Support Vector Machins
Introduction to ML part 2

AGENDA

1. Introduction to SVMs
    1. Background understanding – Visualizing SVMs
    2. Background understanding – “A Little Math”
2. Linear SVMs
    1. Soft margin tuning
    2. Key points & comparison with Logistic Regression
3. Non-Linear Decision Boundaries
    1. Understanding non-linear decision boundaries
    2. Computational efficiency and “The Kernel Trick”
4. Non-Linear Kernel SVMs
    1. Review of popular Kernels
    2. Gaussian Kernel: Conceptualization and Tuning
    3. Wrap up and comparison with Logistic Regression

INTRODUCTION

- What are Support Vector Machines (SVMs)?

INTRODUCTION

- What are Support Vector Machines (SVMs)?

SVMs is one of the most popular binary ML classifiers

INTRODUCTION

- What are Support Vector Machines (SVMs)?

SVMs is one of the most popular binary ML classifiers

- An SVM separates data sets into (+) and (-) classes by "slicing" the feature space into 2 regions:

INTRODUCTION

- What are Support Vector Machines (SVMs)?

SVMs is one of the most popular binary ML classifiers

- An SVM separates data sets into (+) and (-) classes by "slicing" the feature space into 2 regions:
  1. A positive (+) region of the feature space, and

INTRODUCTION

- What are Support Vector Machines (SVMs)?

SVMs is one of the most popular binary ML classifiers

- An SVM separates data sets into (+) and (-) classes by “slicing” the feature space into 2 regions:
  1. A positive (+) region of the feature space, and
  2. A negative (-) region of the feature space.

INTRODUCTION

- What are Support Vector Machines (SVMs)?

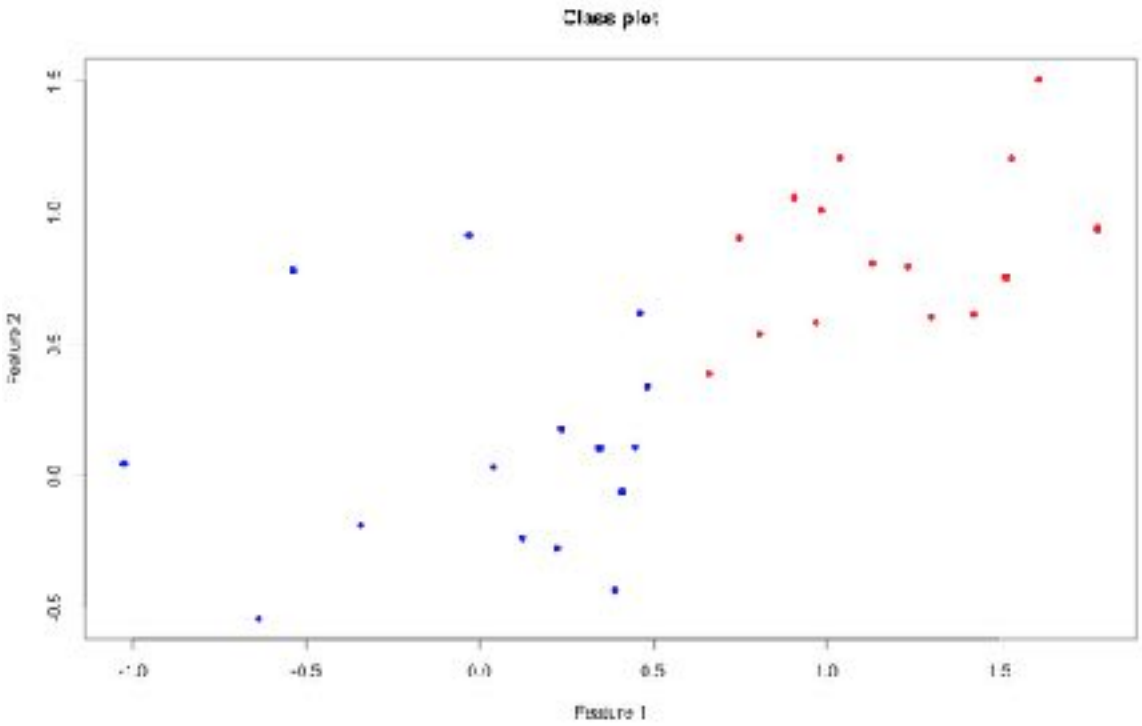
SVMs is one of the most popular binary ML classifiers

- An SVM separates data sets into (+) and (-) classes by “slicing” the feature space into 2 regions:
    1. A positive (+) region of the feature space, and
    2. A negative (-) region of the feature space.

- New data is then classified based on the region in which it lies.

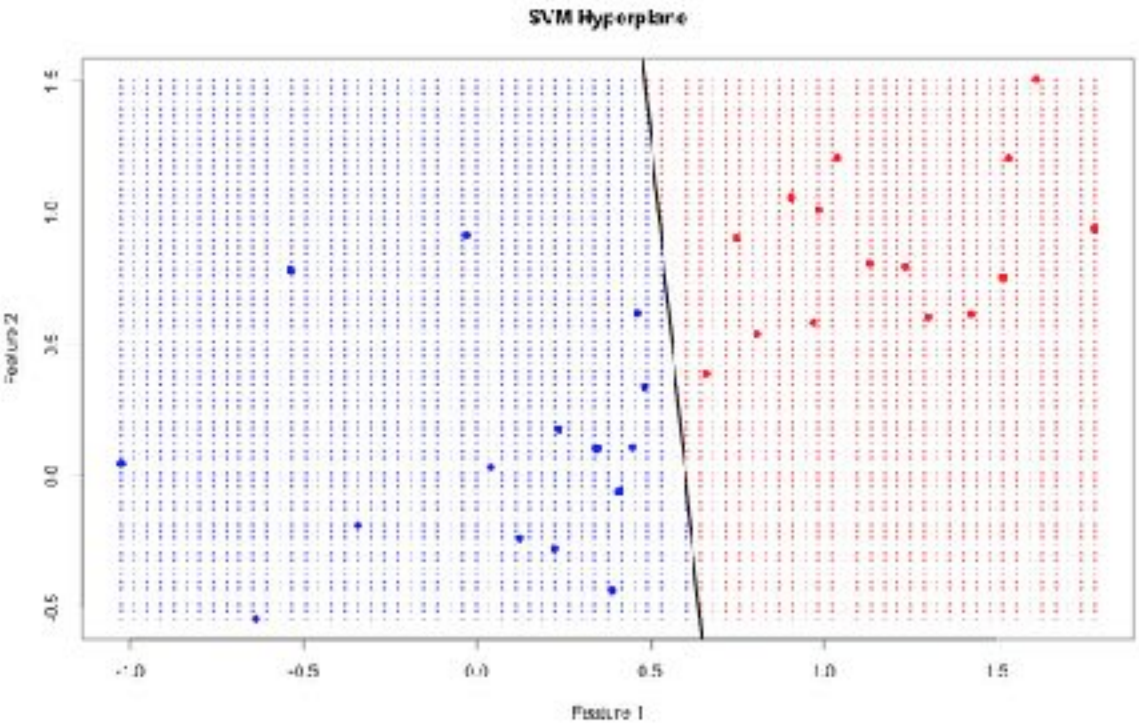
Introduction

- What does SVM classification look like?



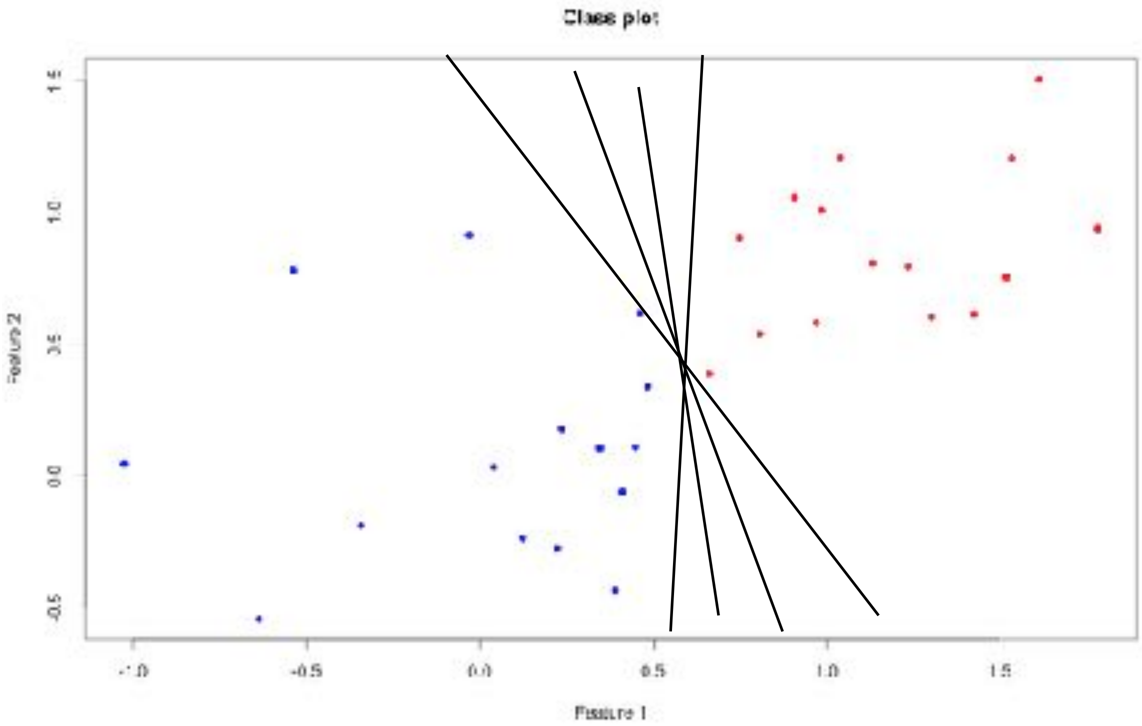
Introduction

- What does SVM classification look like?



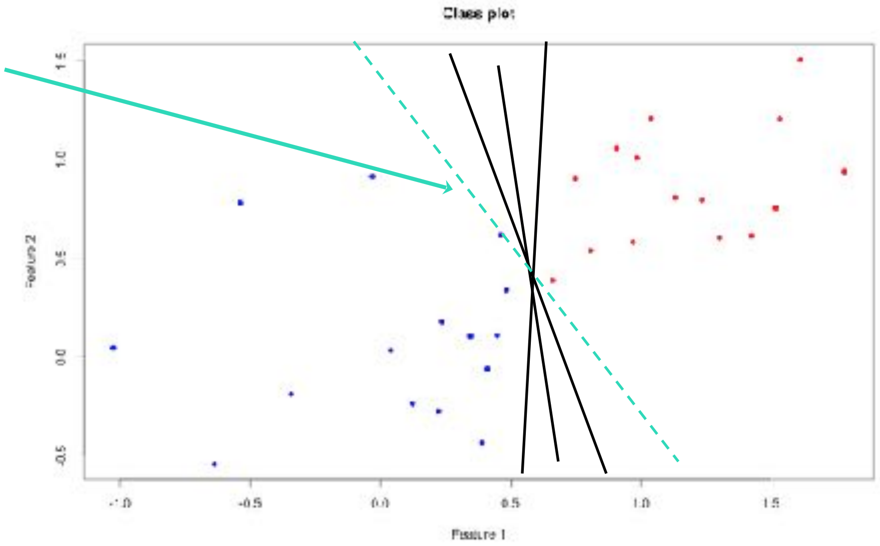
Introduction

- When the data is “separable,” multiple boundary can be defined:



