GENERAL ASSEMBLY

SUPPORT VECTOR MACHINES

Brian Chung

FROM LAST TIME

Model Selection Methods:

- **Best Subset Selection**
- **Forward Stepwise Selection**
- **Backwards Stepwise Selection**

Selection Criterion:

- **Adjusted R^2**
- **AIC**
- **BIC**
- **MDL (Minimum Description Length)**

GOOGLE RELEASES DEEP LEARNING COURSE

https://www.udacity.com/course/deep-learning--ud730

Built by Google

Join thousands of students

Course Summary

Machine learning is one of the fastest-growing and most exciting fields out there, and **deep learning** represents its true bleeding edge. In this course, you'll develop a clear understanding of the motivation for deep learning, and design intelligent systems that learn from complex and/or large-scale datasets.

We'll show you how to train and optimize basic neural networks, convolutional neural networks, and long short term memory networks. Complete learning systems in TensorFlow will be introduced via projects and assignments. You will learn to solve new classes of problems that were once thought prohibitively challenging, and come to better appreciate the complex nature of human intelligence as you solve these same problems effortlessly using deep learning methods.

We have developed this course with Vincent Vanhoucke, Principal Scientist at Google, and technical lead in the Google Brain team.

***Note**: This is an intermediate to advanced level course offered as part of the Machine Learning Engineer Nanodegree program. It assumes you have taken a first course in machine learning, and that you are at least familiar with supervised learning methods.*

SVM AGENDA

I. MAXIMUM MARGIN CLASSIFIER
II. SUPPORT VECTOR CLASSIFIER
III. SUPPORT VECTOR MACHINES
IV. SVM LAB

SVM LEARNING OBJECTIVES

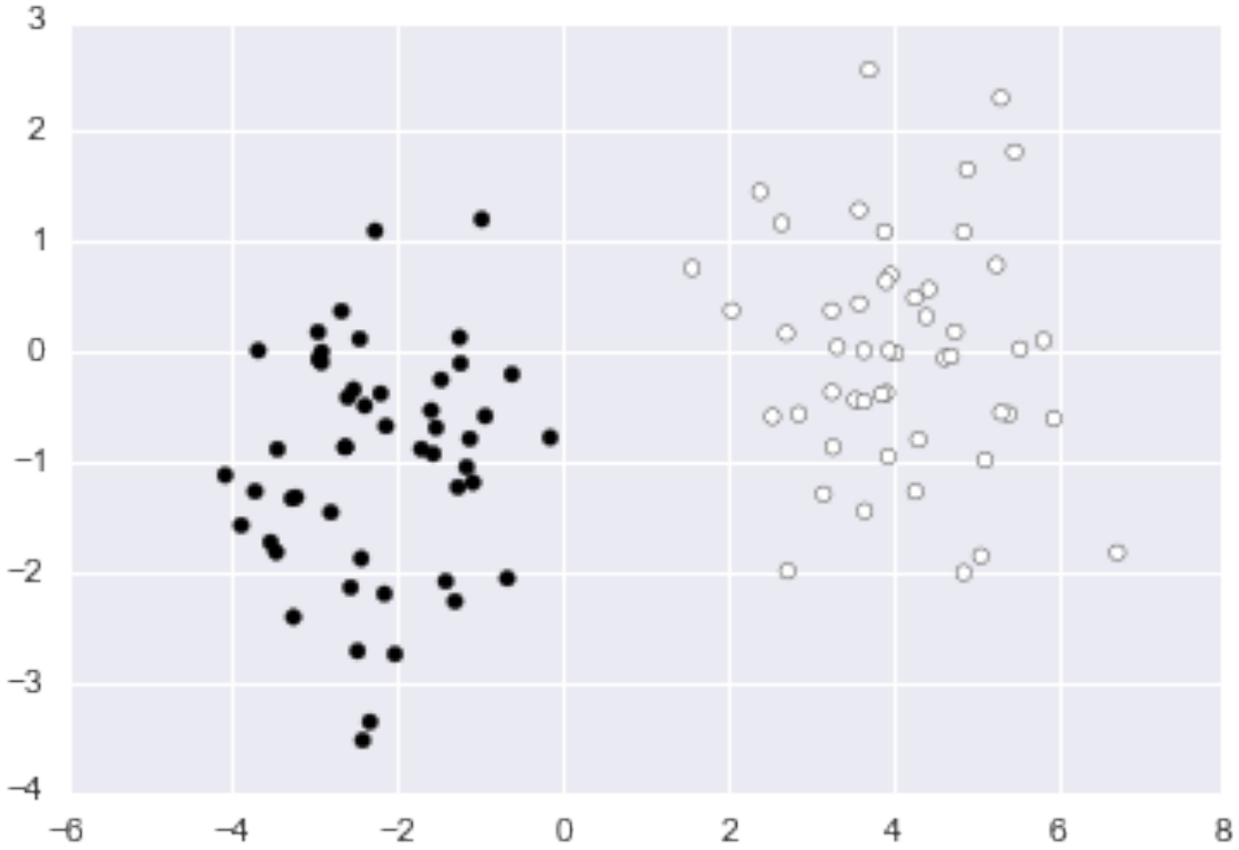
1. Understand how we build upon Maximum Margin Classifier -> Support Vector Classifier -> Support Vector Machines
2. Understand the pros and cons of using an SVM classifier
3. See some of the uses of SVMs
4. Create and interpret SVM classifiers in Python

TYPES OF ML SOLUTIONS

	Continuous	Categorical
Supervised	Regression	Classification
Unsupervised	Dimension Reduction	Clustering

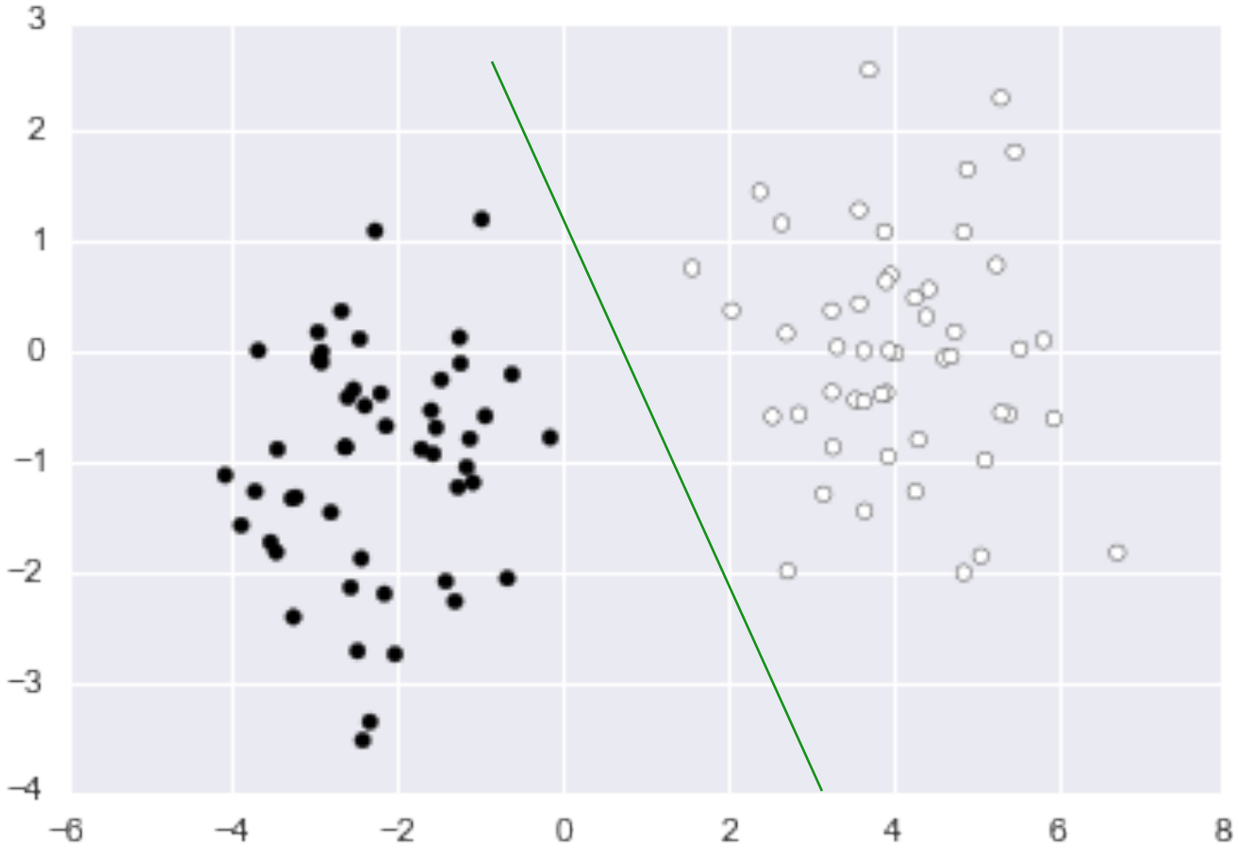
I. MAXIMAL MARGIN CLASSIFIER

MAXIMAL MARGIN CLASSIFIER



Suppose we have data with two features and binary class labels which we would like to classify

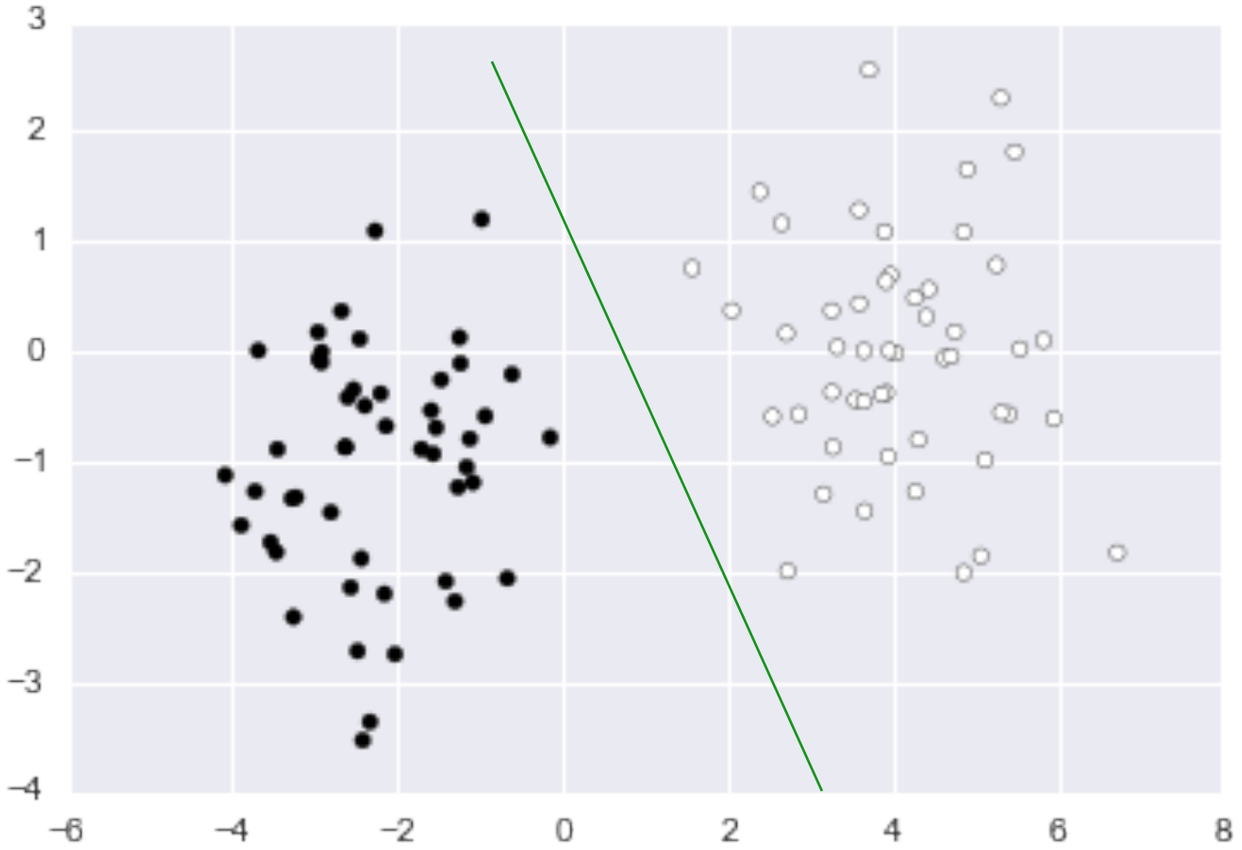
MAXIMAL MARGIN CLASSIFIER



Suppose we have data with two features and binary class labels which we would like to classify

Recall that after fitting a classifier, we can plot the **decision boundary** which separates the two classes

MAXIMAL MARGIN CLASSIFIER

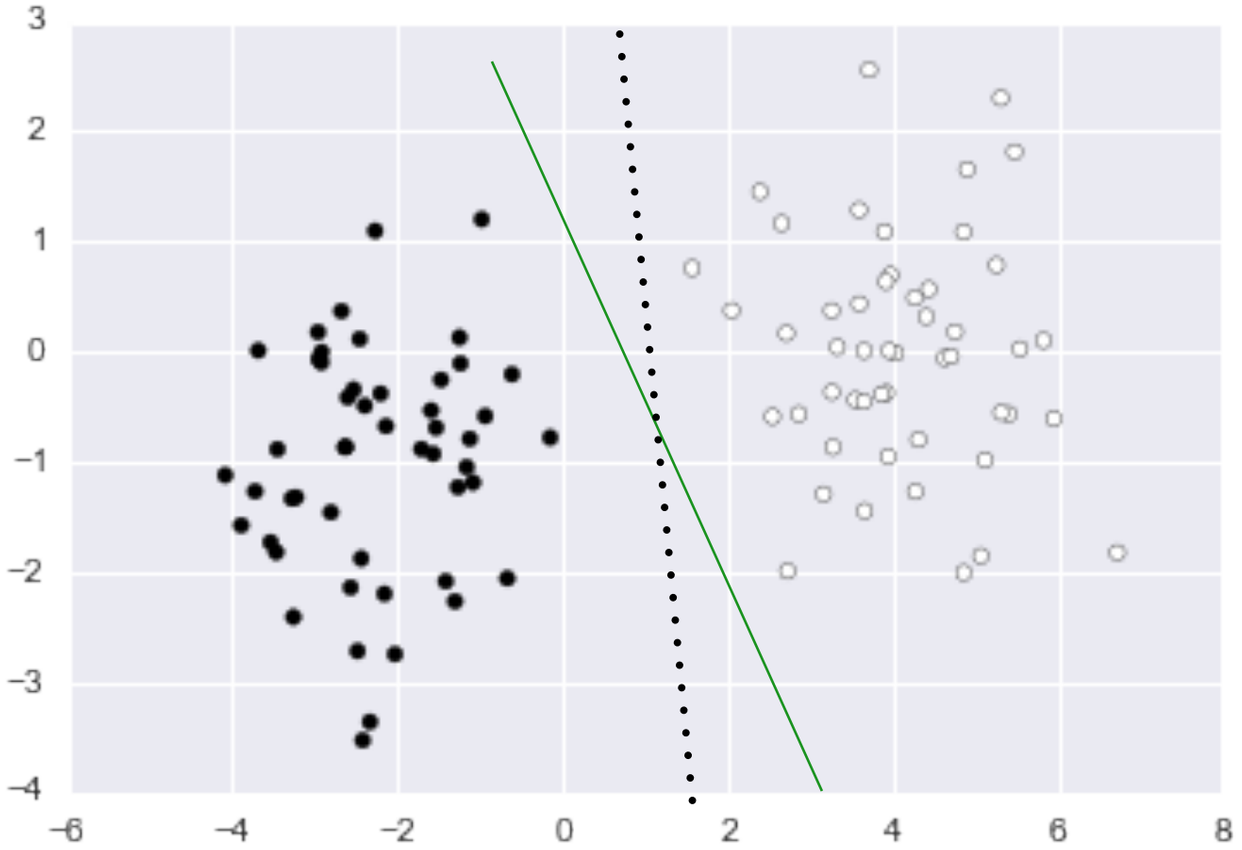


Suppose we have data with two features and binary class labels which we would like to classify

Recall that after fitting a classifier, we can plot the **decision boundary** which separates the two classes

The boundaries depend on the classifier. For example, logistic regression gives a different one than naive bayes

MAXIMAL MARGIN CLASSIFIER

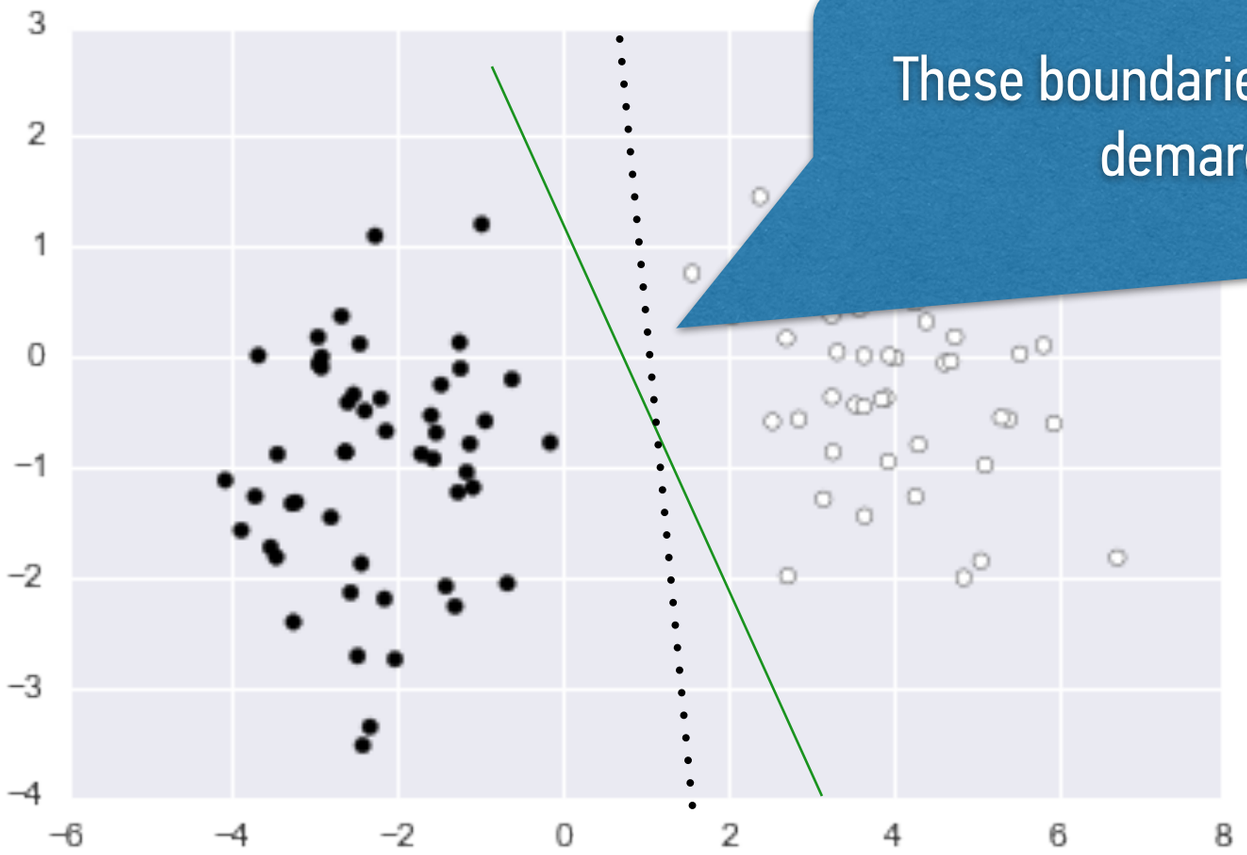


Suppose we have data with two features and binary class labels which we would like to classify

Recall that after fitting a classifier, we can plot the **decision boundary** which separates the two classes

The boundaries depend on the classifier. For example, logistic regression gives a different one than naive bayes

MAXIMAL MARGIN CLASSIFIER

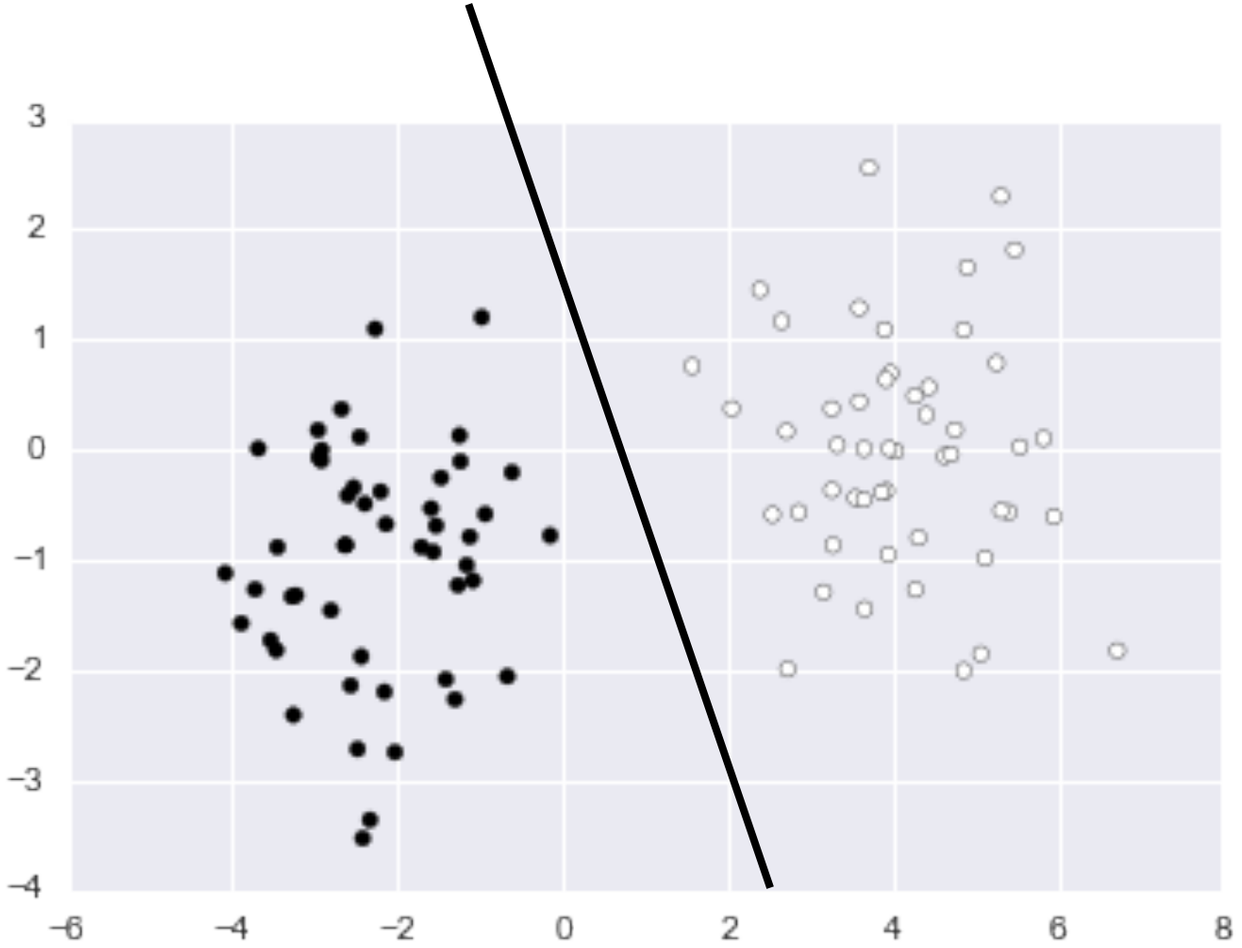


These boundaries are built from probabilistic methods demarcating the P(C==c) = .5 points

Recall that after fitting a classifier, we can plot the **decision boundary** which separates the two classes

