INTRO TO DATA SCIENCE

LECTURE 13: SUPPORT VECTOR MACHINES

I. DECISION TREES
II. FITTING DECISION TREES
III. OBJECTIVE FUNCTIONS
IV. REGULARIZATION
V. ENSEMBLE METHODS

BAGGING BOOSTING RANDOM FORESTS

Questions?

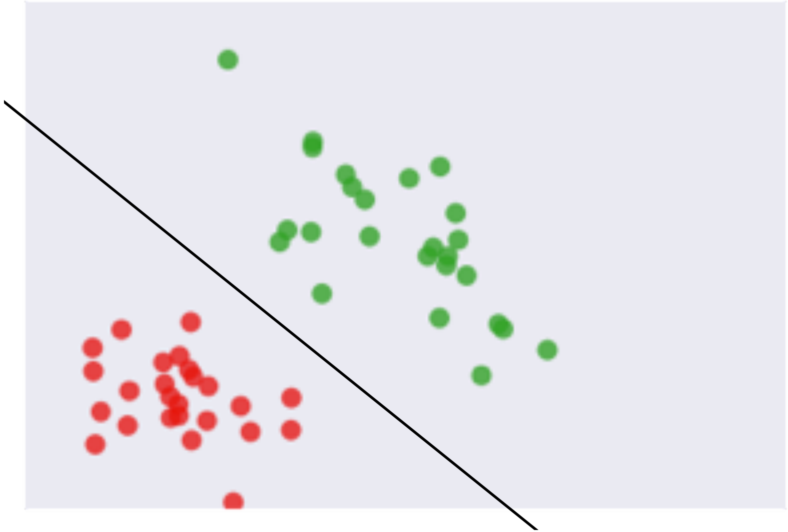
- DATA EXPLORATION
- SUPERVISED LEARNING: REGRESSION
- **SUPERVISED LEARNING: CLASSIFICATION**
  - LOGISTIC REGRESSION
  - NAIVE BAYES
  - RANDOM FORESTS
  - **SUPPORT VECTOR MACHINES**
  - COMPETITION
- UNSUPERVISED LEARNING
- VARIOUS TOPICS

Final outlines for your project are due next lesson

I. SUPPORT VECTOR MACHINES
II. REGULARIZATION
III. KERNELS

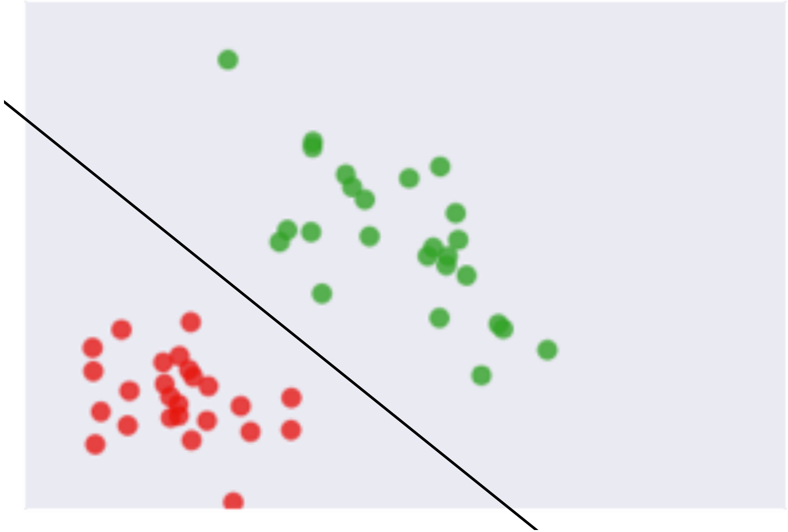
I. SUPPORT VECTORS

Suppose we have data with binary labels, which we would like to classify



Suppose we have data with binary labels, which we would like to classify

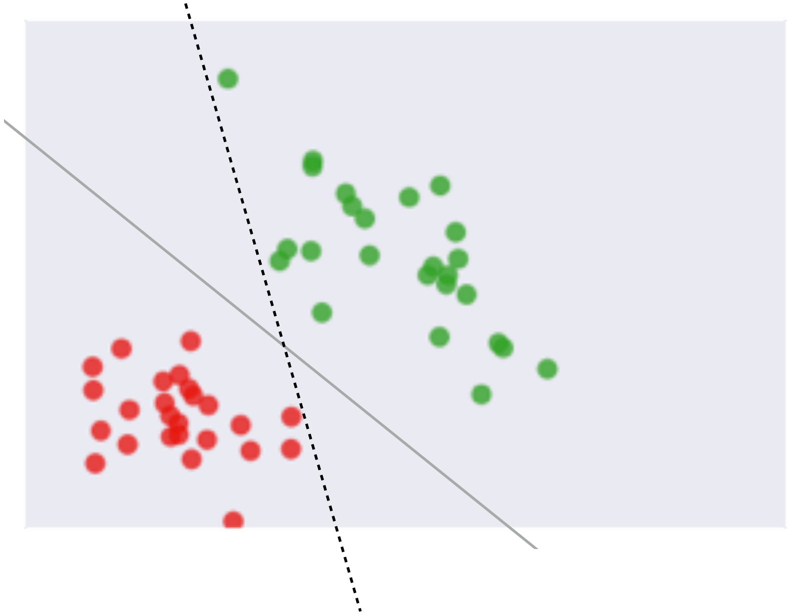
Recall that after fitting a classifier, we can draw the **decision boundary** *which separates the two classes*



Suppose we have data with binary labels, which we would like to classify

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

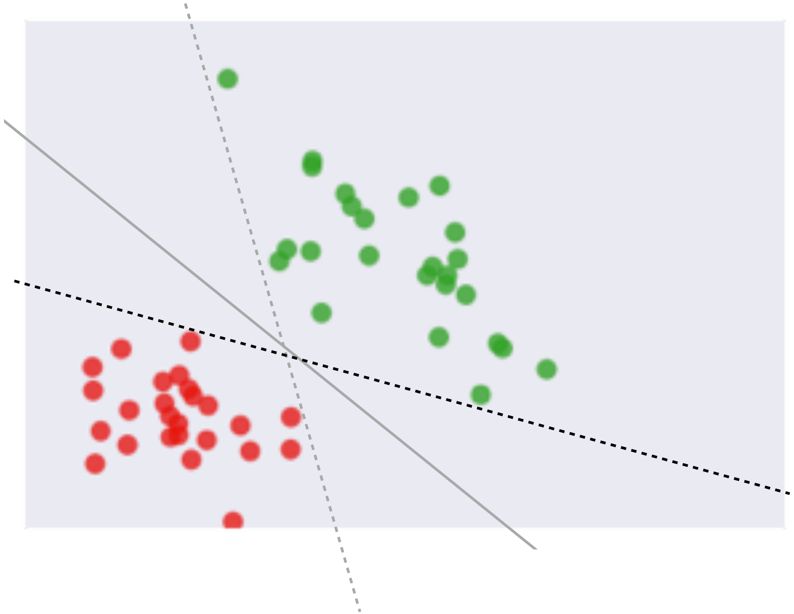
These boundaries depend on the specific optimization method. For example, logistic regression gives a different one than naive Bayes



Suppose we have data with binary labels, which we would like to classify

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

These boundaries depend on the specific optimization method. For example, logistic regression gives a different one than naive Bayes



Suppose we have data with binary labels, which we would like to classify

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

These boundaries depend on the specific optimization method. For example, logistic regression gives a different one than naive Bayes

NOTE

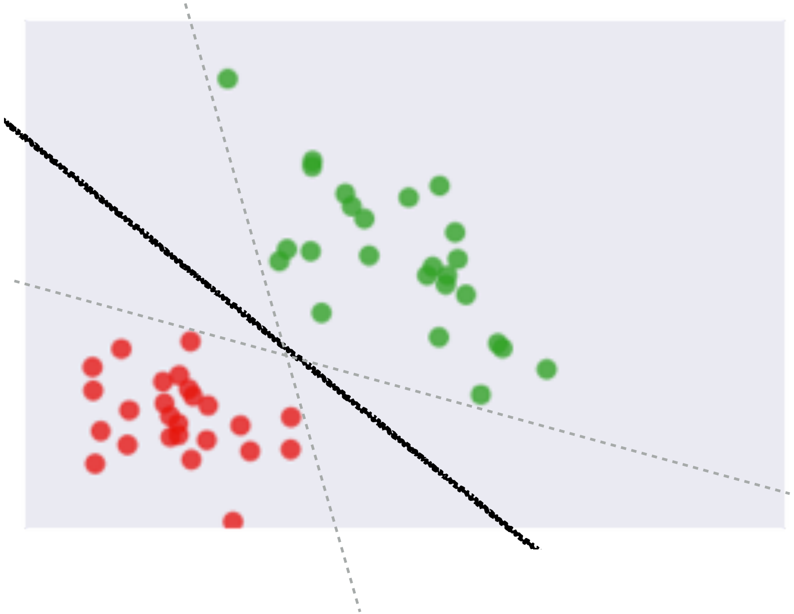
These are all different ways of looking at the same problem.