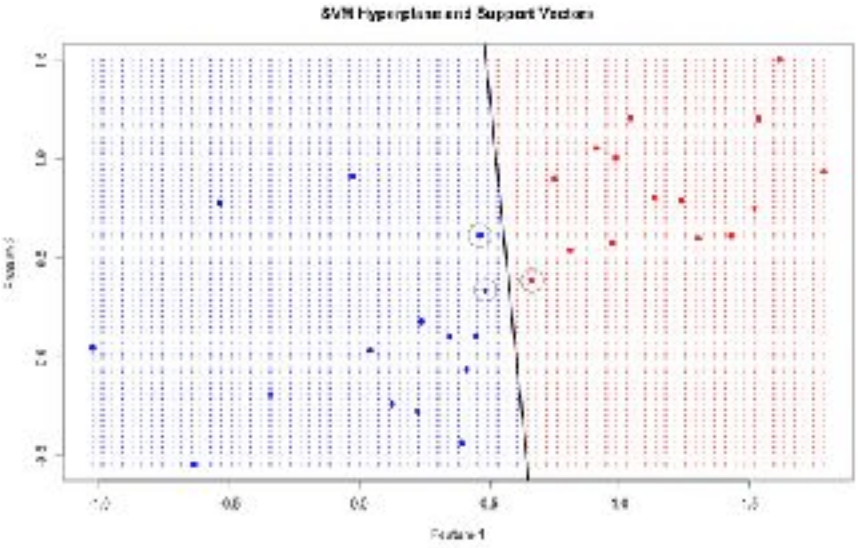
Introduction

- When the data is “separable,” multiple boundary can be defined:
    - *Ceteris paribus,* separators that “cut too close” to potential **boundary points** run a risk of underperforming on new data…



Introduction

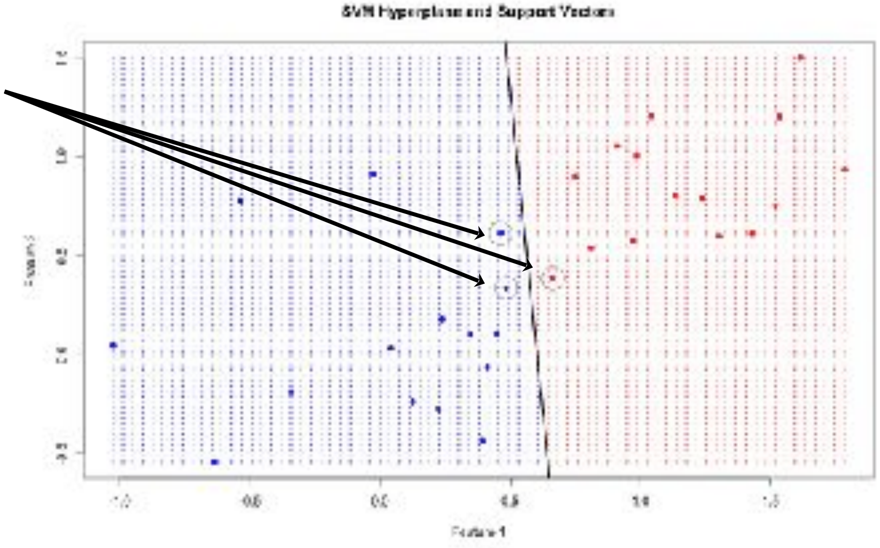
- When the data is “separable,” multiple boundary can be defined:
  - *Ceteris paribus,* separators that “cut too close” to potential **boundary points** run a risk of underperforming on new data…
  - SVMs work by finding the boundary with an optimal chance of generalizing to new data, by avoiding these points.



Introduction

- How is the the hyperplane detected?

  - The hyper-plane is defined by identifying the “***Support Vectors***.”

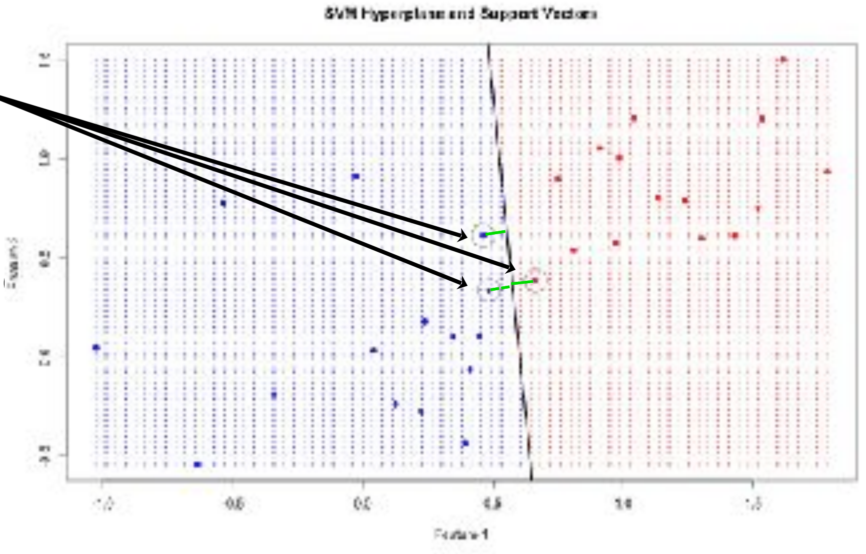


Introduction

- How is the the hyperplane detected?

  - The hyper-plane is defined by identifying the “***Support Vectors***.”

  - And finding the boundary that *optimizes* the the distances from th set of support vectors to the hyperplane.



A little math…

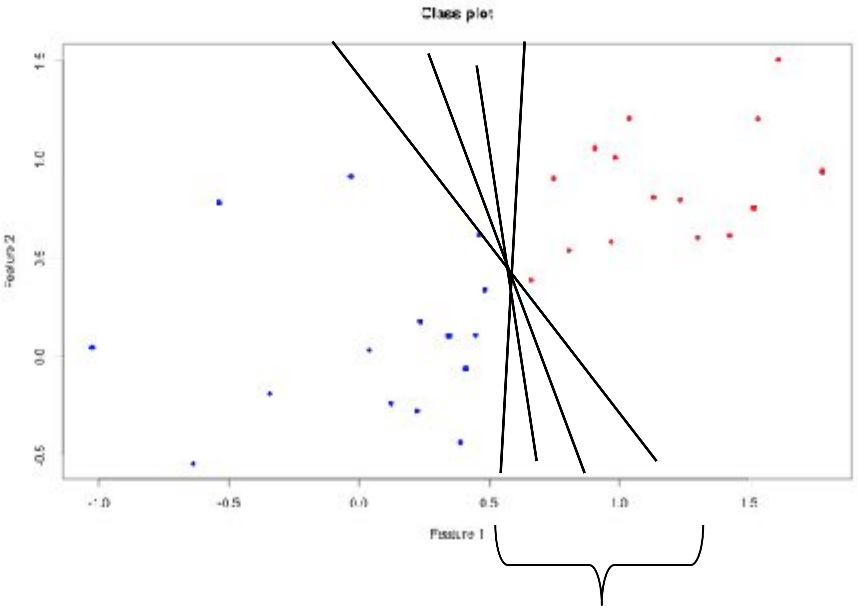
- How is the the hyperplane detected mathematically?
  - A hyperplane is defined as the set of all points x such that:

$$\sum_{i=1}^{n} x_i \beta_i = \beta_0$$

  - or

$$\sum_{i=0}^{n} x_i \beta_i = 0$$

$$\text{where } x_0 \coloneqq 1$$



Hyperplanes defined by different choices of $\vec{\beta}$

A little math…

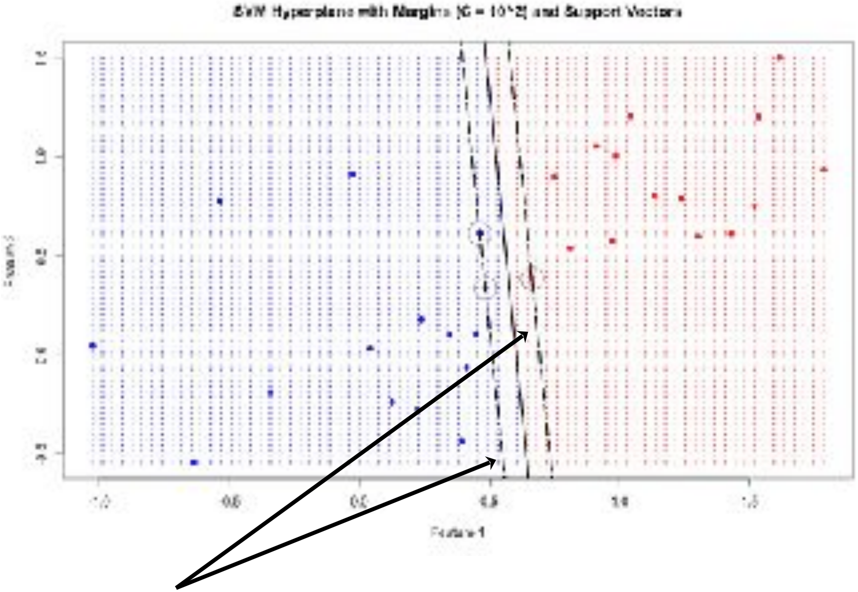
- SVM: find the line ($\vec{\beta}$ values) *maximize the +/- separation (M).*
  - This is a “constrained optimization” problem:

$$\max_{\|\vec{\beta}\| \leq 1} M$$

Such that for all $(\boldsymbol{x}_j, y_j)$:

$$y_j \left(\sum_{i=0}^{n} x_{ji} \beta_i \right) \geq M$$

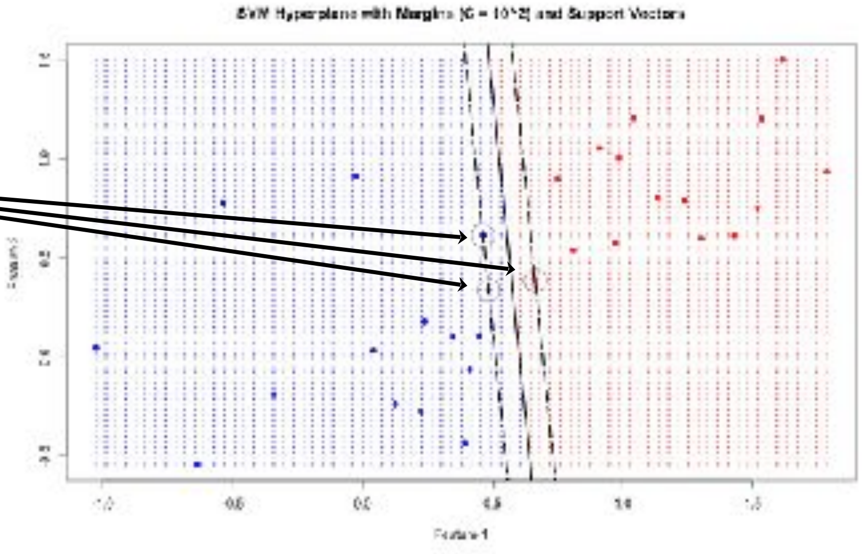
(where $y_j \epsilon \{1, -1\}$)



Margins of optimal distance M from all points.

Key Point

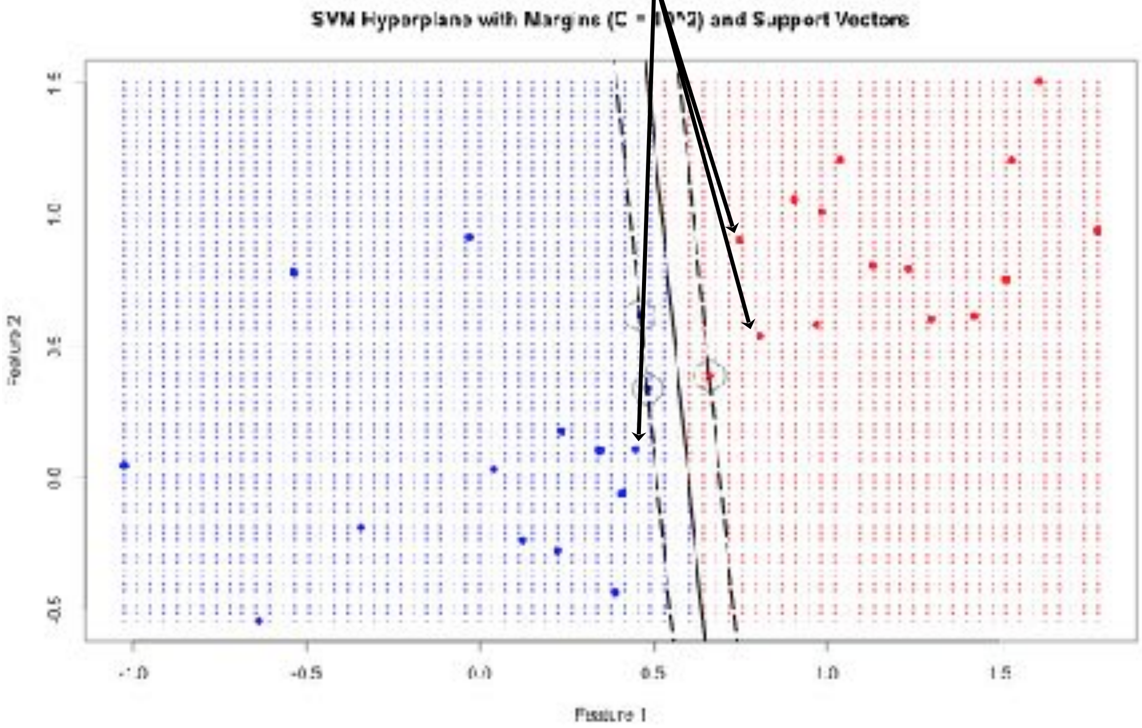
- The **maximized** margin M is based on it’s distance from the “***Support Vectors***.”



