Zurich University
of Applied Sciences

Machine Learning
V04: Support Vector Machines

Basic idea
Mathematical definition and optimization
Kernels
More than two classes

Based on material from Oliver Dürr
See also [ISL, 2014, ch. 9] and [ESL, 2009, ch. 12]

Zurich University of Applied Sciences and Arts
InIT Institute of Applied Information Technology (stdm)

Educational objectives

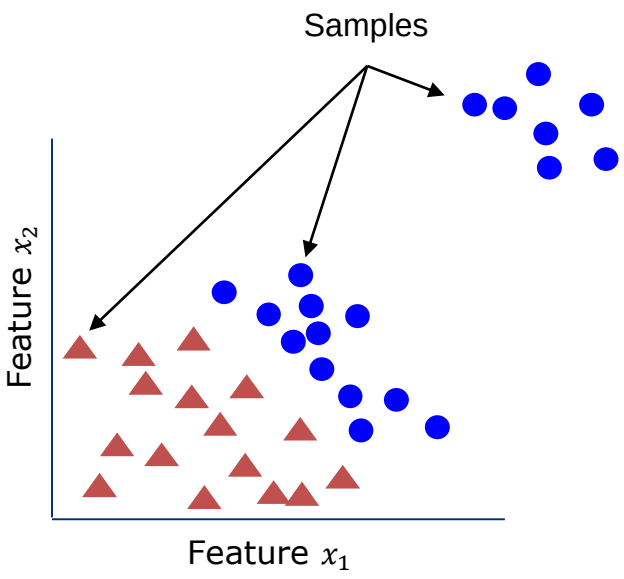
- **Know** the **Support Vector Machine** method in some detail
  - **Explain** the progression from **maximal margin classifier → support vector classifier → SVM**
  - **Explain** the workings of the C and γ parameters

- **Use** **SVM** successfully **on** tutorial-style **examples**, **including** parameter **grid search**

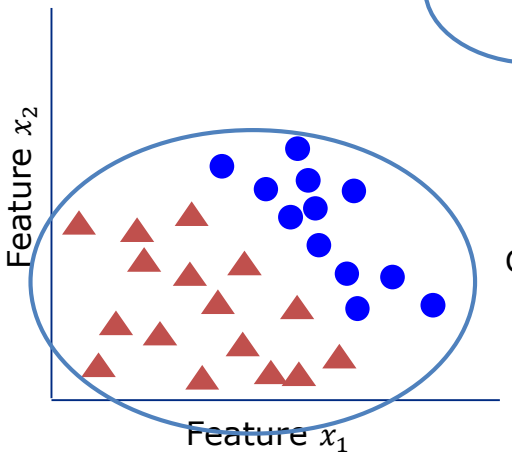
Zurich University
of Applied Sciences

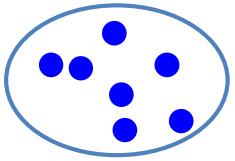
1. BASIC IDEA

Recap: Unsupervised vs. supervised learning



Unsupervised Learning

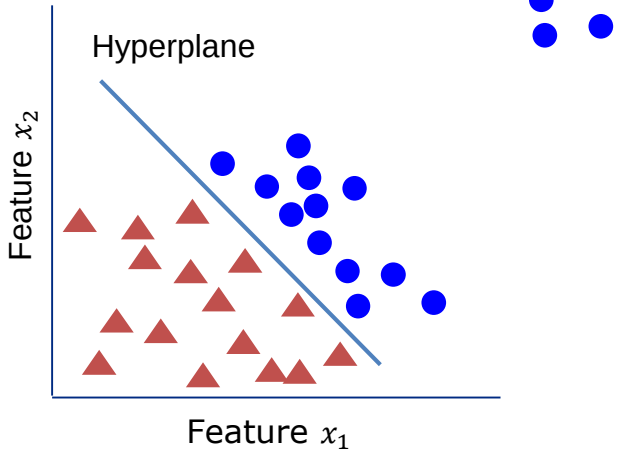




Cluster 1

Cluster 2

Supervised Learning



Probably a misnomer for SV “algorithm”

Zurich University of Applied Sciences

Support Vector Machine (SVM)
Basics & notation

Representation of SVM
- Each observation is a vector of values (p-dimensional)
- SVM **constructs a hyperplane** to separate class members