Familiarity with more than one leads to deeper understanding!

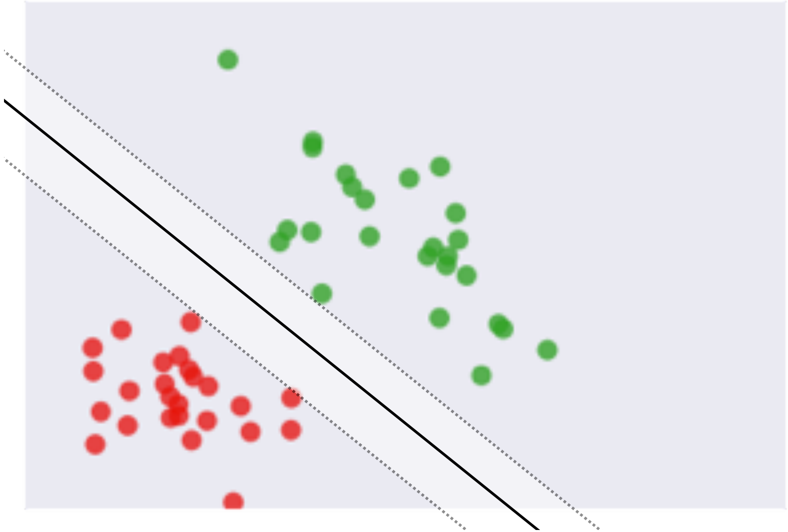
Suppose we have data with binary labels, which we would like to classify

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

These boundaries depend on the specific optimization method. For example, logistic regression gives a different one than naive Bayes

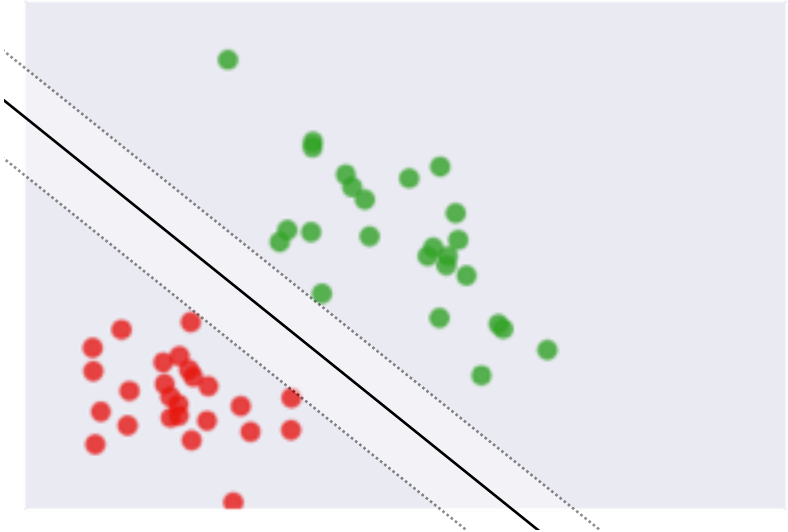


Support Vector Machines is a classifier that explicitly constructs a decision boundary that geometrically "makes the most sense"



Support Vector Machines is a classifier that explicitly constructs a decision boundary that geometrically "makes the most sense"

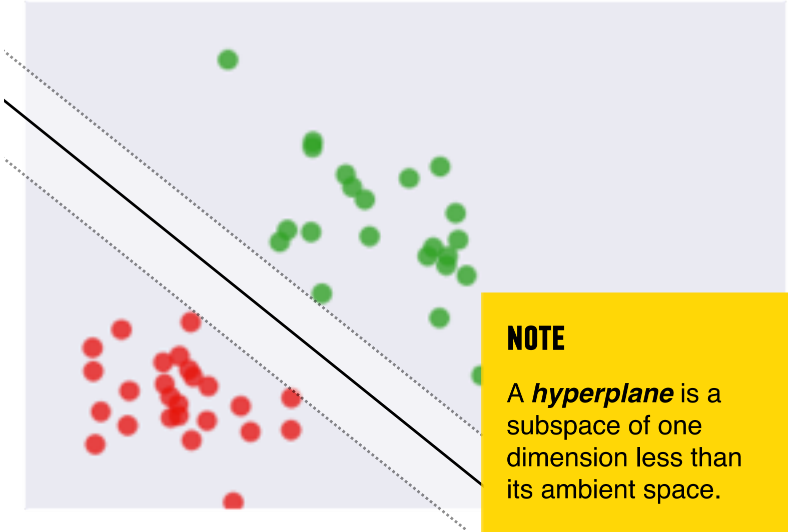
The generalization error is equated with the geometric concept of **margin**, *which is the region along the boundary that is free of points.*



Support Vector Machines is a classifier that explicitly constructs a decision boundary that geometrically "makes the most sense"

The generalization error is equated with the geometric concept of **margin**, *which is the region along the boundary that is free of points.*

The goal of SVM is to create a linear decision boundary with the largest margin. This is called the **maximum margin hyperplane**.



NOTE

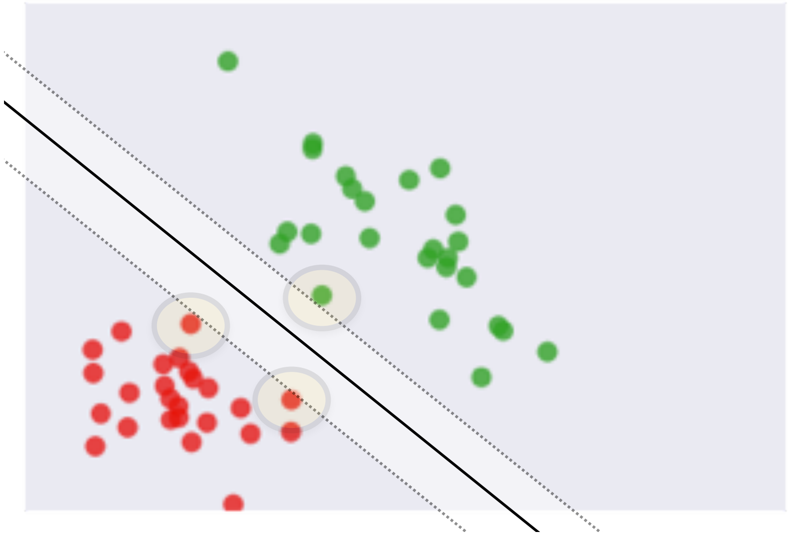
A ***hyperplane*** is a subspace of one dimension less than its ambient space.

In this 2D example, it is a line, and in a 3D space it is an ordinary plane.

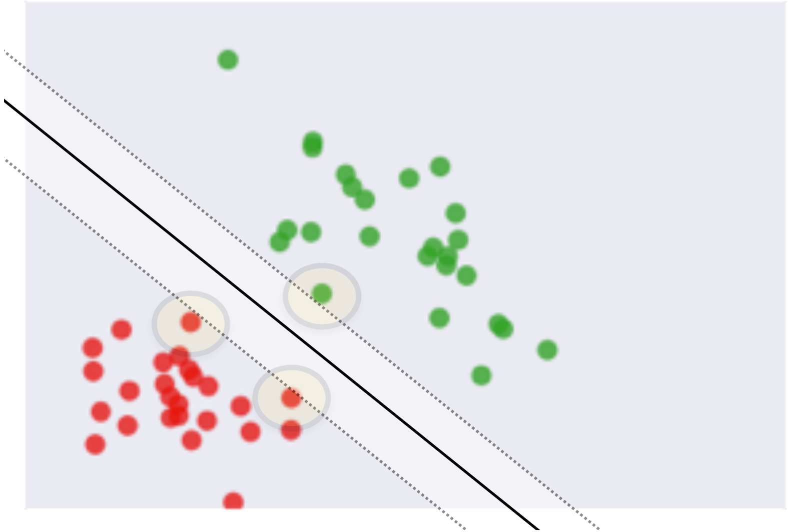
Support Vector Machines is a classifier that explicitly constructs a decision boundary that geometrically "makes the most sense"

The generalization error is equated with the geometric concept of **margin**, *which is the region along the boundary that is free of points.*

The goal of SVM is to create a linear decision boundary with the largest margin. This is called the **maximum margin hyperplane**.