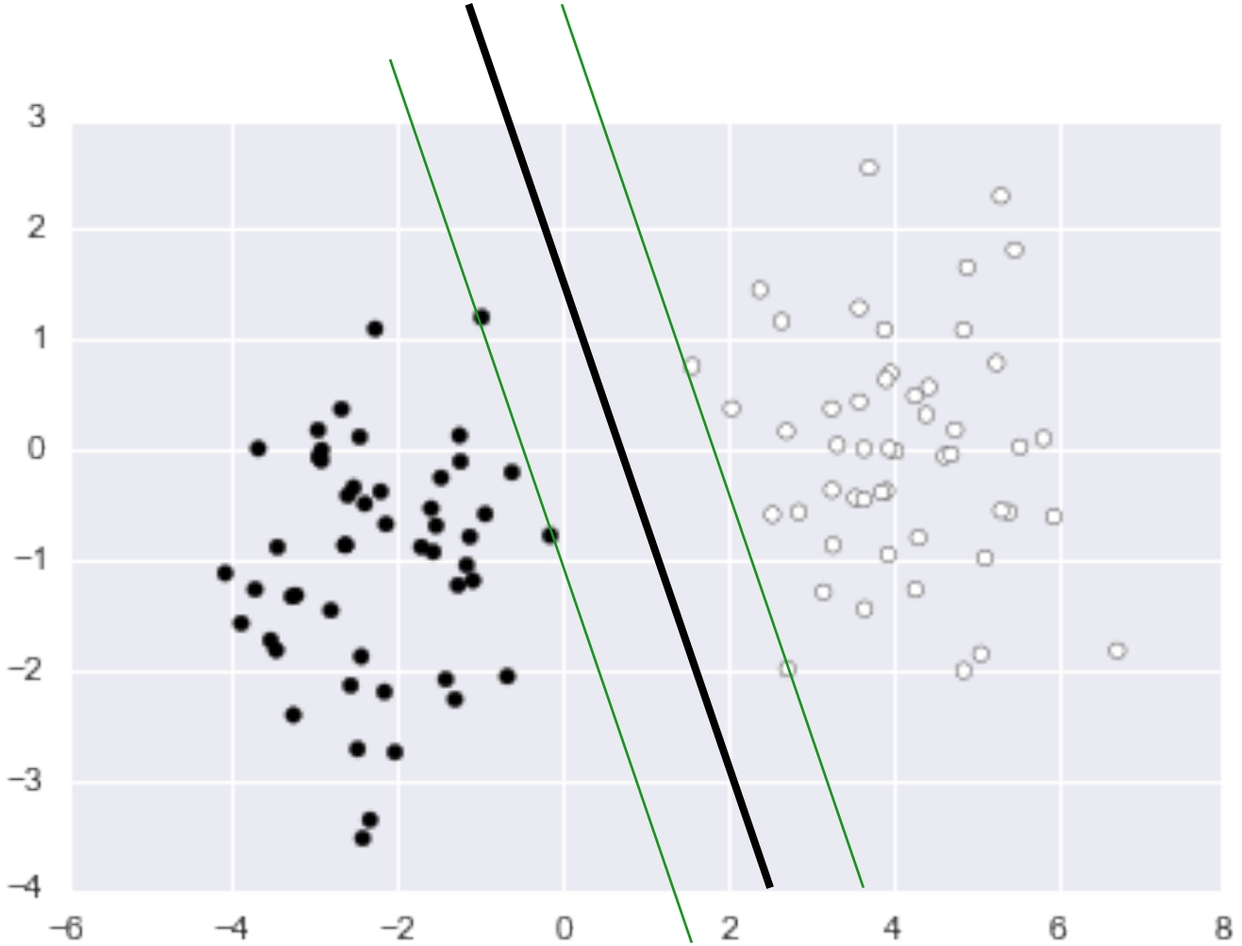
The boundaries depend on the classifier. For example, logistic regression gives a different one than naive bayes

MAXIMAL MARGIN CLASSIFIER



Maximal Margin classifiers explicitly construct a decision boundary that geometrically “makes the most sense”

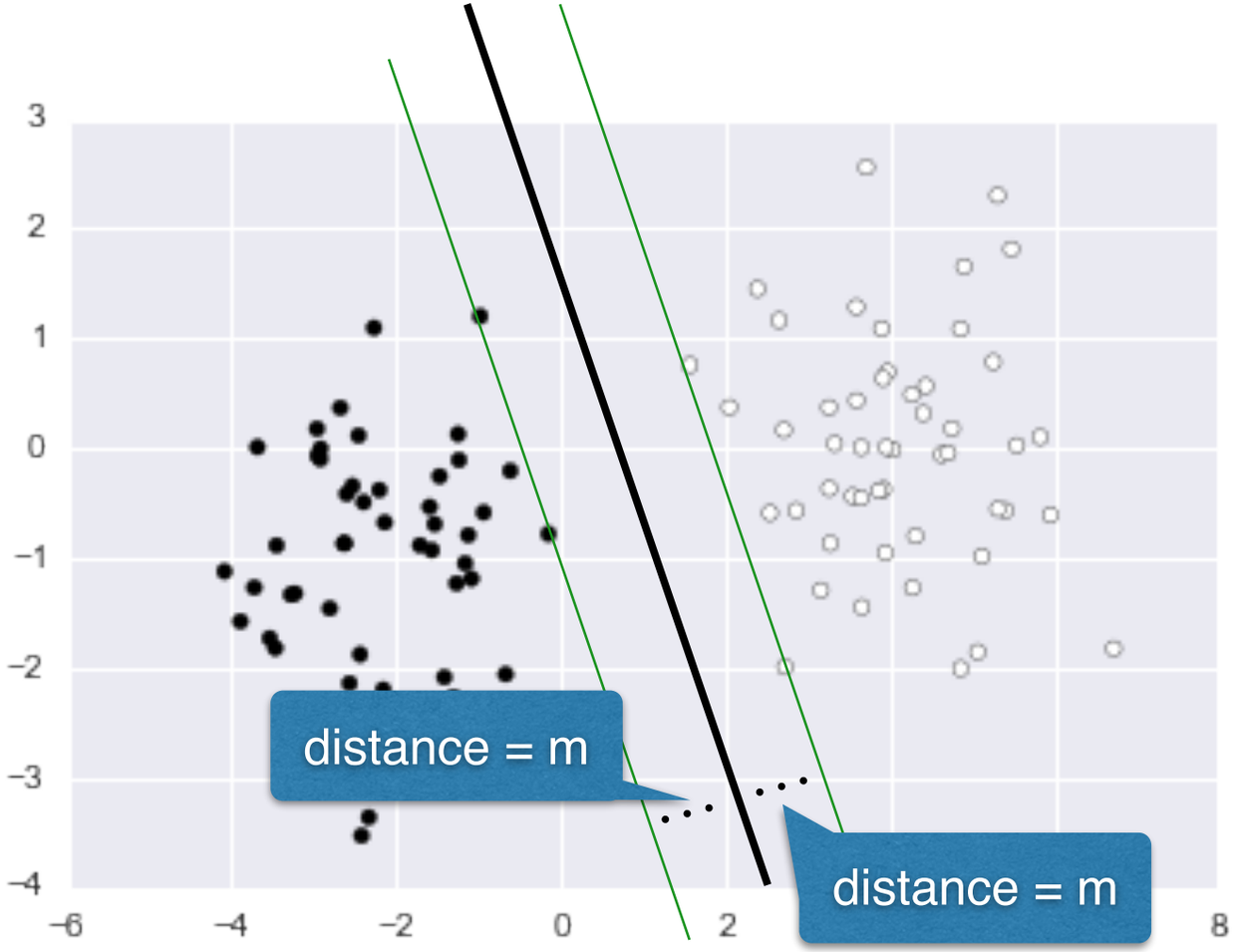
MAXIMAL MARGIN CLASSIFIER



Maximal Margin classifiers explicitly constructs a decision boundary that geometrically “makes the most sense”

It does this by maximizing the **margin**, or the region around the boundary free of points

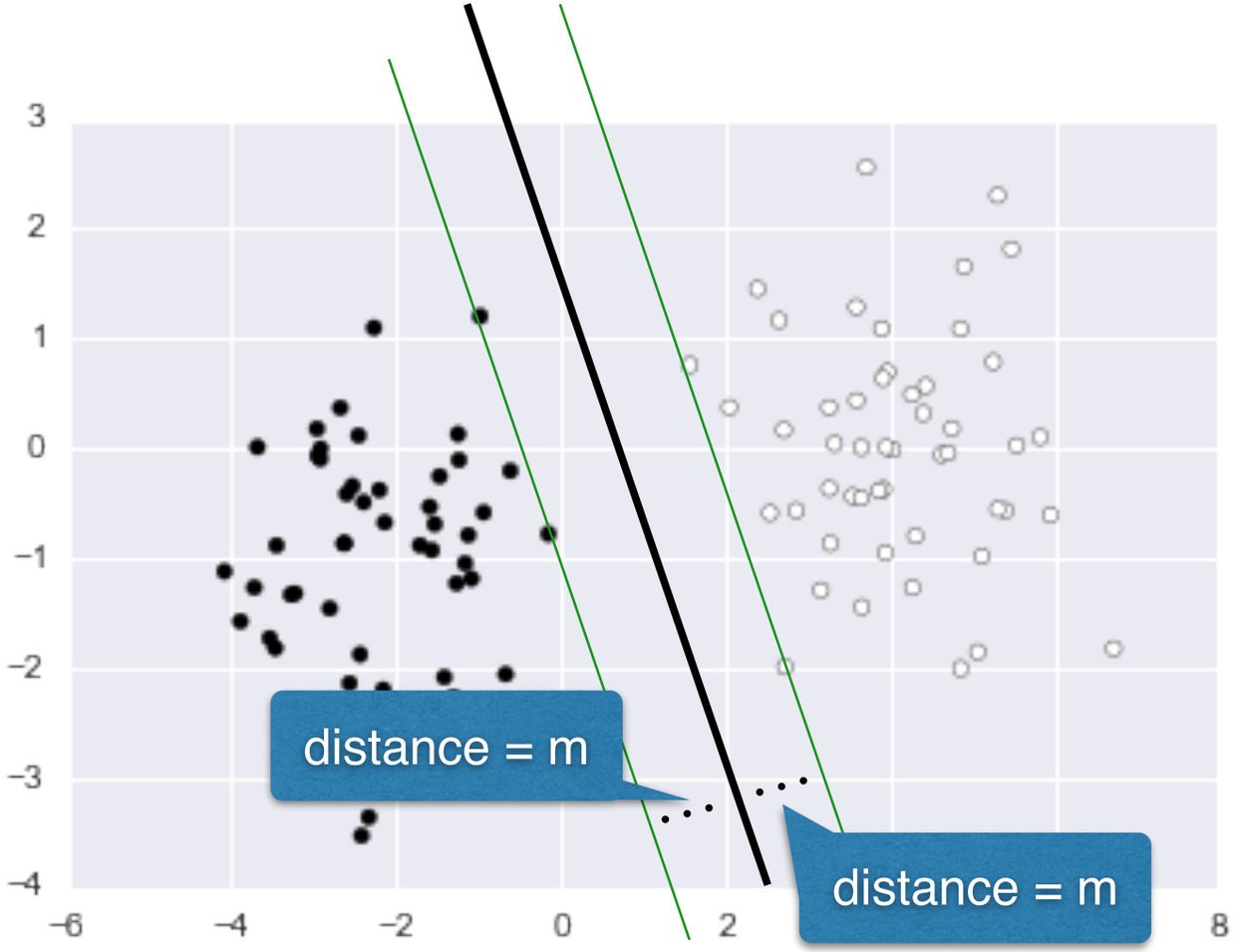
MAXIMAL MARGIN CLASSIFIER



Maximal Margin classifiers explicitly constructs a decision boundary that geometrically “makes the most sense”

It does this by maximizing the **margin**, or the region around the boundary free of points

MAXIMAL MARGIN CLASSIFIER

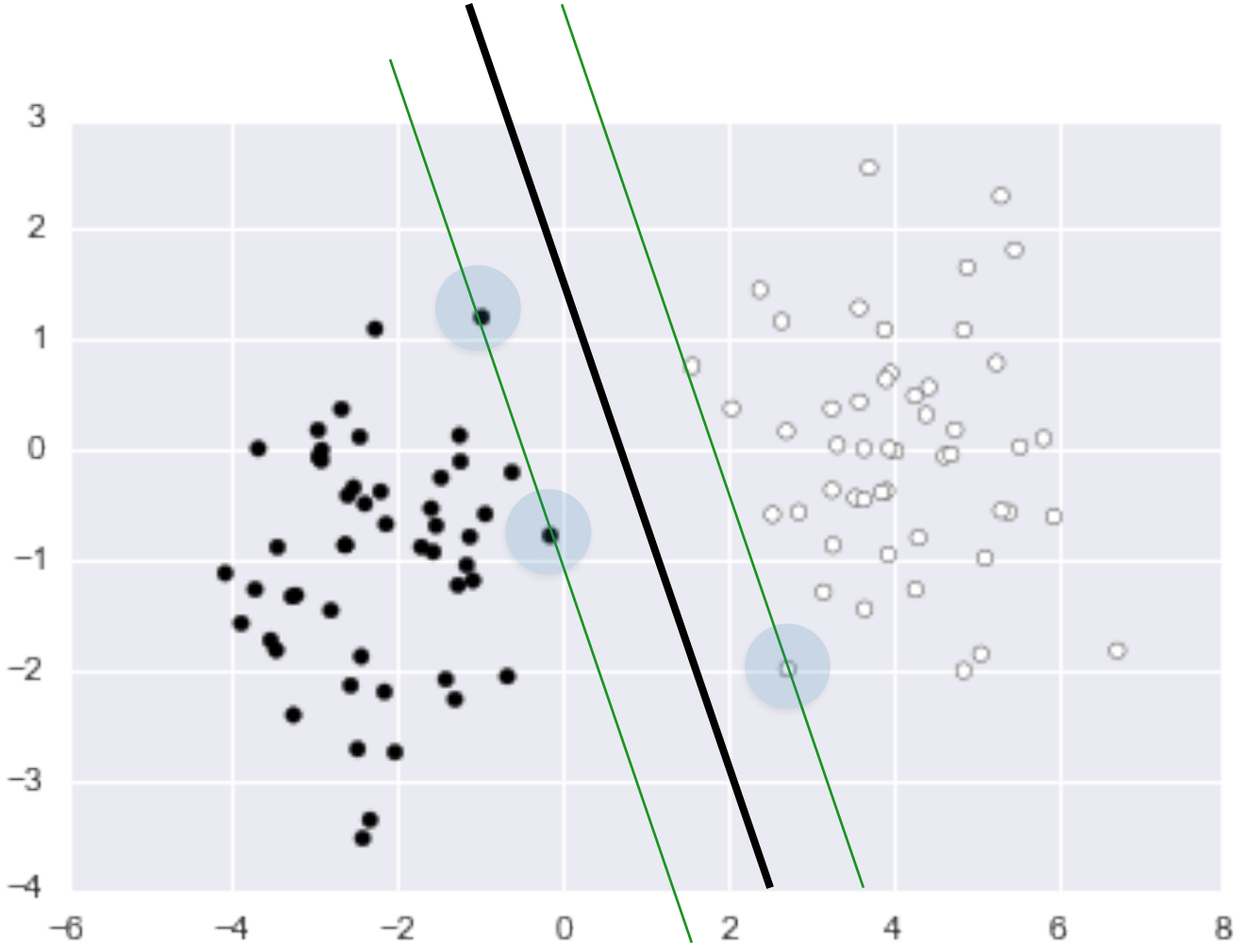


Maximal Margin classifiers explicitly constructs a decision boundary that geometrically “makes the most sense”

It does this by maximizing the **margin**, or the region around the boundary free of points

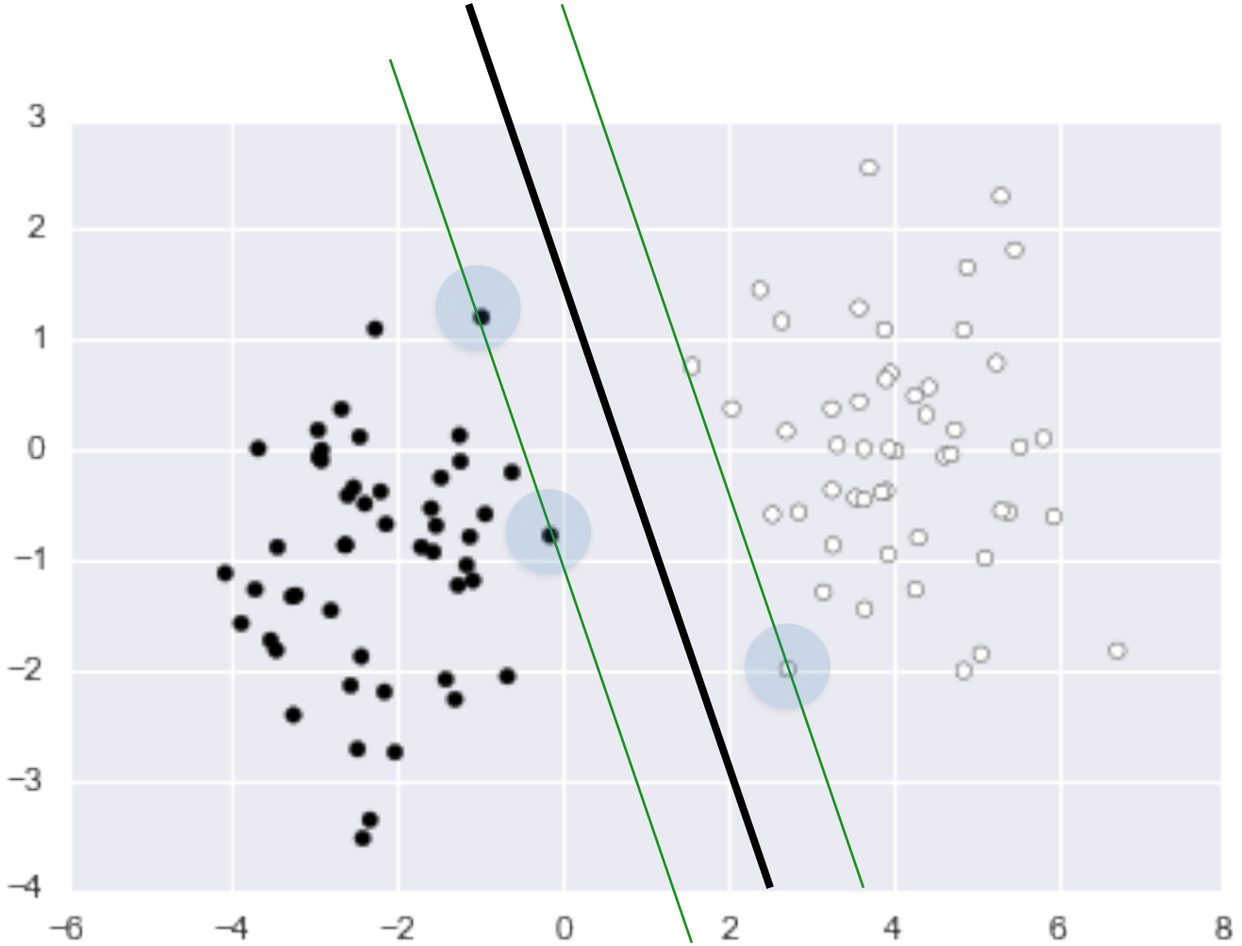
The resulting plane is called the **maximum margin hyperplane**

MAXIMAL MARGIN CLASSIFIER



Notice that the hyperplane only depends on a subset of the training data — i.e only the points on the margins

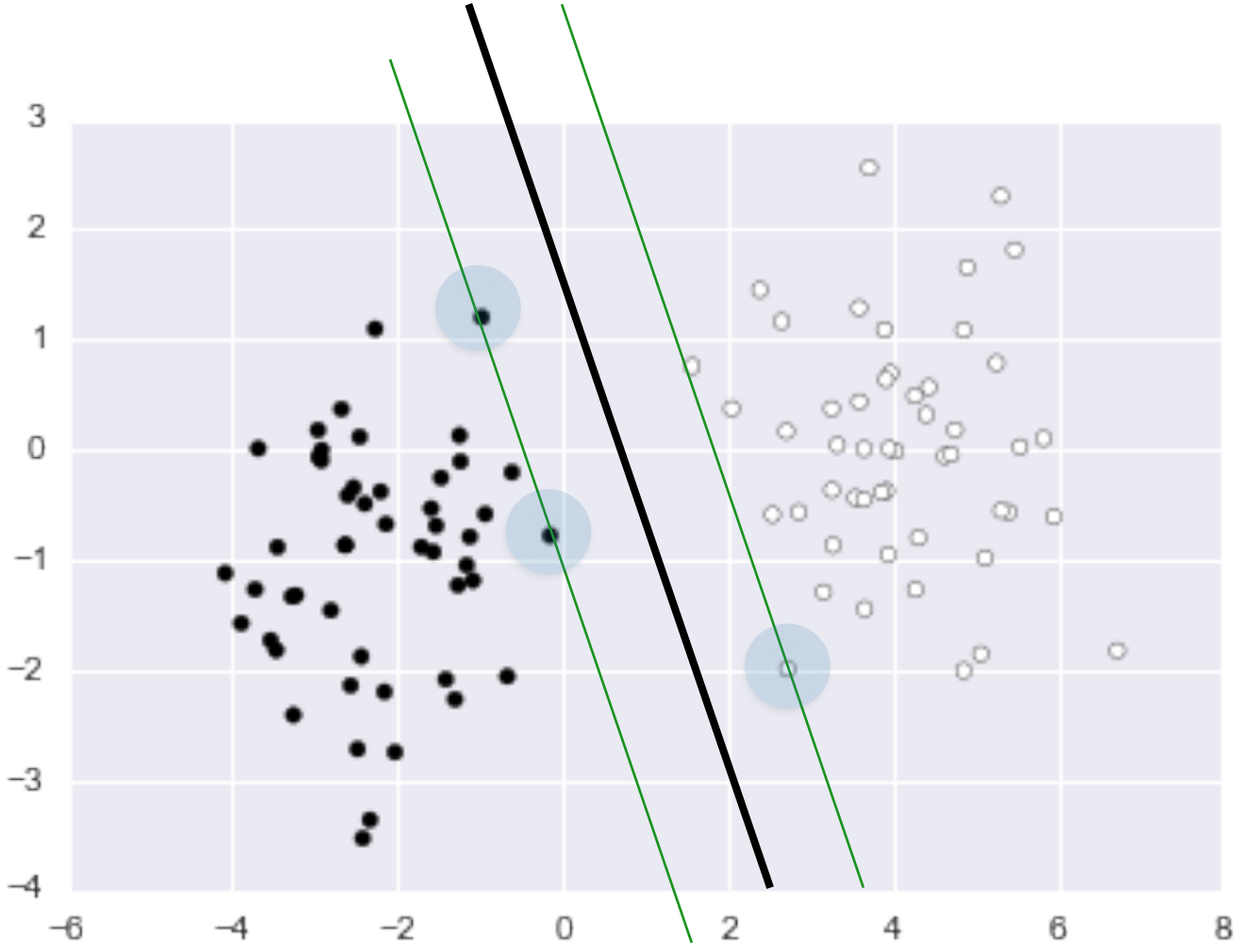
MAXIMAL MARGIN CLASSIFIER



Notice that the hyperplane only depends on a subset of the training data — i.e only the points on the margins