- However, in order to determine which points count as Support Vectors, the inequality must be taken over all data points.

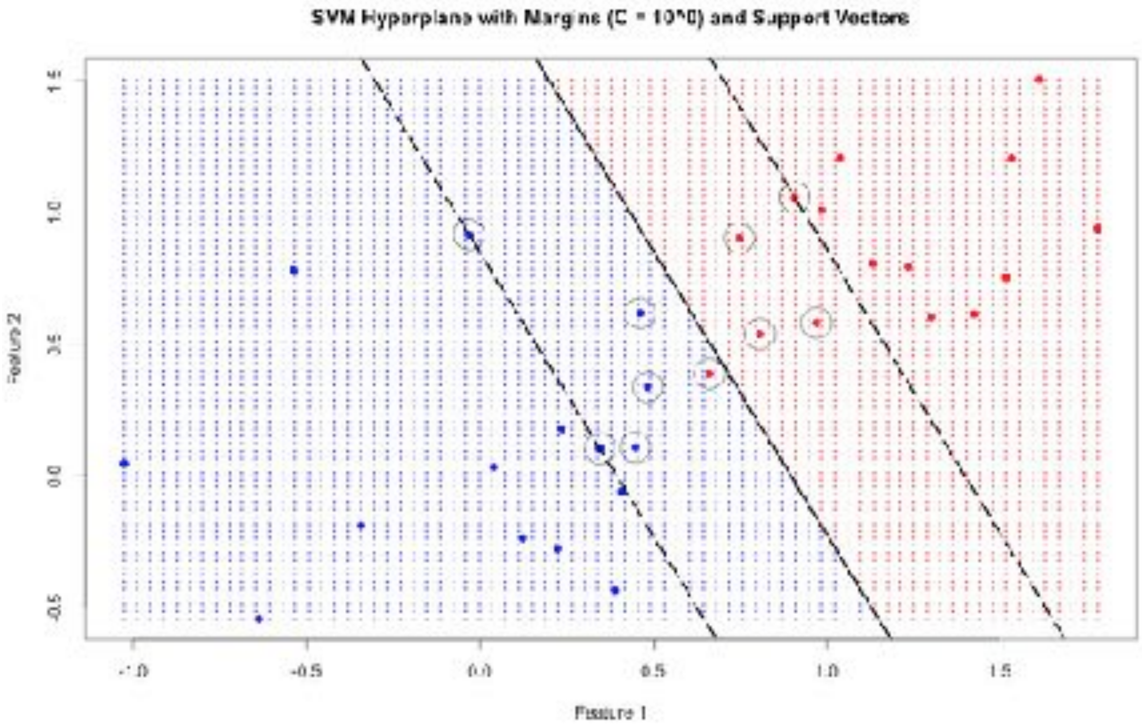
Linear SVM

- What about data points that were “almost” Support Vectors?



Linear SVM

- SVMs allow for a ***tuning parameter*** (C) that “softens” the margins.



Linear SVM

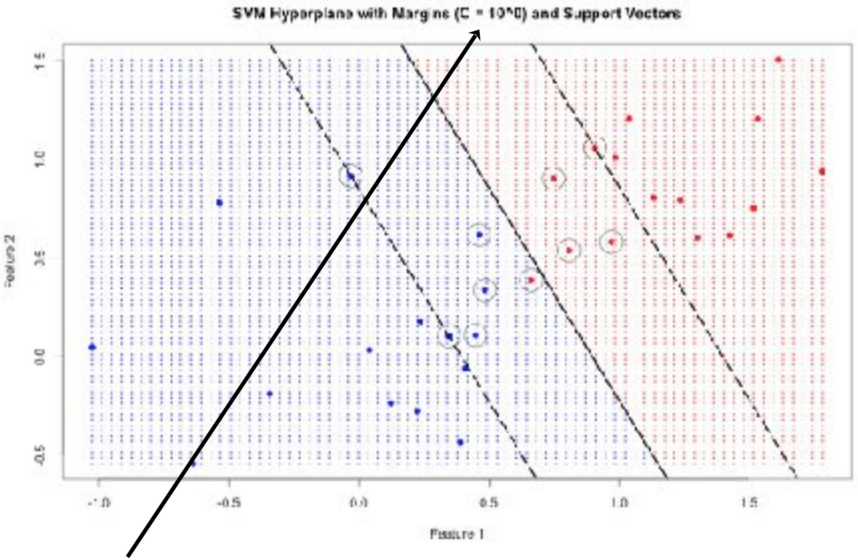
- Margins are softened, by adding error factors   to the constraint equations:
  - Softened constrained optimization:

$$\max_{\|\vec{\beta}\| \leq 1} M$$

Such that for all $(\boldsymbol{x}_j, y_j)$:

$$y_j \left(\sum_{i=0}^{n} x_{ji} \beta_i \right) \geq M(1 - \varepsilon_j)$$

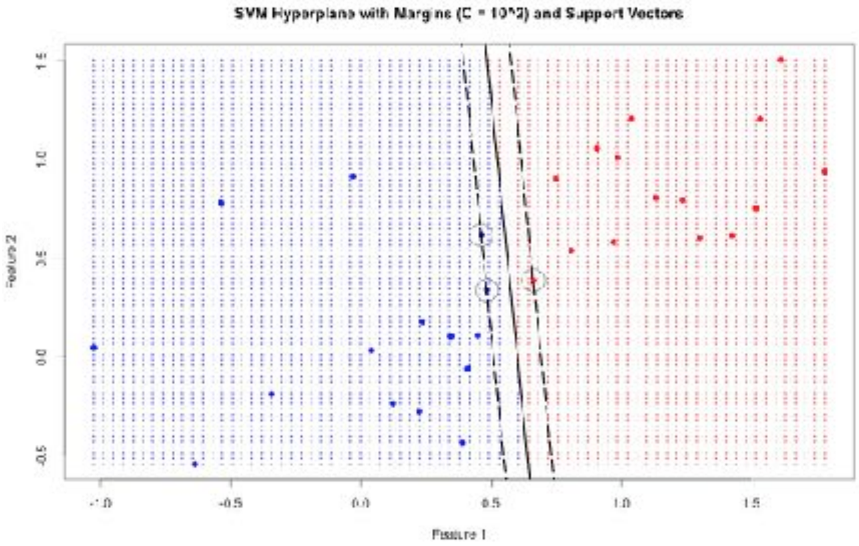
where the $\varepsilon_j \geq 0$ and $\sum \varepsilon_j \propto \frac{1}{C}$.

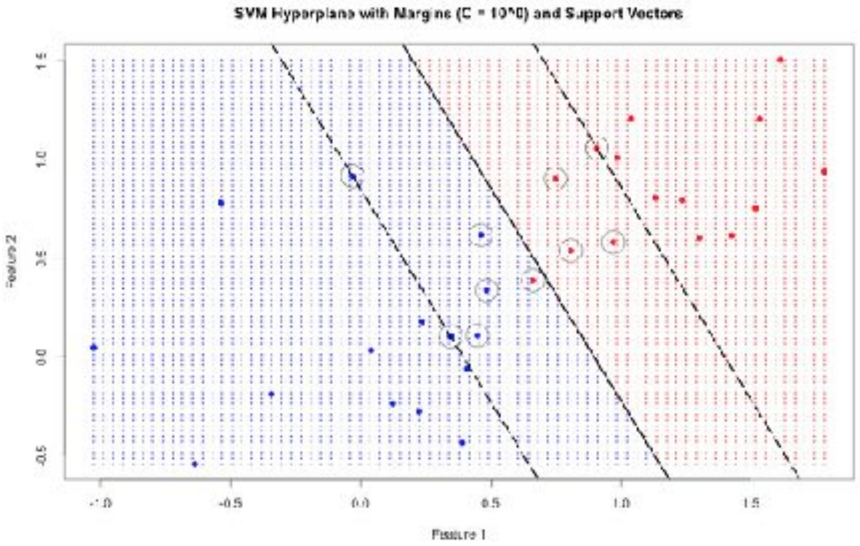


Note: The smaller the value of C the wider the margins become.

Linear SVM

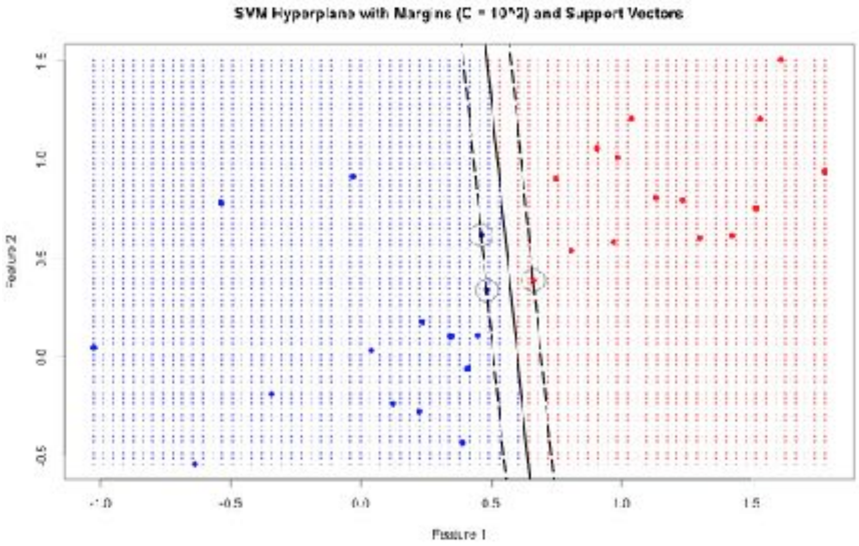
- Tuning the margins (C-value) affects the Hyperplane boundary.