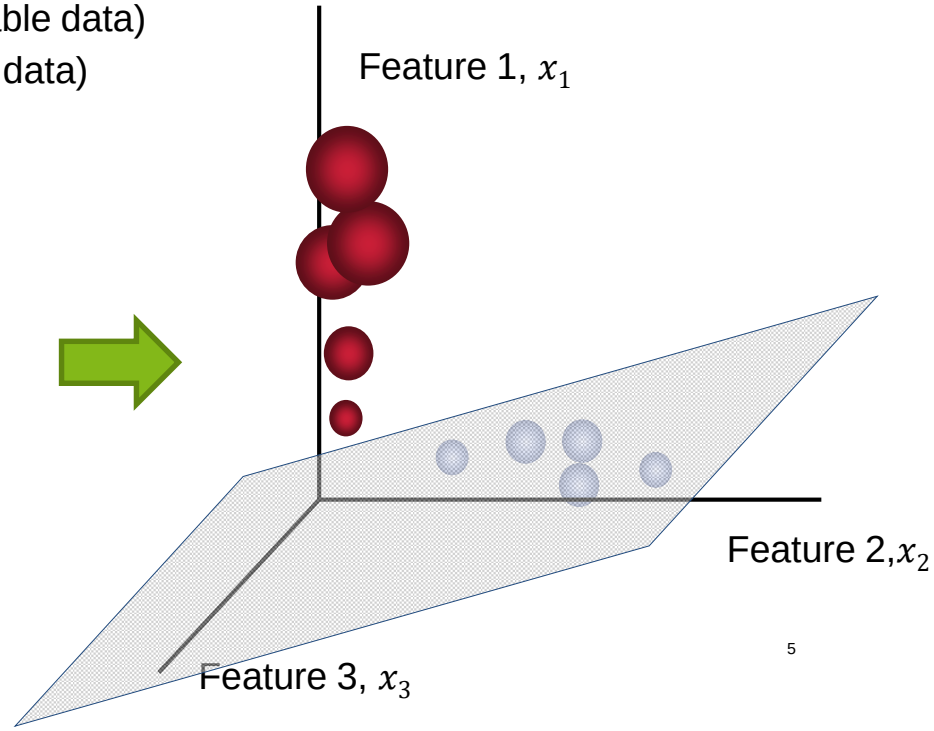
SVM family of classifiers
1. (Maximal margin) **hyperplane** classifier (for linearly separable data)
2. Support vector **classifier** (for almost linearly separable data)
3. Support vector **machine** (for non-linearly separable data)

Feature vectors being **column vectors** here (as opposed to row vectors usually) makes the math more convenient later

N observations

p features

	1	2	3	4	5	6	7	8	9	10
1	2	1	3	4	4	5	7	6	6	3
2	4	2	6	8	10	2	1	2	1	1
3	3	1	5	6	6	2	1	2	2	1
	Class 1					Class 2				