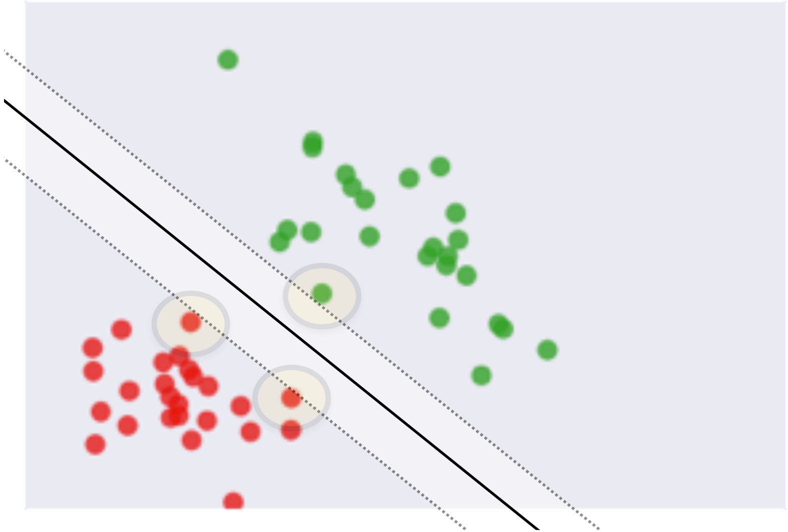


Notice that the margin depends only on a subset of the training data – the points nearest to the decision boundary.



Notice that the margin depends only on a subset of the training data – the points nearest to the decision boundary.

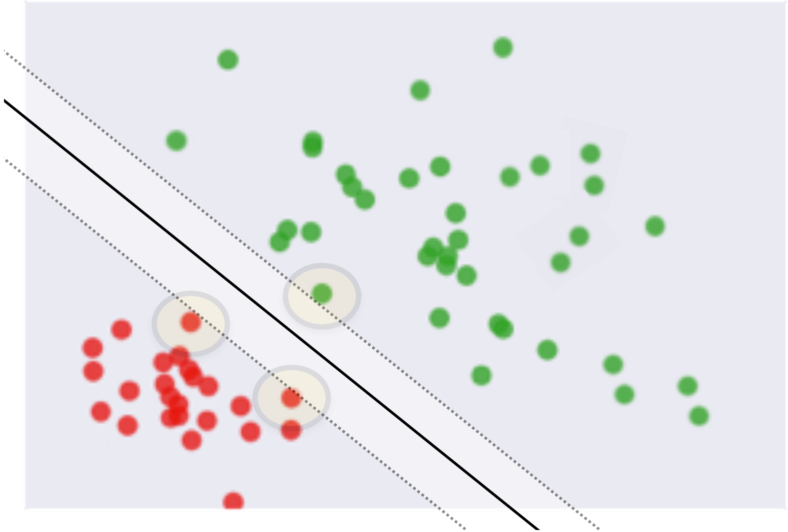
These points are called the **support vectors**.



Notice that the margin depends only on a subset of the training data – the points nearest to the decision boundary.

These points are called the **support vectors***.*

The other points don't affect the construction of the hyperplane at all!



Notice that the margin depends only on a subset of the training data – the points nearest to the decision boundary.

These points are called the **support vectors**.

The other points don't affect the construction of the hyperplane at all!

NOTE

Support vector machines require feature scaling, as we saw earlier with k-nearest neighbors.

Notice that the margin depends only on a subset of the training data – the points nearest to the decision boundary.

These points are called the **support vectors**.

The other points don't affect the construction of the hyperplane at all!

Finding the **maximum margin hyperplane** *is a straightforward exercise in analytic geometry. (We won't go through the details here.)*

Finding the **maximum margin hyperplane** *is a straightforward exercise in analytic geometry. (We won't go through the details here.)*

In short, this requires the optimization of a **convex** *objective function.*

Finding the **maximum margin hyperplane** *is a straightforward exercise in analytic geometry. (We won't go through the details here.)*

In short, this requires the optimization of a **convex** *objective function.*

Convex optimization are guaranteed to give **global optima.**

So to summarize, what is a support vector machine?

So to summarize, what is a support vector machine?

An SVM is a binary linear classifier whose decision boundary is explicitly constructed to minimize generalization error.

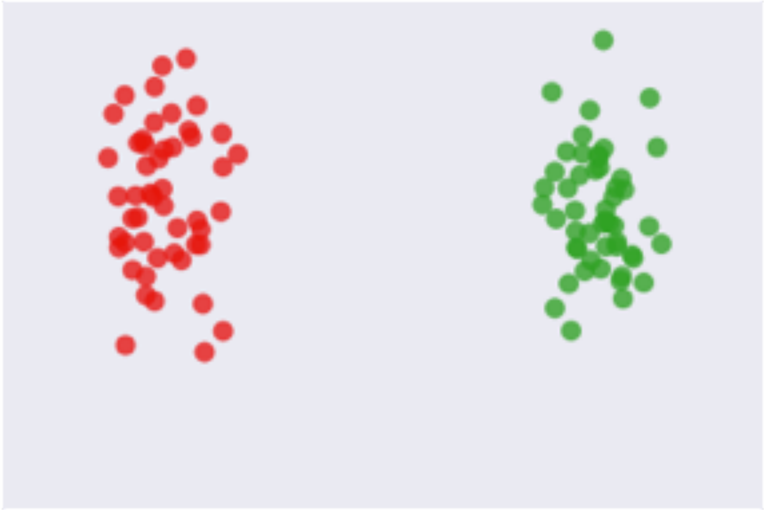
recall:

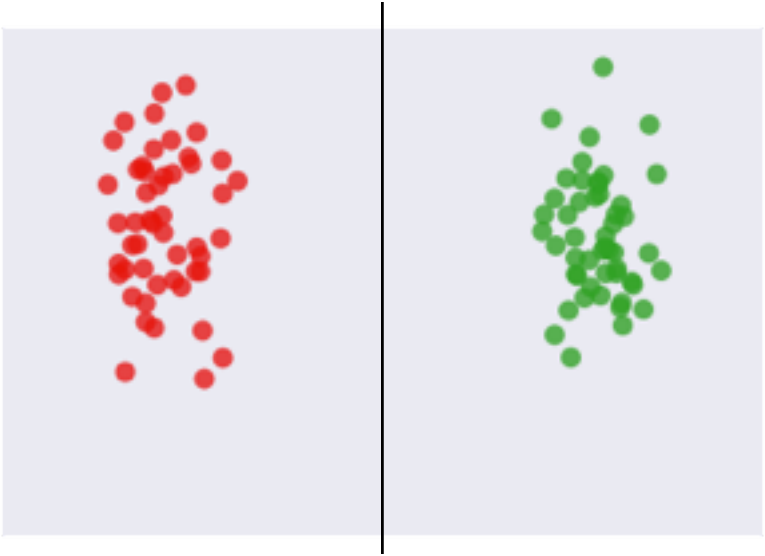
binary classifier – *solves two-class problem*

linear classifier – *creates linear decision boundary*

II. REGULARIZATION

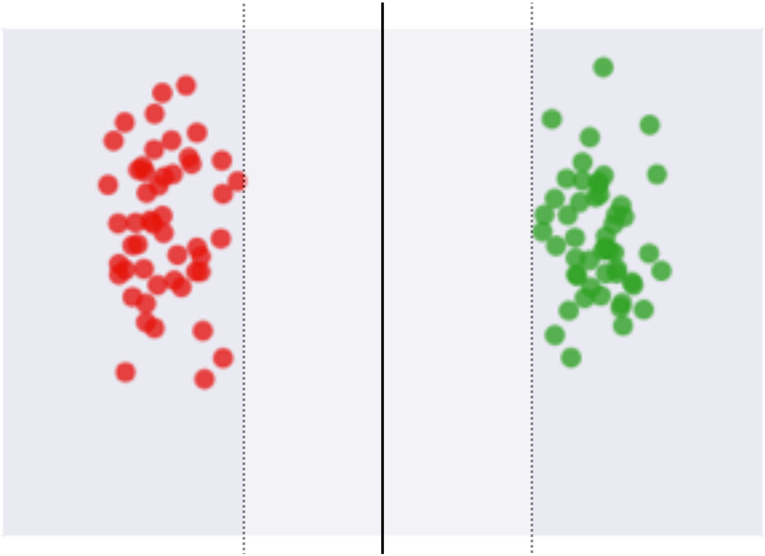
Let's apply a SVM to the dataset on the left





Let's apply a SVM to the dataset on the left.