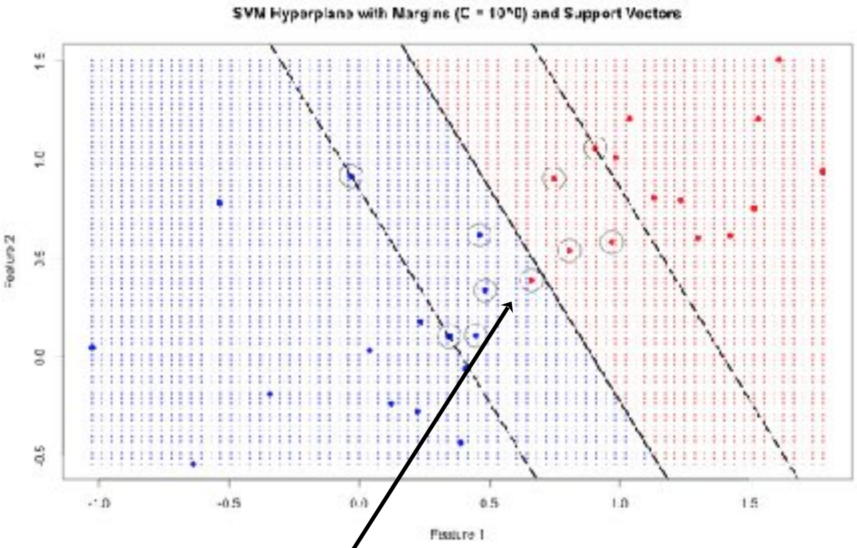




Linear SVM

- Under-fitting C can lead to **Generalization Bias**

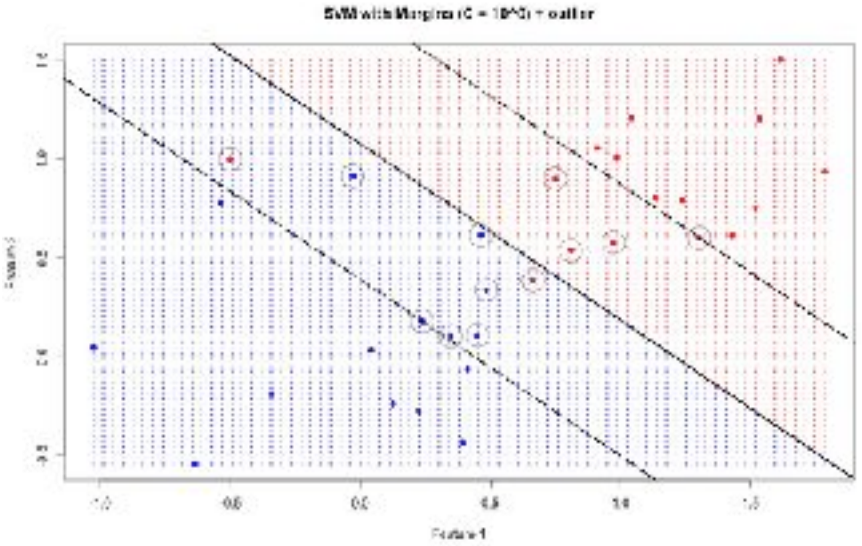


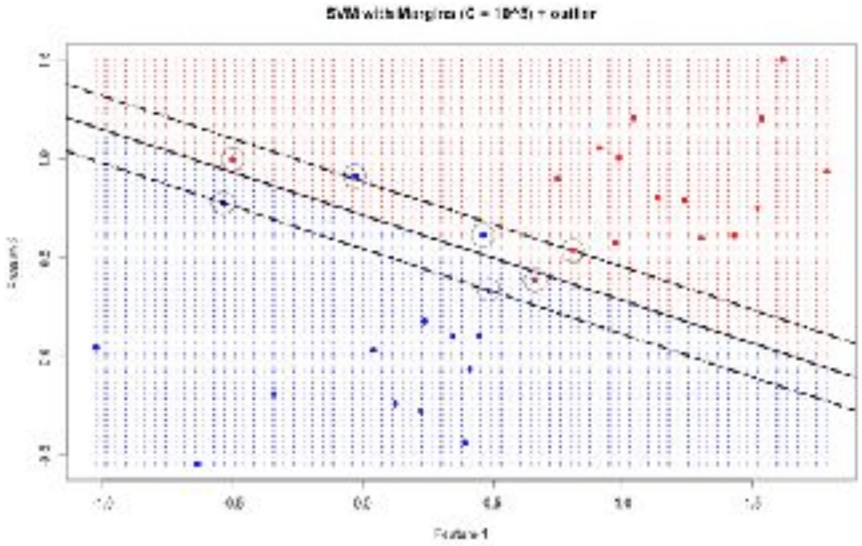


Margins that are too wide can cause miss-classification

Linear SVM

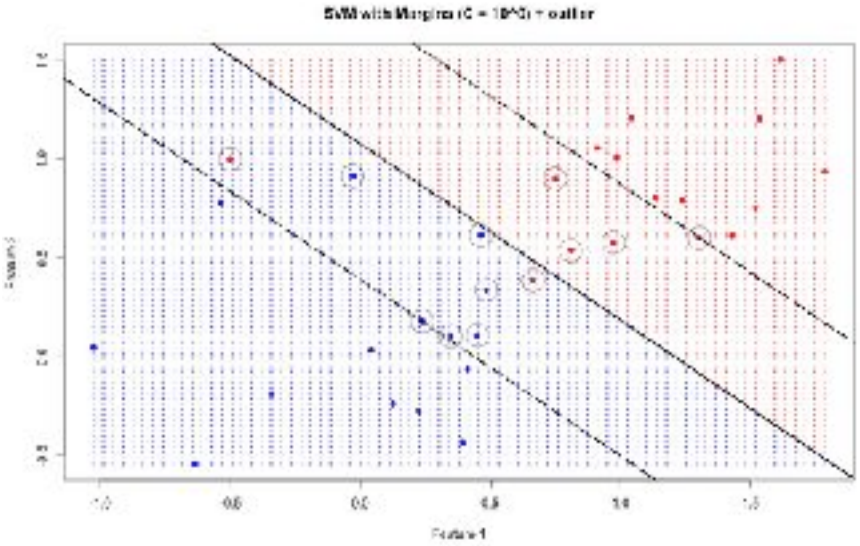
- Over-fitting C can lead to **Generalization Variance**

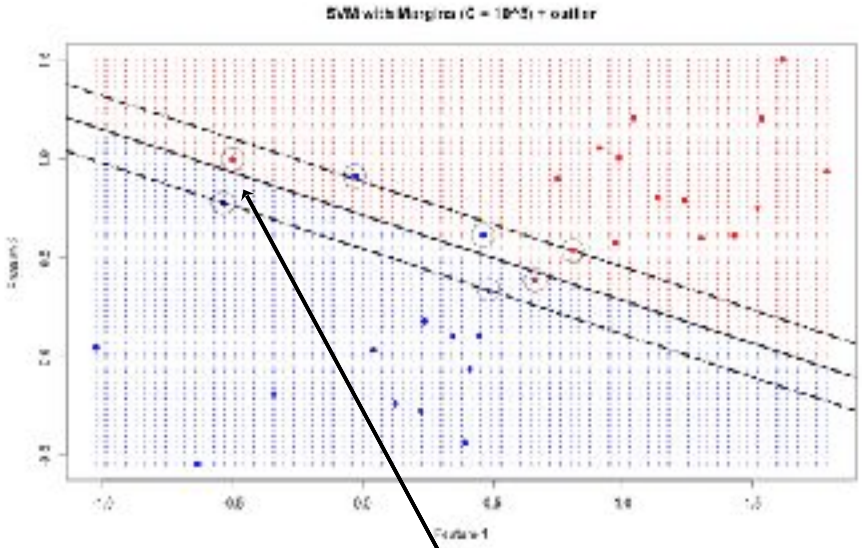




Linear SVM

- Over-fitting C means **Generalization Variance**





Margins that are too narrow are over sensitive to training outliers

Linear SVM - KEY POINTs

1. Support Vector Machines (SVMs) work by detecting decision boundaries that maximally separate (+) from (-) data points.

Linear SVM - KEY POINTs

1. Support Vector Machines (SVMs) work by detecting decision boundaries that maximally separate (+) from (-) data points.

2. Boundaries are determined by “Support Vectors (SVs)” (boundary points).
    1. But non-SV data still plays a role in determining the SVs.

Linear SVM - KEY POINTs

1. Support Vector Machines (SVMs) work by detecting decision boundaries that maximally separate (+) from (-) data points.

2. Boundaries are determined by “Support Vectors (SVs)” (boundary points).
    1. But non-SV data still plays a role in determining the SVs.

3. SVM ***outperforms*** Logistic Regression (LR) for (nearly) separable data.

Linear SVM - KEY POINTs

1. Support Vector Machines (SVMs) work by detecting decision boundaries that maximally separate (+) from (-) data points.

2. Boundaries are determined by “Support Vectors (SVs)” (boundary points).
    1. But non-SV data still plays a role in determining the SVs.

3. SVM ***outperforms*** Logistic Regression (LR) for (nearly) separable data.

4. SVMs do not generate probabilities (as with LR).

Linear SVM - KEY POINTs

1. Support Vector Machines (SVMs) work by detecting decision boundaries that maximally separate (+) from (-) data points.

2. Boundaries are determined by “Support Vectors (SVs)” (boundary points).
    1. But non-SV data still plays a role in determining the SVs.

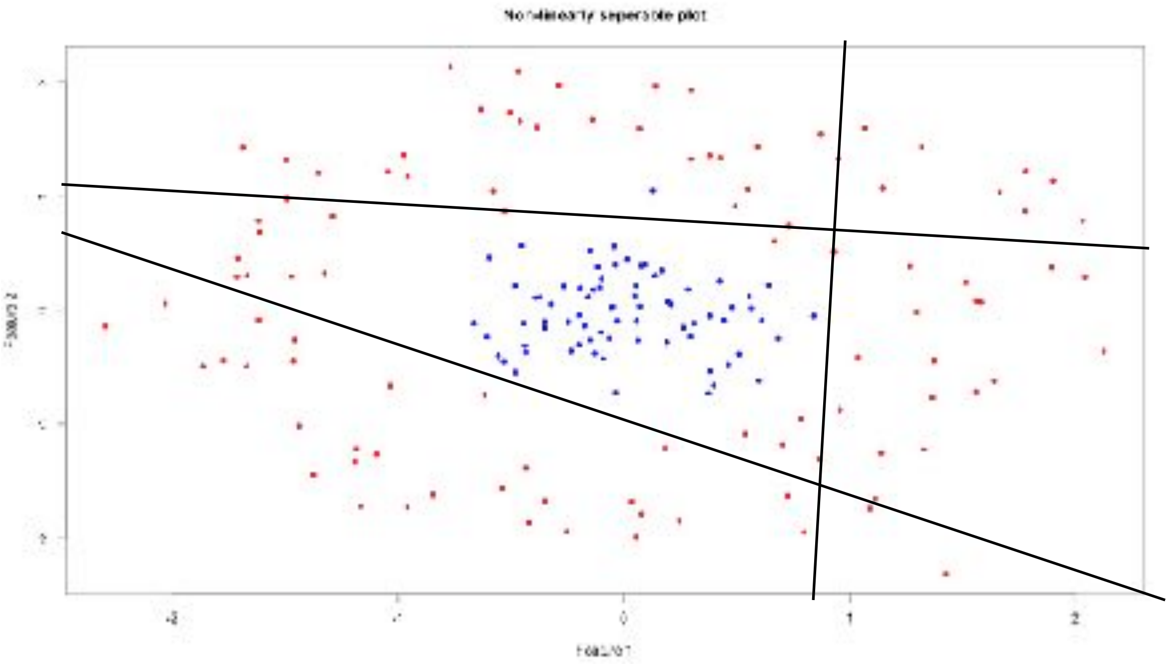
3. SVM ***outperforms*** Logistic Regression (LR) for (nearly) separable data.

4. SVMs do not generate probabilities (as with LR).

5. Linear SVMs are tuned by one parameter (C) to allow for “soft SV margins.”
    1. Compare with lambda ($1/\lambda$) for Lasso or Ridge penalties in LR.
    2. Can be used for feature selection in the Linear SVM case.

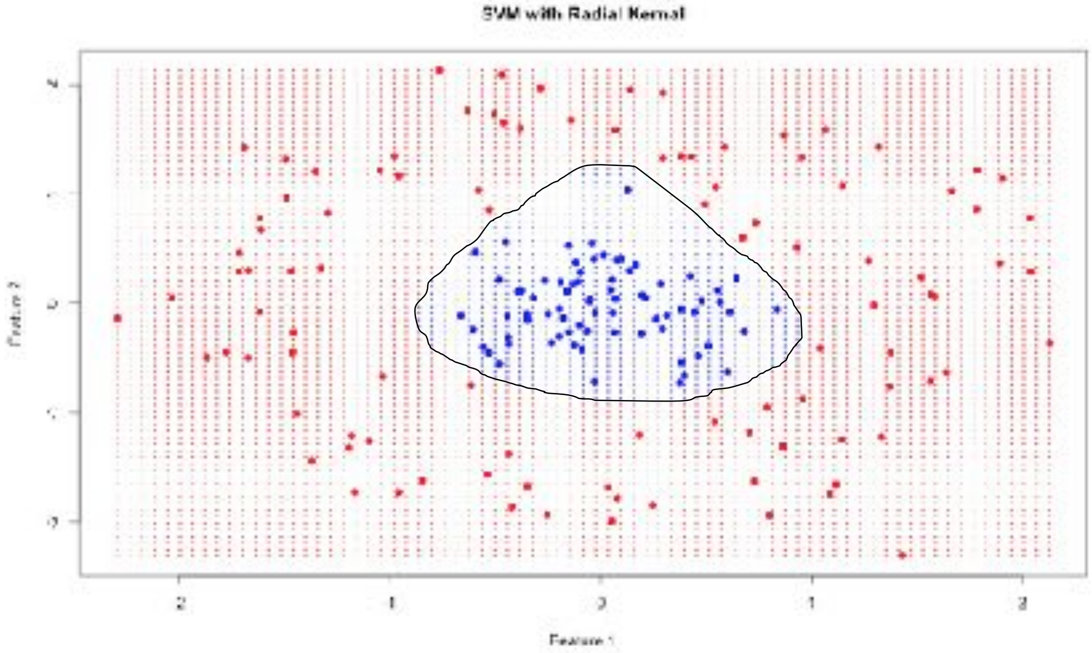
Non-linear SVM

- What if the (+) and (-) data cannot be separated by a single hyperplane?