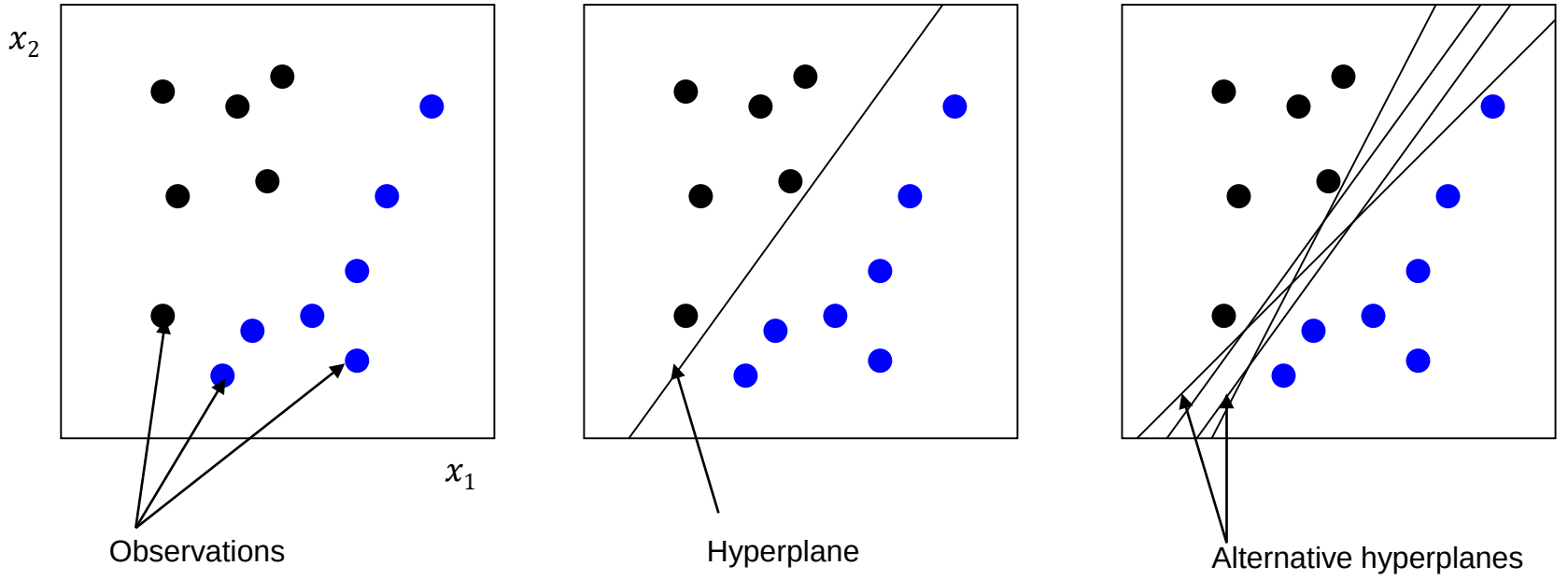


(1) Hyperplane classifiers

Assumption (until further notice): data is linearly separable

Which hyperplane to construct / chose?

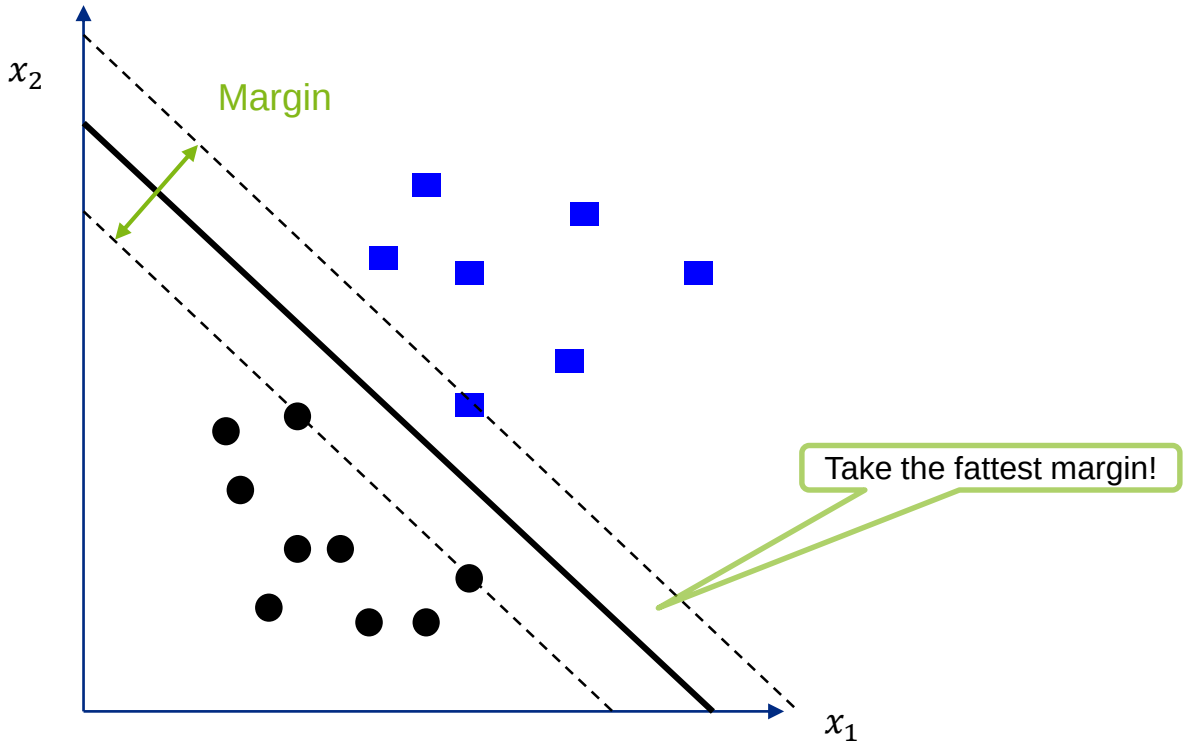
- **Many candidates** could potentially separate the data (given it is linearly separable)