These points are called **support vectors**

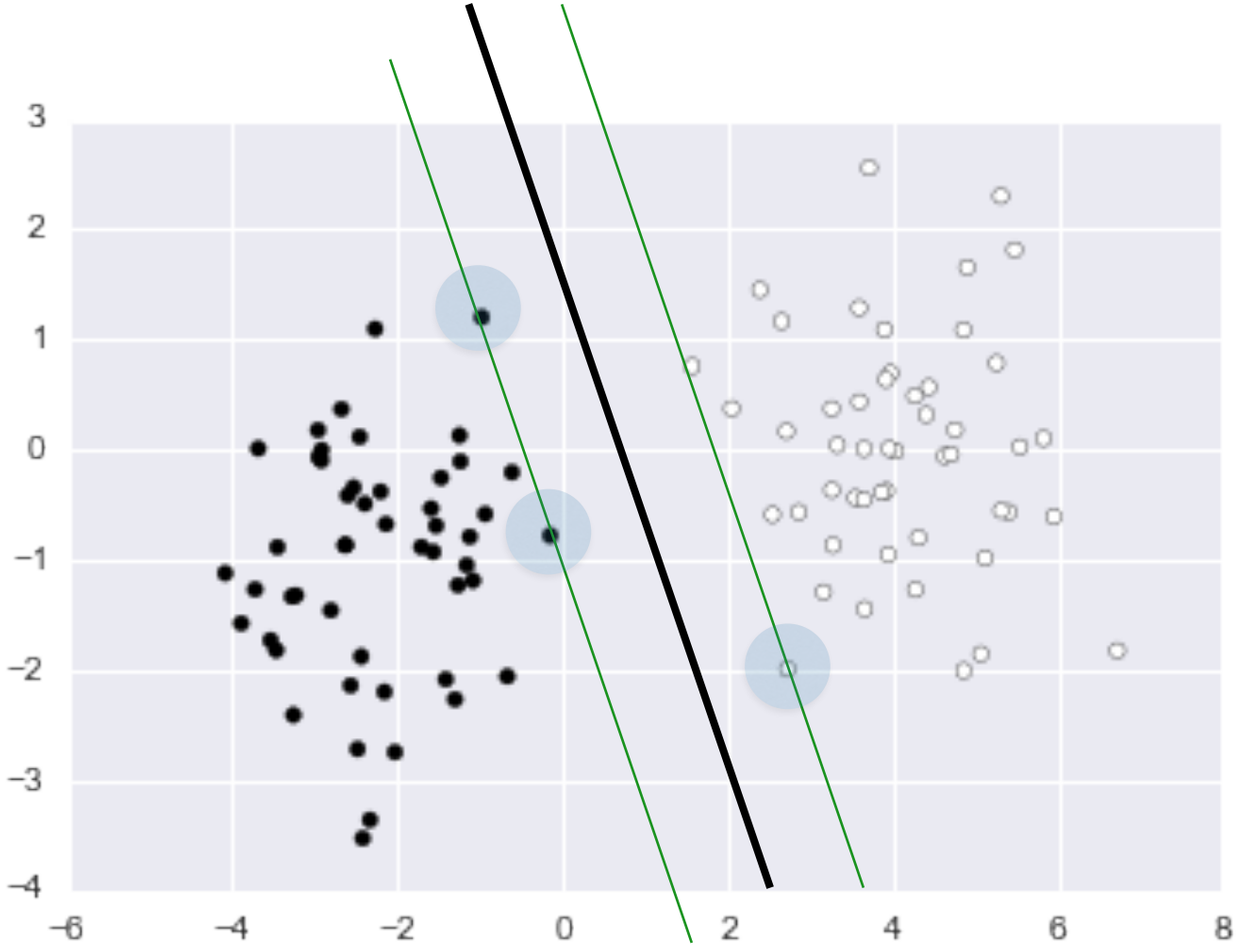
MAXIMAL MARGIN CLASSIFIER



Notice that the hyperplane only depends on a subset of the training data — i.e only the points on the margins

These points are called **support vectors**. They "support" the hyperplane—as in, if they were slightly moved, the hyperplane, and therefore, the decision boundary would change

MAXIMAL MARGIN CLASSIFIER

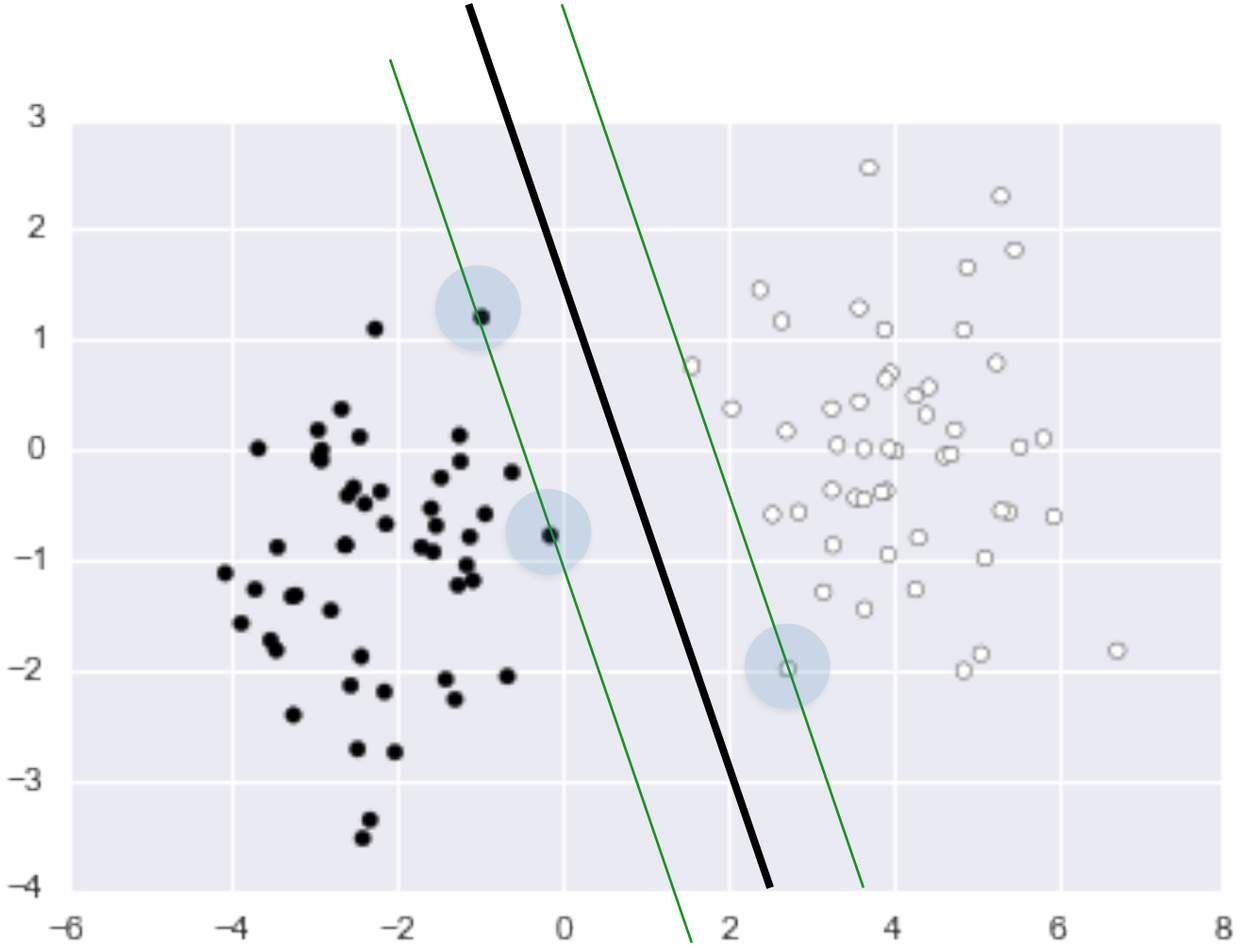


Notice that the hyperplane only depends on a subset of the training data — i.e only the points on the margins

These points are called **support vectors**.

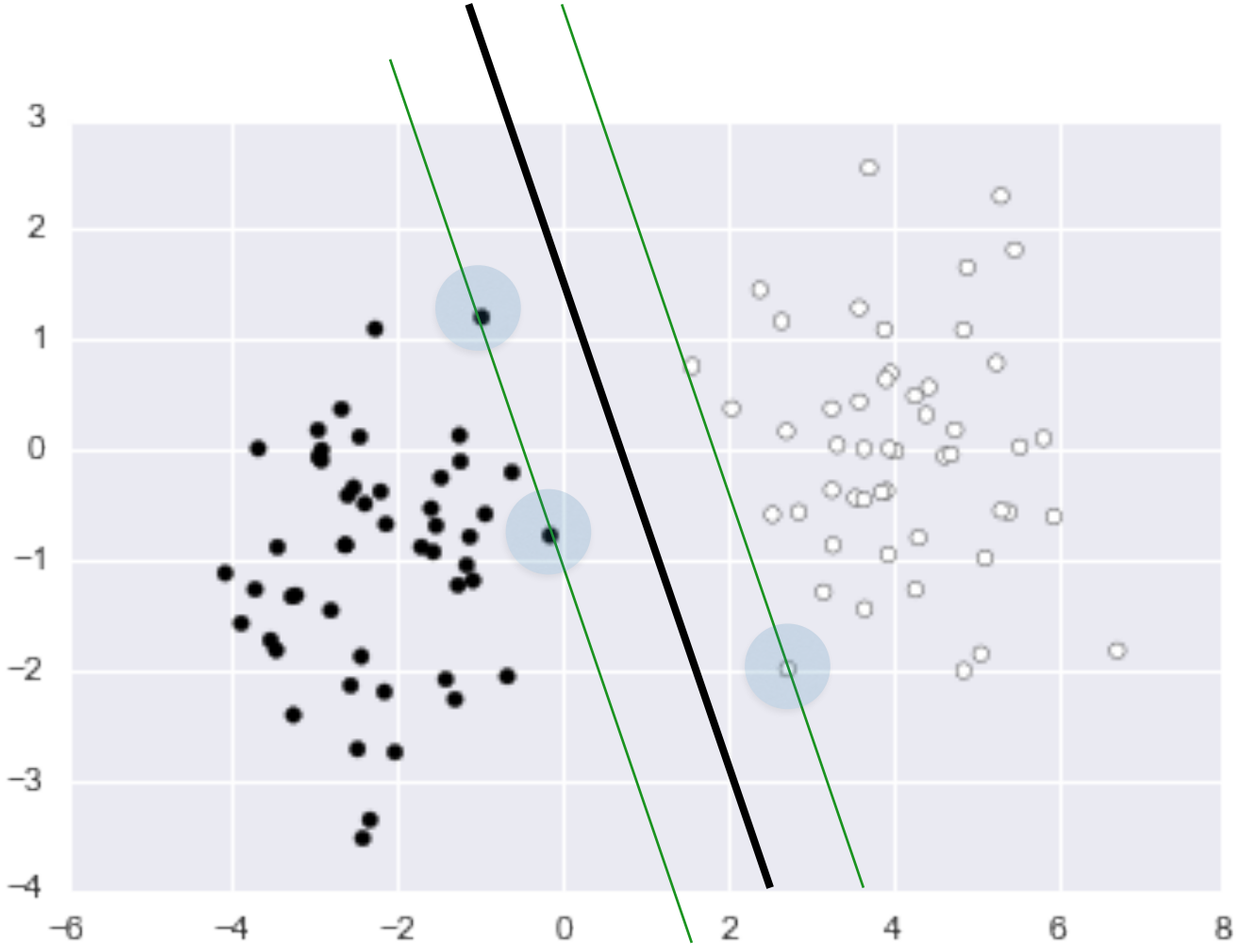
However, the other points do not change the hyperplane at all!

MAXIMAL MARGIN CLASSIFIER



With our new maximum margin hyperplane, we'll classify data points based on what side of the hyperplane they lie on

MAXIMAL MARGIN MATHEMATICS



We won't go into many details, but a few things to note:

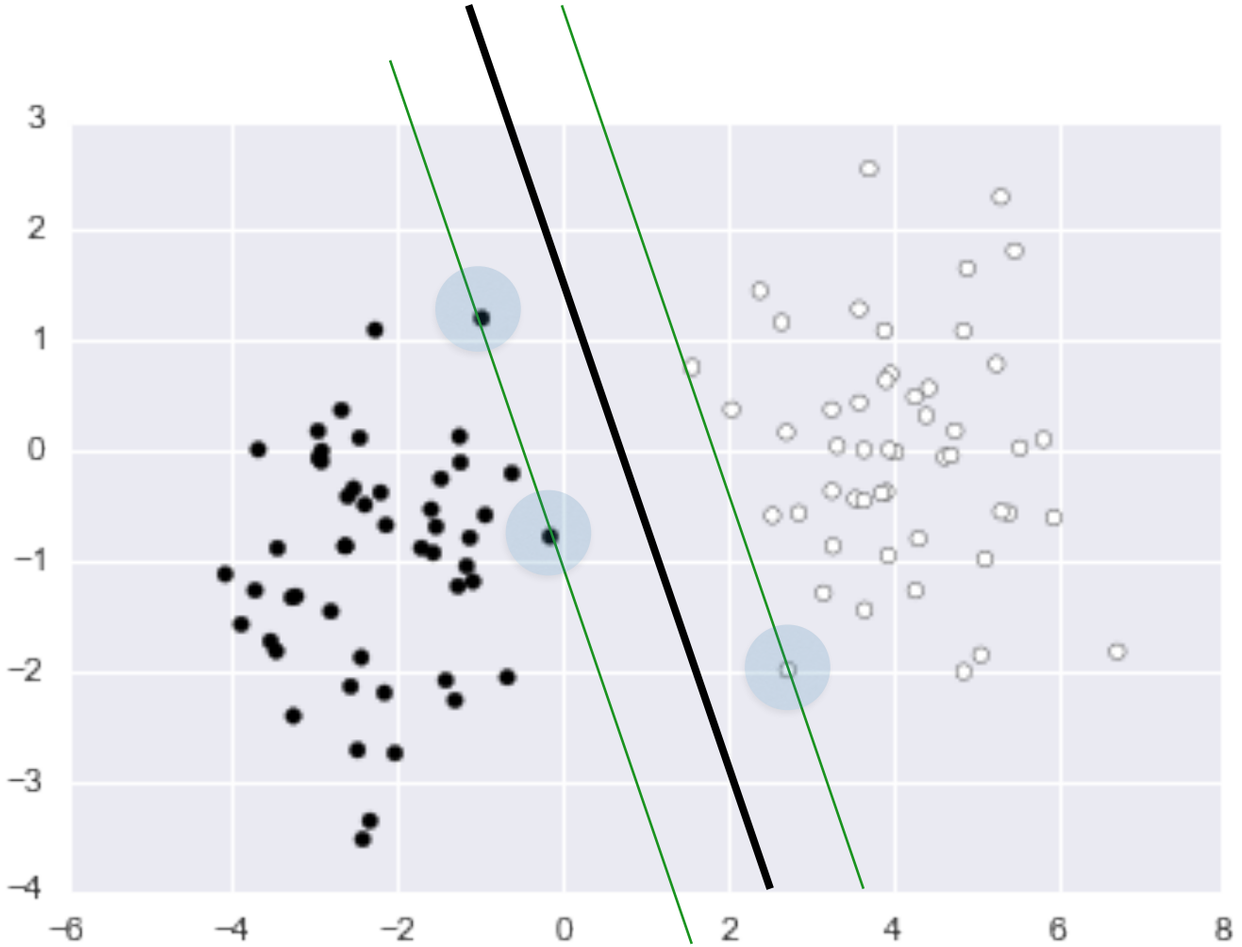
- We will denote the class labels in y as {+1,-1}
- The hyperplane has the form

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n = 0$$

- where

$$\sum_i^n \beta_i^2 = 1$$

MAXIMAL MARGIN MATHEMATICS



Then we will classify using these rules:

Classify y = 1 if

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n > 0$$

Classify y = -1 if

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n < 0$$

MAXIMAL MARGIN MATHEMATICS

The beta values are solved through analytic geometry involving lagrangian multipliers. We won't go into the details, but you should know:

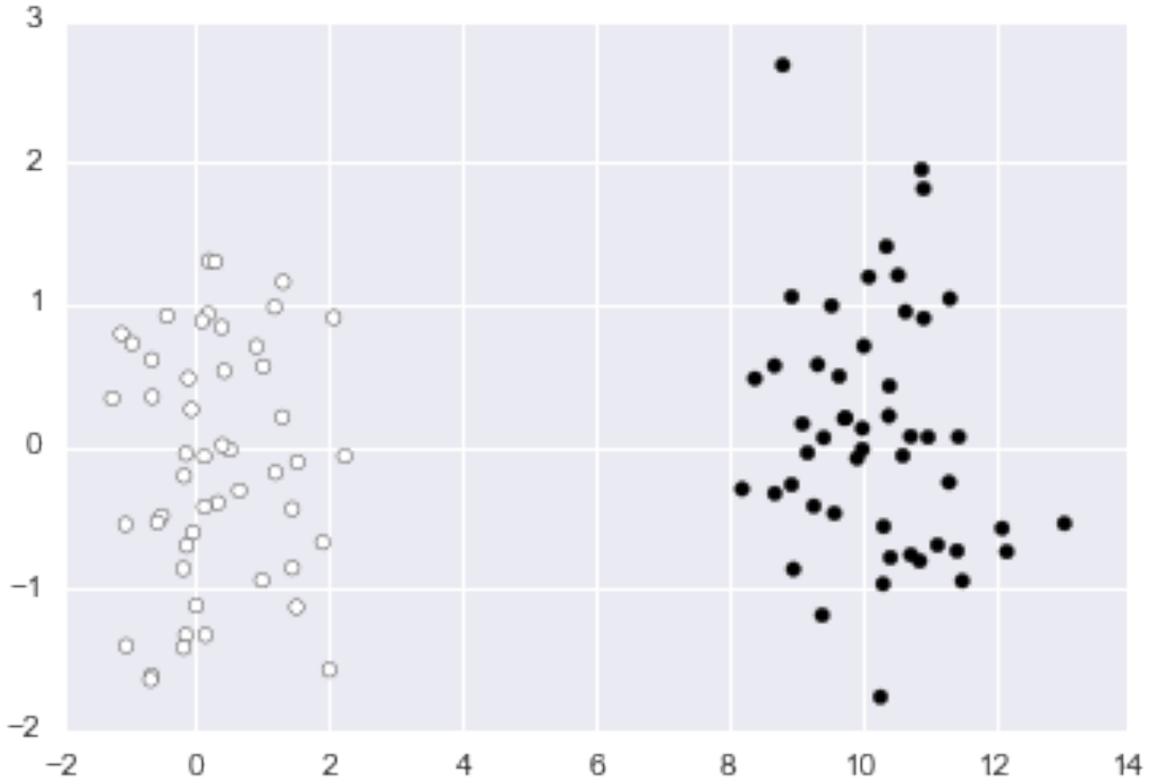
- The objective is convex, so there is a global minima (i.e. one solution)
- The maximal margin classifier is a binary linear classifier whose decision boundary is explicitly constructed to maximize the margins < —— > minimize generalization error

II.SUPPORT VECTOR CLASSIFIER

SUPPORT VECTOR CLASSIFIERS

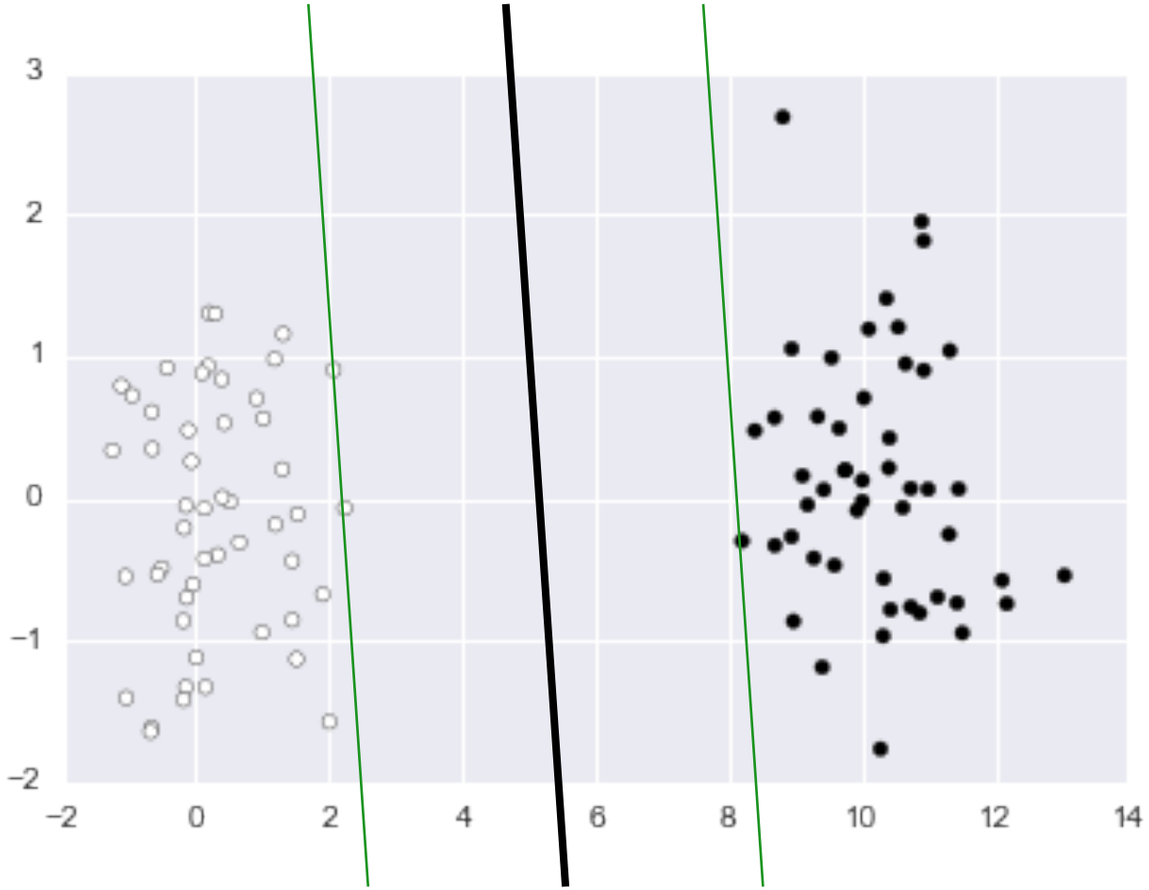
Maximum margin classifiers have actually been around since 1960 (or so), but have seen limited use... let's see why

SUPPORT VECTOR CLASSIFIERS



Let's apply a maximum margin classifier to the left

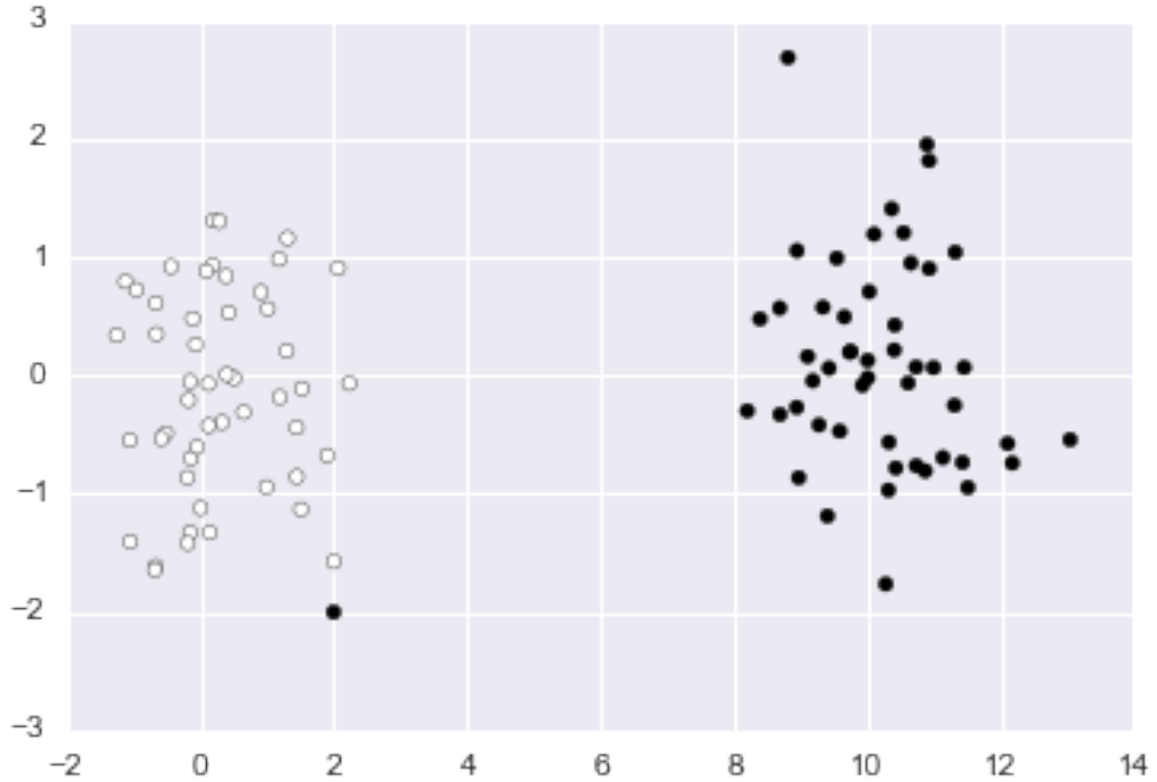
SUPPORT VECTOR CLASSIFIERS



If the data are linearly separable, the training error is zero and we can build the maximum margin classifier

The margin is nice and wide

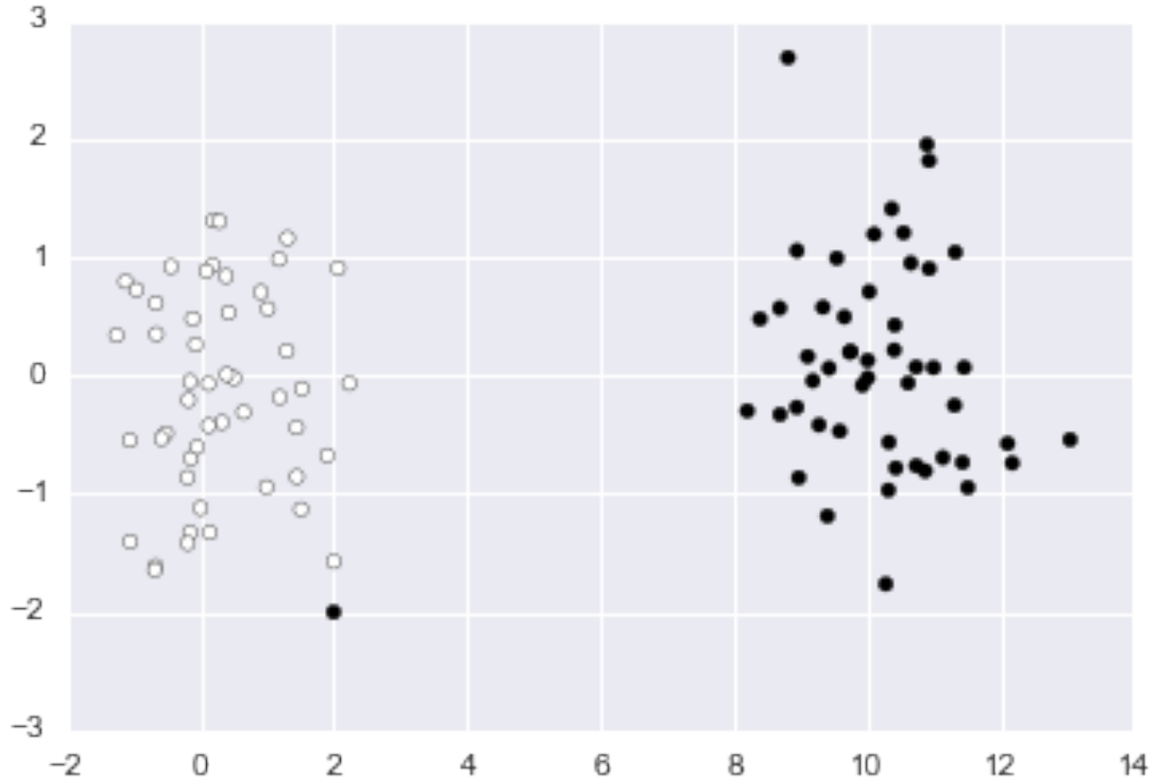
SUPPORT VECTOR CLASSIFIERS



If the data are linearly separable, the training error is zero and we can build the maximum margin classifier

But what if our data has a single outlier?