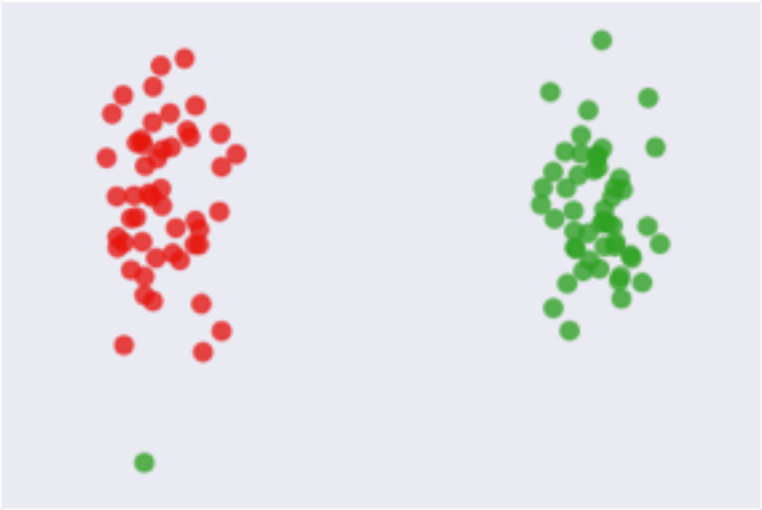
If the data are **linearly separable**, *the training error is zero.*



Let's apply a SVM to the dataset on the left.

If the data are **linearly separable**, *the training error is zero.*

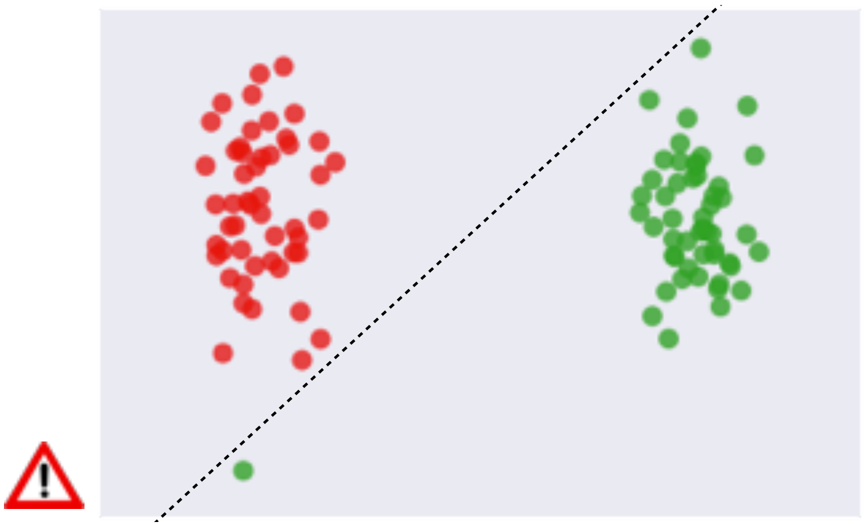
The **margin** *is nice and wide.*



Let's apply a SVM to the dataset on the left.

If the data are **linearly separable**, *the training error is zero.*

But what if our data has a single outlier?



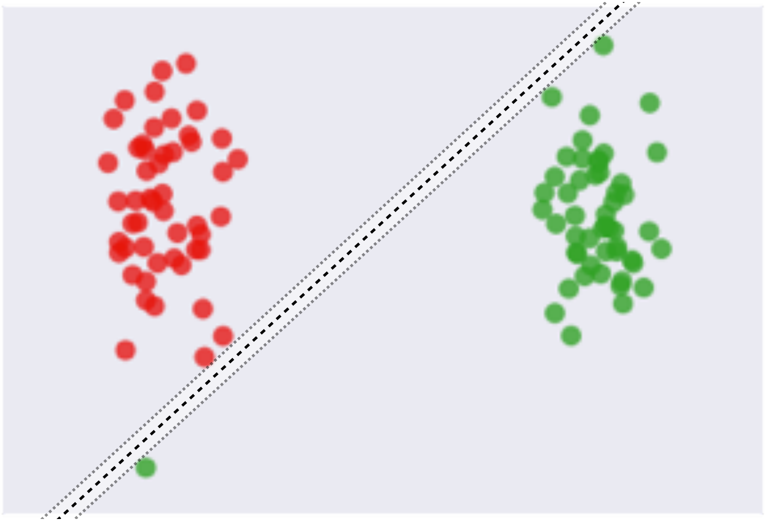
Let's apply a SVM to the dataset on the left.

If the data are **linearly separable**, *the training error is zero.*

But what if our data has a single outlier?

This will disproportionally impact the result, since the SVM tries to linearly separate **all** *data.*

Let's apply a SVM to the dataset on the left.



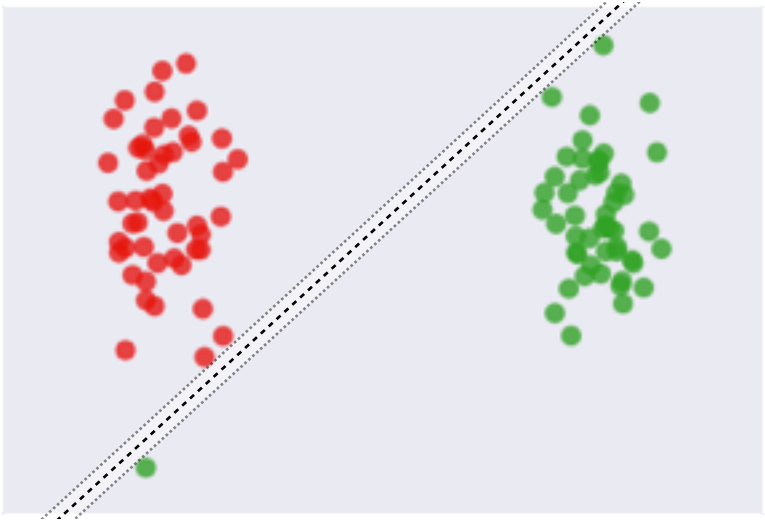
If the data are **linearly separable**, *the training error is zero.*

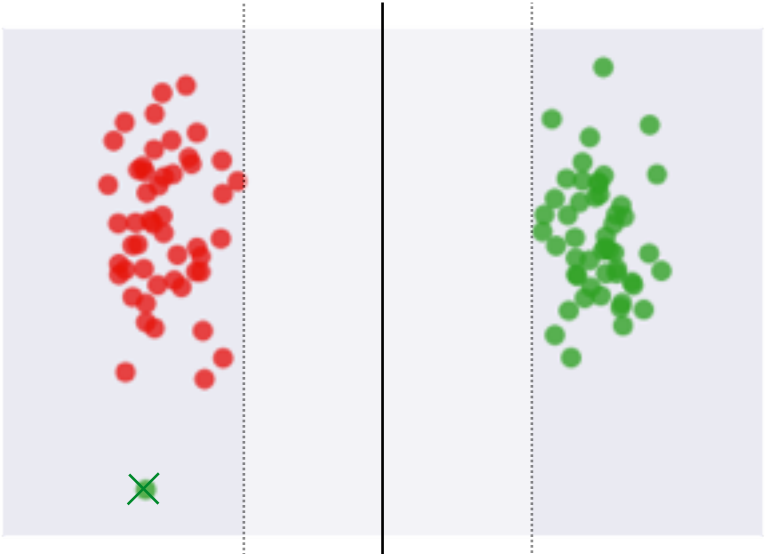
But what if our data has a single outlier?