Maximal margin classifier

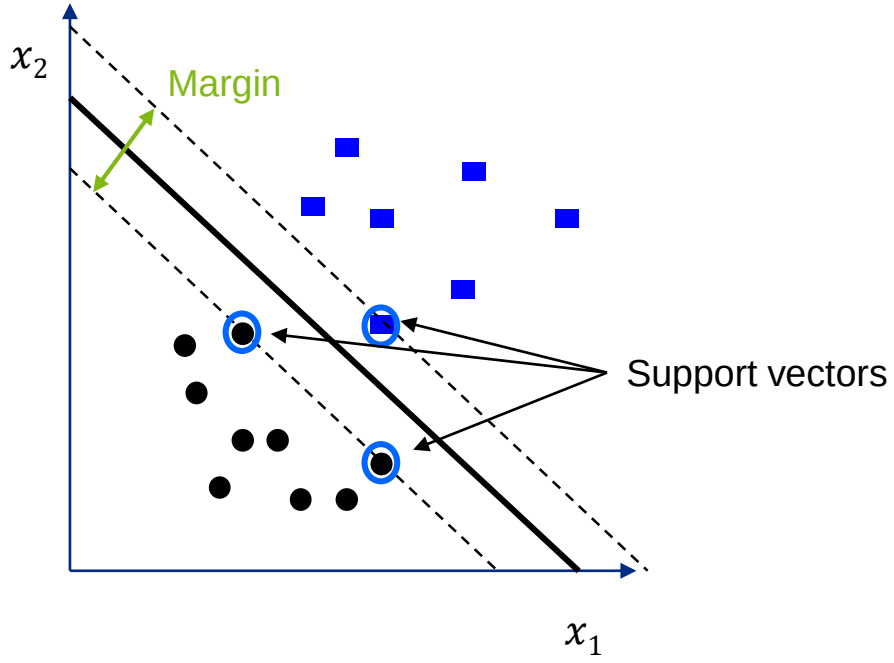
Ideally, choose a specific hyperplane

- The one **maximizing** the **distance from** the **hyperplane to closest** training **point** (on each side)
- Called the maximal margin hyperplane
- Can be represented as a linear combination of only few training points



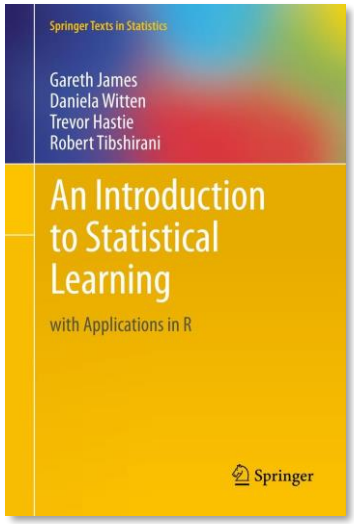
Support vectors
…for the maximum margin classifier

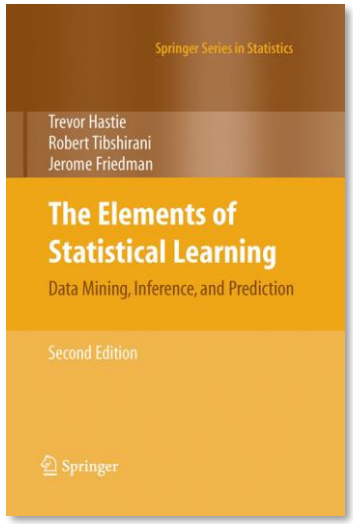
- Close (to the hyperplane) training examples determine the hyperplane
  (➔ Far away training examples do not participate in its specification)
- These examples are called the support vectors
  ➔ removing them would change the location of the separating hyperplane