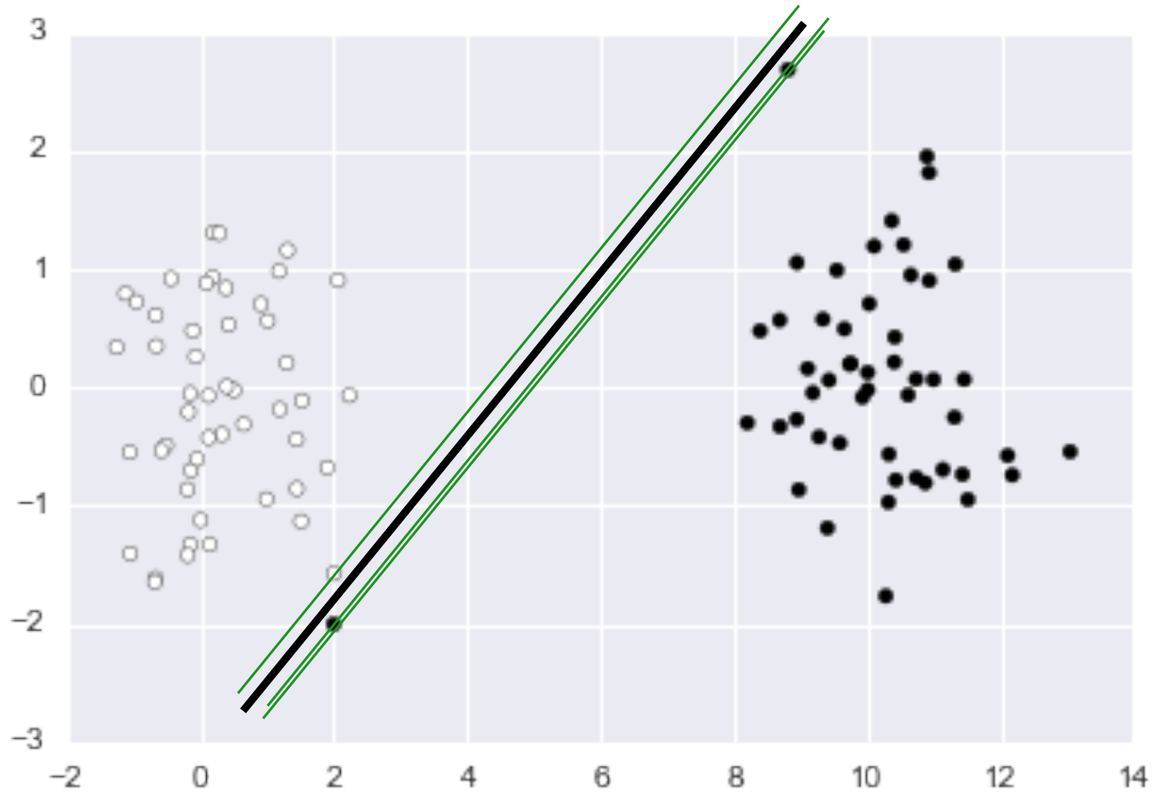
SUPPORT VECTOR CLASSIFIERS



If the data are linearly separable, the training error is zero and we can build the maximum margin classifier

But what if our data has a single outlier? This will disproportionately impact the result since the maximal margin classifier tries to perfectly separate all data

SUPPORT VECTOR CLASSIFIERS

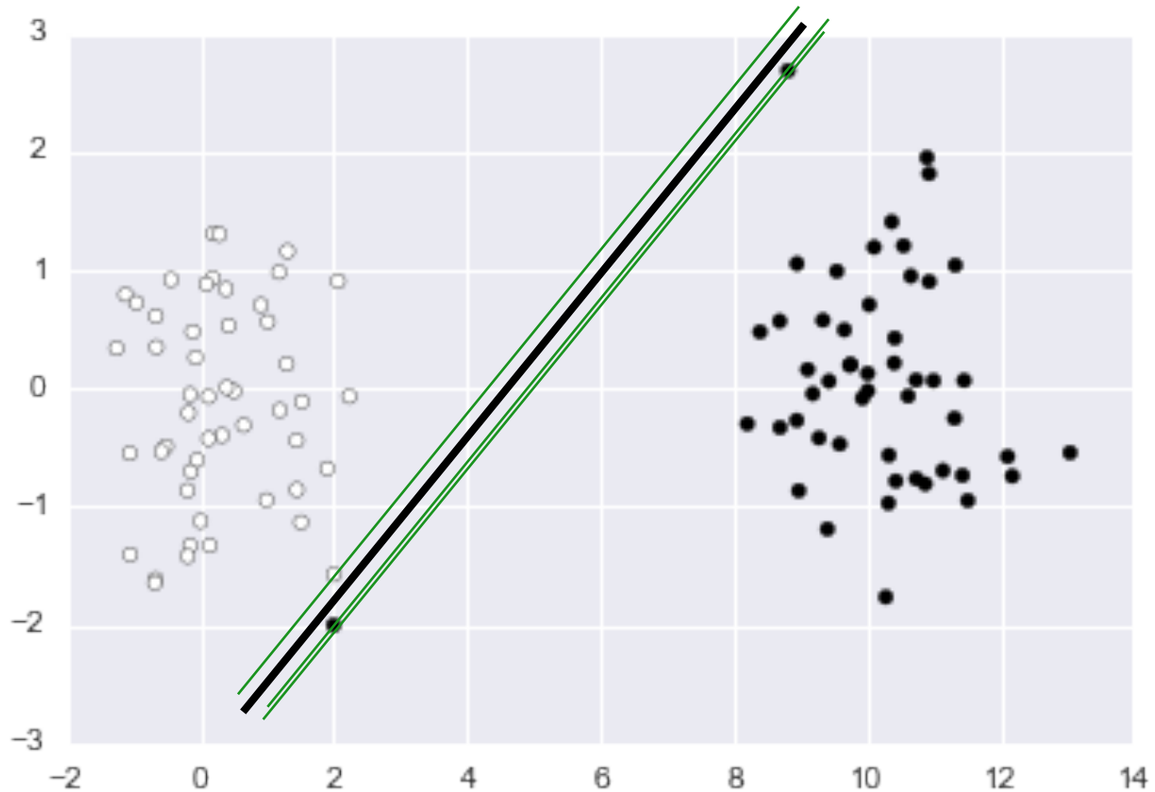


If the data are linearly separable, the training error is zero and we can build the maximum margin classifier

But what if our data has a single outlier? This will disproportionately impact the result since the maximal margin classifier tries to perfectly separate all data

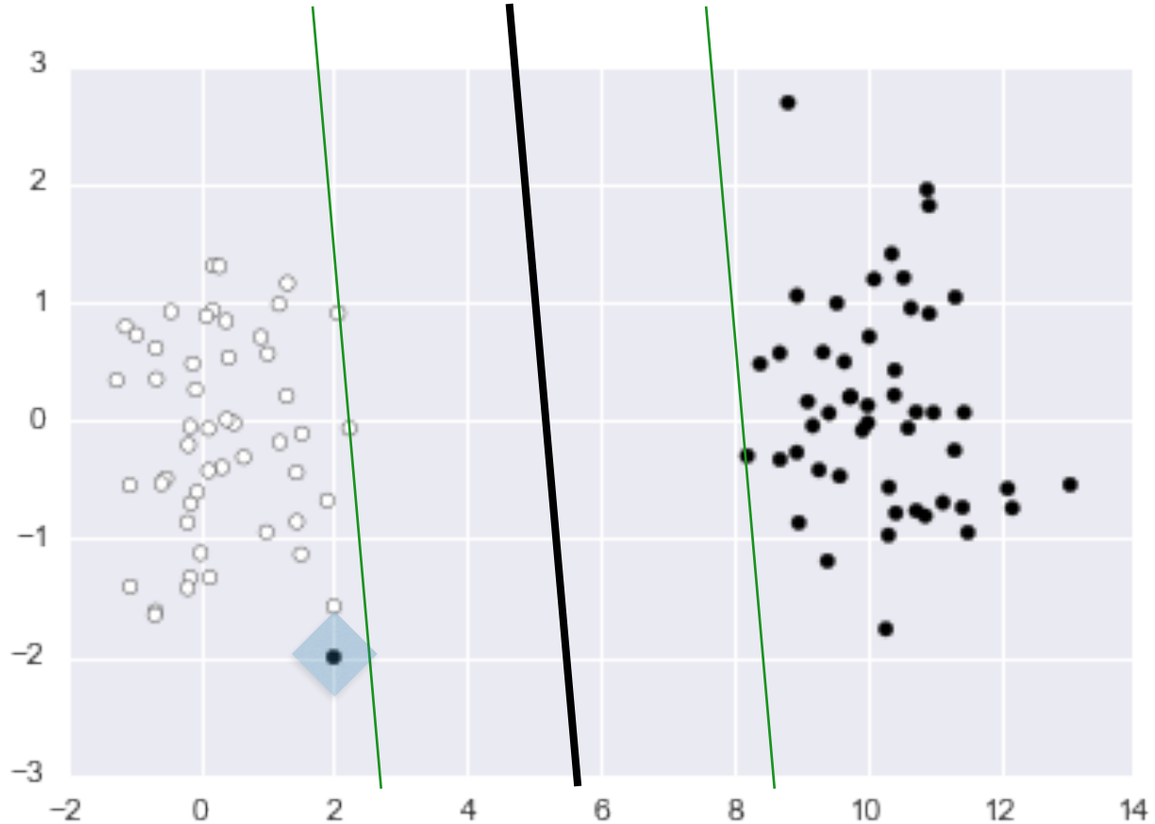
The margin is extremely small

SUPPORT VECTOR CLASSIFIERS



This calls for our good friend **regularization**

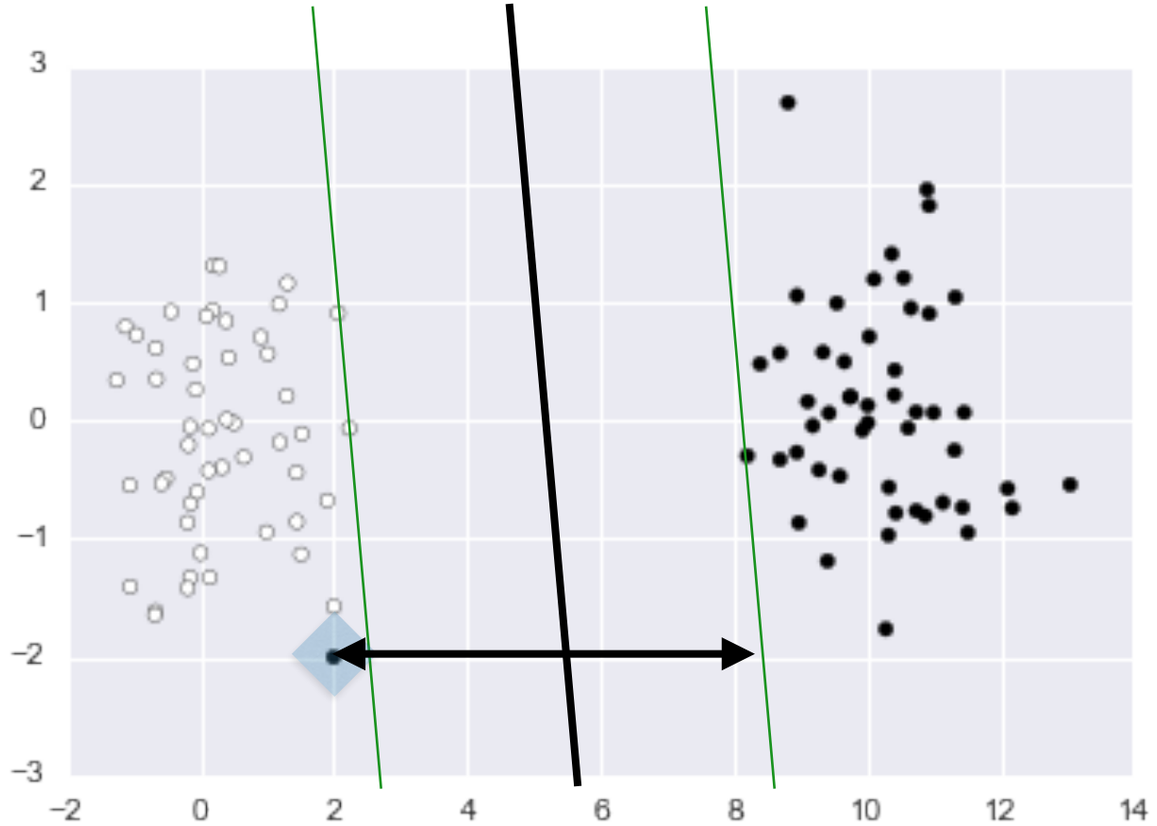
SUPPORT VECTOR CLASSIFIERS



This calls for our good friend **regularization**

We want a tradeoff between the width of the margin and the number of misclassified samples

SUPPORT VECTOR CLASSIFIERS

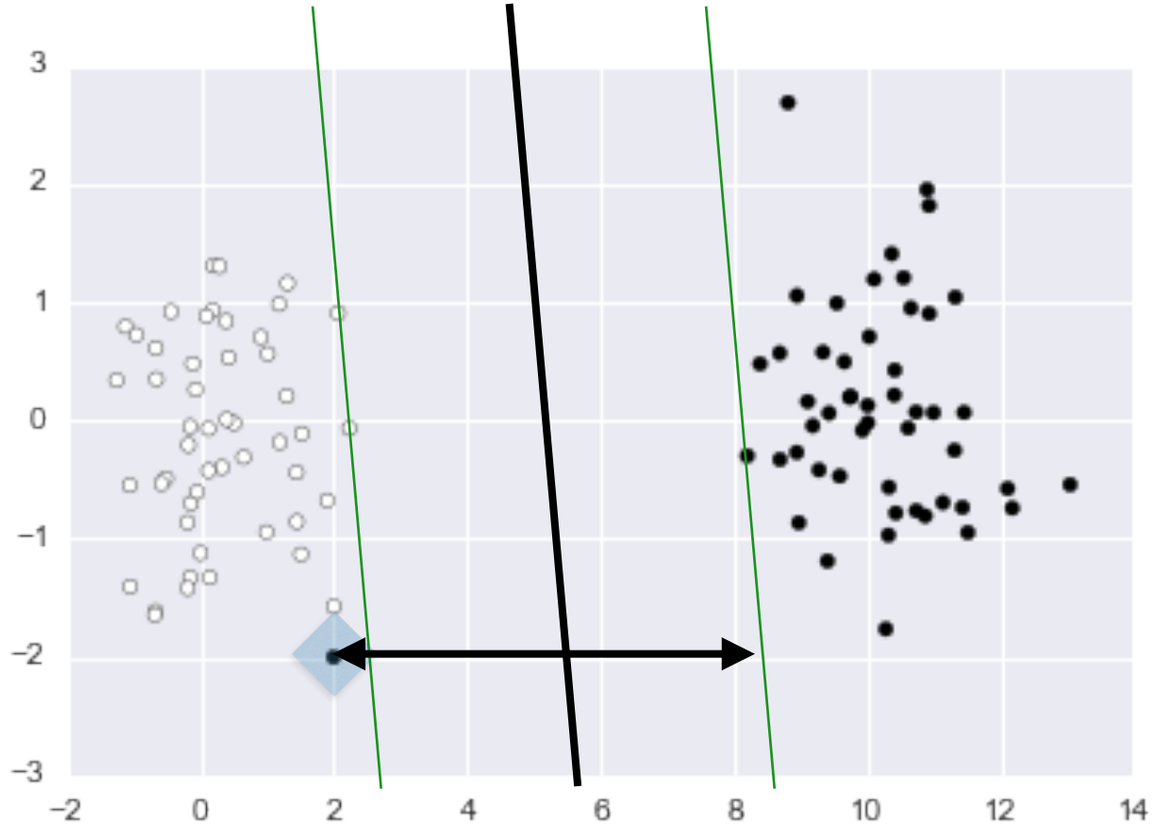


This calls for our good friend **regularization**

We want a tradeoff between the width of the margin and the number of misclassified samples

This is again related to the bias-variance tradeoff, which is done by **slack variables**

SUPPORT VECTOR CLASSIFIERS



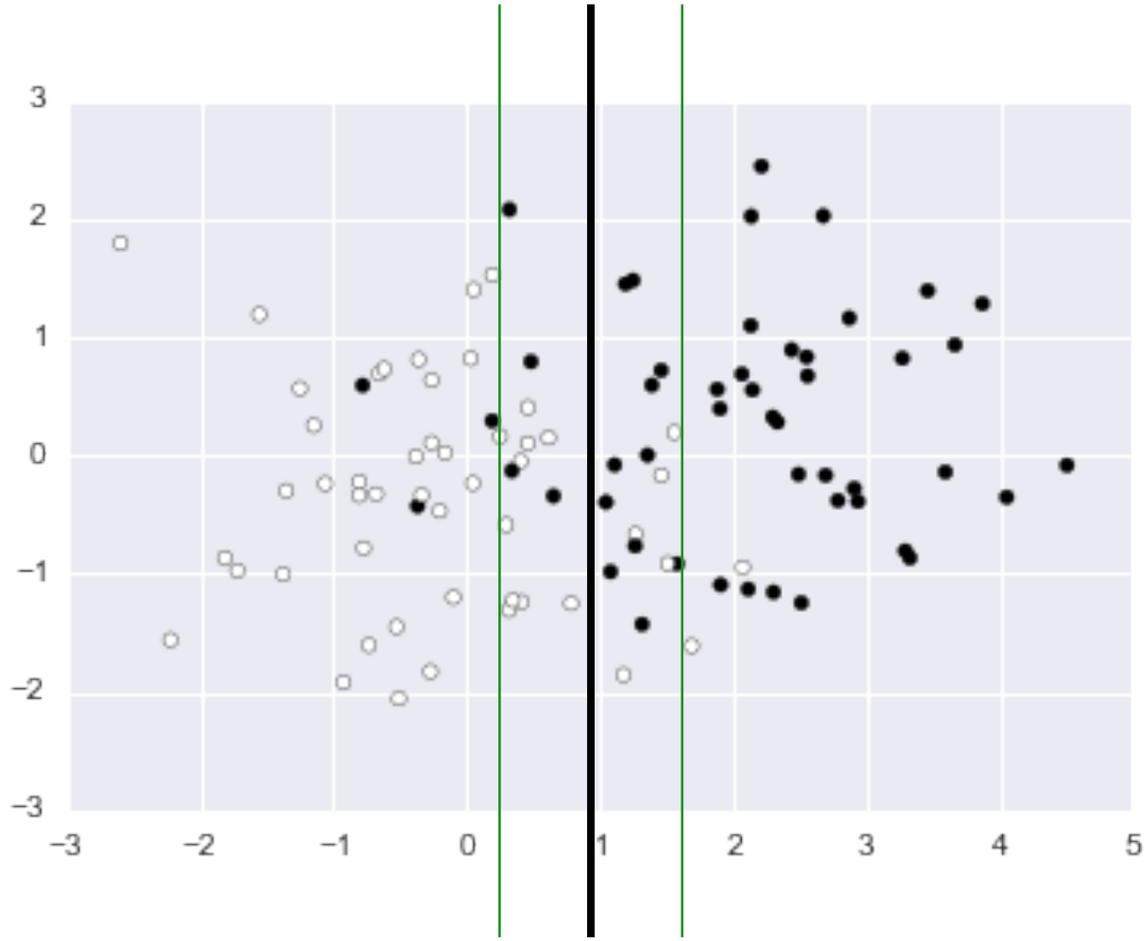
This calls for our good friend **regularization**

We want a tradeoff between the width of the margin and the number of misclassified samples

This is again related to the bias-variance tradeoff, which is done by **slack variables**

