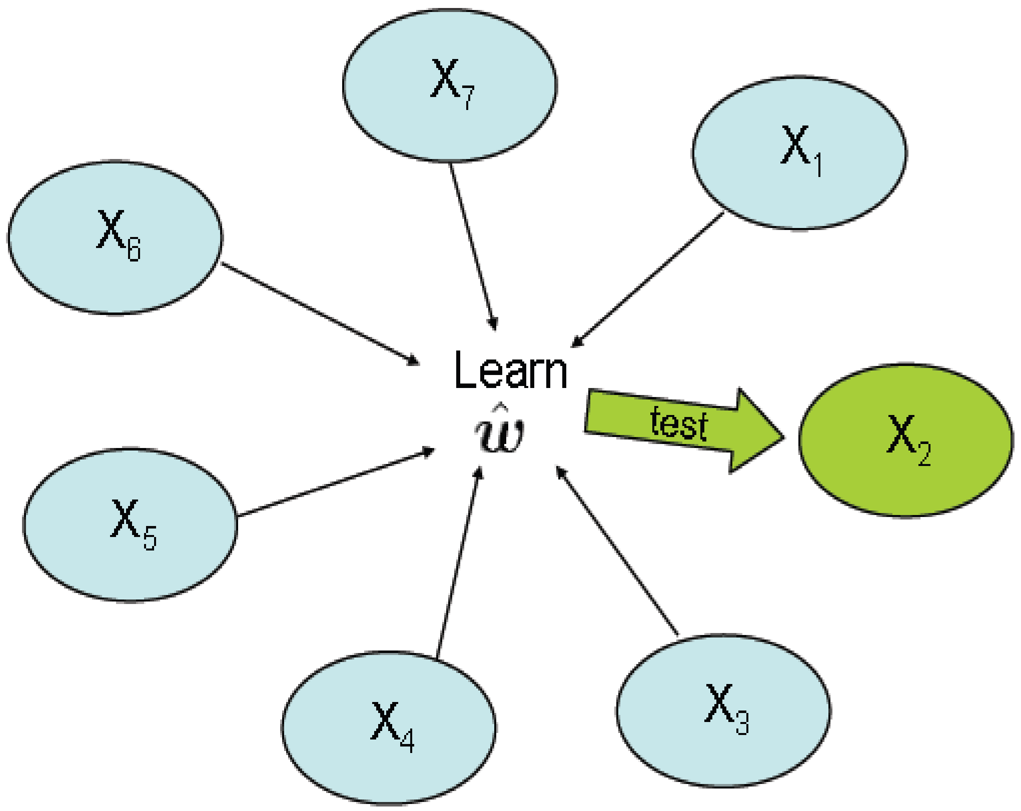
- A technique for estimating test error
- Uses *all* of the data to validate
- Divide data into K groups $\{X_1, X_2, \ldots, X_K\}$.
- Use each group as a validation set, then average all validation errors



$$L_1 = \sum_{(\boldsymbol{x},y)\in X_1} (y - \hat{\boldsymbol{w}}^\top \boldsymbol{x})$$