This will disproportionally impact the result, since the SVM tries to linearly separate **all** *data.*

The **margin** *is very small.*

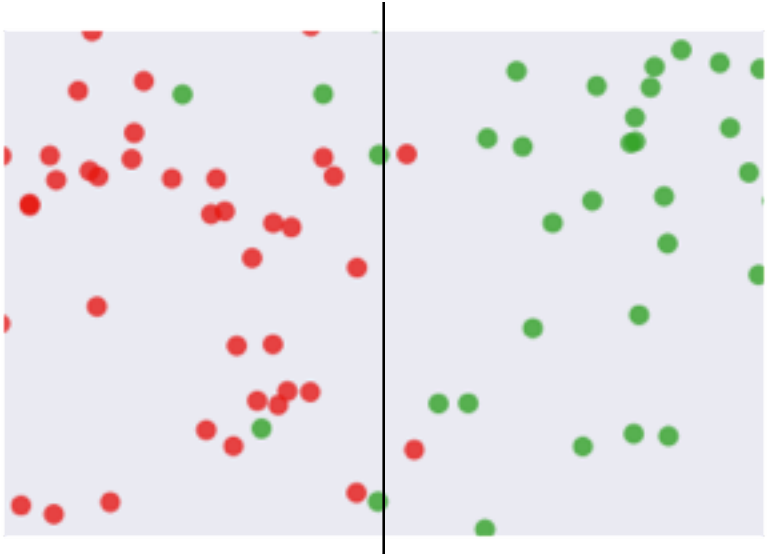
Again, we'll need some sort of regularization.





Again, we'll need some sort of regularization.

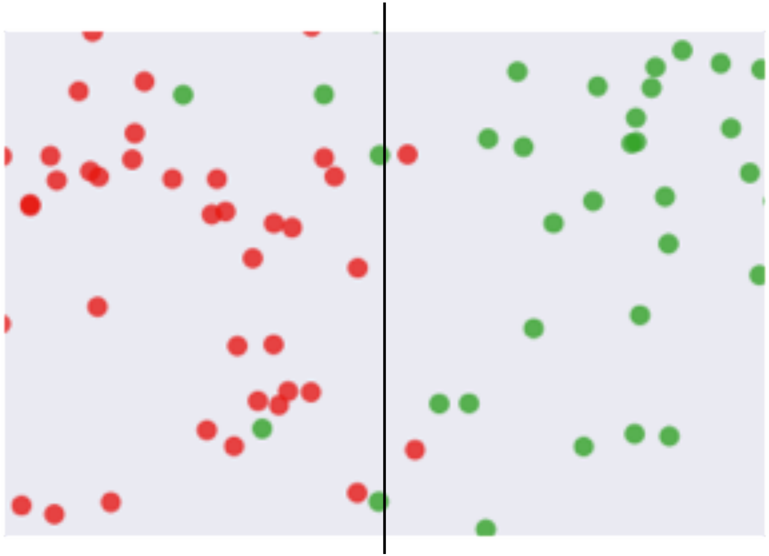
We want a **tradeoff** *of the width of the margin and the number of misclassifications.*



Again, we'll need some sort of regularization.

We want a **tradeoff** *of the width of the margin and the number of misclassifications.*

We definitely need this when our data are not perfectly linearly separable.

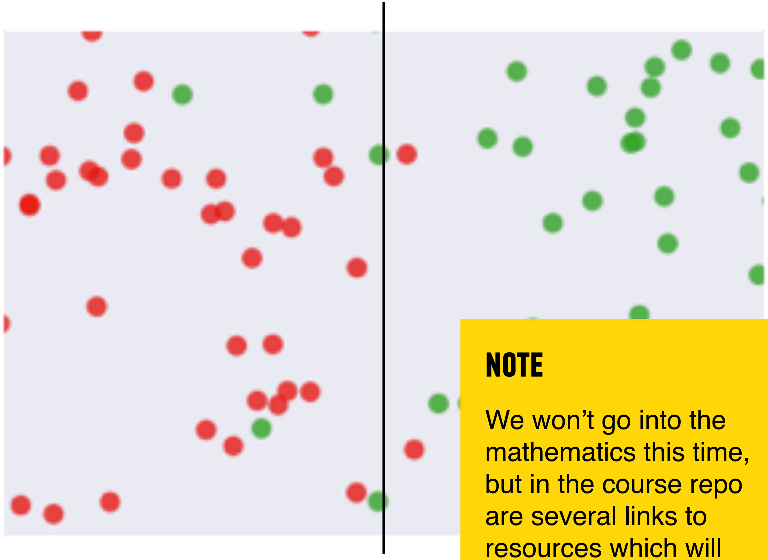


Again, we'll need some sort of regularization.

We want a **tradeoff** *of the width of the margin and the number of misclassifications.*

We definitely need this when our data are not perfectly linearly separable.

This is again a bias-variance tradeoff, which is done by **slack variables***.*



NOTE

We won't go into the mathematics this time, but in the course repo are several links to resources which will explain this if you're interested.

Again, we'll need some sort of regularization.

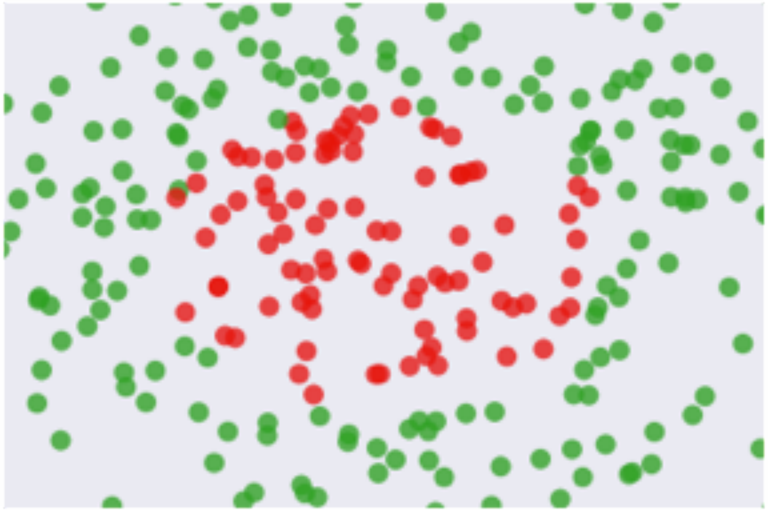
We want a ***tradeoff*** *of the width of the margin and the number of misclassifications.*

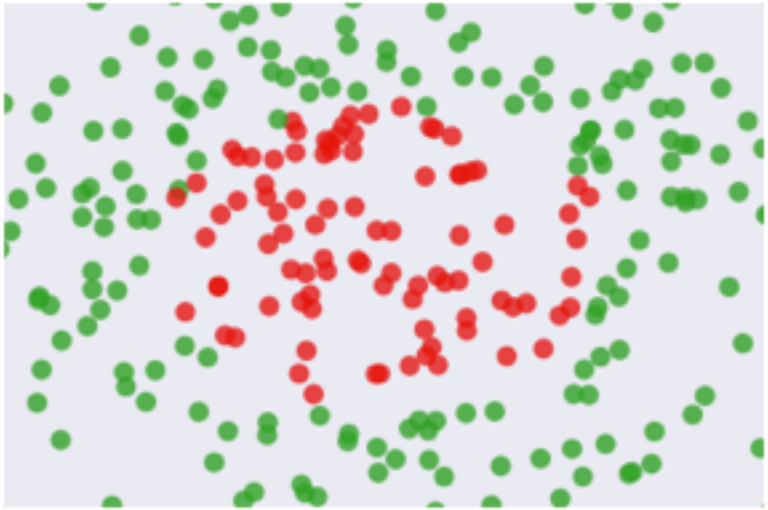
We definitely need this when our data are not perfectly linearly separable.