2. MATHEMATICAL DEFINITION AND OPTIMIZATION

(Just a sketch → details in ISL ch. 9, ESL ch. 12)





Formal: Definition of a hyperplane

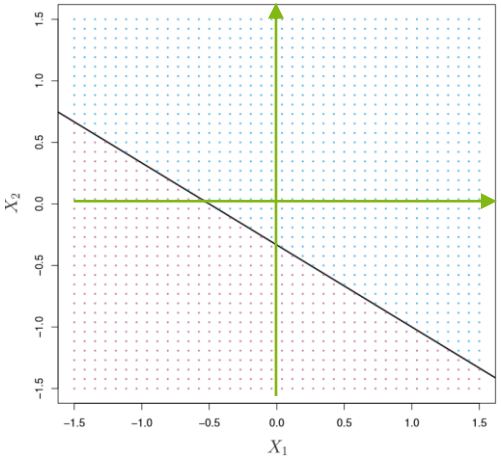
Points x satisfying this equation lie on the hyperplane

Definition of a hyperplane

- $\{\vec{x}: \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = 0\}$

Separating hyperplanes for **classes encoded as ± 1**

- $\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0$ if $y_i = 1$
- $\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0$ if $y_i = -1$



The hyperplane $1 + 2X_1 + 3X_2 = 0$. It divides the 2D space into two areas (classes): red and blue, depending on which side of the straight line a point lies. From ISL Fig. 9.1

More familiar notation:
$X_2 = -\frac{2}{3}X_1 - \frac{1}{3}$

➔ Equivalently: $y_i \cdot (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0$

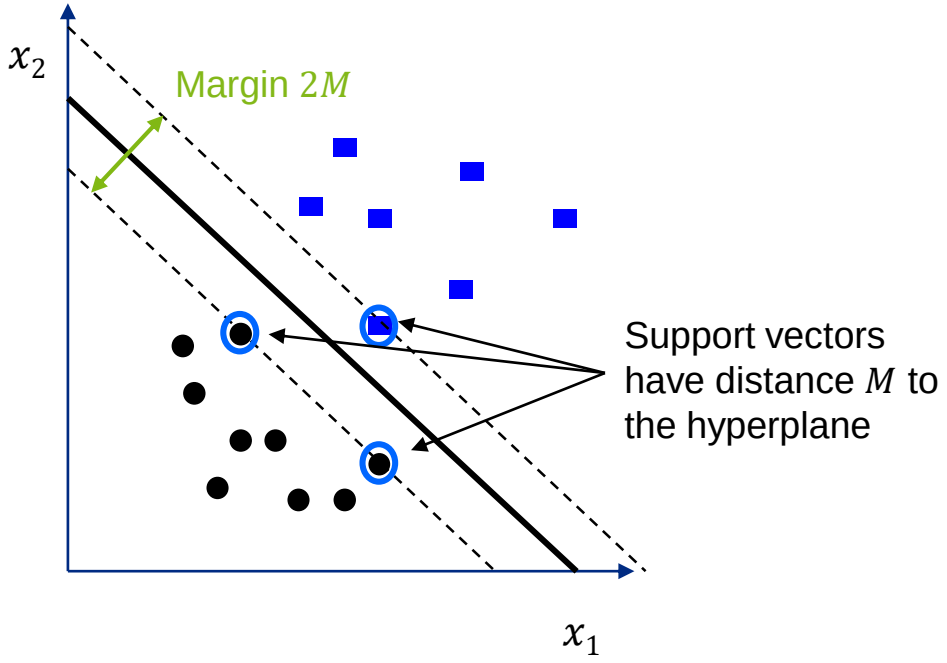
- If β is a **normal vector** (length 1: $\sum_{j=1}^{p} {\beta_j}^2 = 1$), evaluating the hyperplane formula for a point x gives it's **distance to the hyperplane**

Formal: Definition of optimization problem

Intuitive optimization

- $\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$,
  - subject to $\sum_{j=1}^{p} {\beta_j}^2 = 1$,
  - and $y_i \cdot \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M \; \forall i = 1..N$

➔ All vectors have at least distance M; support vectors have distance $= M$



Formal: Reformulation of optimization problem

Intuitive optimization

- $\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$,
  - subject to $\sum_{j=1}^{p} {\beta_j}^2 = 1$,
  - and $y_i \cdot \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M \; \forall i = 1..N$

An **optimization method**, together with quadratic programming used here to find the function's local extrema, subject to several constraints (think of a relative of batch gradient descent for linear regression).