Non-linear SVM

- In these cases we use SVMs with a ***Non-linear Kernal***

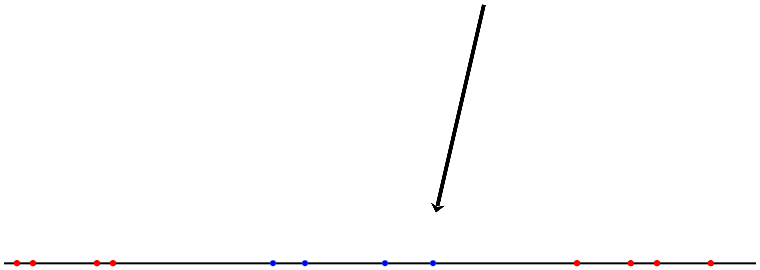


Non-linear SVM

- What is the ***Kernel Trick***?

  - Consider data distributed in 1D…

(+) and (-) data in a 1-d space

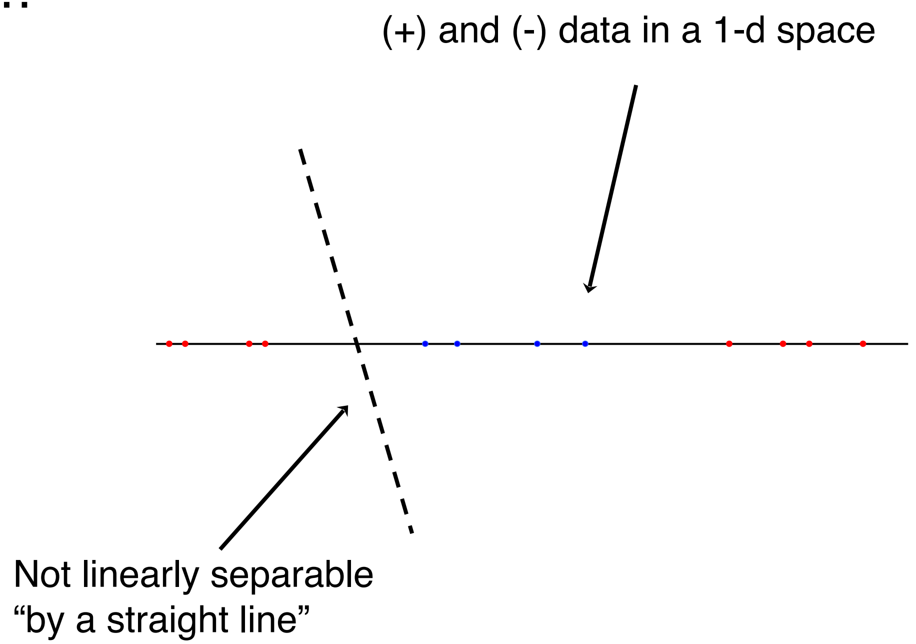


Non-linear SVM

- What is the ***Kernel Trick***?

  - Consider data distributed in 1D…
  - Not separable using terms dependent on "linear" features

    e.g. $x_1, \ldots$

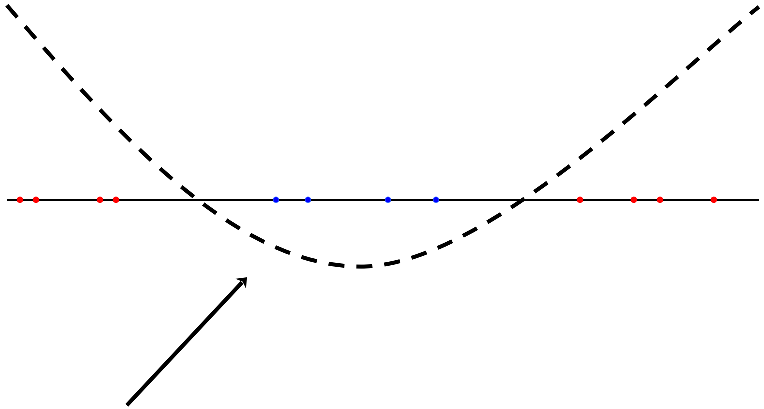


Non-linear SVM

- What is the ***Kernel Trick***?

  - Consider data distributed in 1D…
  - Not separable using terms dependent on "linear" features
    e.g. $x_1, \ldots$

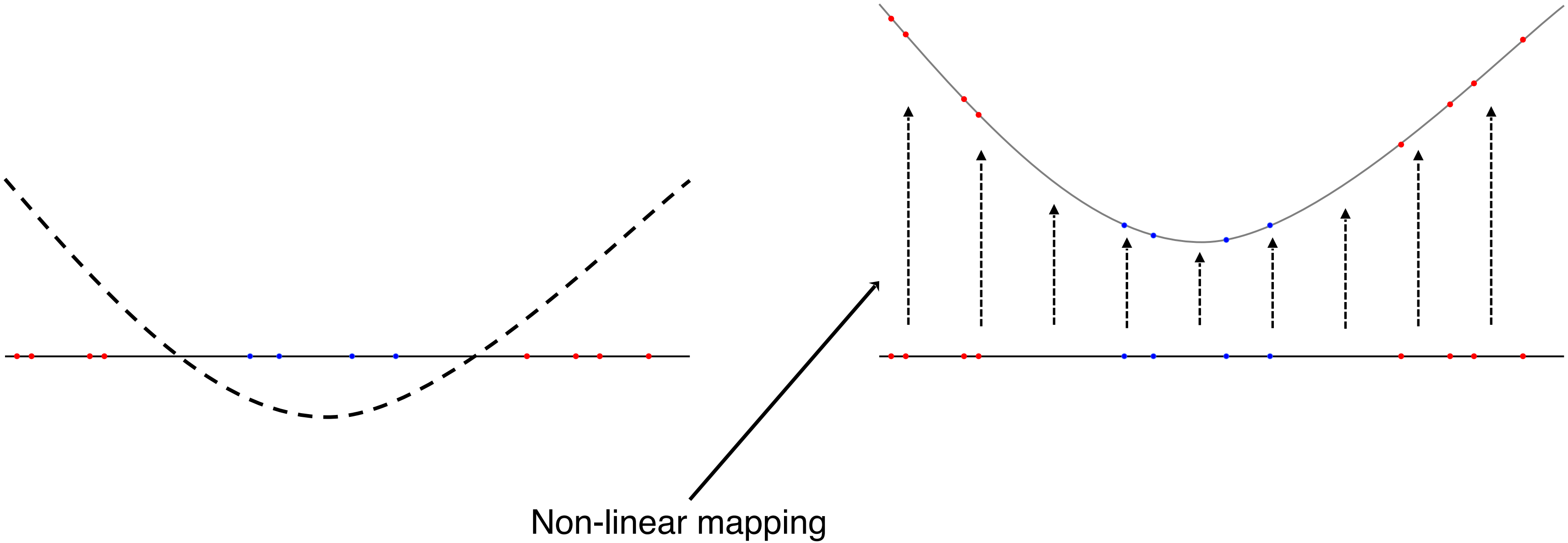
  - However, (+) and (-) points may still be separated by "non-linear" terms:
    e.g. $x_1, x_1^2, x_1^3, \ldots$



Data is separable by a "parabolic dependency on x_1^2"

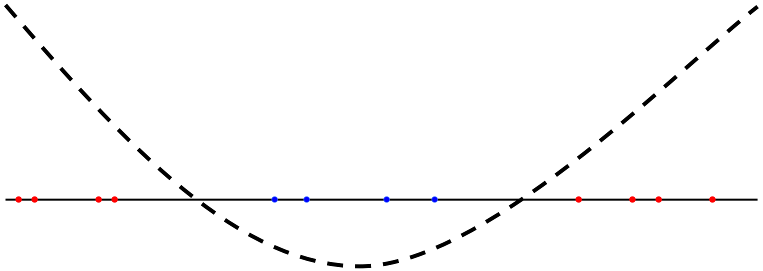
Non-linear SVM

- What is the ***Kernel Trick***?

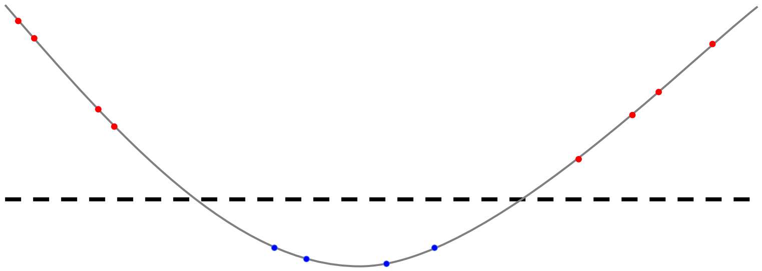


Non-linear SVM

- What is the ***Kernel Trick***?



Non-linear separator defined for the original feature space



Linear separator defined for a ***non-linearly related*** new space

Non-linear SVM

- What is the ***Kernel Trick***?

Steps for non-linear separation:

•Step 1: Map the original dimensions $x_1, x_2, x_3,\ldots$ into a **higher dimensional** feature space

$x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, \ldots$

- (New feature space not linearly dependent on the original space.)

•Step 2: Define a decision boundary in terms of the new higher dimensional feature space

•Step 3: New data points can be defined by mapping to the new higher dimensional feature space and comparing with the high-D boundary

Non-linear SVM

- What is the ***Kernel Trick***?

Classifiers: Logistic Regression, Linear-SVM, Decision Trees*, …

- PROBLEM:
  - The count of features $|\{ x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, \dots \}|$ in the high-D space can grow too quickly with the dimension of the base space .
- With SVM's, non-linear Kernels allow us to skip directly mapping to the high-D space for classification,
  - Significantly reduces the computational complexity.
- That's why it is called the "Kernal Trick"

A little More math…

- How does the ***Kernel Trick*** avoid high-D computation?
  - Recall SVMs are a “constrained optimization” problem:

$$\max_{\|\vec{\beta}\| \leq 1} M$$

Such that for all $(\boldsymbol{x}_j, y_j)$:

$$y_j \left(\sum_{i=0}^{n} x_{ji} \beta_i \right) \geq M$$

A little More math…

- How does the ***Kernel Trick*** avoid high-D computation?
  - Recall SVMs are a "constrained optimization" problem:

$$\max_{\|\vec{\beta}\|\leq 1} M$$

Such that for all $(\boldsymbol{x}_j, y_j)$:

$$y_j\left(\sum_{i=0}^{n} x_{ji}\beta_i\right) \geq M$$

$$\longrightarrow$$

  - Convex optimization in the "dual form":

$$\max_{\vec{\alpha}} 2\sum_i \alpha_i + \sum_{i,j} \alpha_i\alpha_j y_i y_j \left(\sum_k x_{jk}x_{ik}\right)$$

  - In vector form:

$$\max_{\vec{\alpha}} 2\|\vec{\alpha}\|_1 + \sum_{i,j} \alpha_i\alpha_j y_i y_j \langle \vec{x}_j, \vec{x}_i \rangle$$

A little More math…

- How does the ***Kernel Trick*** avoid high-D computation?
  - By replacing the inner product with a non-linear Kernel $K(\vec{x}_j, \vec{x}_i)$.
    - The decision boundary generated can be non-linear
    - Avoids direct calculations in the higher dimensional features space $x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, \ldots$, significantly saving on memory and computation cost.

$$\max_{\vec{\alpha}} 2\|\vec{\alpha}\|_1 + \sum_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{\langle \vec{x}_j, \vec{x}_i \rangle}_{\substack{\text{Euclidian “Inner product,”} \\ \text{a } \textbf{\textit{linear}} \text{ kernal}}} \qquad \max_{\vec{\alpha}} 2\|\vec{\alpha}\|_1 + \sum_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{K(\vec{x}_j, \vec{x}_i)}_{\textbf{\textit{Non-linear}} \text{ Kernal}}$$