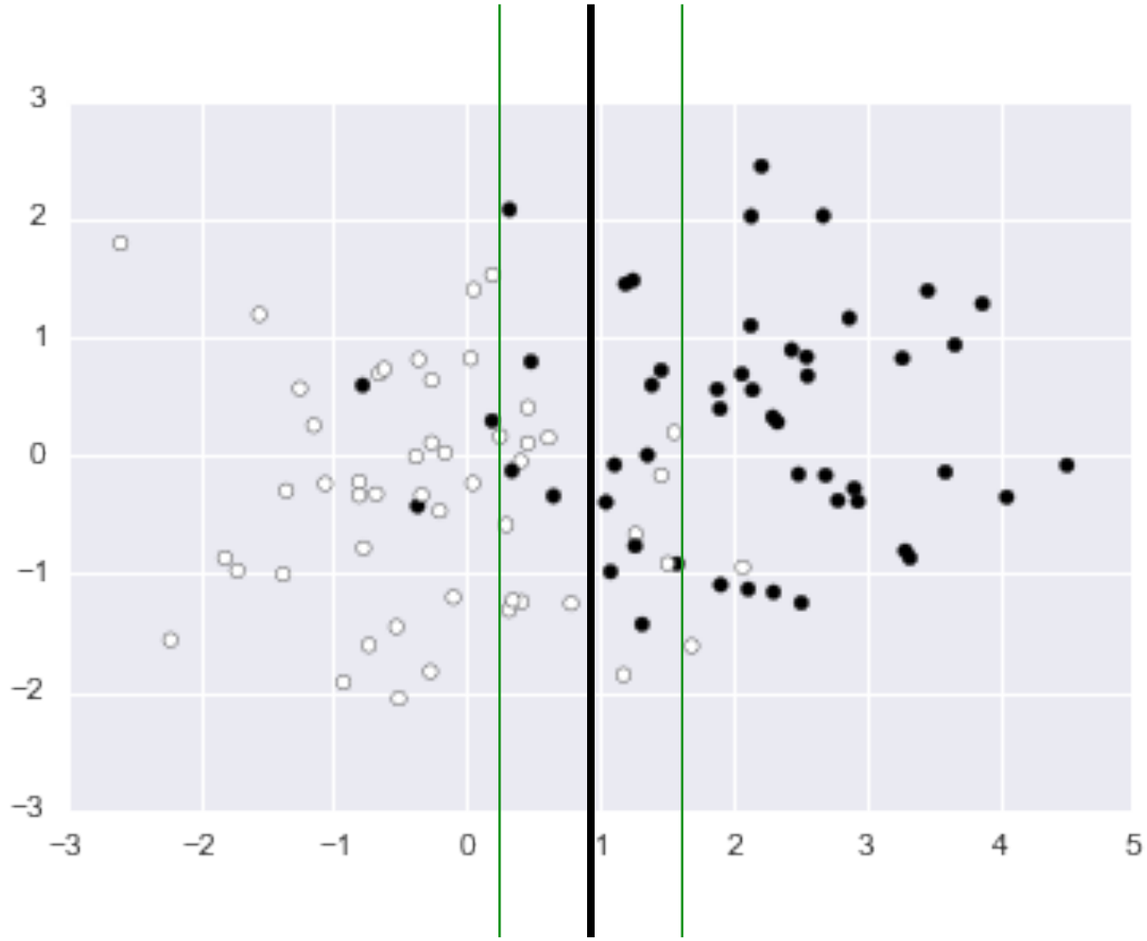
We can solve our original problem, but allow some **budget** for points to be on the wrong side of the margin

SUPPORT VECTOR CLASSIFIERS



When our data is not very well separated, we definitely need to allow for **slack** in generating the separating hyperplane

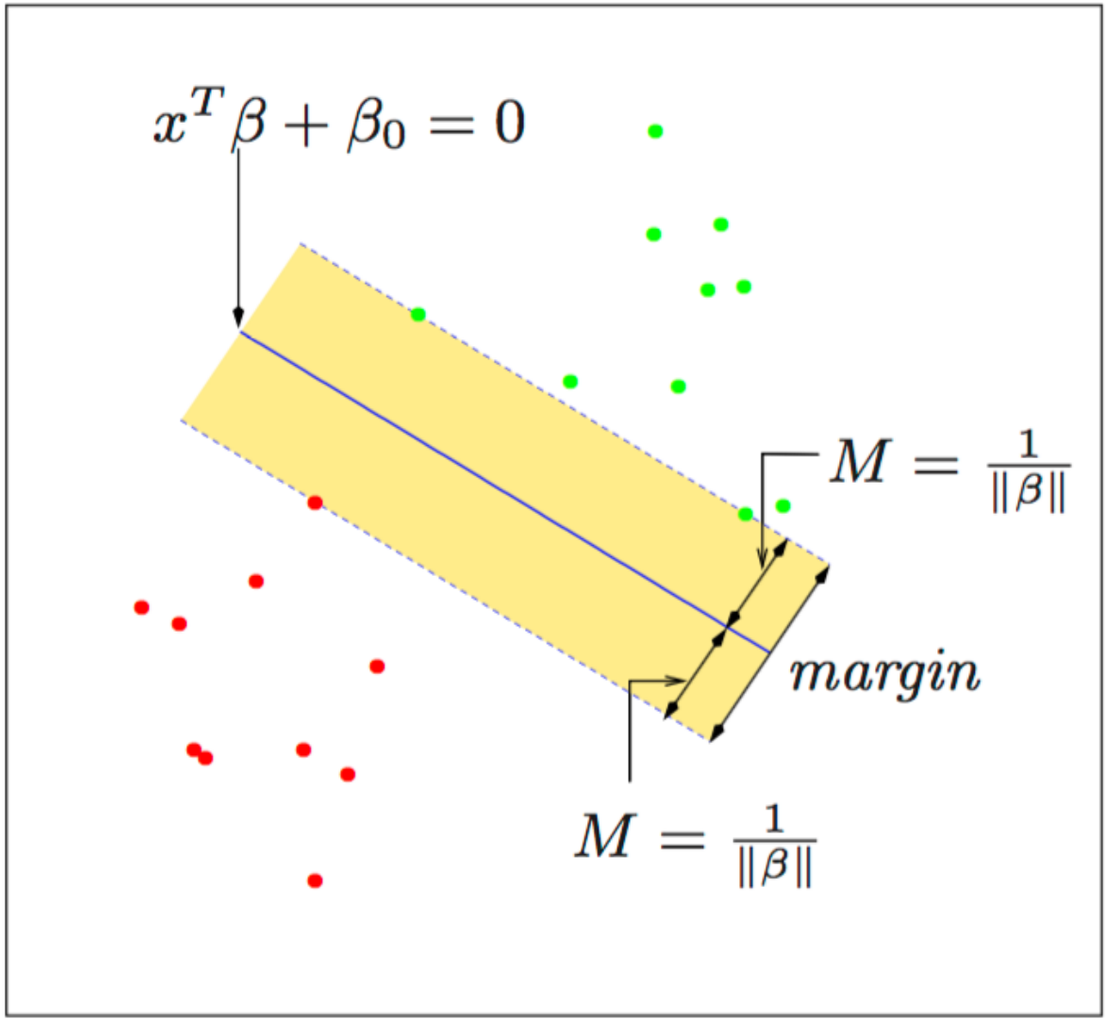
SUPPORT VECTOR CLASSIFIERS



When our data is not very well separated, we definitely need to allow for **slack** in generating the separating hyperplane

The usage of slack variables in generating the separating hyperplane is called **support vector classifiers**

SUPPORT VECTOR CLASSIFIERS

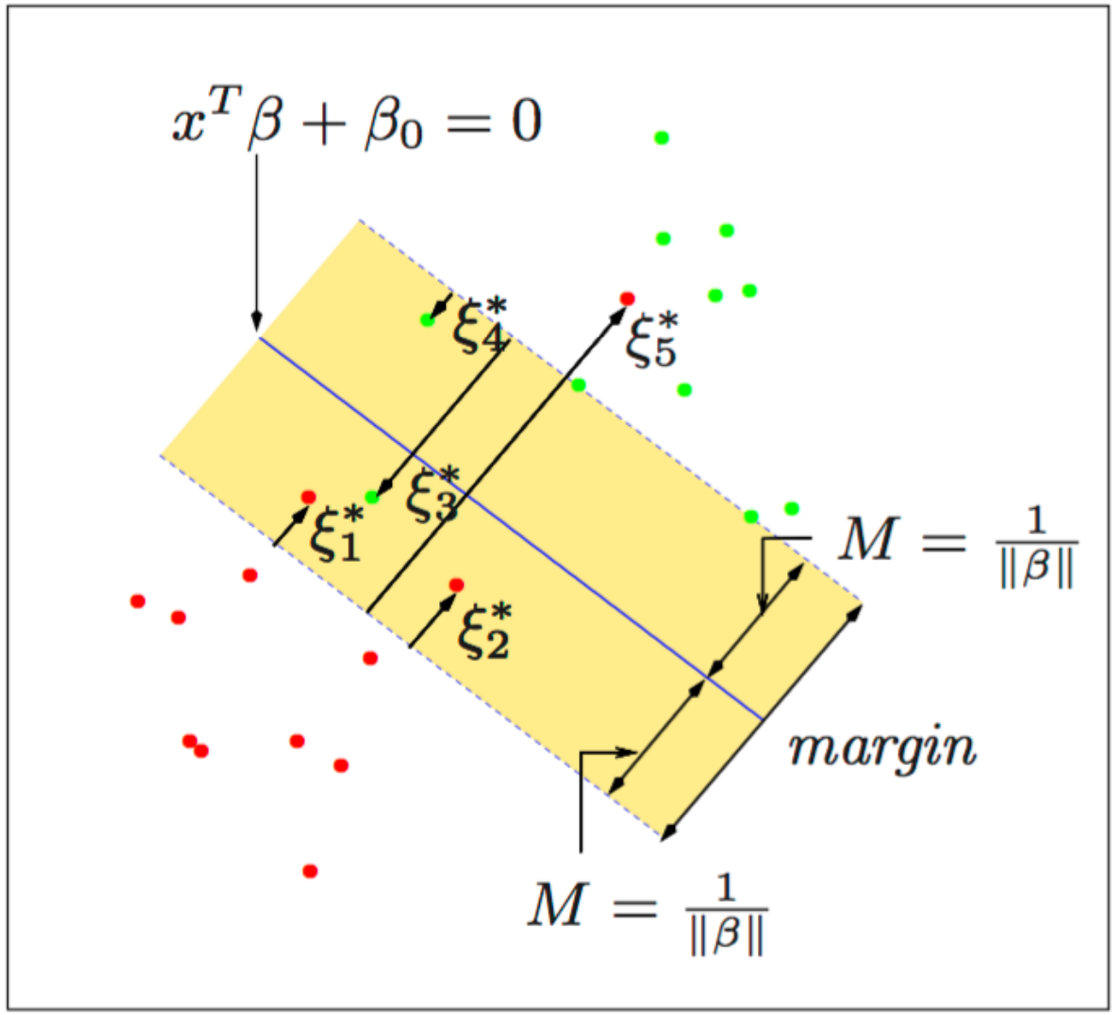


When our data is not very well separated, we definitely need to allow for **slack** in generating the separating hyperplane

The usage of slack variables in generating the separating hyperplane is called **support vector classifiers**

***The support vectors are now the **points on the margin** and the **points on the wrong side of the margin**

SUPPORT VECTOR CLASSIFIERS

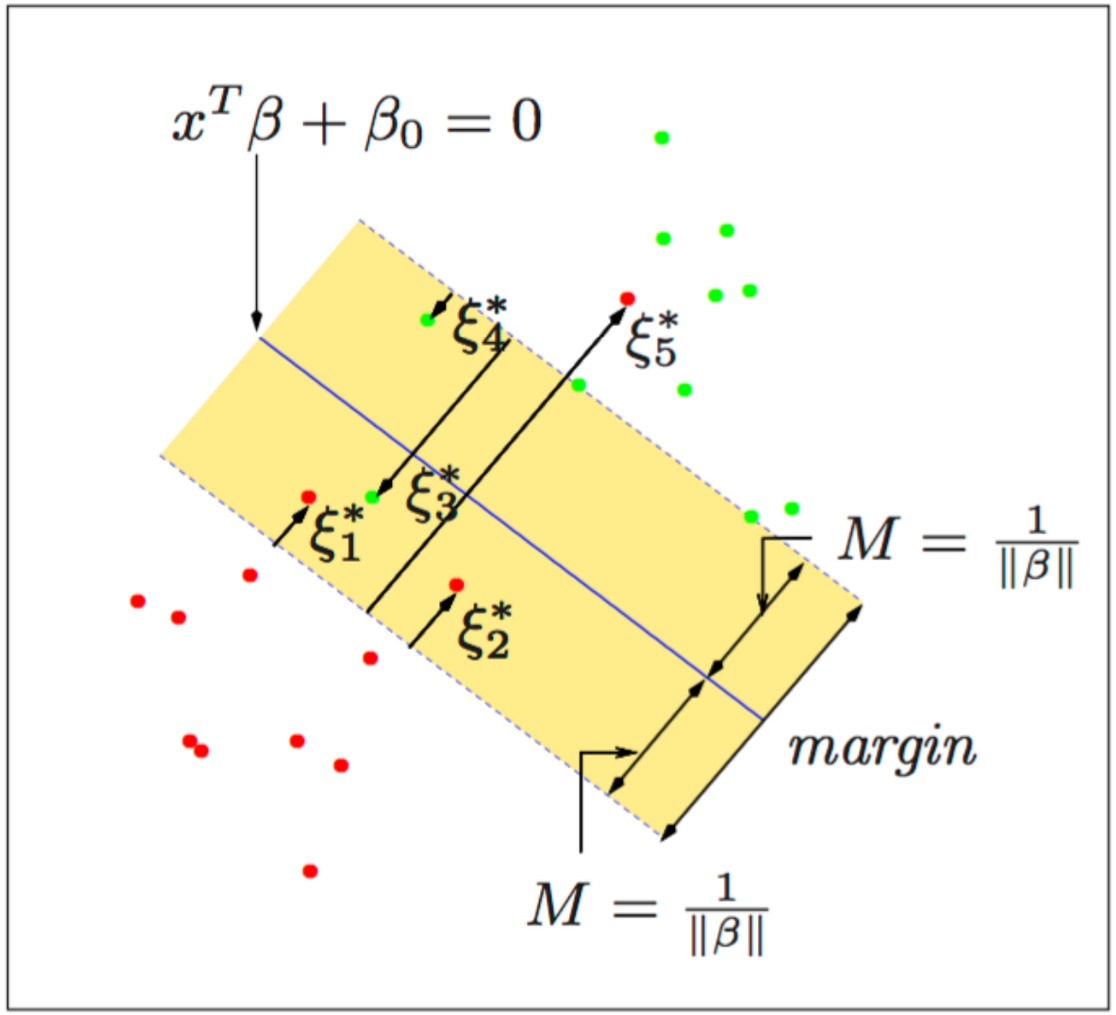


When our data is not very well separated, we definitely need to allow for **slack** in generating the separating hyperplane

The usage of slack variables in generating the separating hyperplane is called **support vector classifiers**

***The support vectors are now the **points on the margin** and the **points on the wrong side of the margin**

SUPPORT VECTOR CLASSIFIERS



Allowing certain points to be on the wrong side of the margin gave Support Vector Classifiers much more use over Maximal margin classifiers.

The new hyperplanes, regularized by **C** (indicating the extent of slack variable error), allowed the classifier to be **more consistent** and in many cases, **solvable**.

III.SUPPORT VECTOR MACHINES

SUPPORT VECTOR MACHINES

We started out with the concept of a **maximal margin classifier.** Its sole purpose was to generate a hyperplane that maximized the margins (distance to the closest point). The hyperplane was defined by **support vectors** that lied on the margins.

SUPPORT VECTOR MACHINES

We started out with the concept of a **maximal margin classifier**. Its sole purpose was to generate a hyperplane that maximized the margins (distance to the closest point). The hyperplane was defined by **support vectors** that lied on the margins.

Then, we introduced **slack variables** that allowed a certain amount of points to be on the wrong side of the hyperplane. The new margin was defined by **support vectors** which were points lying on the margin as well as points on the wrong side of the margin

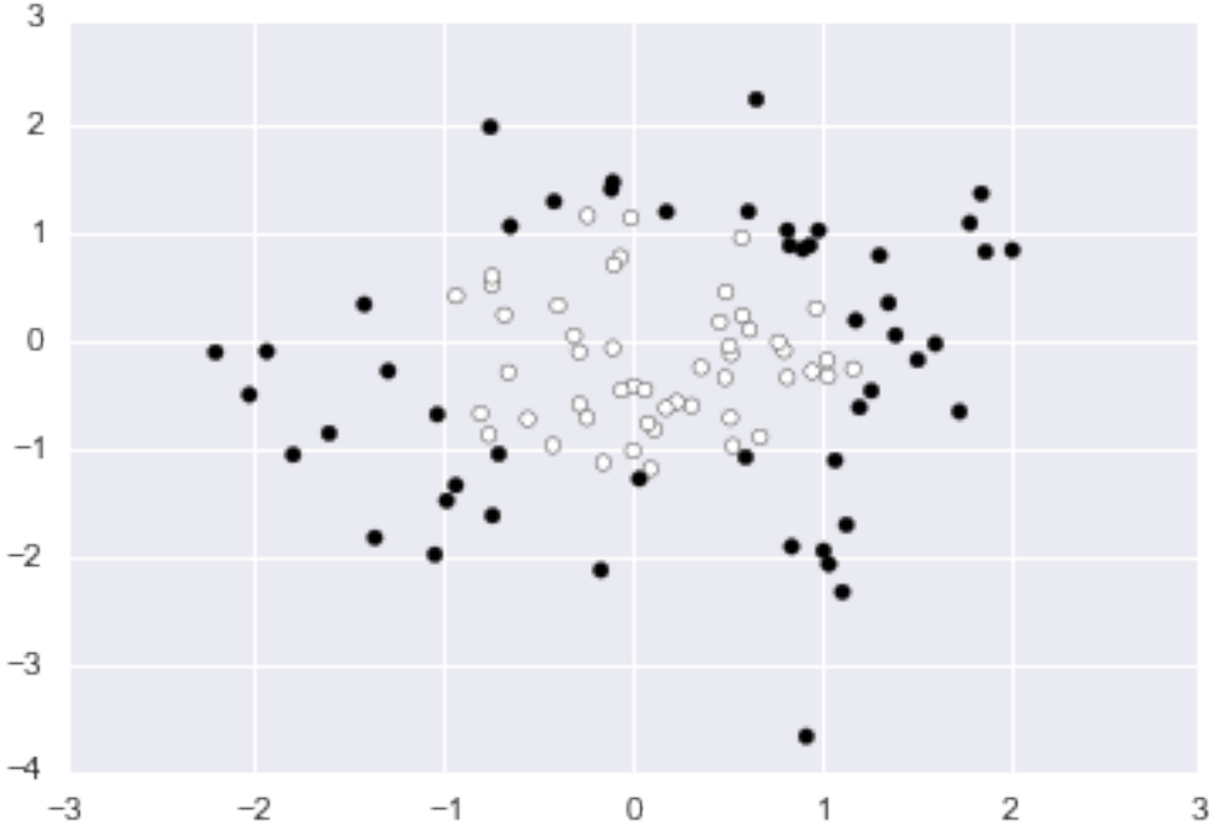
SUPPORT VECTOR MACHINES

We started out with the concept of a **maximal margin classifier.** Its sole purpose was to generate a hyperplane that maximized the margins (distance to the closest point). The hyperplane was defined by **support vectors** that lied on the margins.

Then, we introduced **slack variables** that allowed a certain amount of points to be on the wrong side of the hyperplane. The new margin was defined by **support vectors** which were points lying on the margin as well as points on the wrong side of the margin

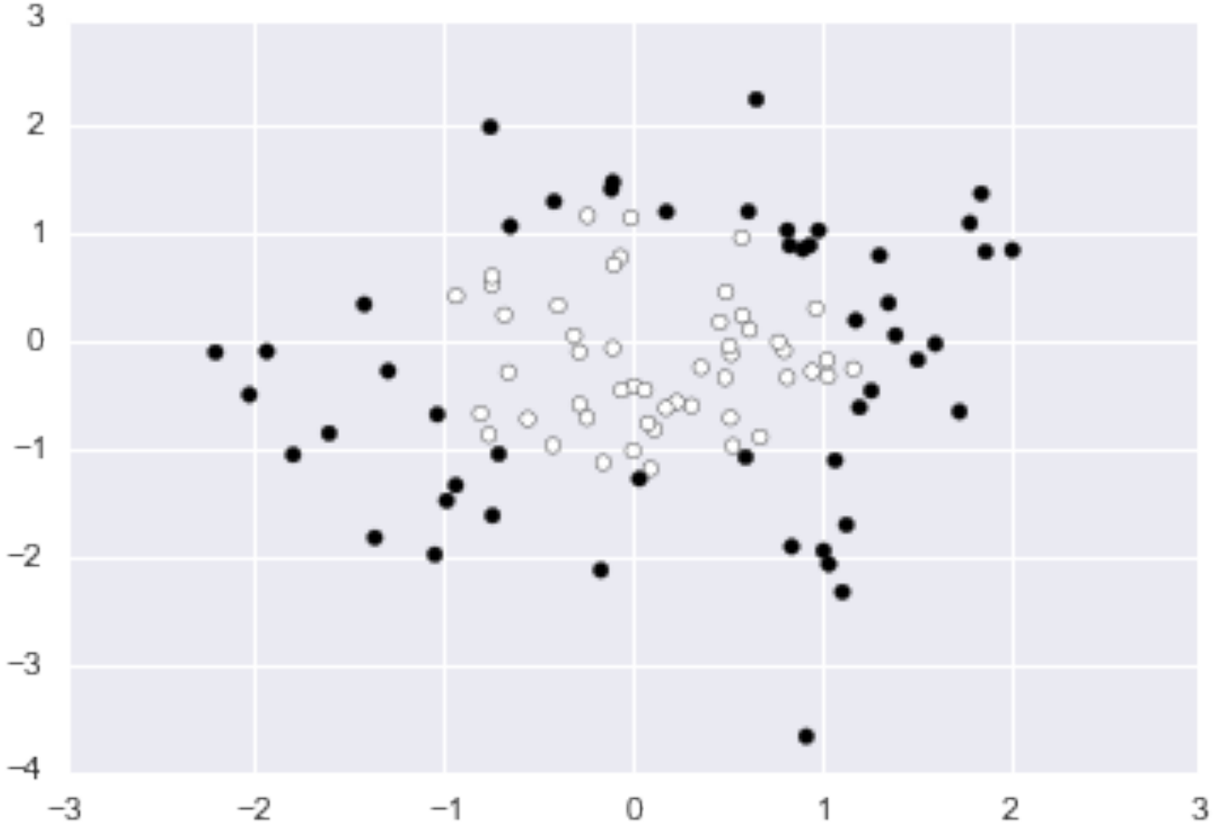
We're ready to introduce our final transition into **support vector machines**

KERNELS



What if your classes aren't linearly separable (even with slack)