K-fold cross validation

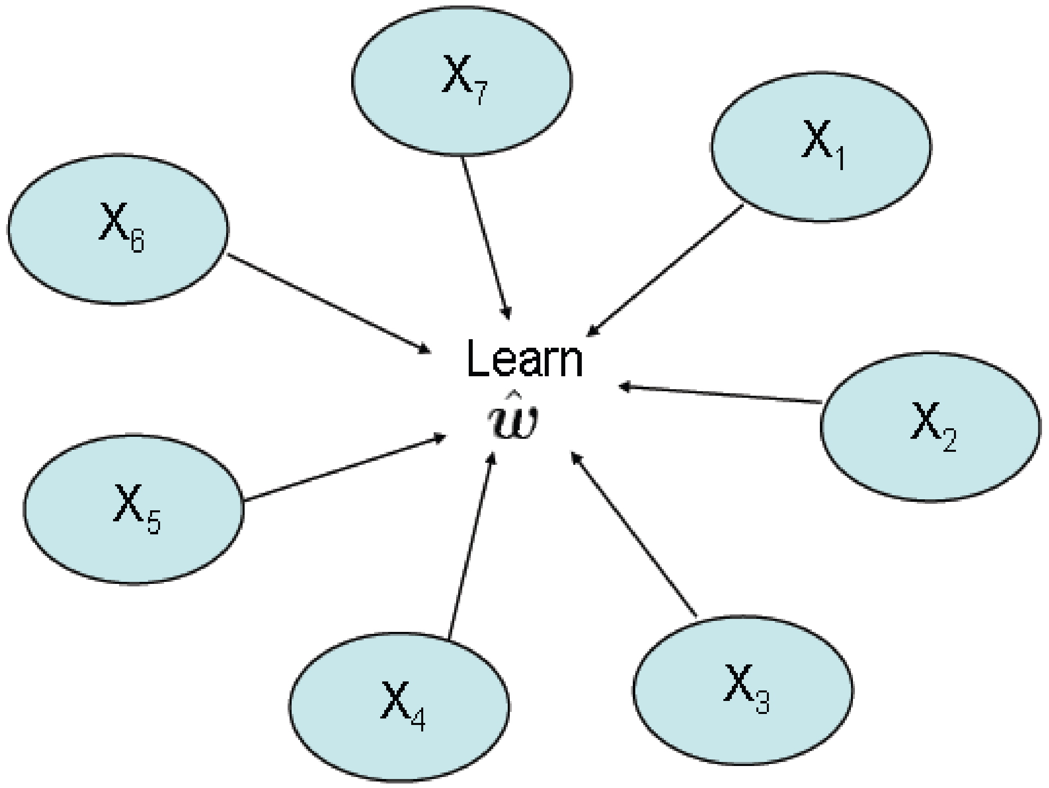
- A technique for estimating test error
- Uses *all* of the data to validate
- Divide data into K groups $\{X_1, X_2, \ldots, X_K\}$.
- Use each group as a validation set, then average all validation errors



$$L_2 = \sum_{(\boldsymbol{x},y)\in X_2} (y - \hat{\boldsymbol{w}}^\top \boldsymbol{x})$$