Non-linear SVM

- Popular SVM Kernels:
  - Linear Kernel:

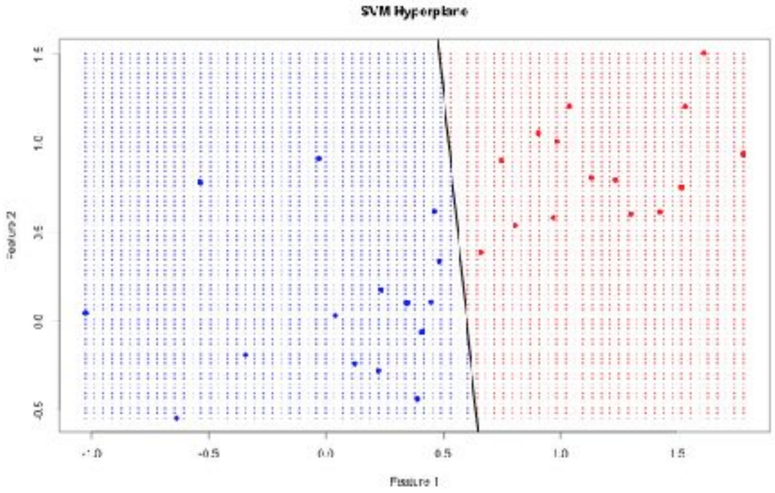
$$K(\vec{x}_j, \vec{x}_i) = \langle \vec{x}_j, \vec{x}_i \rangle + c$$

Non-linear SVM

- Popular SVM Kernels:
  - Linear Kernel:

$$K(\vec{x}_j, \vec{x}_i) = \langle \vec{x}_j, \vec{x}_i \rangle + c$$

Linearly separable data



Non-linear SVM

- Popular SVM Kernels:
  - Linear Kernel:

$$K(\vec{x}_j, \vec{x}_i) = \langle \vec{x}_j, \vec{x}_i \rangle + c$$

$$K(\vec{x}_j, \vec{x}_i) = \left(b\langle \vec{x}_j, \vec{x}_i \rangle + c\right)^d$$

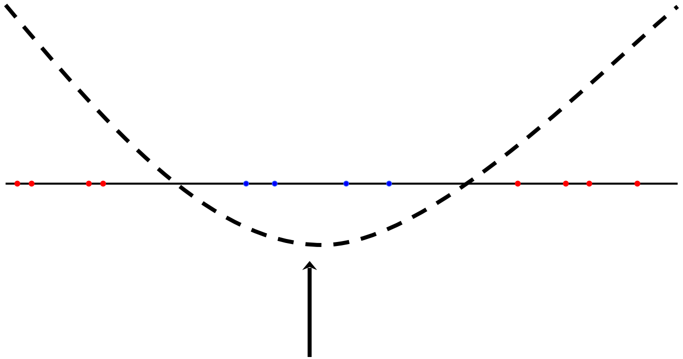
Non-linear SVM

- Popular SVM Kernels:
  - Linear Kernel:

$$K(\vec{x}_j, \vec{x}_i) = \langle \vec{x}_j, \vec{x}_i \rangle + c$$

$$K(\vec{x}_j, \vec{x}_i) = \left(b\langle \vec{x}_j, \vec{x}_i \rangle + c\right)^d$$

Data separable by a decision boundary with a ***finite*** d



parabolic decision boundary (d=2)

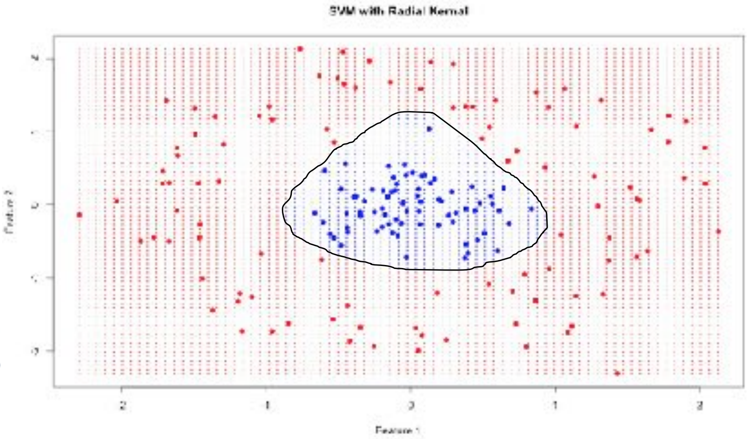
Non-linear SVM

- Popular SVM Kernels:
  - Linear Kernel:

$$K(\vec{x}_j, \vec{x}_i) = \langle \vec{x}_j, \vec{x}_i \rangle + c$$

$$K(\vec{x}_j, \vec{x}_i) = \left(b\langle \vec{x}_j, \vec{x}_i \rangle + c\right)^d$$

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



Data separable by a decision boundary with possibly unbounded degrees of freedom.

Non-linear SVM

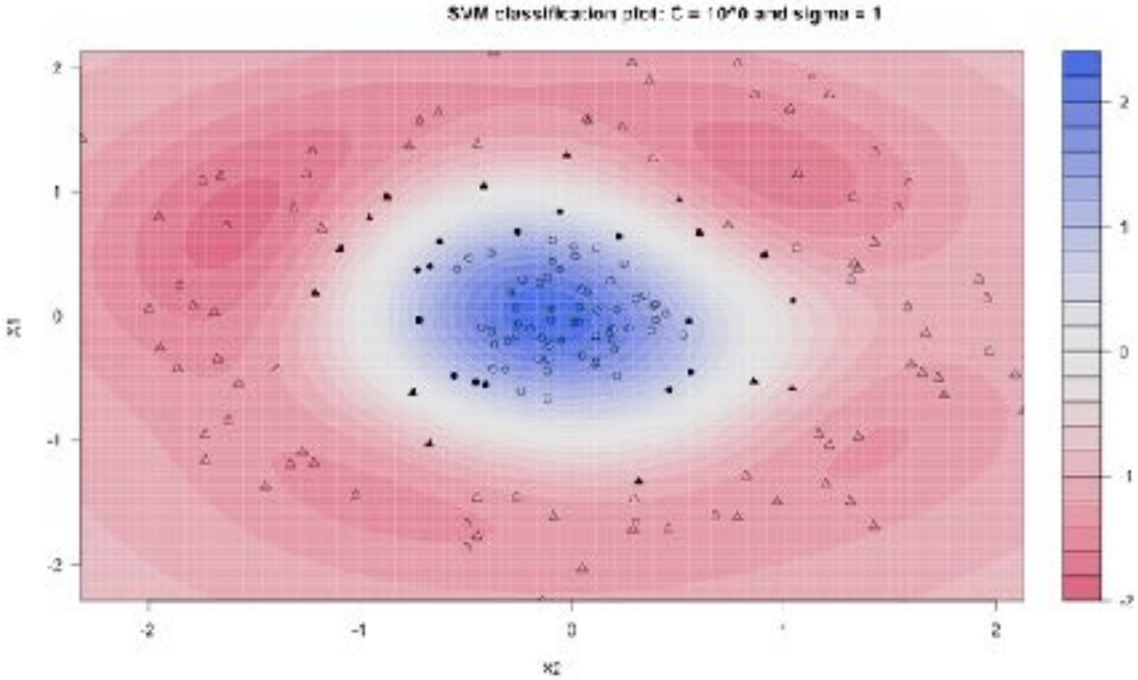
- Gaussian Kernels Key points:

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

  - Very popular non-linear Kernel
  - Gaussian Radial Kernels will outperform Polynomial Kernel for most data sets.
  - Often computationally cheaper.
  - In practice, Data Scientist will check linear and Gaussian Kernels when trying SVMs.
  - The σ parameter in K must also be tuned for Gaussian Kernel SVMs…

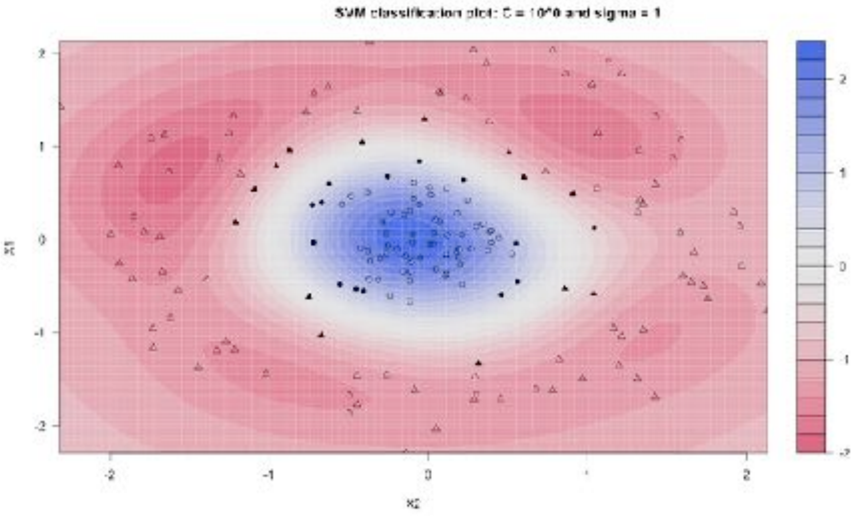
Gaussian-Kernel tuning

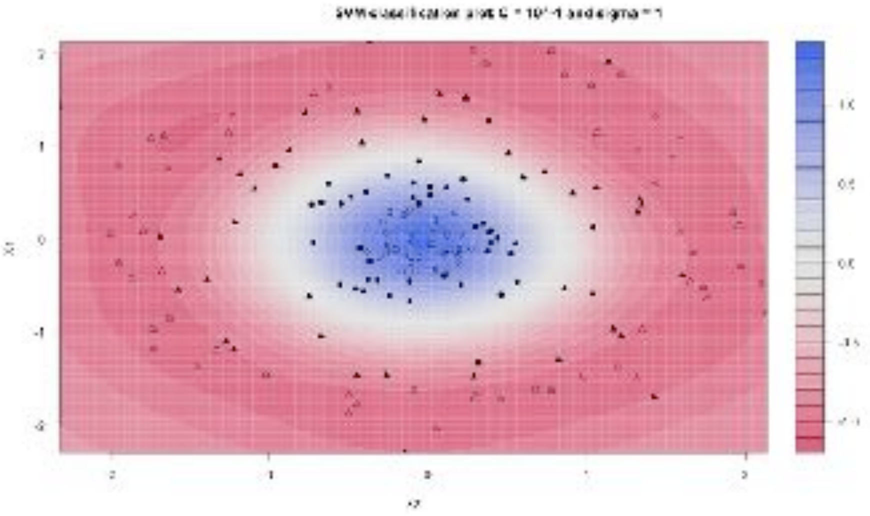
- Gaussian Kernel tuning example:



Gaussian-Kernel tuning

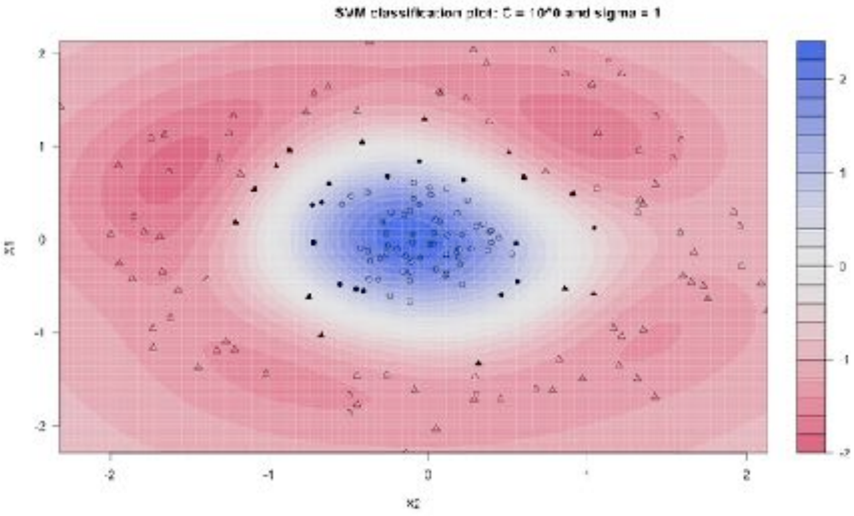
- Under-fitting C can lead to **Generalization Bias**

