Live Session 2

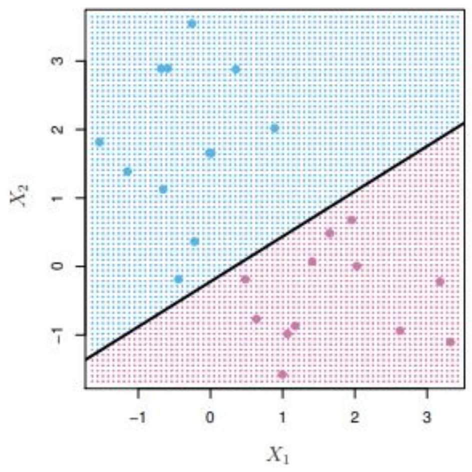
Machine Learning II: Support Vector Machines

Designed by: Sudipta Dasmohapatra & Babak Zafari
Delivered by: Tommy Jones

GEORGETOWN UNIVERSITY McDonough SCHOOL of BUSINESS

Support Vector Machine (SVM)

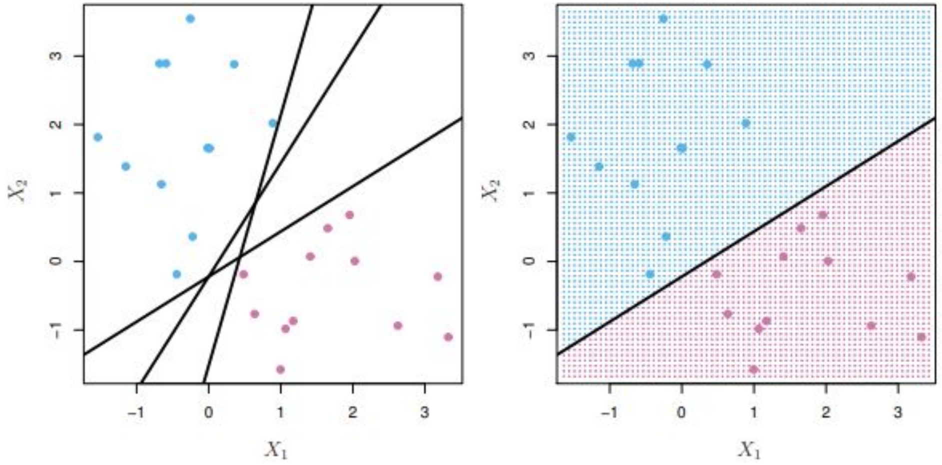
- A Support Vector Machine (SVM) is a machine learning algorithm that is commonly used for classification problems.
- SVM is based on the idea of finding a hyperplane that divides the dataset into the classes.
  - In two dimensions, a hyperplane is a line, in three dimensions, it is a plane.



A separating hyperplane and the classification rule based on this hyperplane

Support Vector Machine (SVM)

- A Support Vector Machine (SVM) is a machine learning algorithm that is commonly used for classification problems.
- SVM is based on the idea of finding a hyperplane that divides the dataset into the classes.
  - In two dimensions, a hyperplane is a line, in three dimensions, it is a plane.



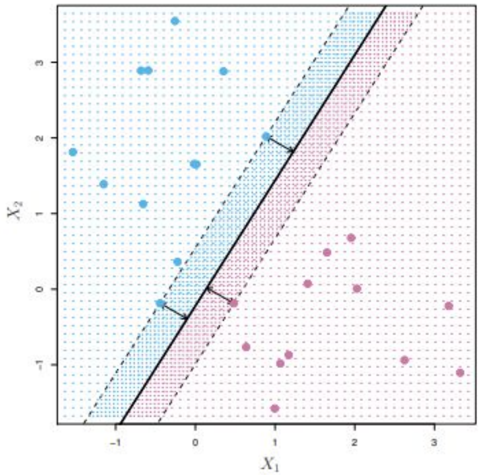
Three separating hyperplanes, out of many possible

A separating hyperplane and the classification rule based on this hyperplane

- In general, if our data can be perfectly separated using a hyperplane, then there will exist an infinite number of such hyperplanes.
  - Because a separating hyperplane can be shifted a tiny bit up/down/rotated without coming into contact with any of the observations.