…can be reformulated using Lagrange multipliers

- Technically, find $L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k {x_i}^T x_k$,
  - subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$
  - Once the α's are computed, $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$

Note: **Only** the **inner products** (dot- or scalar product) between feature vectors enter: this opens the door for "kernel trick" (→ see below)

(2) Towards linearly non-separable data

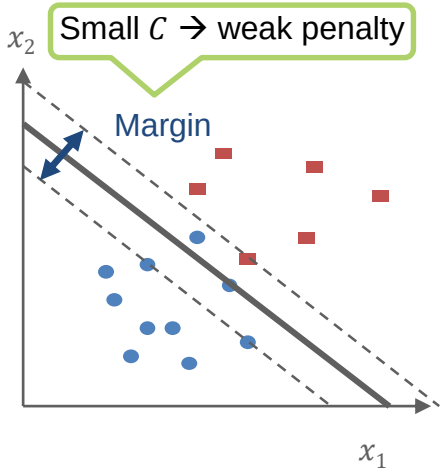
General idea of the C penalty factor on misclassifications

New assumption: Data is not linearly separable any more

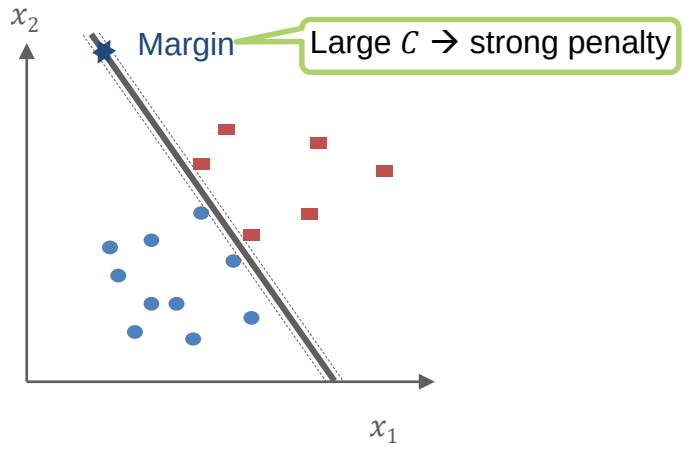
➔ Idea of soft margins that allows for ***some* misclassifications**

- Data may contain untypical or mislabeled samples;
  process might really be (moderately) non-linear
- The number of misclassifications is controlled by a *penalty factor* C

➔ Sometimes a larger margin is worth having some misclassified observations



Low penalty: high # of misclassified training examples



High penalty: low # of misclassified training examples

Formal: Support vector classifier optimization

Intuitive optimization

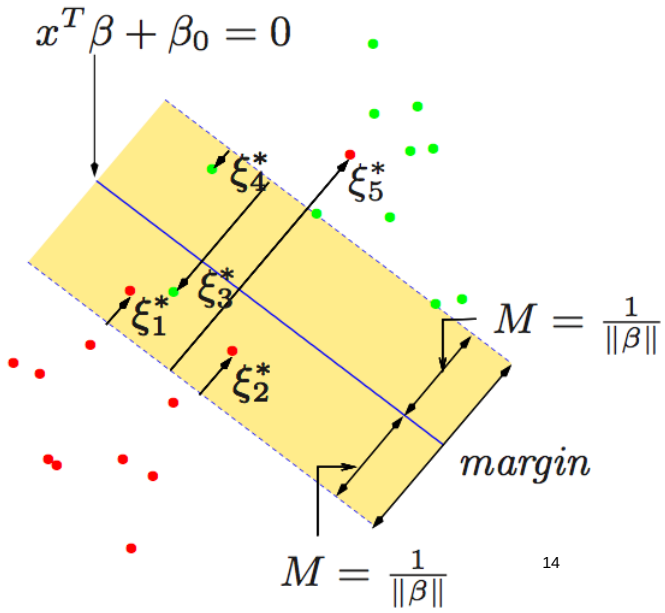
- $\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$,
  - subject to $\sum_{j=1}^{p} {\beta_j}^2 = 1$,
  - and $y_i \cdot \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1 - \xi_i)\ \forall i = 1..N$,
  - and $\xi_i \geq 0$, $\sum_{i=1}^{N} \xi_i \leq \tilde{C}$
  - where $\boldsymbol{\xi_i}$ is a **slack variable** introduced to **allow** instance i to lie on the **wrong side** of the margin