K-fold cross validation

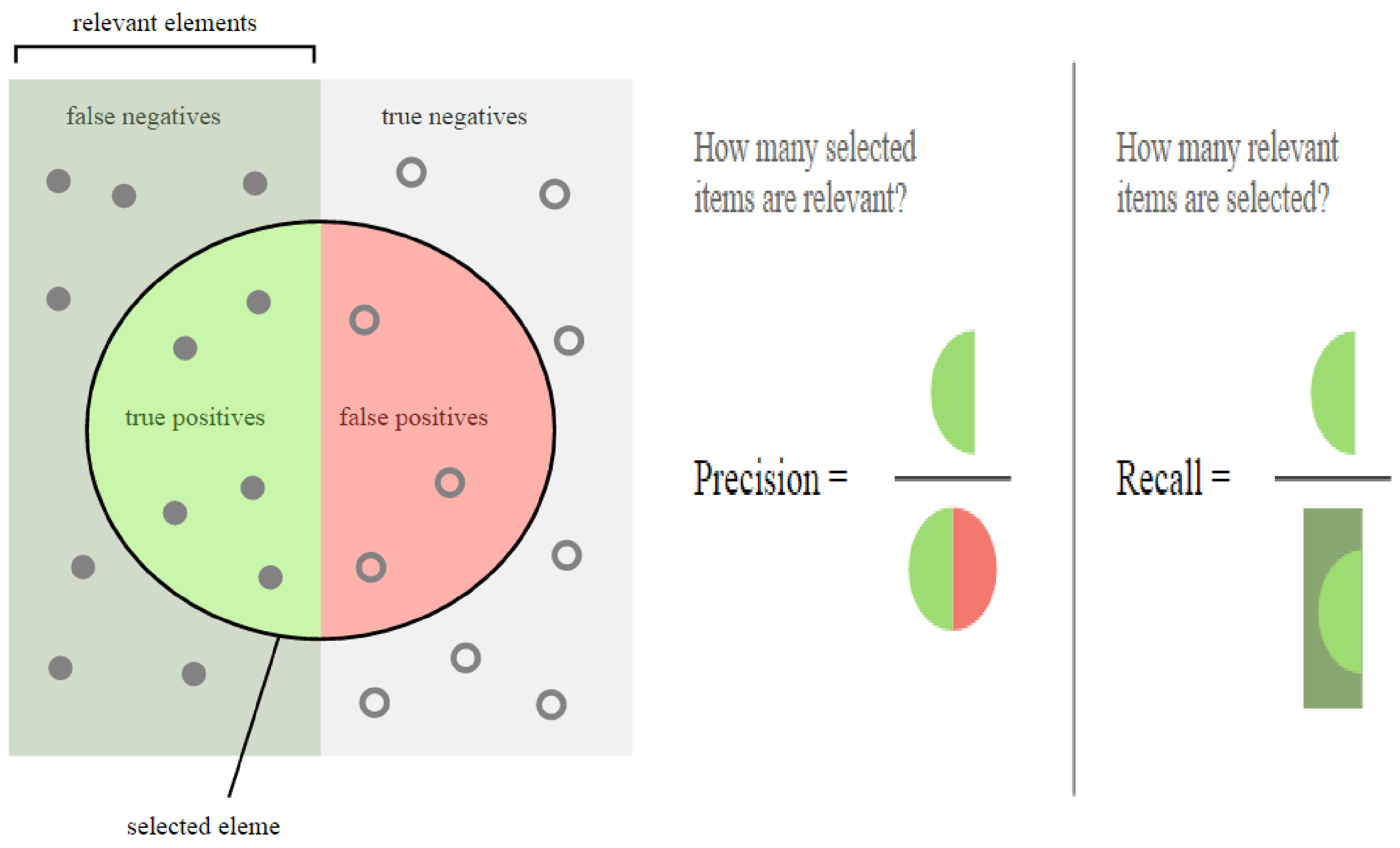
- A technique for estimating test error
- Uses *all* of the data to validate
- Divide data into K groups $\{X_1, X_2, \ldots, X_K\}$.
- Use each group as a validation set, then average all validation errors



$$CV(s) = \frac{1}{K}\sum_{i=1}^{K} L_i$$

MODEL EVALUATION

Precision and Recall Definition



$$F\_measure = 2 * \frac{(Precision * Recall)}{(Precision + Recall)}$$

PROBLEM

Assume the following:

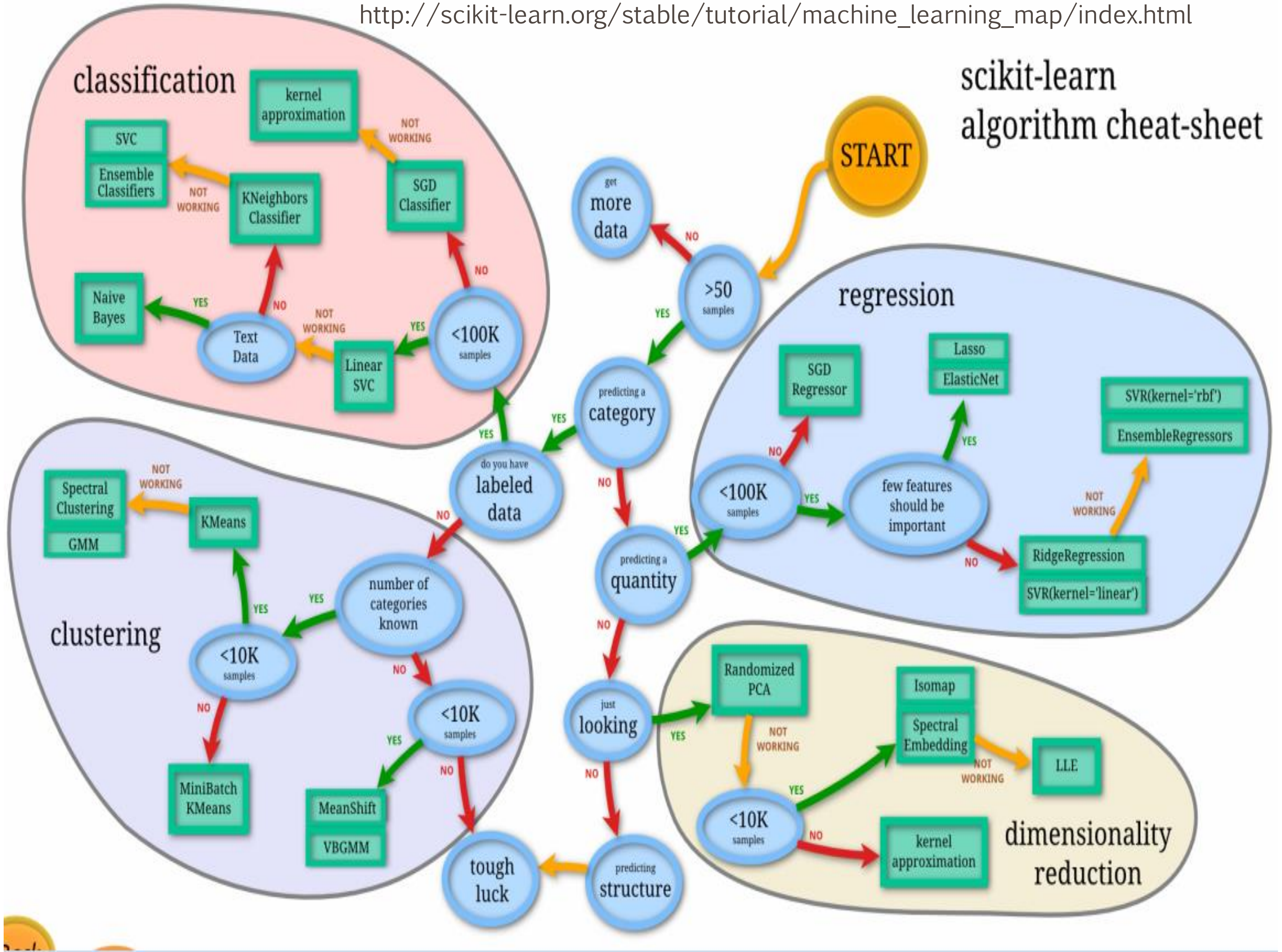
- A database contains **80 records** on a particular topic
- A search was conducted on that topic and **60 records** were retrieved.
- Of the 60 records retrieved, **45 were relevant**. Calculate the precision and recall scores for the search

<u>Solution:</u>

- TP = The number of relevant records retrieved, 45
- SE = The number of retrieved records, 60
- FP = The number of irrelevant records retrieved. (60-45)

In this example
Recall = 45/80 * 100% = 56%
Precision = (45 / (45 + 15)) * 100% => 45/60 * 100% = 75%
F_Measure=(2(0.56*0.75)/(0.56+0.75))*100=64.12%

http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html


scikit-learn
algorithm cheat-sheet
START
>50 samples
NO
get more data
YES
predicting a category
YES
do you have labeled data
YES
<100K samples
YES
Linear SVC
NOT WORKING
Text Data
YES
Naive Bayes
NO
KNeighbors Classifier
NOT WORKING
SVC
Ensemble Classifiers
NO
SGD Classifier
NOT WORKING
kernel approximation
classification
NO
number of categories known
YES
<10K samples
YES
KMeans
NOT WORKING
Spectral Clustering
GMM
NO
MiniBatch KMeans
NO
<10K samples
YES
MeanShift
VBGMM
NO
tough luck
clustering
NO
predicting a quantity
YES
<100K samples
NO
SGD Regressor
YES
few features should be important
YES
Lasso
ElasticNet
NO
RidgeRegression
SVR(kernel='linear')
NOT WORKING
SVR(kernel='rbf')
EnsembleRegressors
regression
NO
just looking
YES
Randomized PCA
NOT WORKING
<10K samples
YES
Isomap
Spectral Embedding
NOT WORKING
LLE
NO
kernel approximation
dimensionality reduction
NO
predicting structure