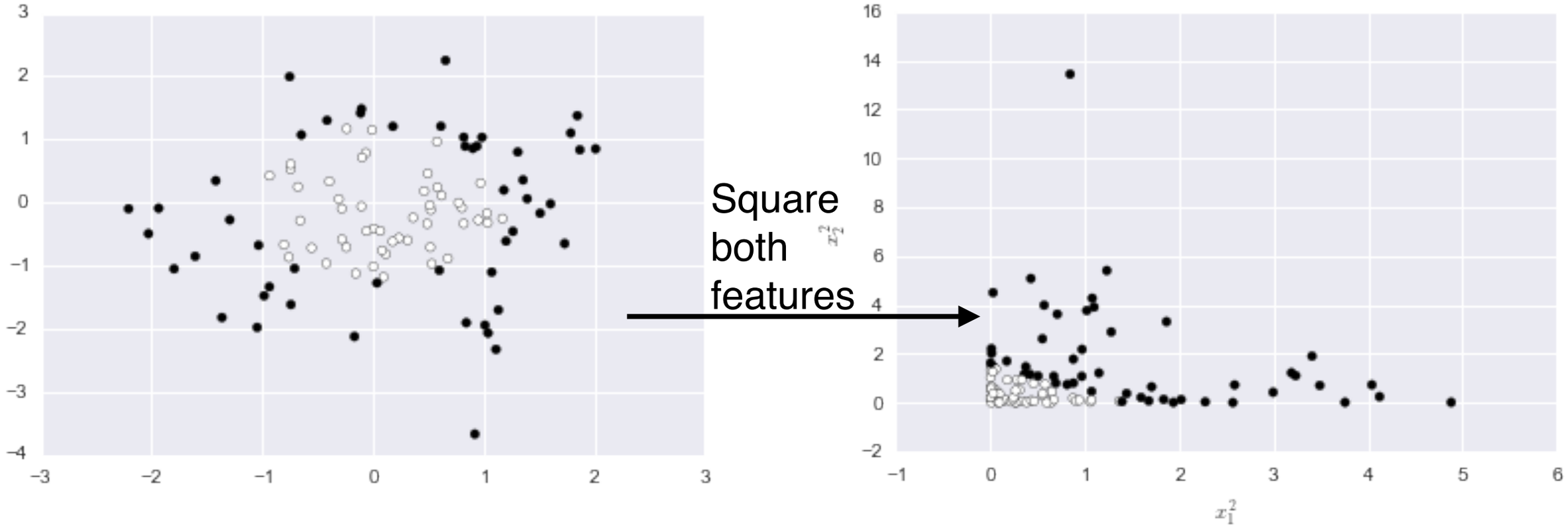
KERNELS



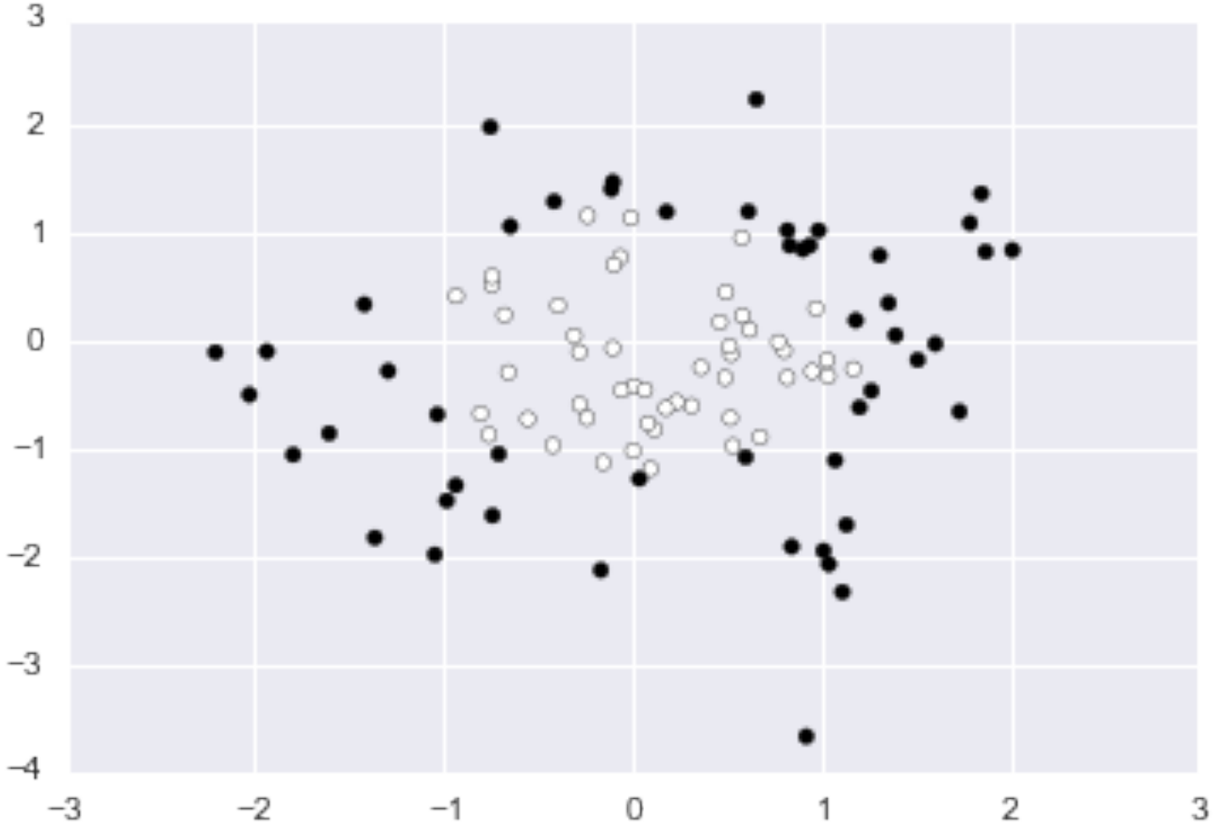
What if your classes aren't linearly separable (even with slack)

We could try transforming the input features by adding polynomial terms

KERNELS



KERNELS

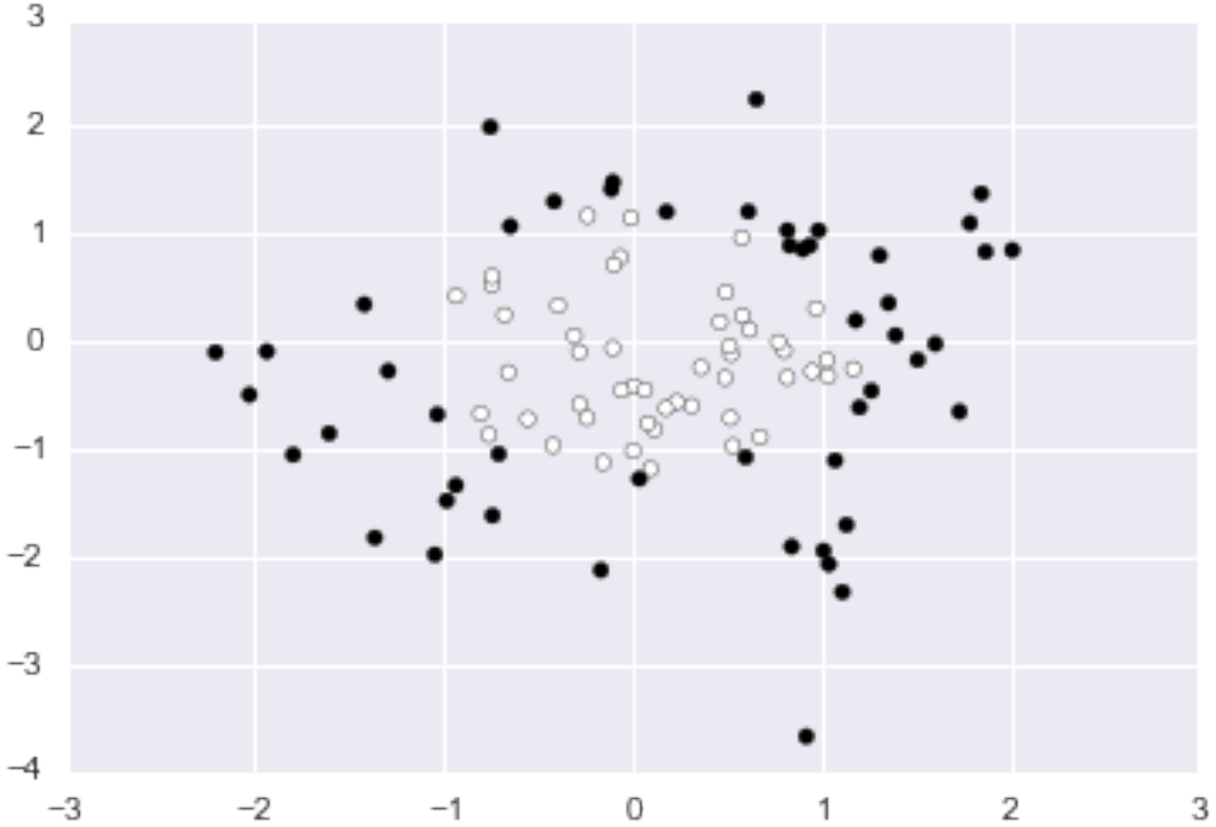


What if your classes aren't linearly separable (even with slack)

We could try transforming the input features by adding polynomial terms

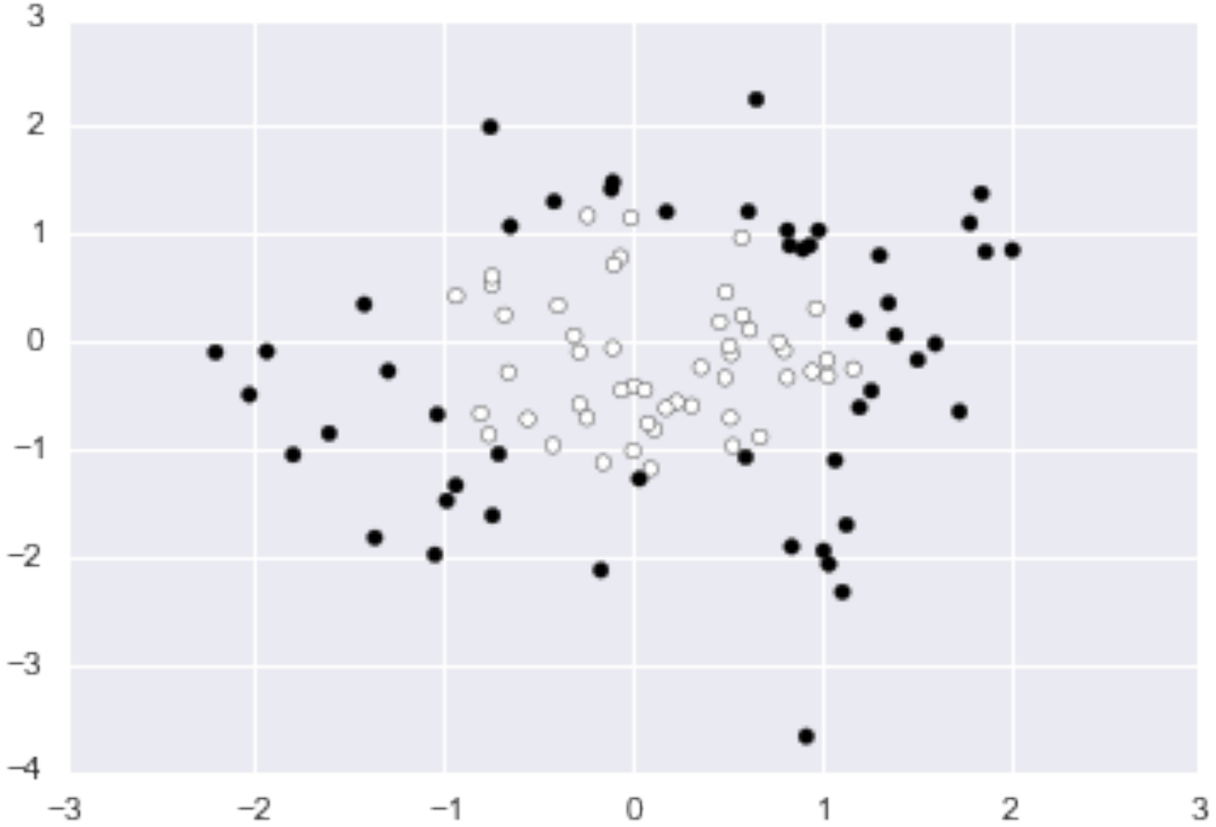
However as the data becomes more nonlinear, this becomes more of an exercise in adding x^2, x^3, x^4 and so on terms

KERNELS



We're going to introduce something cool called the "kernel trick" to create highly nonlinear features for us automatically

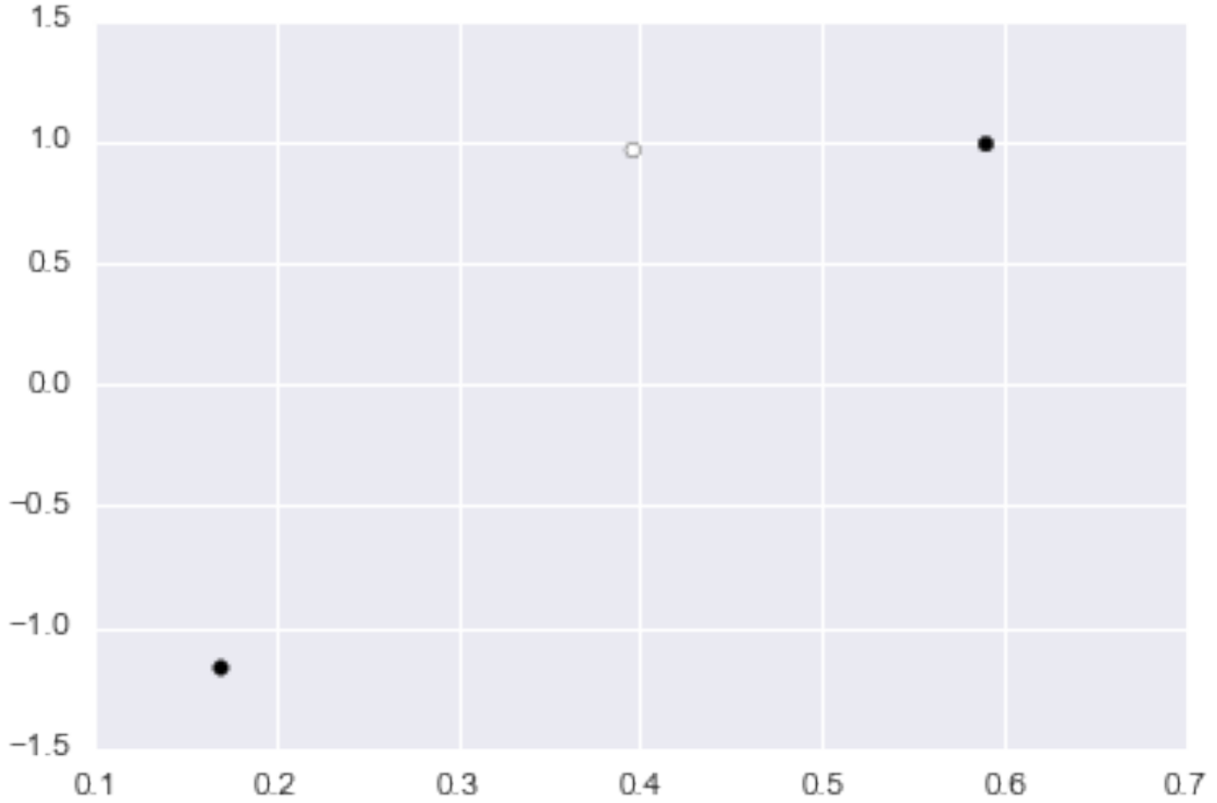
KERNELS



We're going to introduce something cool called the "kernel trick" to create highly nonlinear features for us automatically

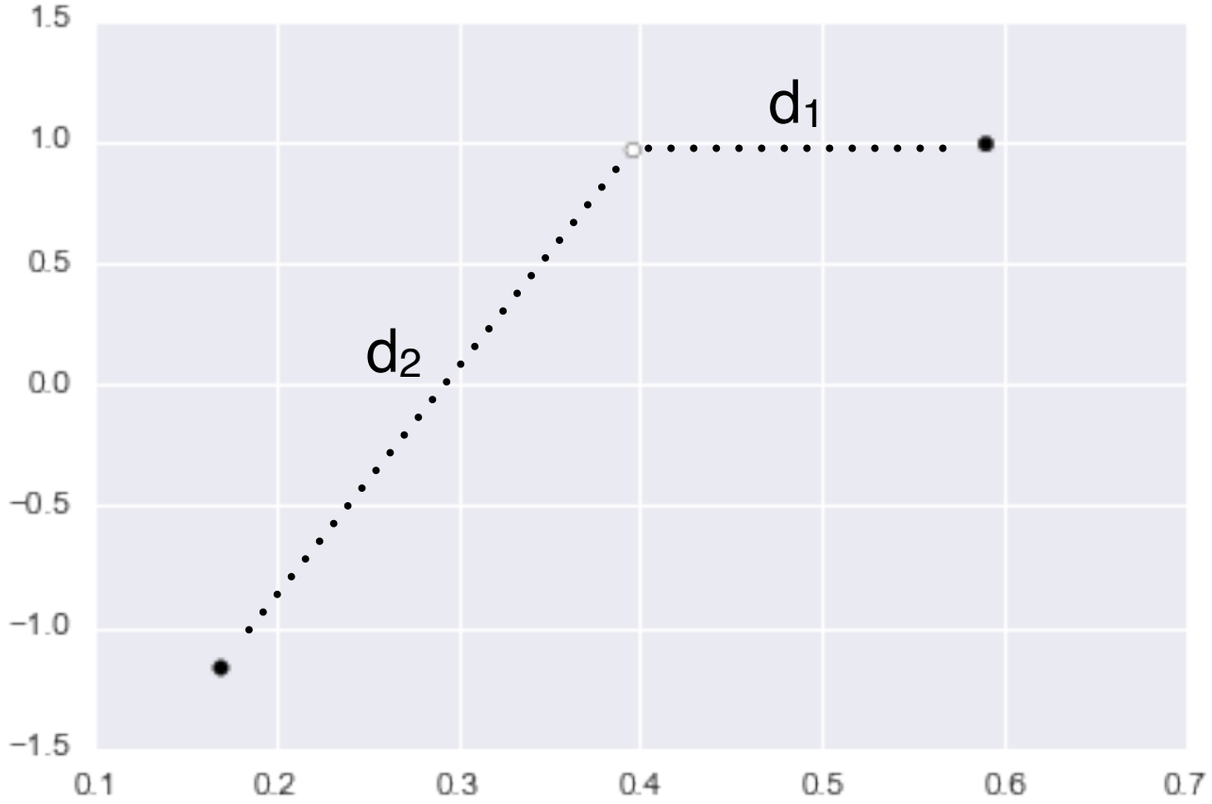
Maximal margin classifier + slack + kernel = **support vector machine**

KERNELS



First we add a **landmark** to the feature space (in white)

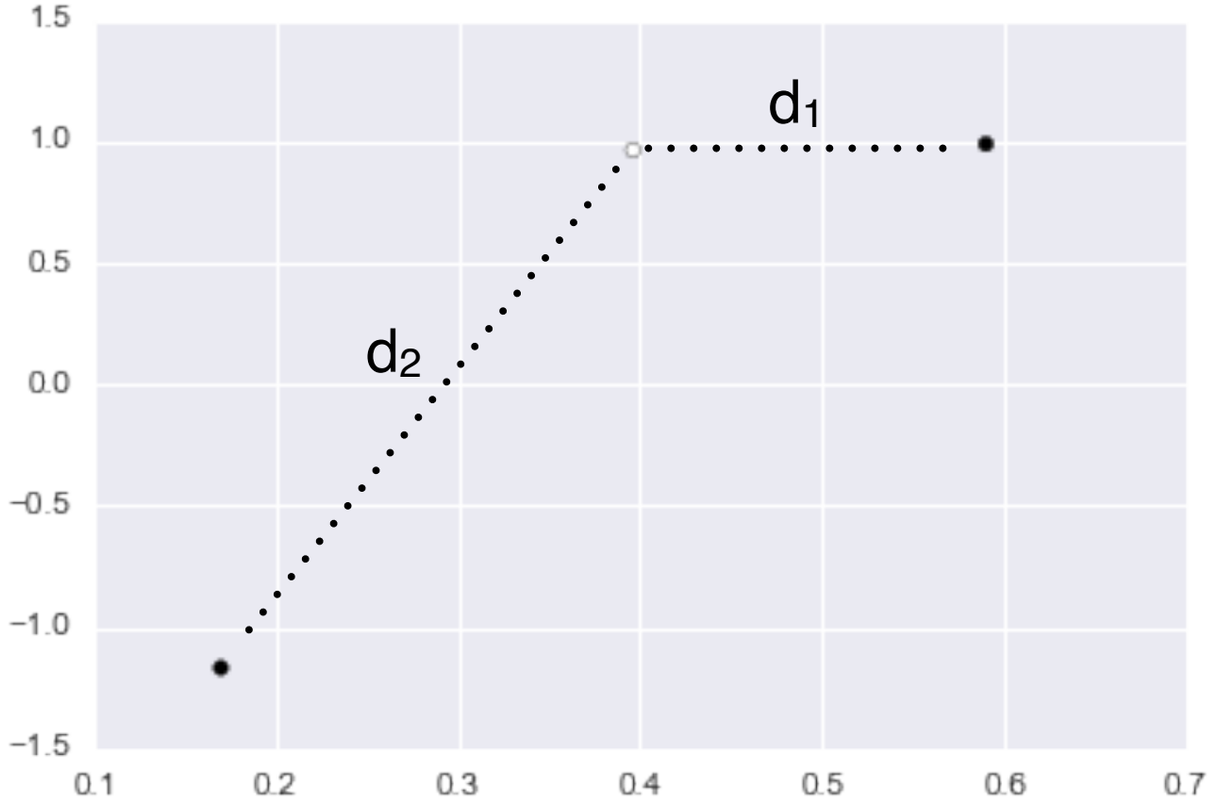
KERNELS



First we add a **landmark** to the feature space (in white)

For each point, compute the distance to this landmark:
||x - l|| **Feature scaling first!!**

KERNELS



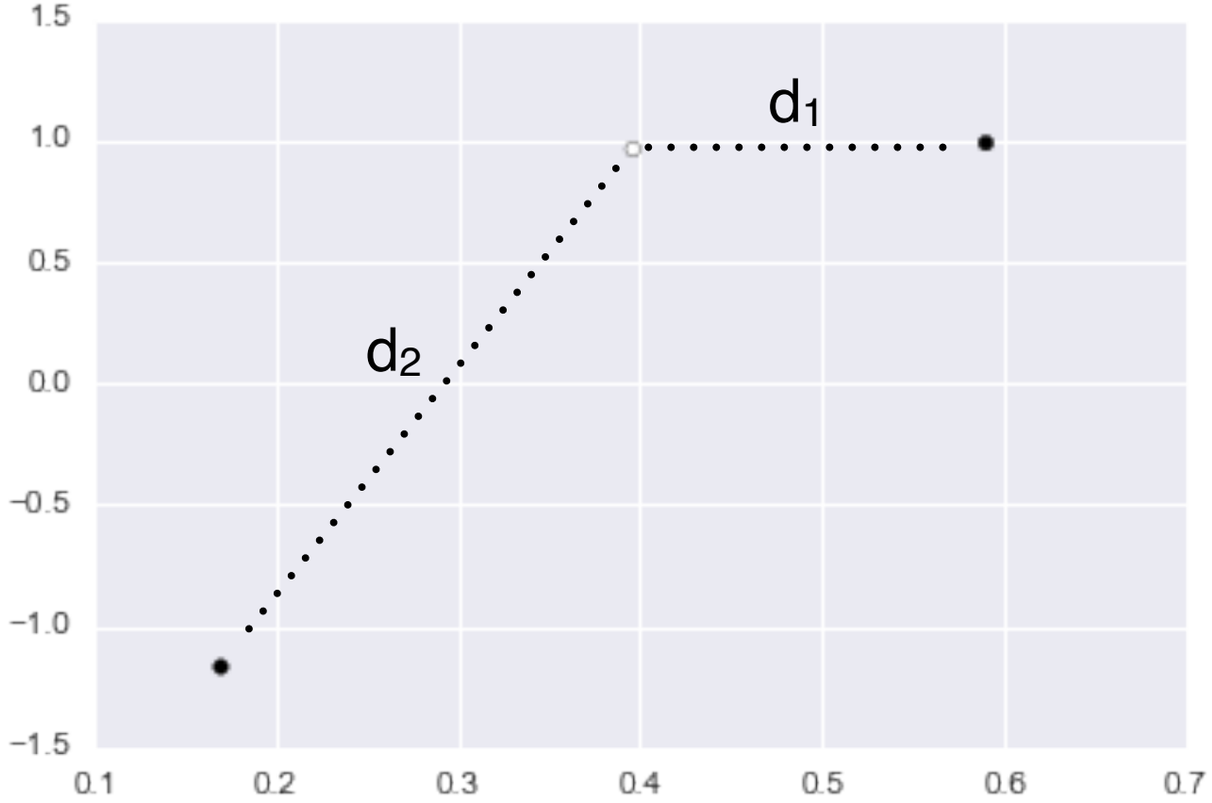
First we add a **landmark** to the feature space (in white)

For each point, compute the distance to this landmark: ||x - l|| **Feature scaling first!!**

Then define the similarity between a landmark and a point as the **radial basis function (rbf):**

$$K(x,l) = e^{-\frac{\|x-l\|^2}{2\sigma^2}}$$

KERNELS



First we add a **landmark** to the feature space (in white)