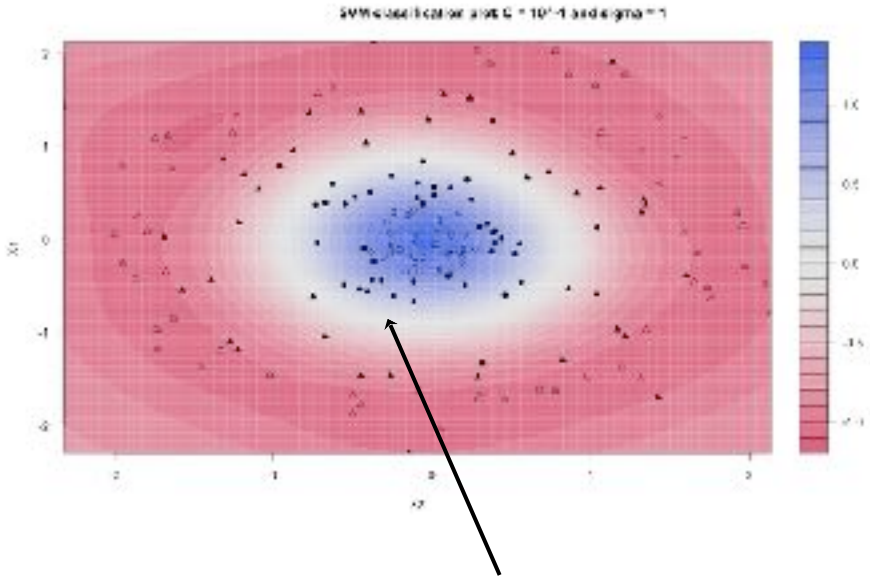




Gaussian-Kernel tuning

- Under-fitting C can lead to **Generalization Bias**

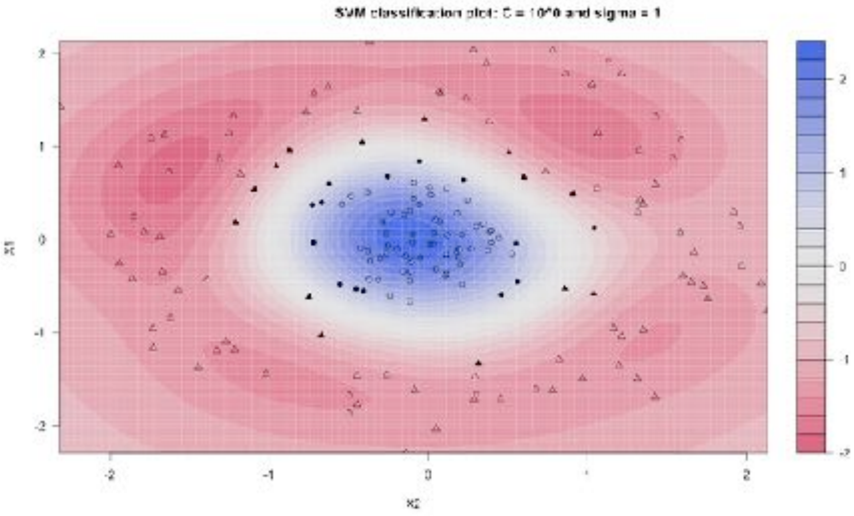


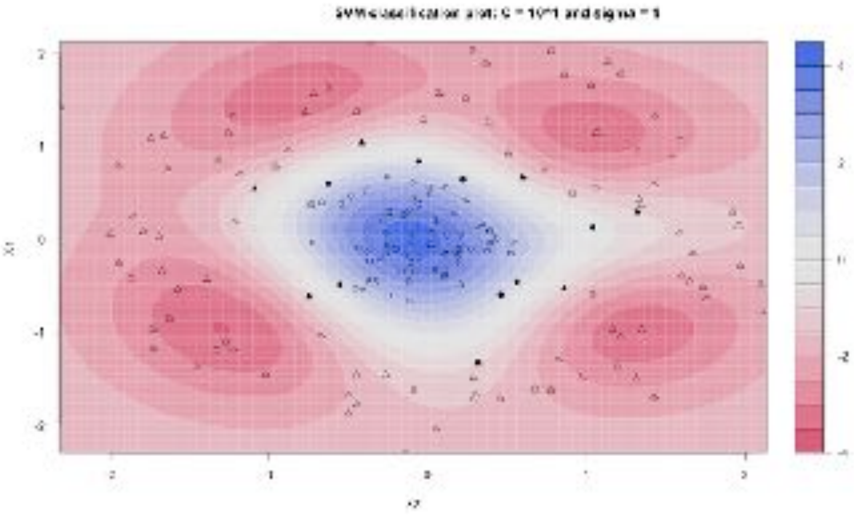


Lower C value means wider margins so ***more*** support vectors

Gaussian-Kernel tuning

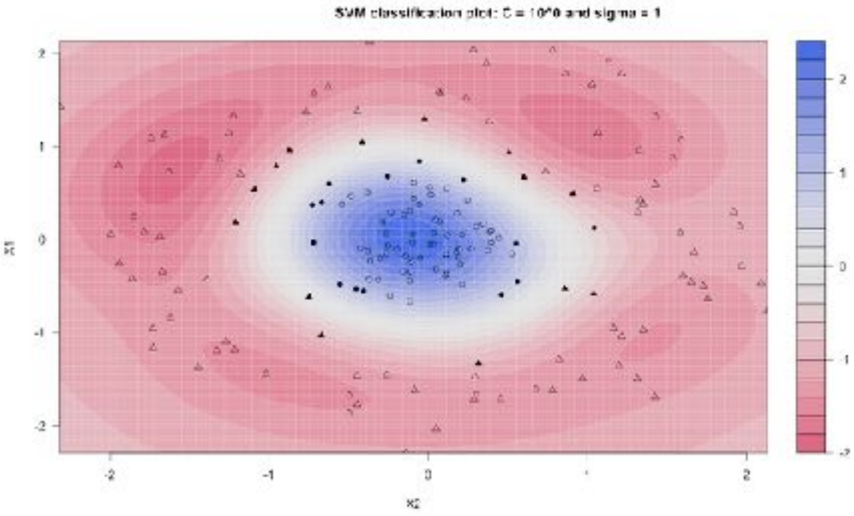
- Over-fitting C can lead to **Generalization Variance**

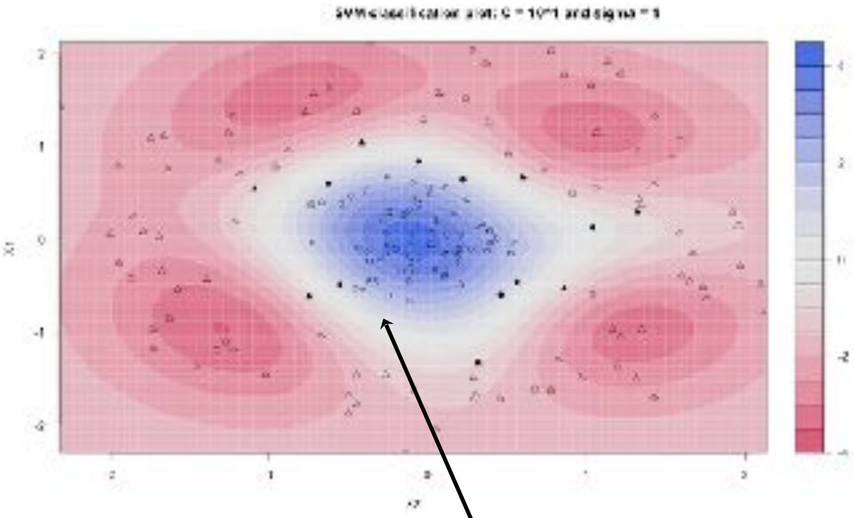




Gaussian-Kernel tuning

- Over-fitting C can lead to **Generalization Variance**

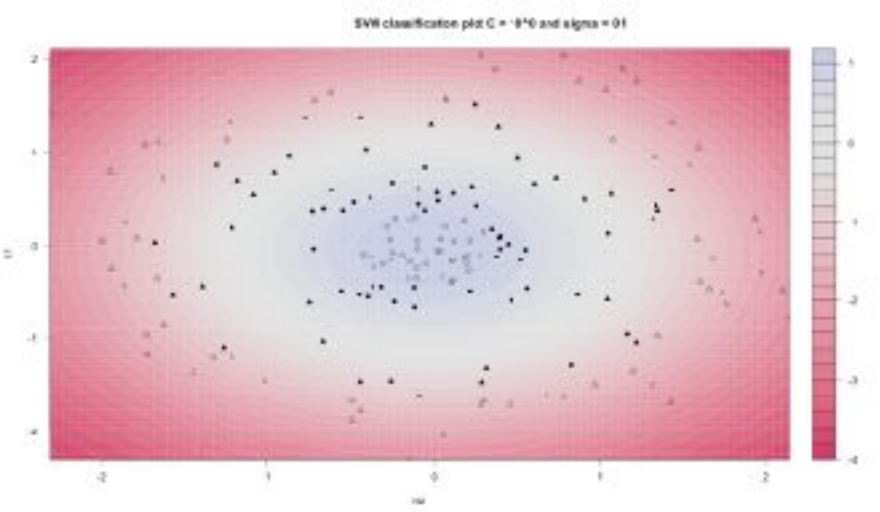


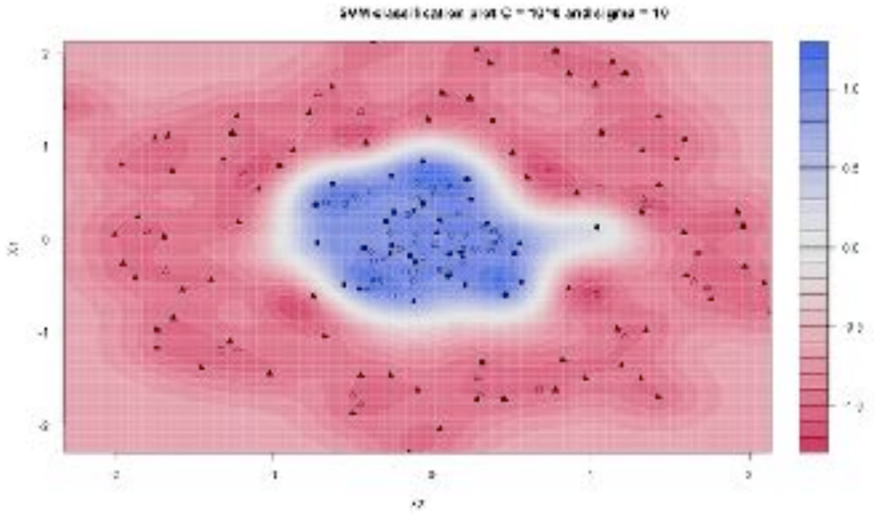


Higher C value can mean missing support vectors

Gaussian-Kernel tuning

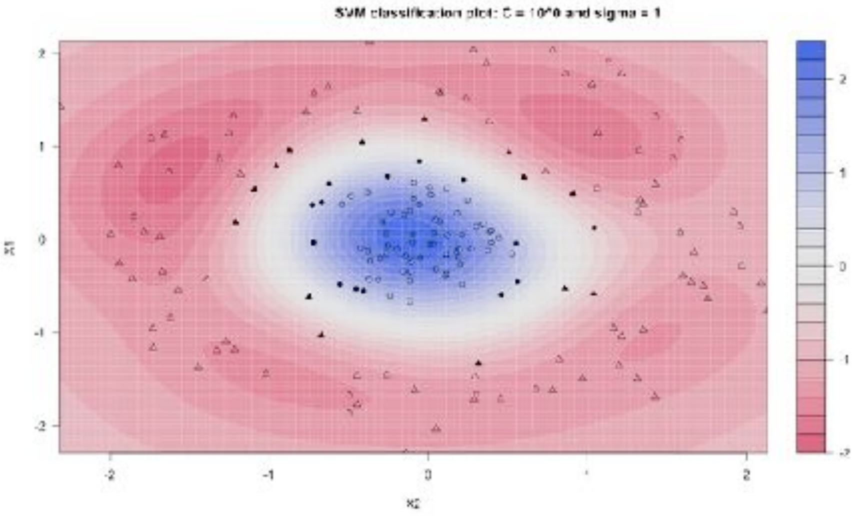
- Tuning sigma (σ) controls the “narrowness” of each point’s effect