This is again a bias-variance tradeoff, which is done by ***slack variables****.*

III. KERNELS

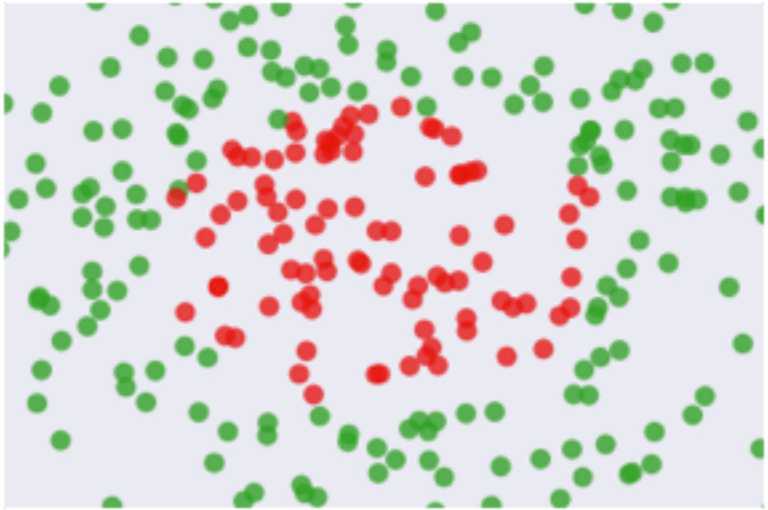
What if our data is not linearly separable at all?





What if our data is not linearly separable at all?

Again, we could add polynomial features. (This might be computationally expensive.)

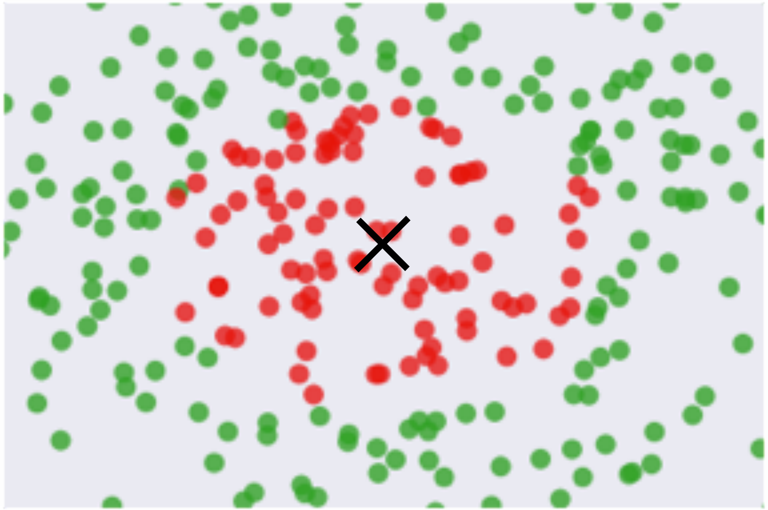


What if our data is not linearly separable at all?

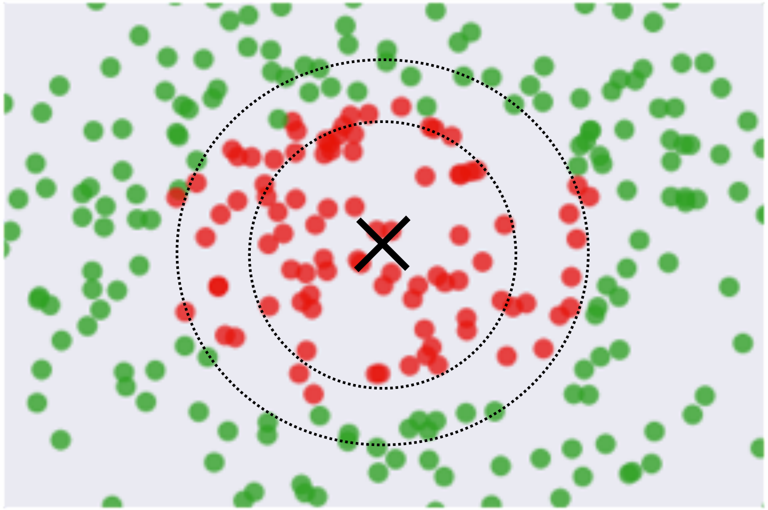
Again, we could add polynomial features. (This might be computationally expensive.)

We could also use ***kernels****.*

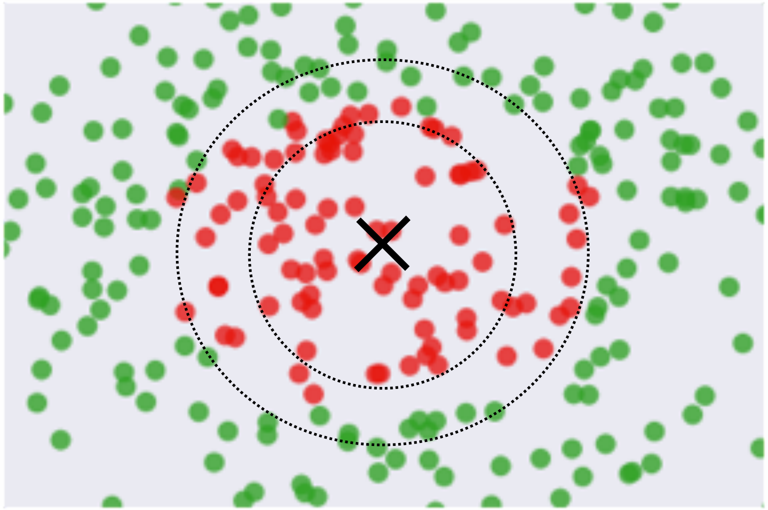
Add a **landmark** *to the feature space.*



Add a **landmark** *to the feature space.*



For each point, compute the distance to this landmark: $||x - l||$

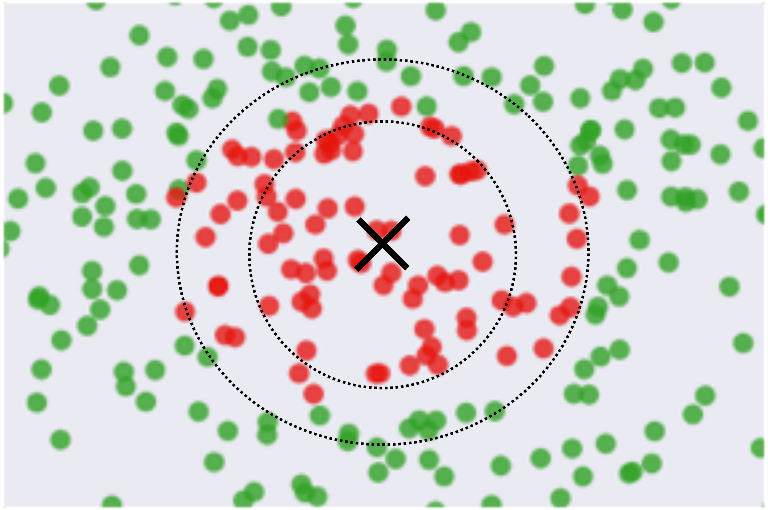


Add a **landmark** *to the feature space.*

For each point, compute the distance to this landmark: $\|x - l\|$

Then define the similarity as the **radius basis function (rbf)**

$$e^{-\frac{\|x-l\|^2}{2\sigma^2}}$$



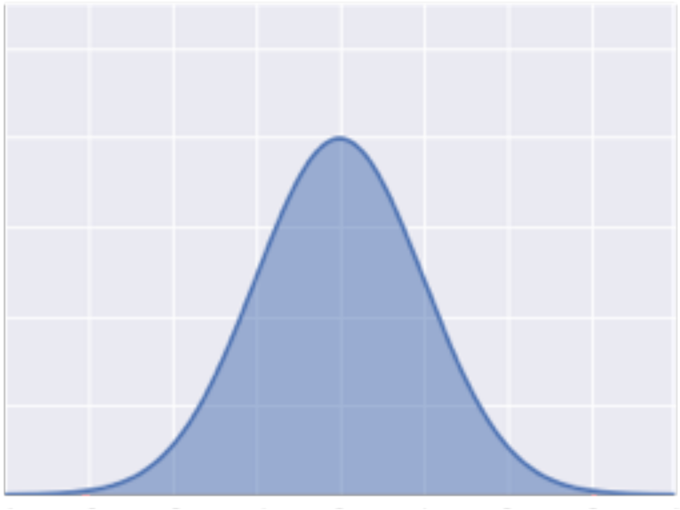
Add a **landmark** *to the feature space.*

For each point, compute the distance to this landmark: $||x - l||$

Then define the similarity as the **radius basis function (rbf)**

$$e^{-\frac{||x-l||^2}{2\sigma^2}}$$

This is called a Gaussian kernel.