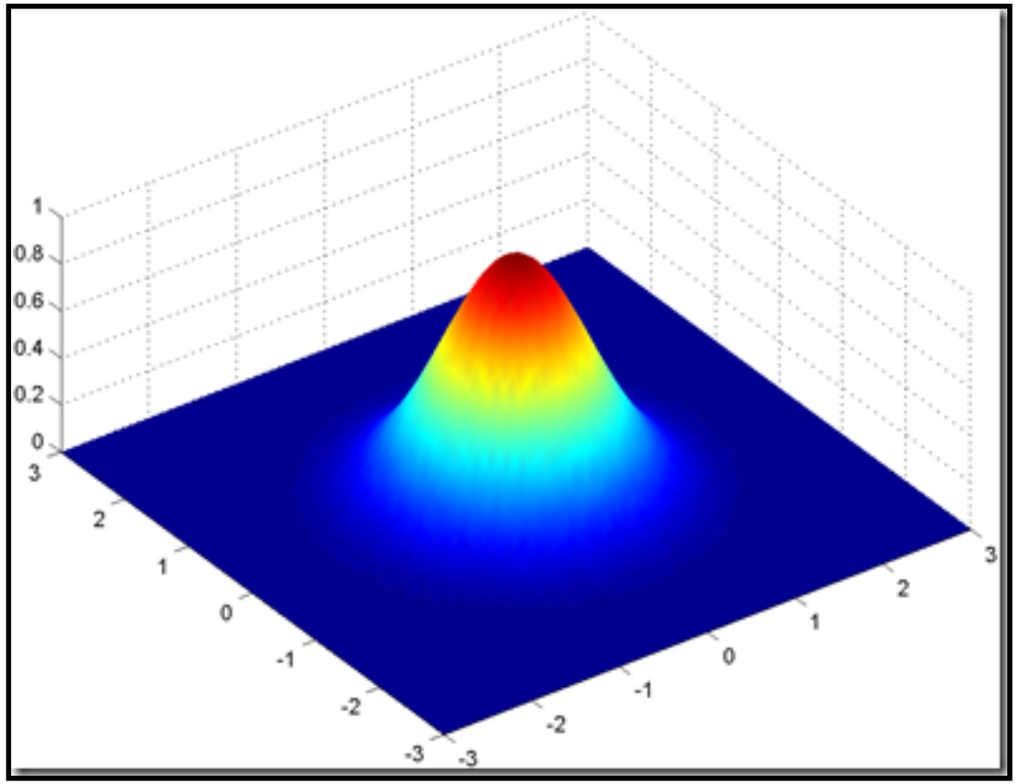
For each point, compute the distance to this landmark: ||x - l|| **Feature scaling first!!**

Then define the similarity between a landmark and a point as the **radial basis function (rbf):**

$$K(x,l) = e^{-\frac{\|x-l\|^2}{2\sigma^2}}$$

Let's see on the board how this function behaves when points are close and far from the landmark

KERNELS



First we add a **landmark** to the feature space (in white)

For each point, compute the distance to this landmark: ||x - l|| ****Feature scaling first!!****

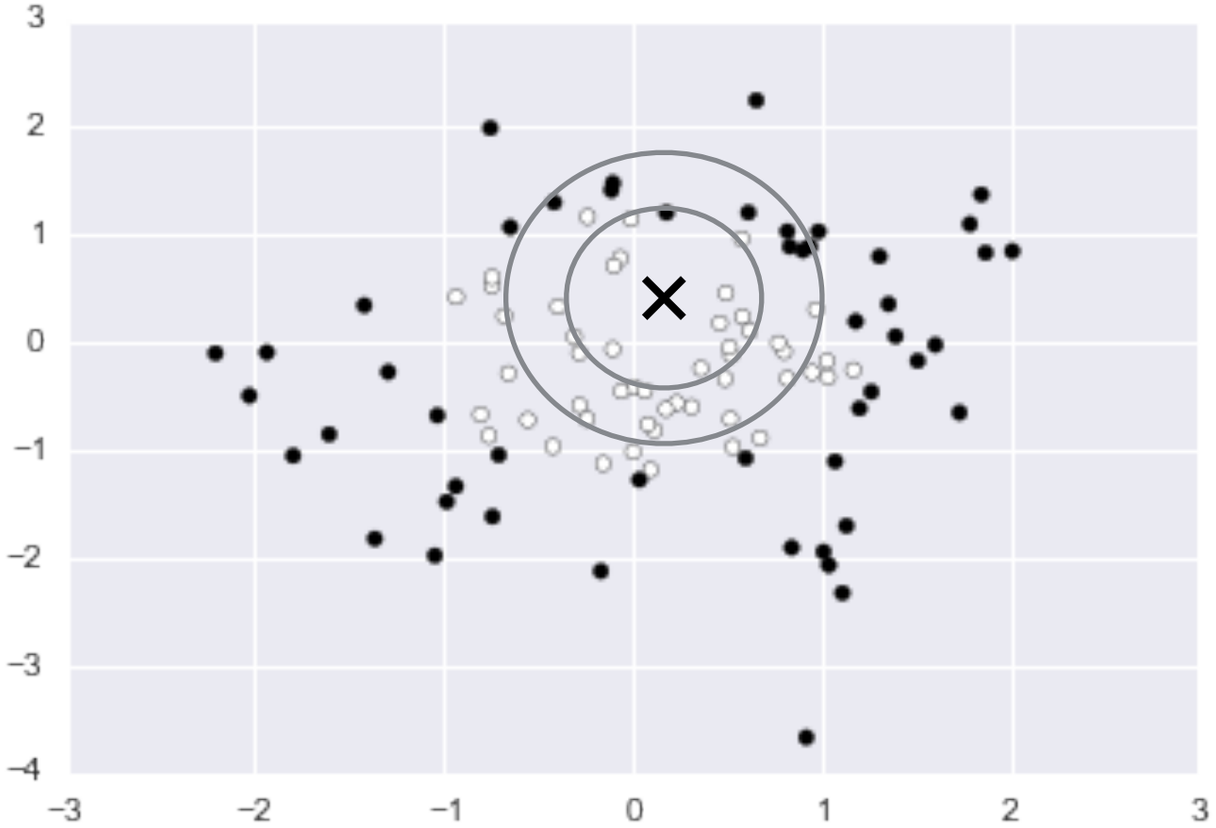
Then define the similarity between a landmark and a point as the **radial basis function (rbf):**

$$K(x,l) = e^{-\frac{\|x-l\|^2}{2\sigma^2}}$$

Let's see on the board how this function behaves when points are close and far from the landmark

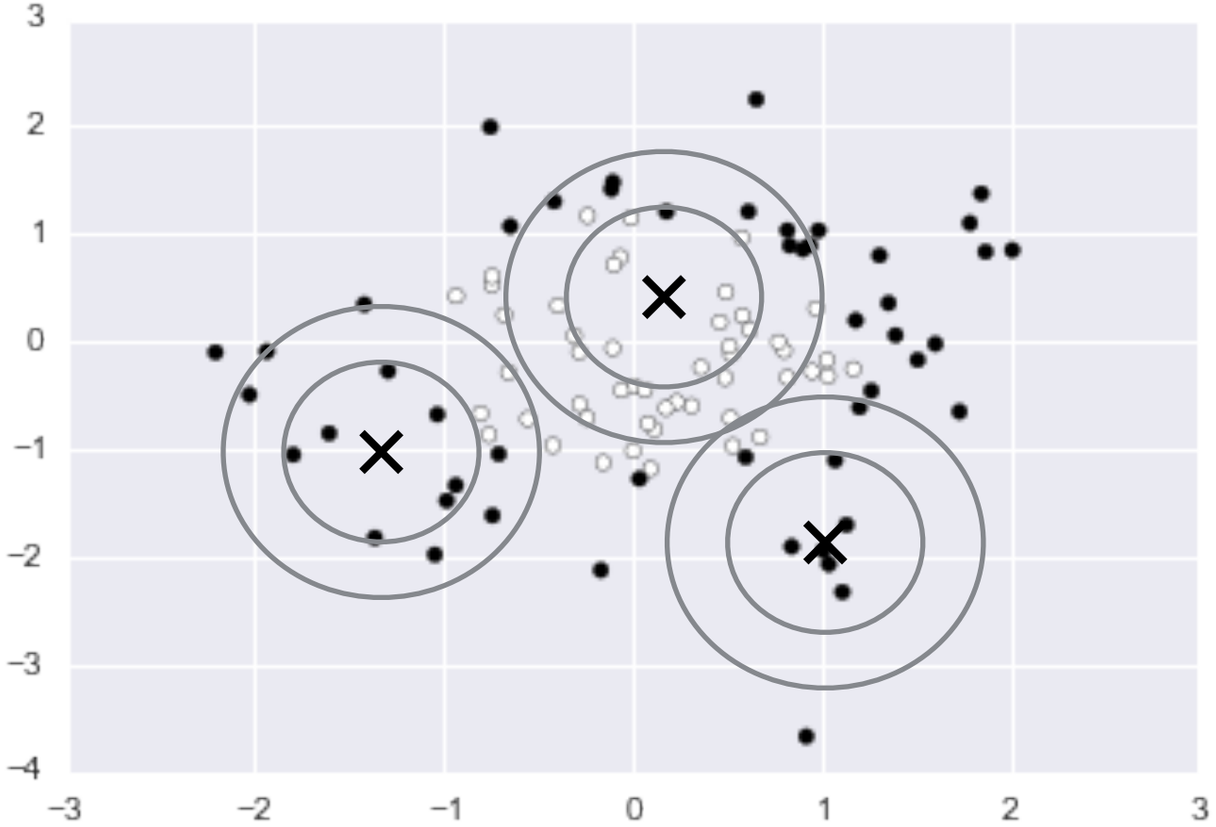
This is called a **Gaussian kernel.** The value of points from landmarks fall away according to sigma squared

KERNELS



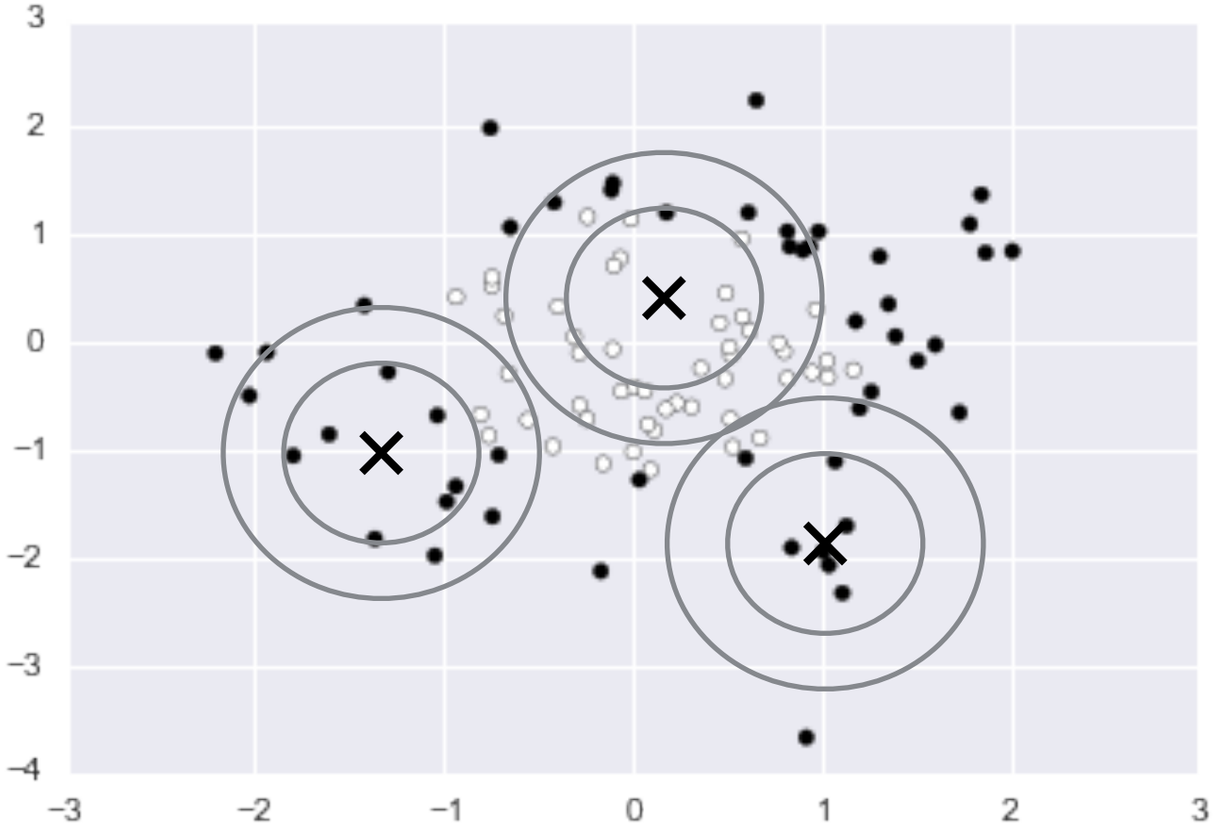
So how do we choose the right landmark in the original point space?

KERNELS



So how do we choose the right landmark in the original point space?

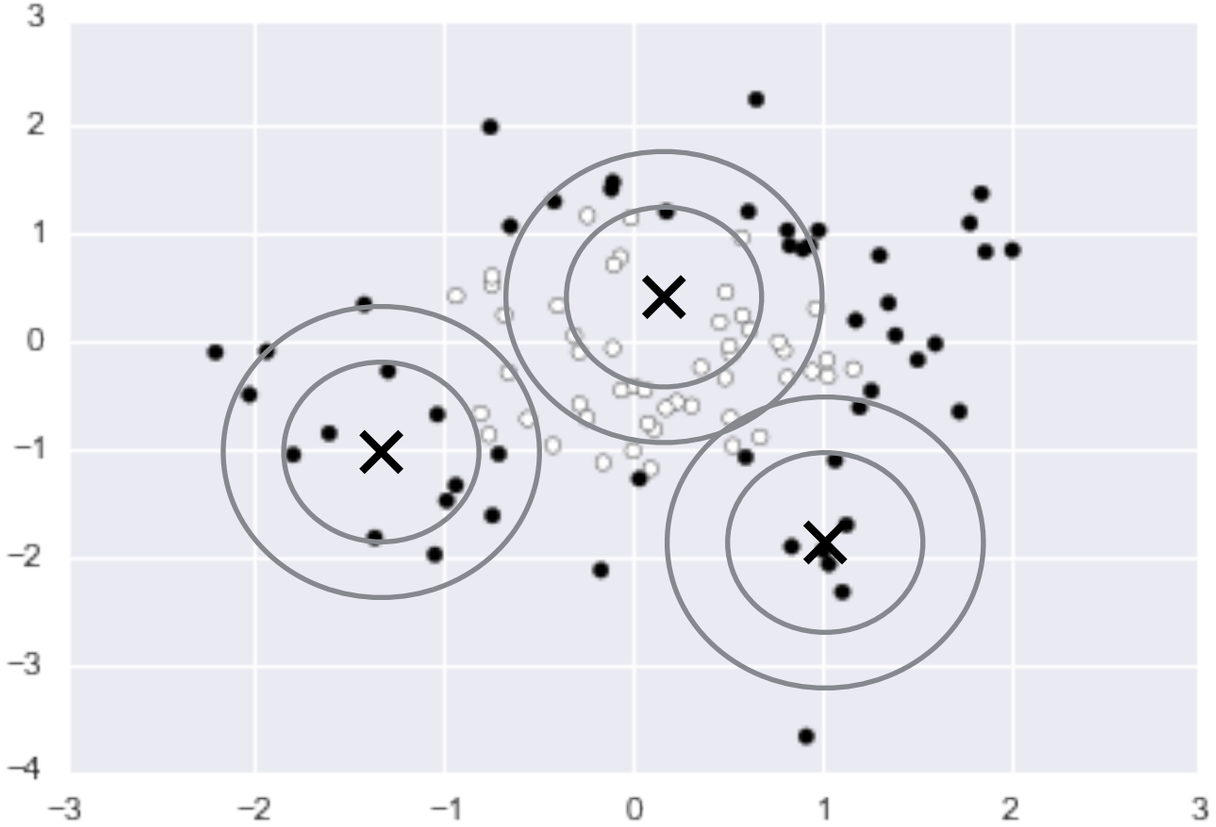
KERNELS



So how do we choose the right landmark in the original point space?

Choose every sample as a landmark

KERNELS

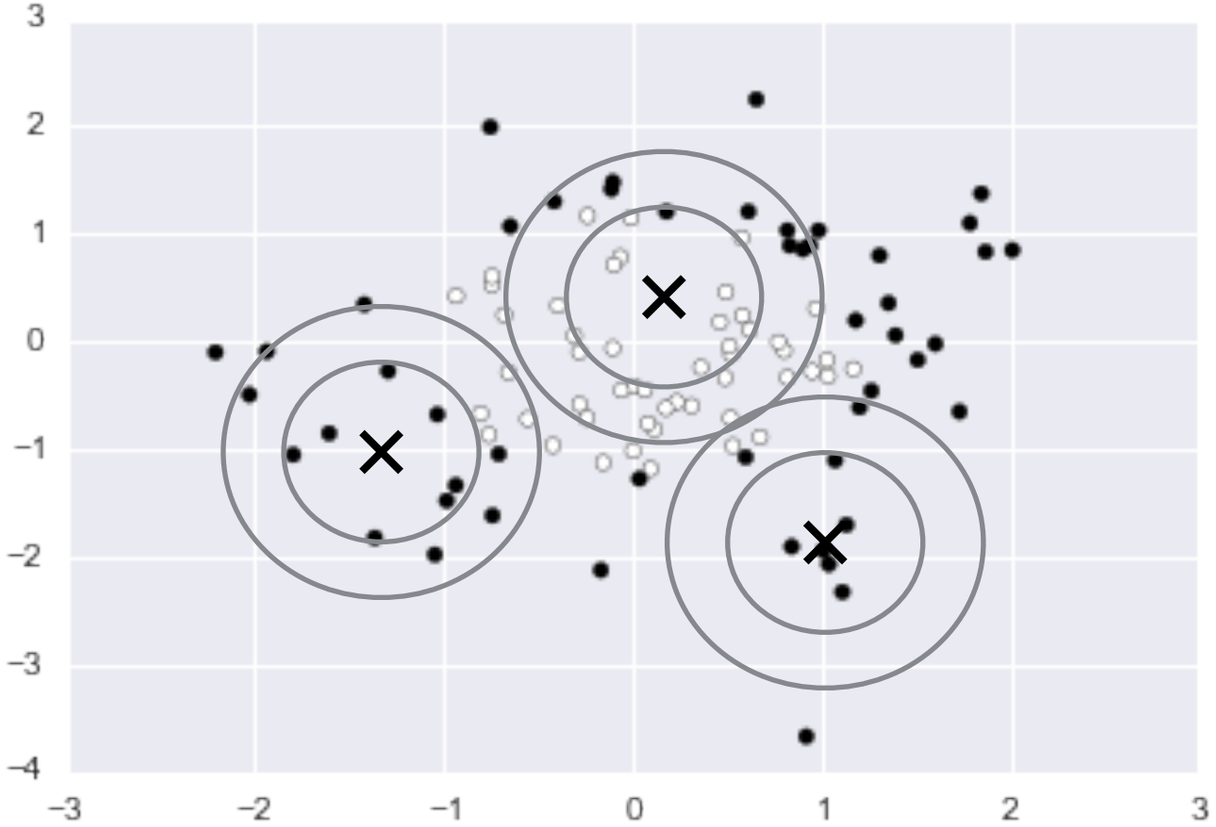


So how do we choose the right landmark in the original point space?

Choose every sample as a landmark

Then replace all samples with the kernel values

KERNELS



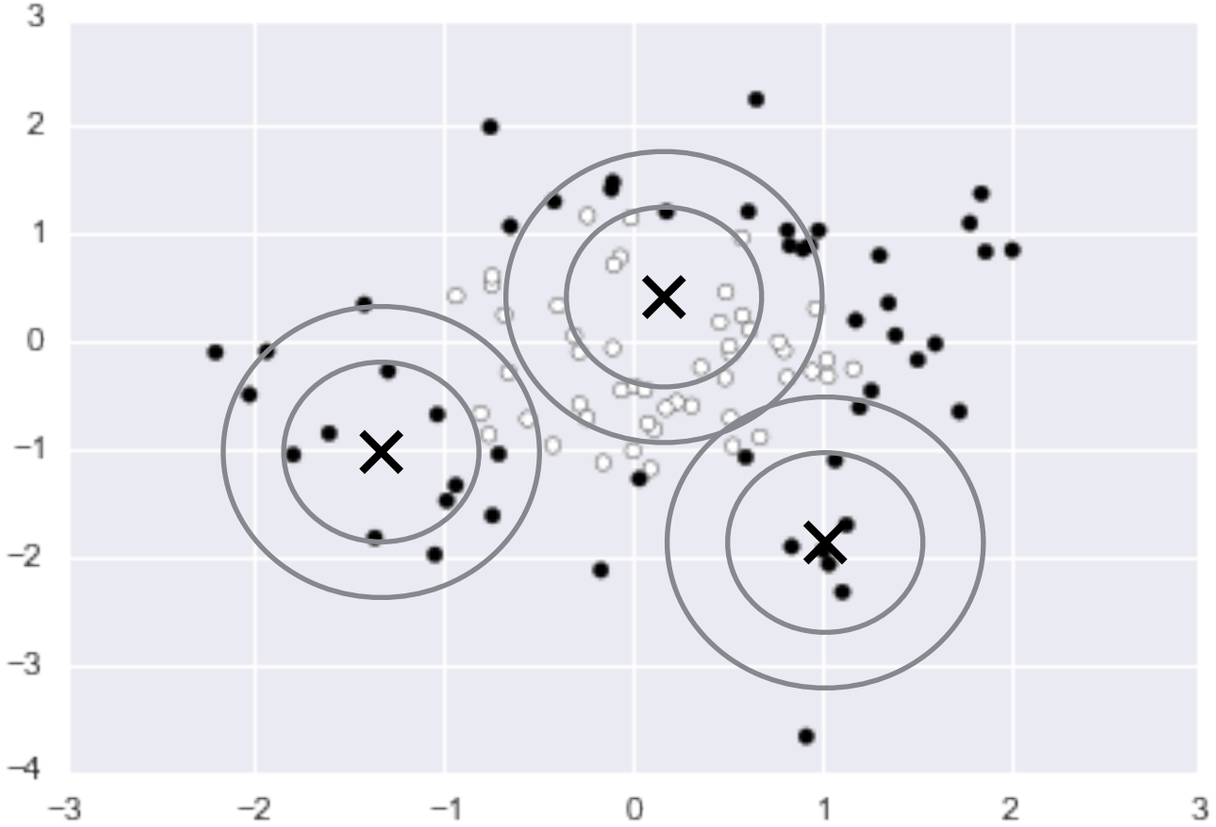
So how do we choose the right landmark in the original point space?

Choose every sample as a landmark

Then replace all samples with the kernel values

This will (nonlinearly) transform your matrix **X** from an N x n matrix into an N x N matrix.

KERNELS



So how do we choose the right landmark in the original point space?

