Data Mining 2 – Lecture 2a
Support Vector Machines (SVMs)

Anthony Breitzman, PhD
9/12/18

Overview

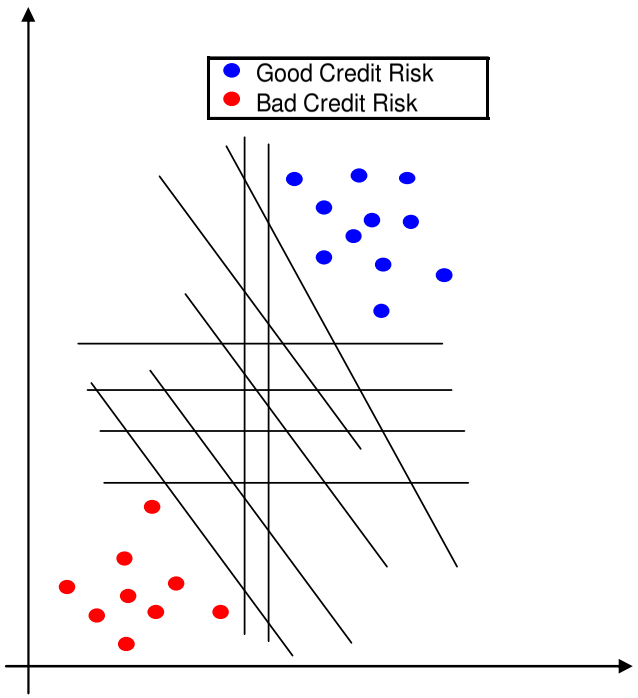
- The first paper on SVMs was presented in 1992 by Vapnik, Boser, and Guyon
- It actually builds on things from the 1960s and earlier
- SVMs are actually kind of in vogue now. They have a lot of good properties
  - Not prone to overfitting like NNs or decision trees
  - Highly accurate
  - Good with non-linear complex decision boundaries
  - Can be used with numeric prediction as well as classification
  - Have been applied successfully to digit recognition, object recognition, speaker identification, etc.
- The only downside is they can be incredibly slow for large data sets

Warning!

- SVMs are not really my area of expertise
- I have a passing knowledge of roughly how they work and I can get R to “somewhat” implement them
- I’m showing you what I know because they are an important classification tool, but since they aren’t really practical for very large data sets I’ve never used them a lot
- So if you want to become experts in SVMs it’s a good topic for a Master’s Thesis or PhD Thesis.
- Then I’ll invite you back to teach my next class!