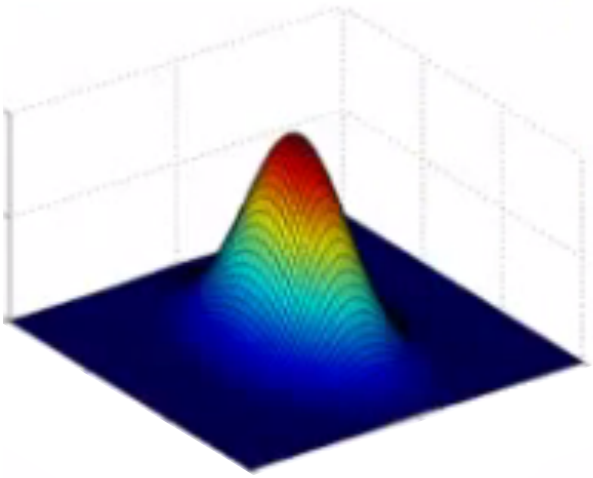
Add a **landmark** *to the feature space.*

For each point, compute the distance to this landmark: $||x - l||$

Then define the similarity as the **radius basis function (rbf)**

$$e^{-\frac{||x-l||^2}{2\sigma^2}}$$

This is called a Gaussian kernel.



Add a **landmark** *to the feature space.*

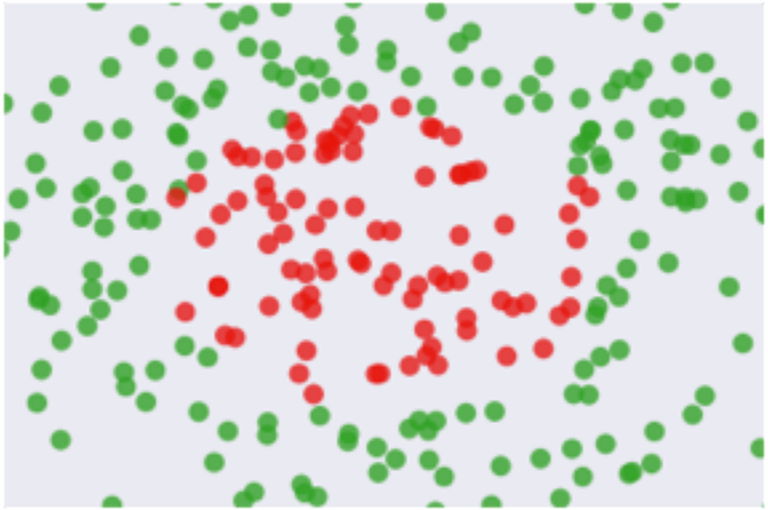
For each point, compute the distance to this landmark: $||x - l||$

Then define the similarity as the **radius basis function (rbf)**

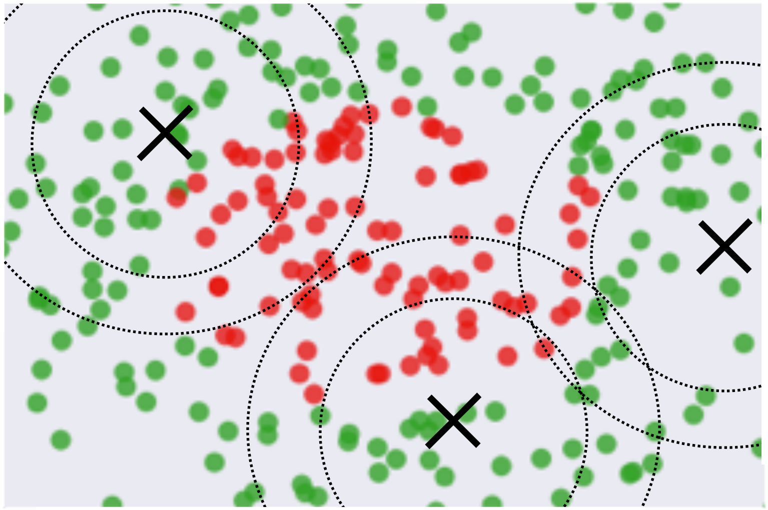
$$e^{-\frac{||x-l||^2}{2\sigma^2}}$$

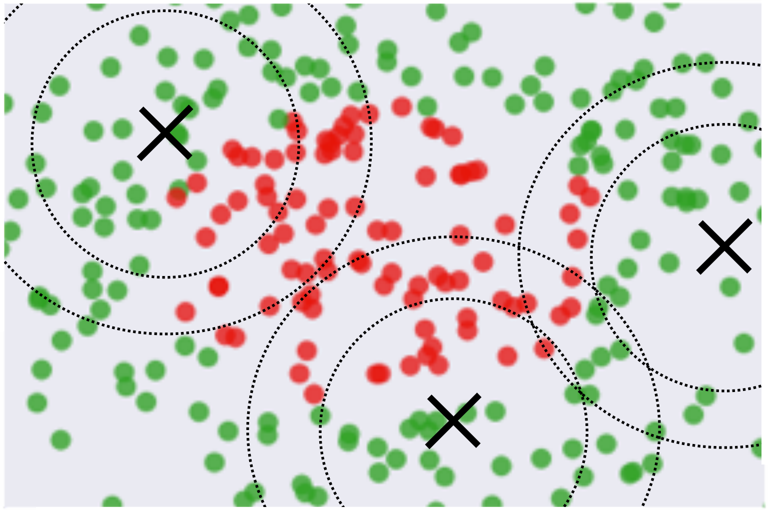
This is called a Gaussian kernel.

How do we choose the right landmark?



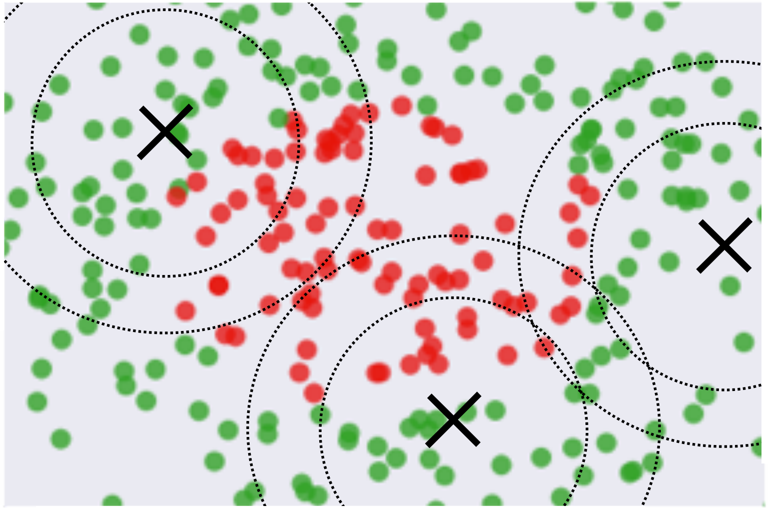
How do we choose the right landmark?





How do we choose the right landmark?

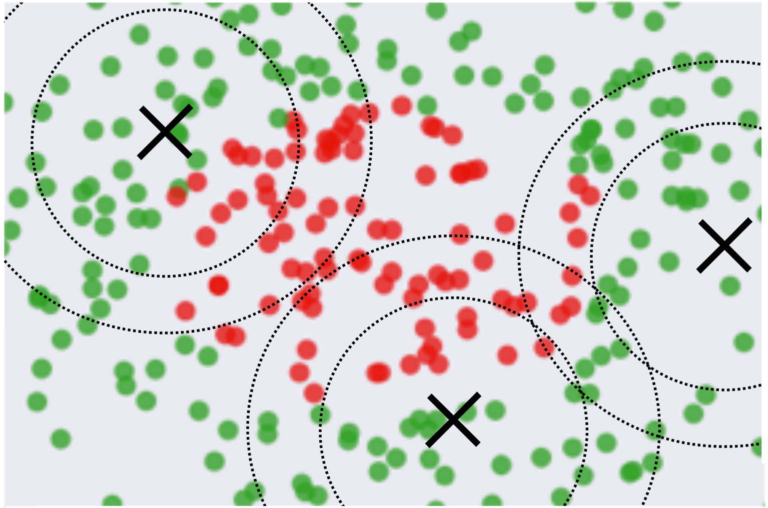
Choose each sample as a landmark.