Maximal Margin Hyperplane

- In order to construct a classifier based upon a separating hyperplane, we must decide which of the infinite possible separating hyperplanes to use.
  - A natural choice is the maximal margin hyperplane (also known as optimal separating hyperplane), which is the separating hyperplane that is farthest from the training observations.
  - In a sense, the maximal margin hyperplane represents the mid-line of the widest “slab” that we can insert between the two classes.



The maximal margin hyperplane is shown as a solid line

  - The **margin** is the distance from the solid line to either of the dashed lines.
  - The two blue points and the purple point that lie on the dashed lines are the **support vectors**.

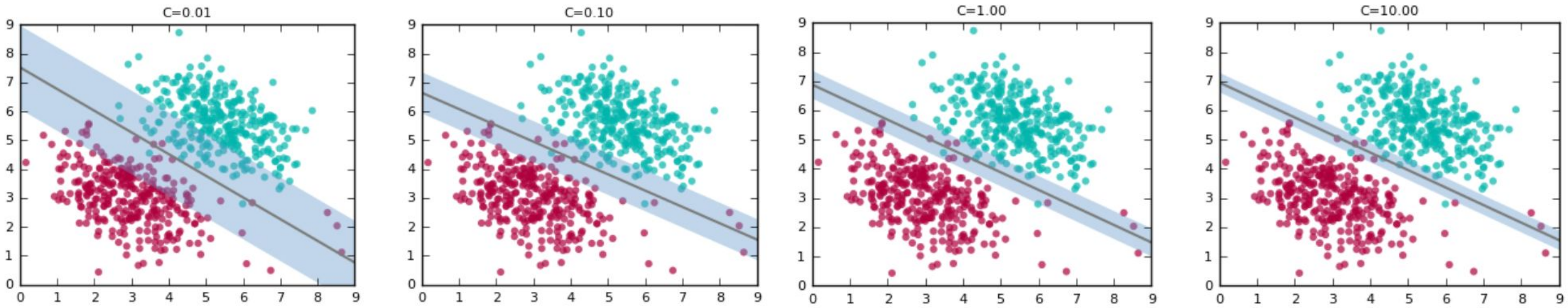
Cost Parameter

- If the datapoints are not linearly separable, we have a soft margin hyperplane.
- You can provide a parameter called C (regularization parameter) that helps tradeoff between:
  - Having a wide margin
  - Correctly classifying data
- A high value of C implies you want less errors!

Cost Parameter

- If the datapoints are not linearly separable, we have a soft margin hyperplane.
- You can provide a parameter called C (regularization parameter) that helps tradeoff between:
  - Having a wide margin
  - Correctly classifying data

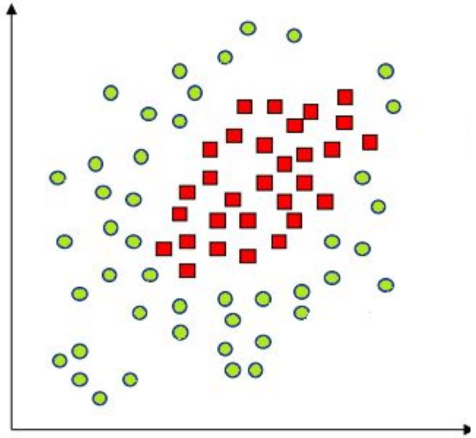
- A high value of C implies you want less errors!



- An important practical problem is to decide on a good value of C. Since real-world data is almost never cleanly separable, this need comes up often.
- We use **cross-validation to pick a good value for C.**

Kernel Trick

- So what happens when the data is not linearly separable, for example, for a two-dimension feature space?