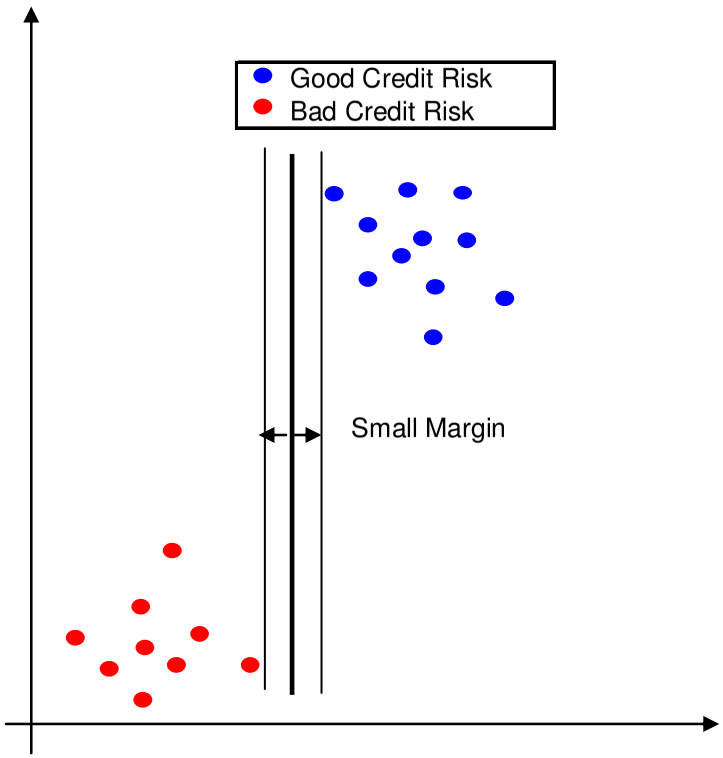
Case 1: Data Linearly Separable

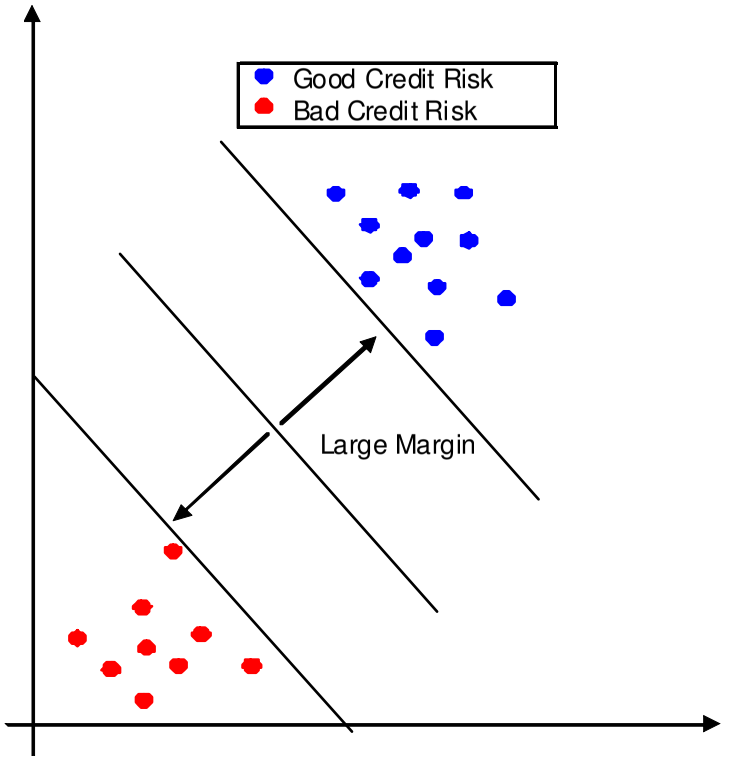
- We talked about linear separable data last week when we talked about perceptrons
- The picture on the right shows data that is linear separable in numerous ways
- The various lines separating the data are called support vectors
- Obviously there are an infinite number of support vectors
- The trick is to find the best
- If we visualized these points in 3-space then the support vectors are actually planes
- Since most of the time the solutions are in n-dimensions for n large it is easiest to consider the support vectors as hyperplanes in every case



Case 1: Continued

- The “best” support vector is the one that has the widest margin between points of each class
- That way new points to be classified are more likely to be correctly classified as being on one side or the other of the hyperplane





How do we find the Support Vectors and Measure the Margins?