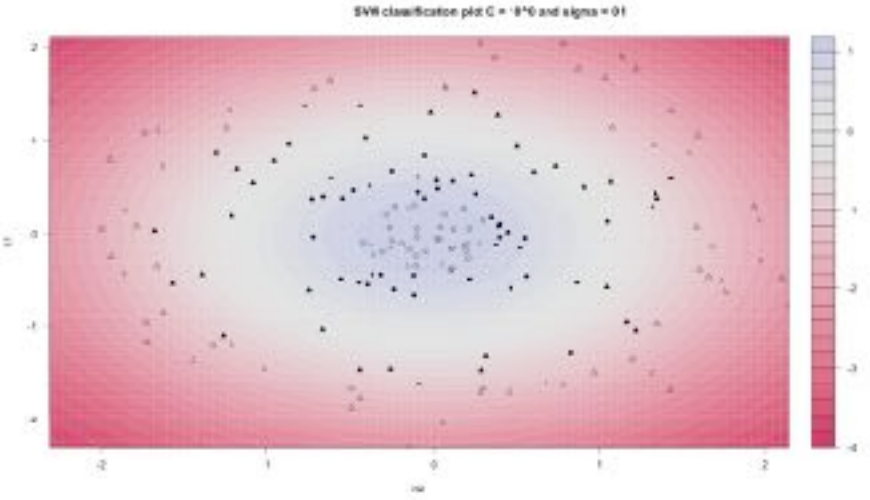




Gaussian-Kernel tuning

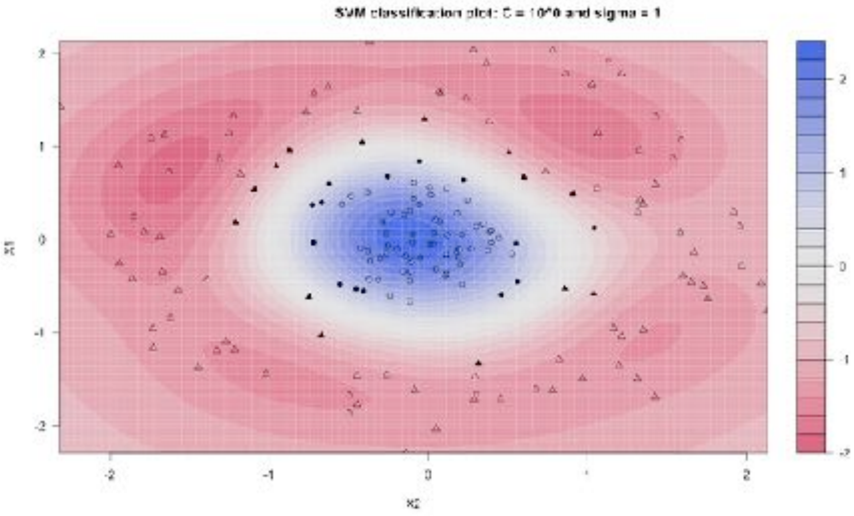
- Small σ (large γ) can lead to **Generalization Bias** (under-fitting)

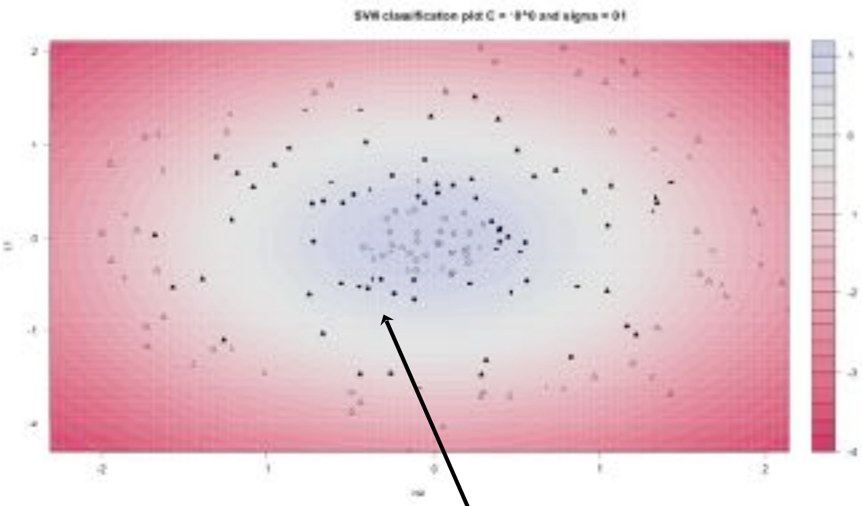




Gaussian-Kernel tuning

- Small σ (large γ) can lead to **Generalization Bias** (under-fitting)

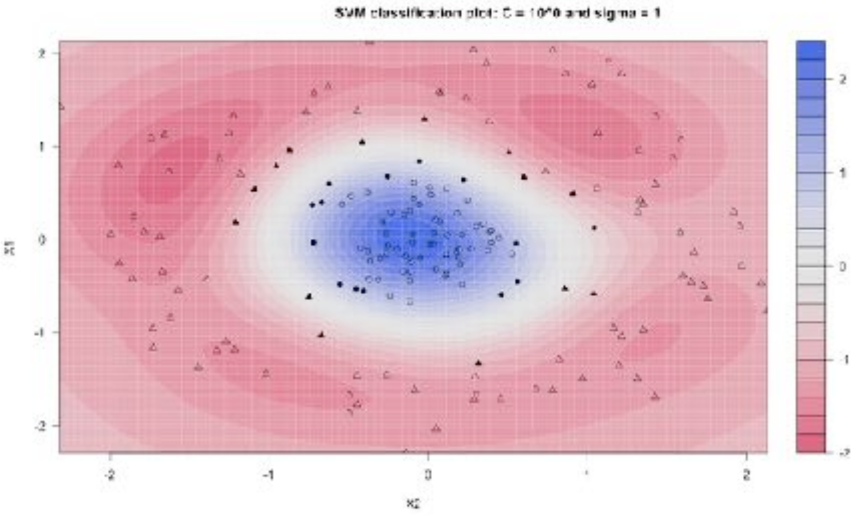


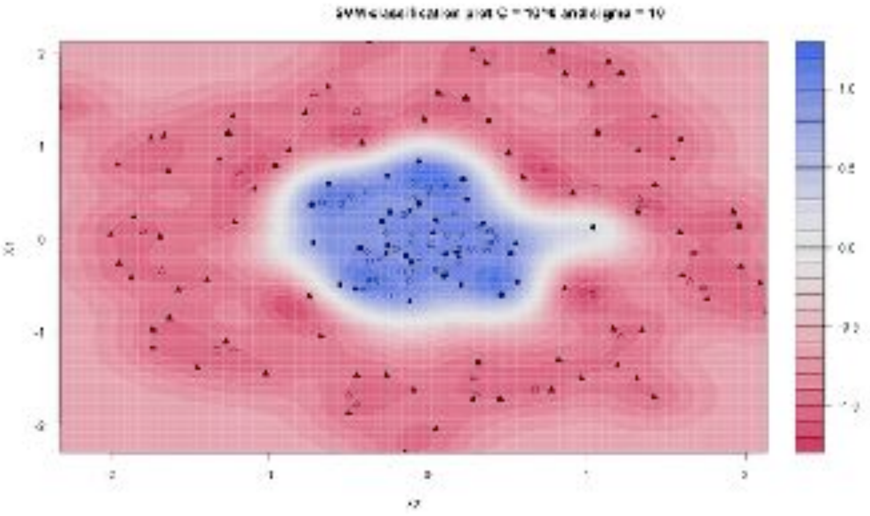


Lower σ value means the influence of each point is more dispersed.

Gaussian-Kernel tuning

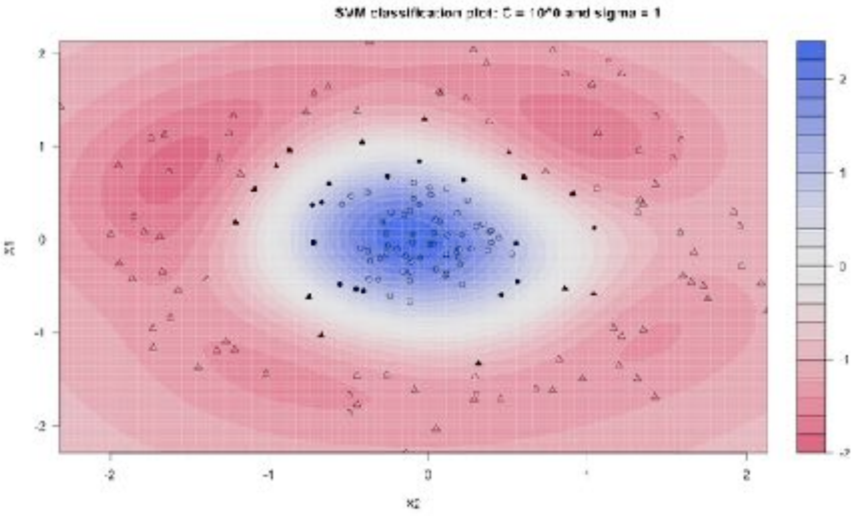
- Large σ (small γ) can lead to **Generalization Variance** (over-fitting)

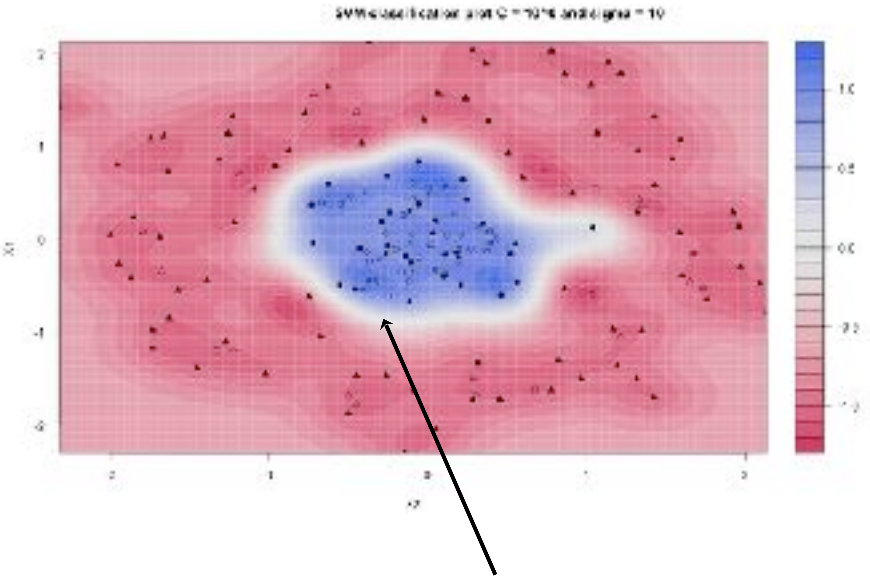




Gaussian-Kernel tuning

- Large σ (small γ) can lead to **Generalization Variance** (over-fitting)





Higher σ value means higher "peaks" generated by individual points.

Non-linear SVM - KEY POINTs

1. Non-linear Kernel SVMs allow for decision boundaries that do not have to be “straight.”

Non-linear SVM - KEY POINTs

1. Non-linear Kernel SVMs allow for decision boundaries that do not have to be “straight.”
2. Logistic Regression (LR) can do non-linear decision boundaries, but at high computation cost.

Non-linear SVM - KEY POINTs

1. Non-linear Kernel SVMs allow for decision boundaries that do not have to be “straight.”

2. Logistic Regression (LR) can do non-linear decision boundaries, but at high computation cost.

3. The ***kernel trick*** allows SVMs to have non-linear decision boundaries without paying the high computation cost of an explosively high-D feature space.

Non-linear SVM - KEY POINTs

1. Non-linear Kernel SVMs allow for decision boundaries that do not have to be “straight.”

2. Logistic Regression (LR) can do non-linear decision boundaries, but at high computation cost.

3. The ***kernel trick*** allows SVMs to have non-linear decision boundaries without paying the high computation cost of an explosively high-D feature space.

4. Linear SVMs are tuned by two parameters C and σ:
    1. C is used to tune the “soft margins” for determining Support Vectors.
        1. C ***cannot*** be used be used for feature selection with non-linear kernels.
    2. σ (or γ) is used to tune how disperse the effect of individual data points are in influencing the decision boundaries.