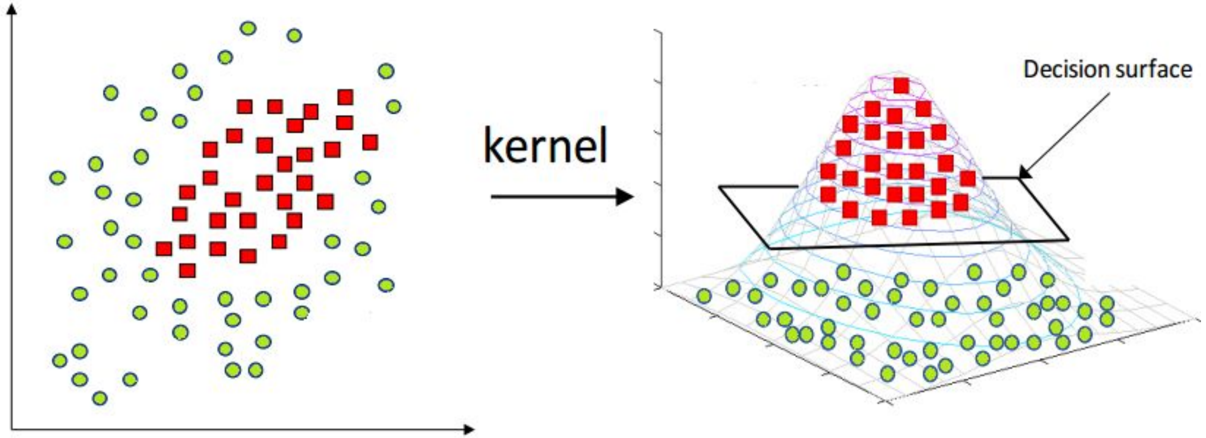
https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

Kernel Trick

- So what happens when the data is not linearly separable, for example, for a two-dimension feature space?

- In this case, instead of modeling in two-dimensional space, we add a third dimension – called feature space.
- We add a non-linear transformation to the input variable in this feature space.
  - The idea is to gain linearly separation by mapping the data to a higher dimensional space

- Common kernel functions
  - Linear
  - Nonlinear/Polynomial
  - Radial Basis Function (RBF)
  - Sigmoid



https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

SVM Summary

1. For linearly separable data SVMs works well.

2. For data that's almost linearly separable, SVMs can still be made to work pretty well by using the right value of the cost parameter, C.

3. For data that's not linearly separable, we can project data to a space where it is perfectly/almost linearly separable, which reduces the problem to 1 or 2.

- Parameters to tune (through cross validation):
  - The penalty C (regularization term) for data that is not completely linearly separable.
  - The kernel function and its parameters.

Categorical Variables - Revisited

- Categorical variables can not be used directly in distance functions, such as Euclidean. Also, many other models only accept numerical features.
- One way to use categorical variables is to convert them to dummy/indicator variables (one-hot encoding).

- In R, we can use the **dummy_cols( )** function from the “fastDummies” package.
  - This creates one dummy variable for each level of specified categorical variables. For example, for a binary-coded variable Sex, this will create two new variables, Sex_male and Sex_female
  - The original categorical variable(s) will also be included in the generated dataset. You can use **remove_selected_columns=T** to remove the original variable and **remove_first_dummy=T** to remove the first dummy variable.
  - Also, while some models can accept all levels of dummy variables as input, some such as linear regression, may run into multi-collinearity and linear dependence issues. A practical approach is to delete one dummy level/variable of each categorical variable.

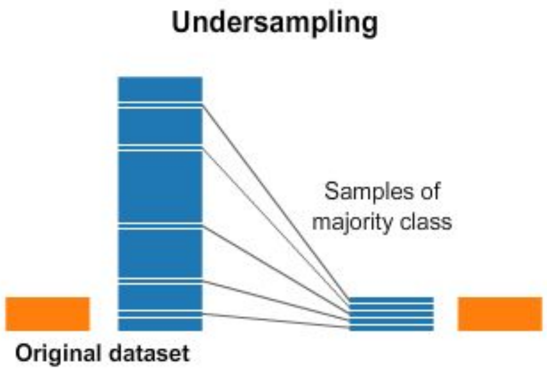
- **NOTE**: In Week 1, **Pclass** (ticket class) variable from the titanic dataset was treated as a numerical variable. A better approach is to treat this as a categorical variable and convert it into dummies.