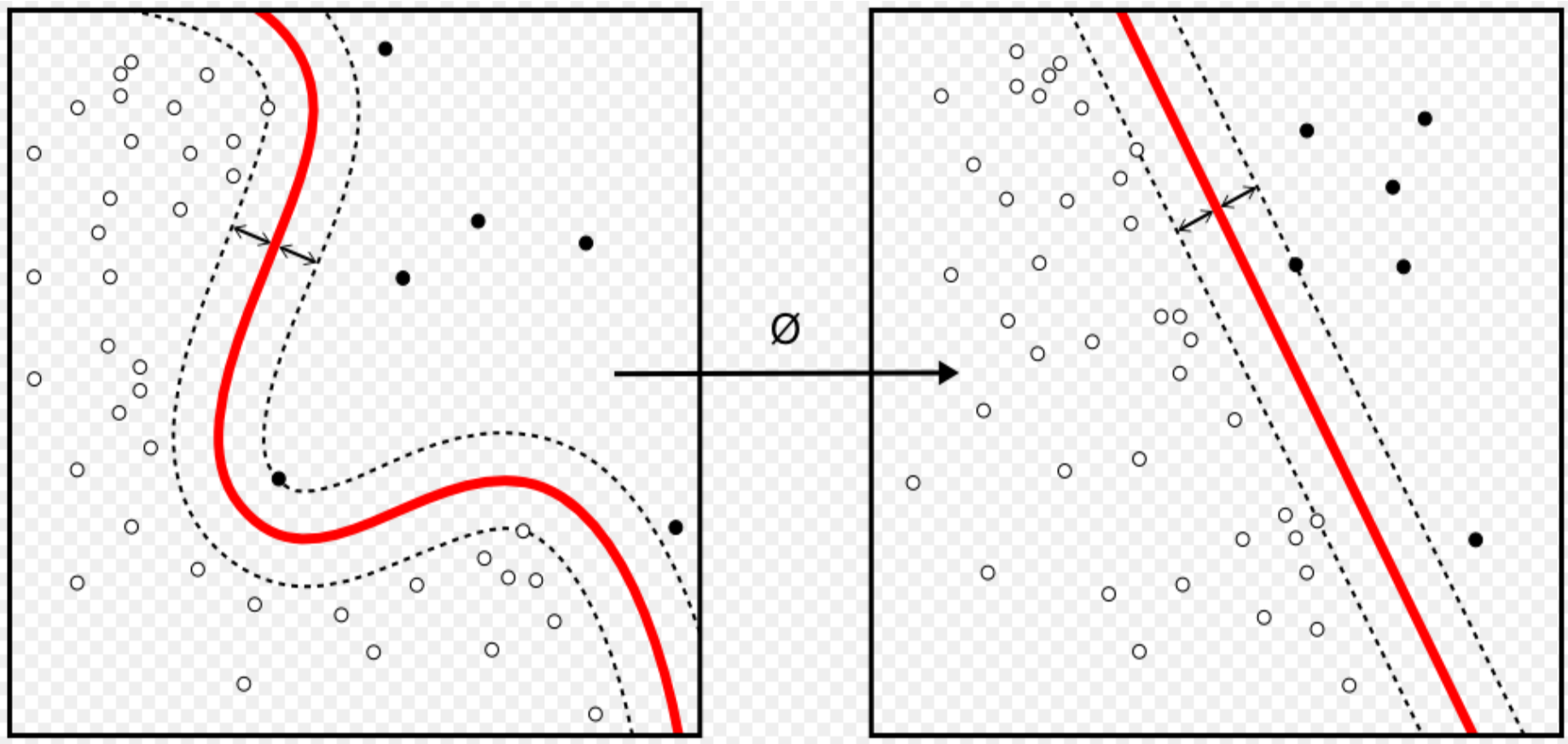
- It's actually a bit beyond the scope of this course.
- If you end up getting a PhD and studying Support Vector Machines you will learn about it
- The rough idea is that some basic linear algebra gets you to the point where the problem reduces to what's known as a 'Constrained (Convex) Quadratic Optimization Problem"
- The interested reader can look up solving for solutions using the "Karush-Kuhn-Tucker (KKT) conditions"
- If you want to learn the beginning of the math go to https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-2/ but note the math gets pretty out of hand pretty quickly
- At some point I intend to take a few weeks and actually dive into the math, but I've been saying that for 3 years but I keep running into different more interesting problems to work on.
- If the data are small (under 2000 training tuples) then any optimization software package for solving constrained convex quadratic problems can be used. For larger data there are more advanced algorithms which again are beyond the scope of this class

A Fundamental Point

- Unlike a Neural Net, we are not actually fitting anything to the data. We are picking a hyperplane that separates the data. If tuples are added or removed to training, the support vector is likely to stay the same
  - That is the complexity of the algorithm is related to the number of support vectors to eliminate and not the dimensionality of the data
  - This makes SVMs less prone to overfitting than other methods

  - Reread those last 2 bullet points!