How do we choose the right landmark?

Choose each sample as a landmark.

Then replace all samples with the kernel values.

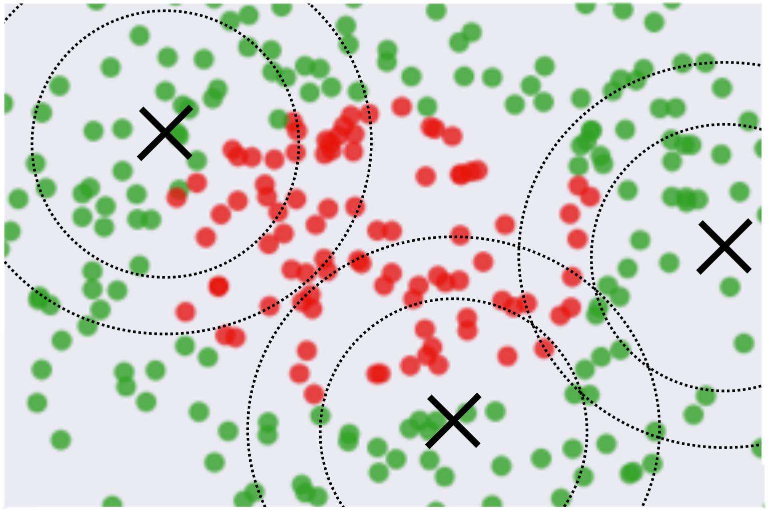


How do we choose the right landmark?

Choose each sample as a landmark.

Then replace all samples with the kernel values.

*This will (non-linearly) transform your feature matrix **X** from an $N \times n$ matrix to an $N \times N$ matrix.*



How do we choose the right landmark?

Choose each sample as a landmark.

Then replace all samples with the kernel values.

This will (non-linearly) transform your feature matrix ***X*** *from an N×n matrix to an N×N matrix.*

On the diagonal we have ones (each sample compared with itself).

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

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

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

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

NOTE

These conditions are contained in a result called *Mercer's theorem*.

Kernels can be used as preprocessing step at other models as well.

Kernels can be used as preprocessing step at other models as well.

However, they tend to be very computationally expensive.

Kernels can be used as preprocessing step at other models as well.

However, they tend to be very computationally expensive.

The SVM is far more efficient, so using kernels is more practical.

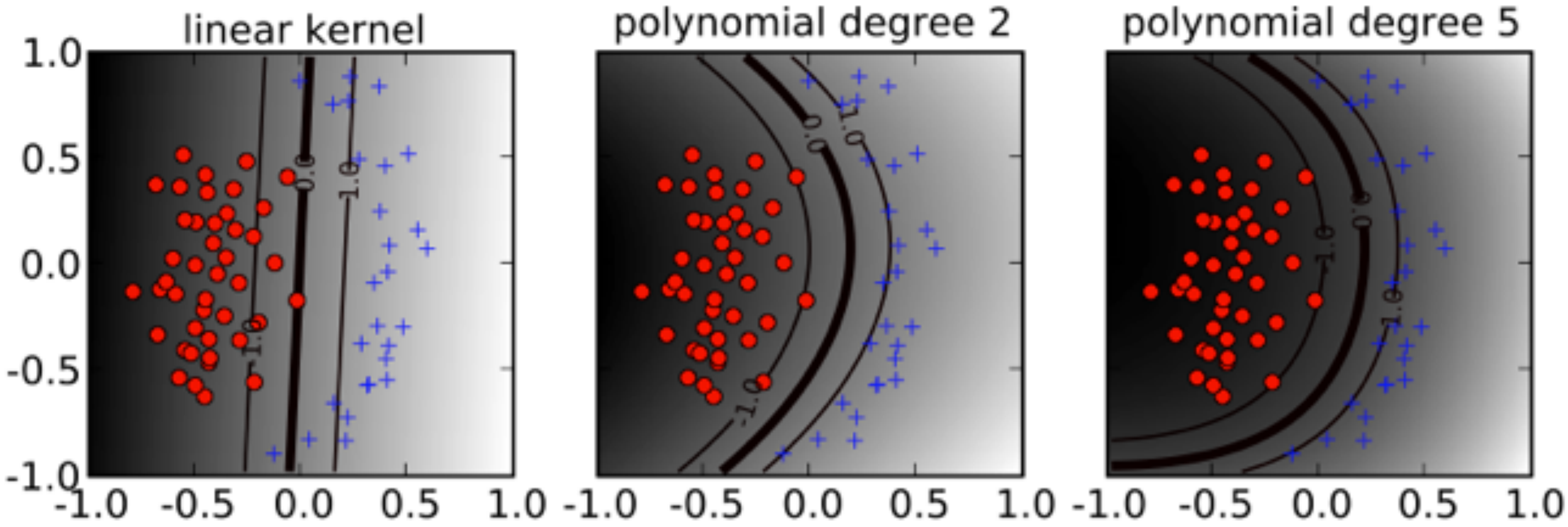
Some popular kernels

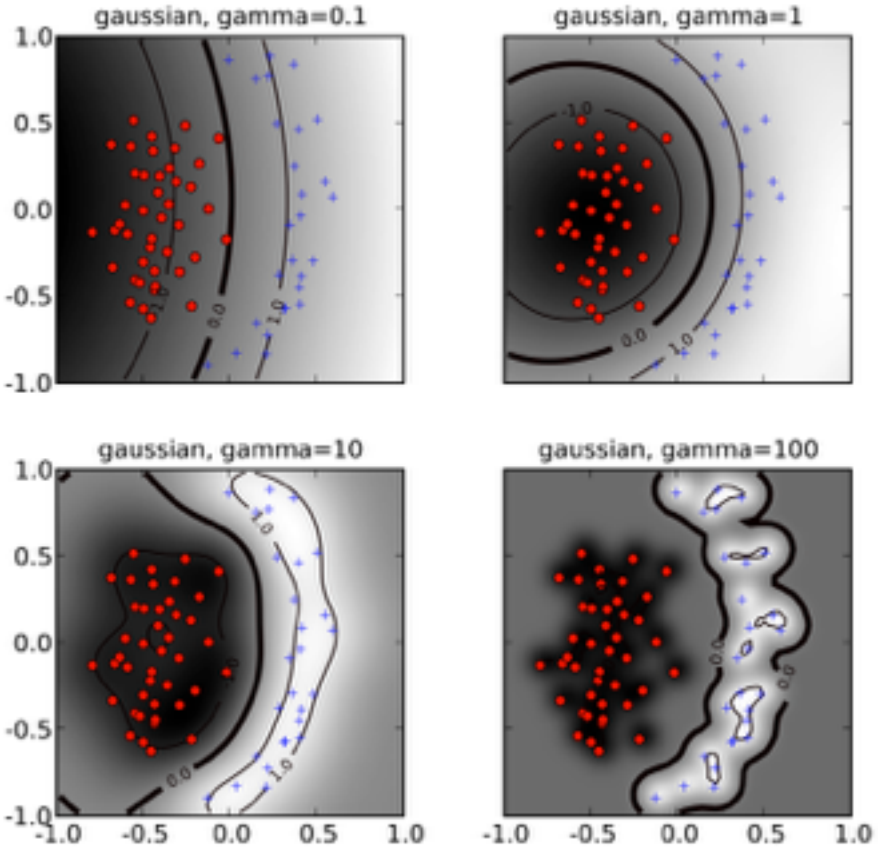
linear kernel $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$

polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^d$

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

The **hyperparameters** *d and γ affect the flexibility of the dec. boundary*


linear kernel
polynomial degree 2
polynomial degree 5
1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0


gaussian, gamma=0.1
gaussian, gamma=1
gaussian, gamma=10
gaussian, gamma=100
1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0

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

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

The main disadvantage of SVMs is the lack of intuition they produce.

These models are truly black boxes!

DISCUSSION