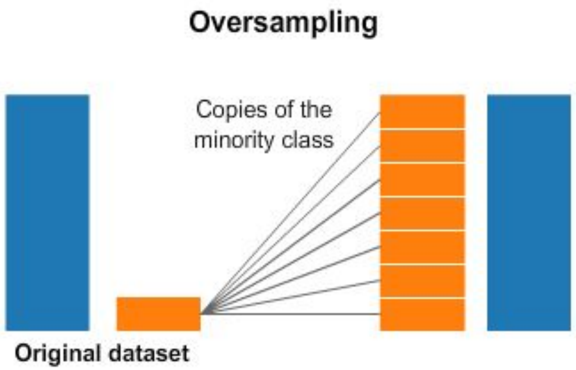
Imbalanced Datasets (in Classification)

- A classification data set with skewed class proportions is called imbalanced.

Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

- The main two resampling methods that are used to tackle the class imbalance are:
  – **downsampling/undersampling**: reducing the count of training samples falling under the majority class
  – **upsampling/oversampling**: creating artificial or duplicate data points of the minority class to balance the class label





- https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data
- https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets/notebook

Imbalanced Datasets (in Classification)

- Shortcomings:
  - The simplest implementation of oversampling is to duplicate random records from the minority class, which can cause overfitting.
  - In undersampling, the simplest technique involves removing random records from the majority class, which can cause loss of information.

- In R:
  - "caret" package contains a function **downSample()** to randomly subset all the classes in the training set so that their class frequencies match the least prevalent class.
  - "caret" package contains a function **upSample()** to randomly sample (with replacement) the minority class to be the same size as the majority class.

- **NOTE**: The <u>sampling process is applied only to the training set</u> and no changes are made to the validation and testing data. "You would never want to artificially balance the test set; its class frequencies should be in-line with what one would see "in the wild"."
- Check lines 48-49 of the (updated) R code.

Example: Titanic Data

Exercise

Load and save ***redwine.csv*** dataset. You plan to apply the svm algorithm to this data.

1. Convert “quality” variable to two categories: "low" if wine quality is <= 6 otherwise, "high“. This is the target variable of interest.

2. Split the data into train (70%) and test (30%).
   - Check the ratio of low and high in the training dataset. If the ratio of the minority class is less than 20%, upsample the training data.
   - Check the ratio of the upsampled dataset.
   - Check for any potential missing value, and if applicable, replace them with meaningful values.
   - Check if the data needs to be scaled or whether the svm function(s) have an option to do that automatically.
3. Use the caret package (with 10-fold cv) to search for an optimal value of C using radial kernel.

4. Go back to step 3. This time down-sample the majority class, and re-run the models. Compare the results.

Backup

Coding SVM in R – e1071 Package

In coding SVM in our class, we try two different approaches. The following is using the e1071 package:

- **Using the svm() function from the "e1071" package.**
  - This function can automatically scale the variables using **scale=TRUE** (default setting).
  - It automatically applies a 10-fold cross-validation and reports the best parameter.
  - The default method is classification **type = 'C-classification'.** For regression, you can use "eps-regression".
  - The cost parameter (C) can be set through the "**cost**" argument (default is 1).
  - You can use different kernels. If radial kernel is used, pass "**gamma**" parameter. If polynomial is used, pass "**degree**".

- **Using the tune() function from the "e1071" package we can search for the best hyperparameters.**
  - The METHOD should be set as SVM.
  - It automatically applies a 10-fold cross-validation and reports the best set of parameters (i.e. the one with the lowest misclassification/error rate).
  - You can search over different hyperparameters. For example, to search for cost parameter, we can define a list of values to search over: **ranges=list(cost=c(0.01,0.1,1))**.

Coding SVM in R – caret Package

In coding SVM in our class, we try two different approaches. The following is using the caret package:

- **Using the train() function from the "caret" package we can search for the best hyperparameters.**
  - We can set up the desired cross validation parameters using trainControl() function.
  - To have the model generate estimated probabilities as well, we should use **classProbs=TRUE** argument in the trainControl() function, where we set the cross validation parameters.
  - The train() function can either use already-scaled datasets or scale the variables on its own using the preProcess argument.
  - The tuneLength argument allows us to search over a set of Cost values (c) in search of the best one (i.e., the one with the highest accuracy for classification problems).
  - For kernels that have other parameters to tun (such as gamma parameter for radial kernel), a default value will be used. If you'd like to search over multiple parameters at the same time, you should set up a grid search using expand.grid() function.