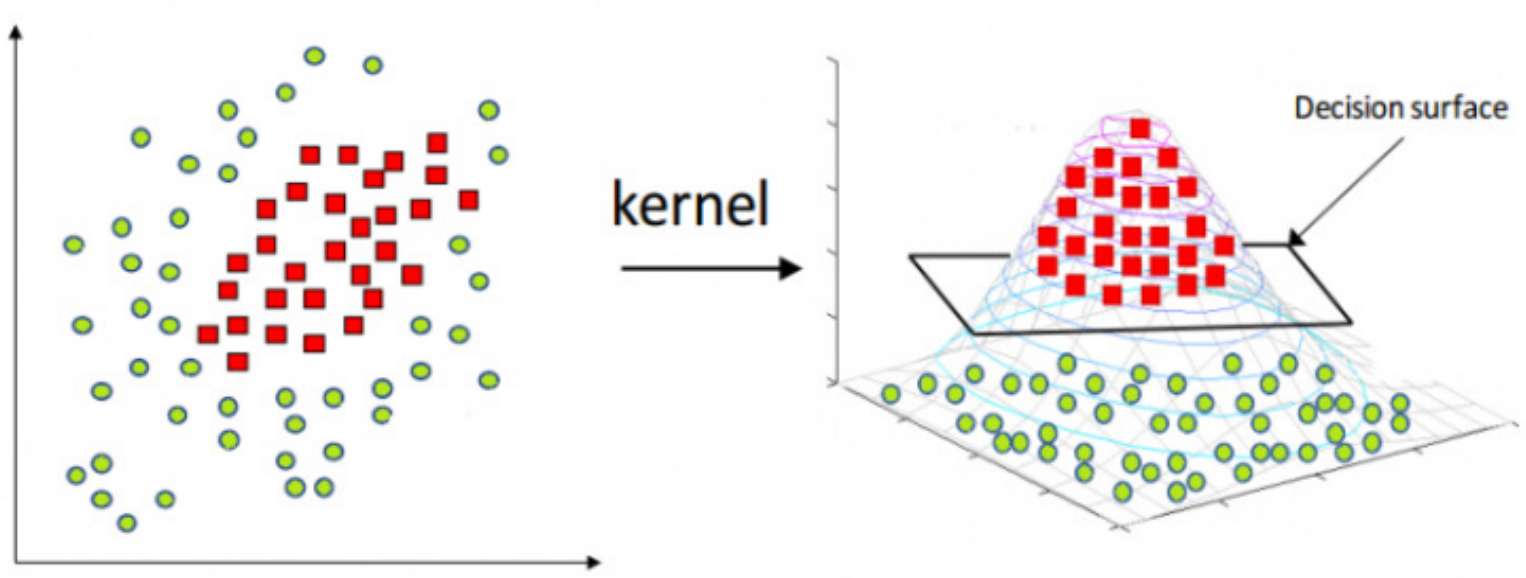
Case II: Linearly Inseparable Data



Source: https://en.wikipedia.org/wiki/Support_vector_machine Accessed 1/29/2017

- By using what's called a 'Kernel Trick' or 'Kernel Function' Vapic et al. 1992 showed that a linearly inseparable set in one dimension can be mapped to a linearly separable set in a higher dimension
- For example the polynomial kernel $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d$ maps a non-linear function in 2 dimensions to a linear function in dimension d.
- See https://en.wikipedia.org/wiki/Support_vector_machine for other common kernel functions

Another Picture of the Kernel Trick



- See https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/

Python Implementation

- I have not tested this….
- Source: https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/
- The most widely used library for implementing machine learning algorithms in Python is scikit-learn. The class used for SVM classification in scikit-learn is svm.SVC()
- sklearn.svm.SVC (C=1.0, kernel='rbf', degree=3, gamma='auto')
- Parameters are as follows:
- C: It is the regularization parameter, C, of the error term.
- kernel: It specifies the kernel type to be used in the algorithm. It can be 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed', or a callable. The default value is 'rbf'.
- degree: It is the degree of the polynomial kernel function ('poly') and is ignored by all other kernels. The default value is 3.
- gamma: It is the kernel coefficient for 'rbf', 'poly', and 'sigmoid'. If gamma is 'auto', then 1/n_features will be used instead.

R Packages

- Tonight's examples will come from the R package 'e1071'
- It's actually a front-end to a famous C++ package libsvm by Chang and Lin, 2001
- Other packages available in R include 'kernlab' and 'klaR'
- The interested reader can get more background on SVMs and these packages at *J. Stat. Software* - v15i09.pdf (included in canvas)
- Note when using the e1071 package if you make the predicted variable a factor (as.factor(y)) then the SVM will attempt classification. Otherwise it does regression or there are other options.
- Go to breastcancer2a.pdf
- Then to regressionTests1.pdf