Choose every sample as a landmark

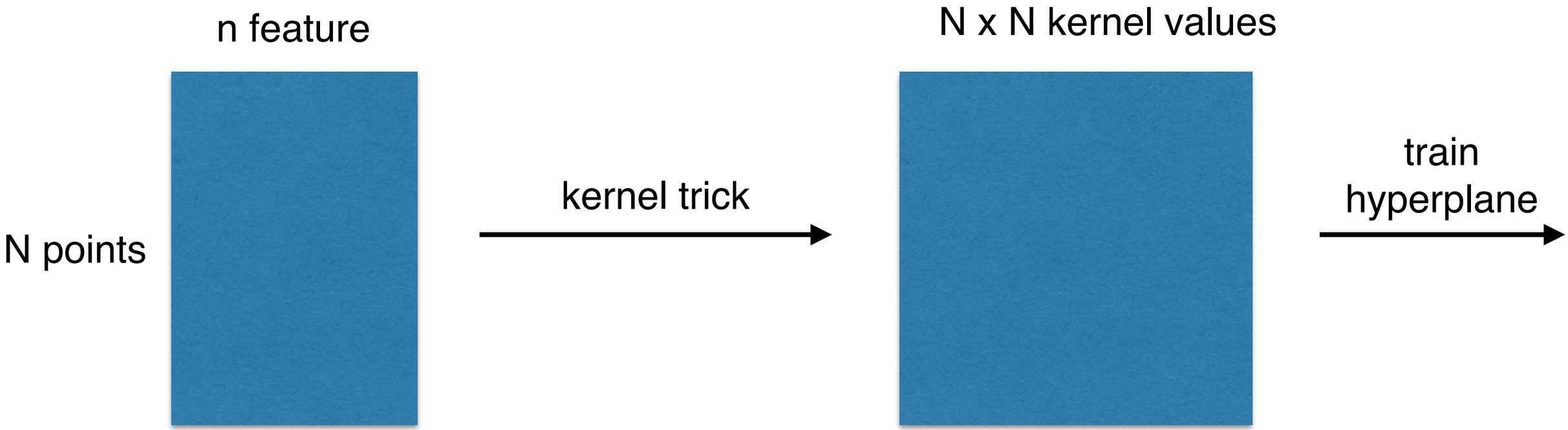
Then replace all samples with the kernel values

This will (nonlinearly) transform your matrix **X** from an N x n matrix into an N x N matrix

The diagonal values are one (i.e. the point is zero distance from the landmark, which is itself), while the off diagonals are the kernel values to that landmark

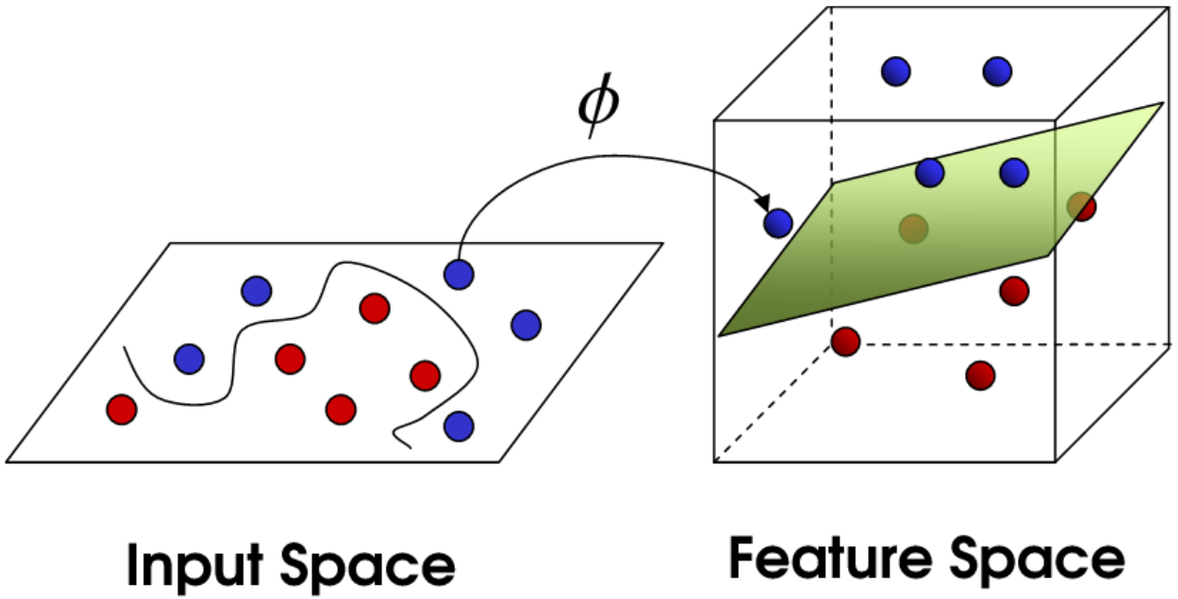
KERNELS

We can use a kernel to implicitly train our model in a high-dimensional feature space, without incurring additional computational complexity



KERNELS

We can use a kernel to implicitly train our model in a high-dimensional feature space, without incurring additional computational complexity



http://stackoverflow.com/questions/9480605/what-is-the-relation-between-the-number-of-support-vectors-and-training-data-and

KERNELS

We can use a kernel to implicitly train our model in a high-dimensional feature space, **without** incurring additional computational complexity

As long as the kernel function satisfies certain conditions, we can perform the same methods for the maximum margin hyperplane

KERNELS

We can use a kernel to implicitly train our model in a high-dimensional feature space, **without** incurring additional computational complexity

As long as the kernel function satisfies certain conditions, we can perform the same methods for the maximum margin hyperplane

Nonlinear classification is then obtained by creating a linear decision boundary in the higher dimensional space

KERNELS

Kernels can also be used as a preprocessing step in other models as well (such as logistic regression)

However, they tend to be very computationally expensive. The SVM solver naturally lends itself to be paired with kernel methods

KERNELS

Some popular kernels:

linear kernel $k(x,x') = \langle x, x' \rangle$

polynomial kernel $k(x,x') = (x^T x' + 1)^d$

Gaussian kernel $k(x,x') = \exp(-\gamma ||x - x'||^2)$

The hyperparameters d and γ affect the flexibility of the decision boundary

KERNELS

Some popular kernels:

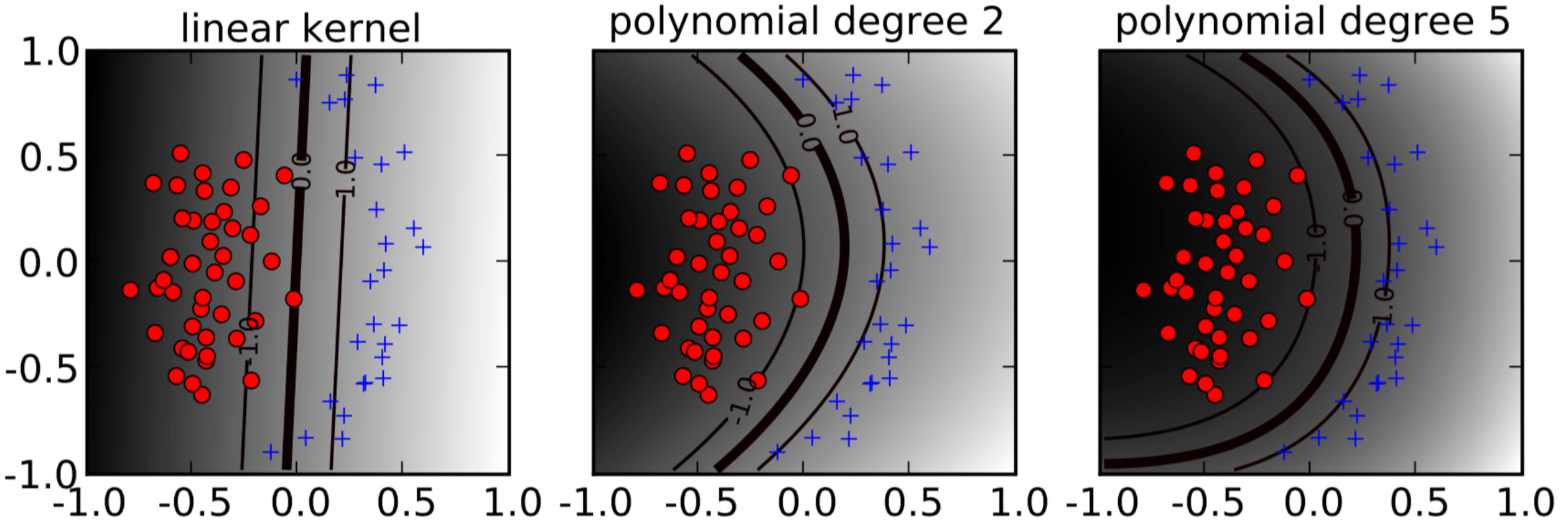
linear kernel $k(x,x') = \langle x, x' \rangle$

polynomial kernel $k(x,x') = (x^T x' + 1)^d$

Gaussian kernel $k(x,x') = \exp(-\gamma ||x - x'||^2)$

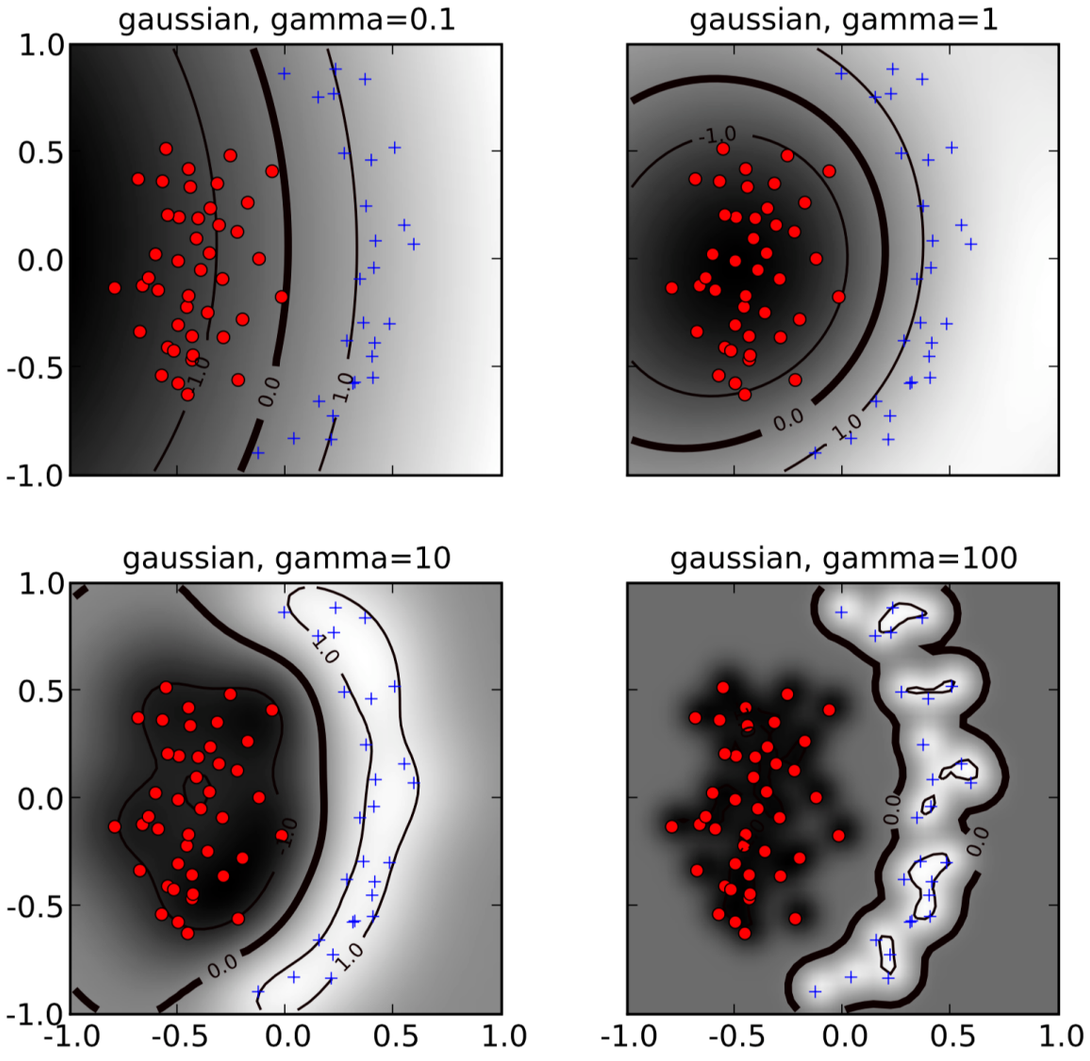
The hyperparameters d and γ affect the flexibility of the decision boundary (which can be chosen using cross validation)

KERNELS



User's Guide to SVM http://pyml.sourceforge.net/doc/howto.pdf

KERNELS



User's Guide to SVM http://pyml.sourceforge.net/doc/howto.pdf

SUPPORT VECTOR MACHINES

We're at the end of our theoretical journey. Starting with **maximal margin classifiers**, we built hyperplanes that perfectly separate the two classes geometrically.

Then, we allowed for a certain amount of error through **slack variables**, and gave us much more regularized decision boundaries. We called these **support vector classifiers.**

Finally, we used **kernel methods** to transform point space into a high dimensional feature space, and created the hyperplanes in that space. Thus, we end with **support vector machines**

SUPPORT VECTOR MACHINES

SVMs (and kernel methods in general) are versatile, powerful, and popular techniques that can produce very accurate results for a wide array of classification problems.

SUPPORT VECTOR MACHINES

SVMs (and kernel methods in general) are versatile, powerful, and popular techniques that can produce very accurate results for a wide array of classification problems.

They allow for highly nonlinear boundaries which we may not intuitively see

SUPPORT VECTOR MACHINES

SVMs (and kernel methods in general) are versatile, powerful, and popular techniques that can produce very accurate results for a wide array of classification problems.

They allow for highly nonlinear boundaries which we may not intuitively see

This is their main drawback. There is a lack of intuition, and they can be seen as **black boxes**.

SUPPORT VECTOR MACHINES

SVMs (and kernel methods in general) are versatile, powerful, and popular techniques that can produce very accurate results for a wide array of classification problems.

They allow for highly nonlinear boundaries which we may not intuitively see

This is their main drawback. There is a lack of intuition, and they can be seen as **black boxes**.

Also, unlike logistic regression or naive bayes, SVMs do not provide probability estimates

SUPPORT VECTOR MACHINES

SVMs (and kernel methods in general) are versatile, powerful, and popular techniques that can produce very accurate results for a wide array of classification problems.

They allow for highly nonlinear boundaries which we may not intuitively see

This is their main drawback. There is a lack of intuition, and they can be seen as **black boxes**.

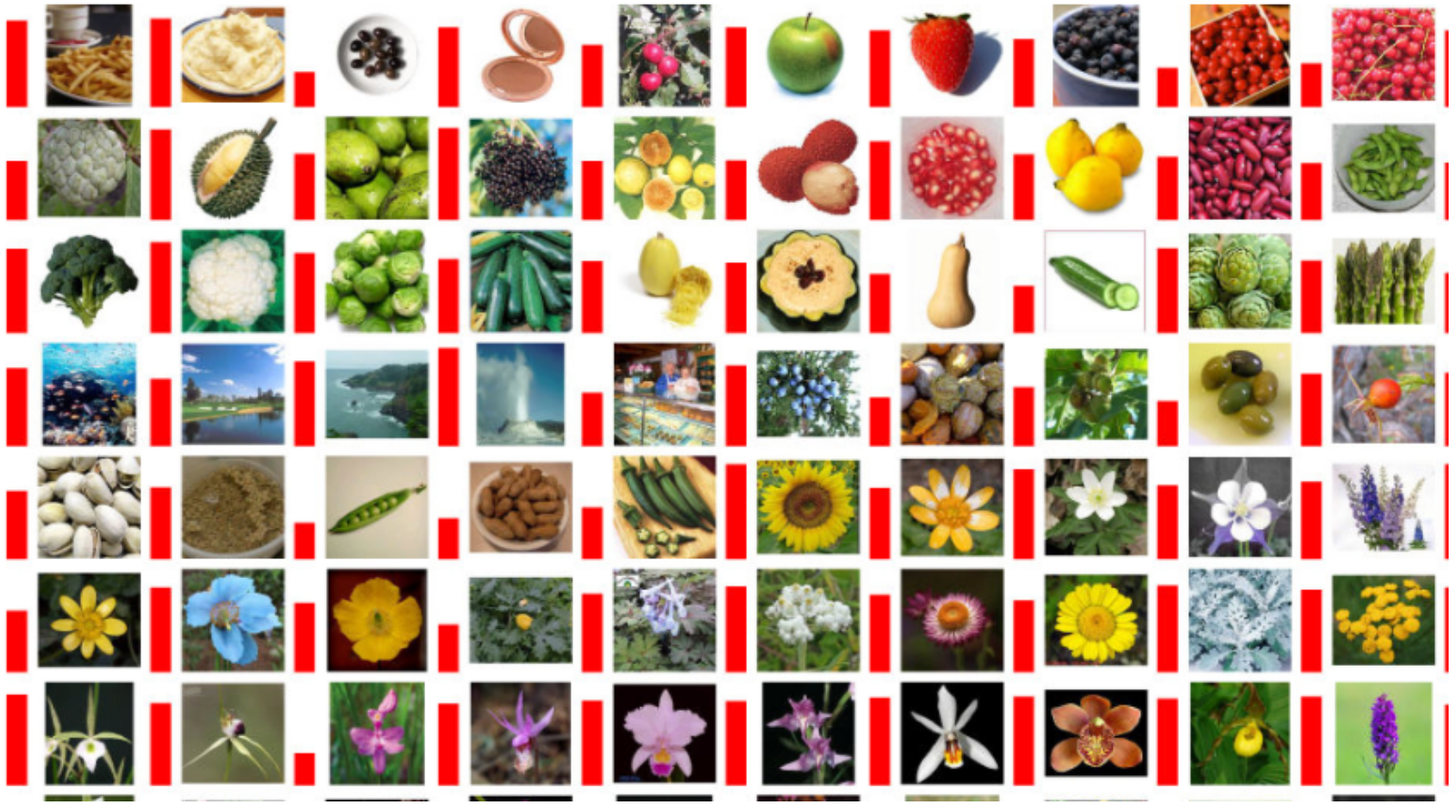
Also, unlike logistic regression or naive bayes, SVMs do not provide probability estimates

Lastly, because we only need to store the support vectors, memory and computationally, they are very efficient, up to a certain extent

SUPPORT VECTOR MACHINES

They’re used in:

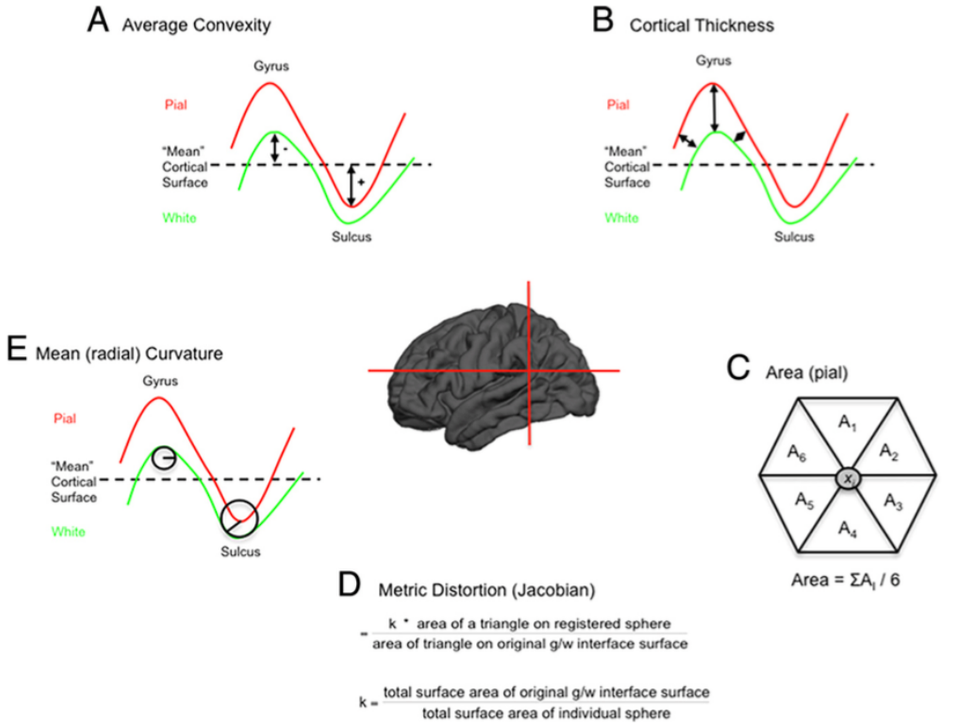
- Large Scale (> 48 gb) Image Classification - http://www.dbs.ifi.lmu.de/~yu_k/cvpr11_0694.pdf



SUPPORT VECTOR MACHINES

They’re used in:

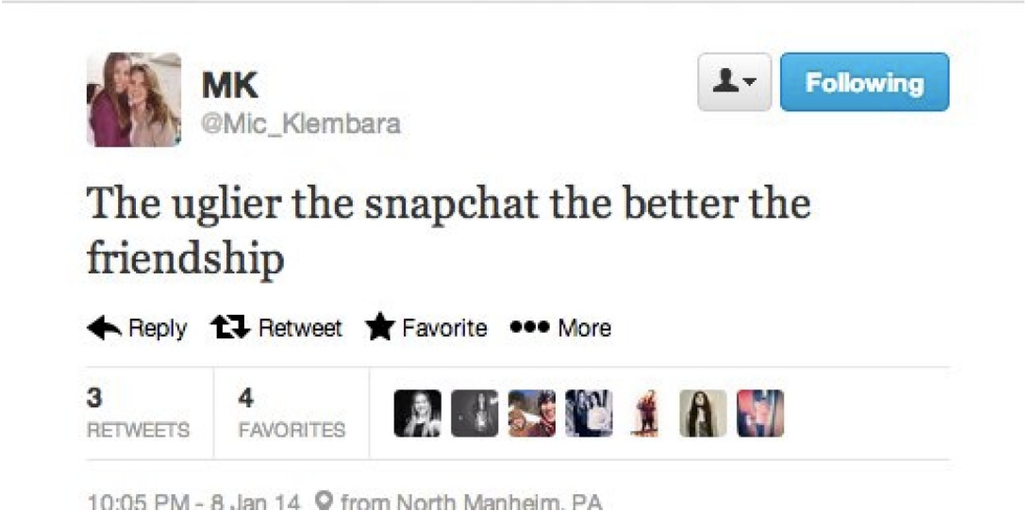
- Large Scale (> 48 gb) Image Classification - http://www.dbs.ifi.lmu.de/~yu_k/cvpr11_0694.pdf
- Describing autism (spatial boundaries!) - http://www.jneurosci.org/content/30/32/10612.full.pdf+html



SUPPORT VECTOR MACHINES

They’re used in:

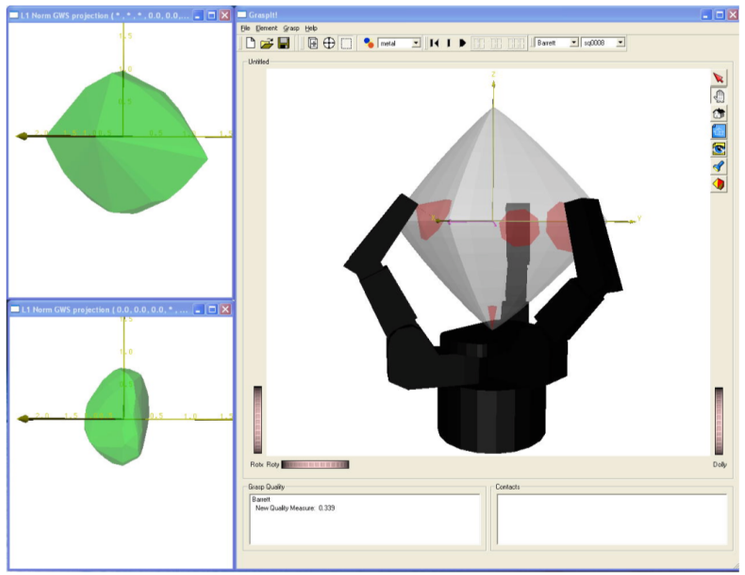
- Large Scale (> 48 gb) Image Classification - http://www.dbs.ifi.lmu.de/~yu_k/cvpr11_0694.pdf
- Describing autism (spatial boundaries!) - http://www.jneurosci.org/content/30/32/10612.full.pdf+html
- Twitter sentiment analysis (positive/negative)



SUPPORT VECTOR MACHINES

They’re used in:

- Large Scale (> 48 gb) Image Classification - http://www.dbs.ifi.lmu.de/~yu_k/cvpr11_0694.pdf
- Describing autism (spatial boundaries!) - http://www.jneurosci.org/content/30/32/10612.full.pdf+html
- Twitter sentiment analysis (positive/negative)
- Robotics http://www.cs.columbia.edu/~allen/PAPERS/ra04submit.pdf



SUPPORT VECTOR MACHINES

They’re used in:

- Large Scale (> 48 gb) Image Classification - http://www.dbs.ifi.lmu.de/~yu_k/cvpr11_0694.pdf
- Describing autism (spatial boundaries!) - http://www.jneurosci.org/content/30/32/10612.full.pdf+html
- Twitter sentiment analysis (positive/negative)
- Robotics http://www.cs.columbia.edu/~allen/PAPERS/ra04submit.pdf
- Any classification scheme where interpretability is not a #1 priority

IV.SVM LAB

THAT'S IT!

- Exit Tickets: DAT1 - Lesson 11 - SVM
- HW 6 cancelled. Free 2/2 for everyone. Is everyone's project on track?
- In lieu of homework, for extra reading:
- SVM Reading: http://pyml.sourceforge.net/doc/howto.pdf
- Introduction To Statistical Learning - SVM chapter
- Learning in video form: https://www.youtube.com/watch?v=HwQQXs4Nyjo