➔ There's a **total budget of $\tilde{C}$ for misclassifications** (distance from the hyperplane for wrong points)

Note the parameter name $\tilde{C}$ here: It is inversely related to the actual hyper parameter C below (i.e., $C \backsim \frac{1}{\tilde{C}}$). C is also exposed in the APIs of SVM implementations we will use (i.e., scikit-learn).

Equivalent technical optimization («dual form»)

- Maximize $L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k {x_i}^T x_k$,
  - subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$
  - Again, $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$ is minimized



The figure is taken from fig. 12.1 in "The Elements of Statistical Learning" (Springer, 2009)

Visualizing the effect of the C parameter

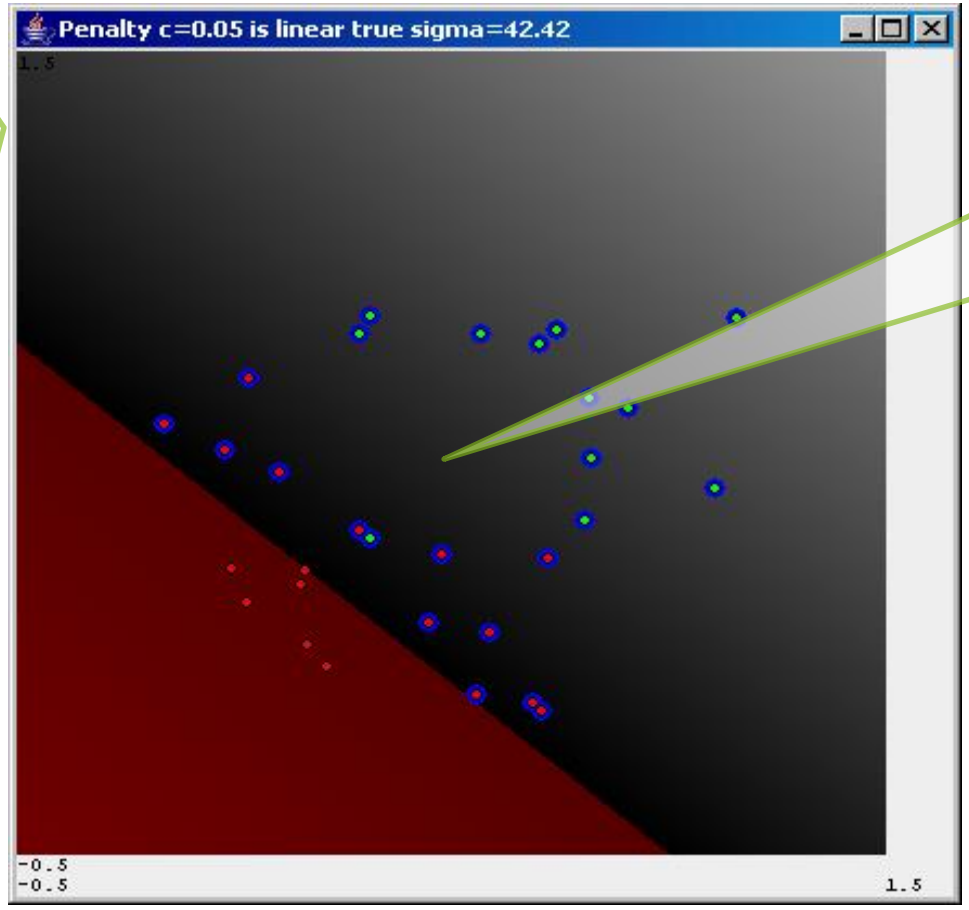
Penalty for misclassifications

White line:
the hyperplane (starts upper right corner)

Black lines:
Margins around hyperplane

Red/green dots:
training instances

Blue circled dots:
support vectors



Very low C, nearly no penalty for misclassifications

➔ Big margin

Let's increase C and see what happens.

Visualizing the effect of the C parameter

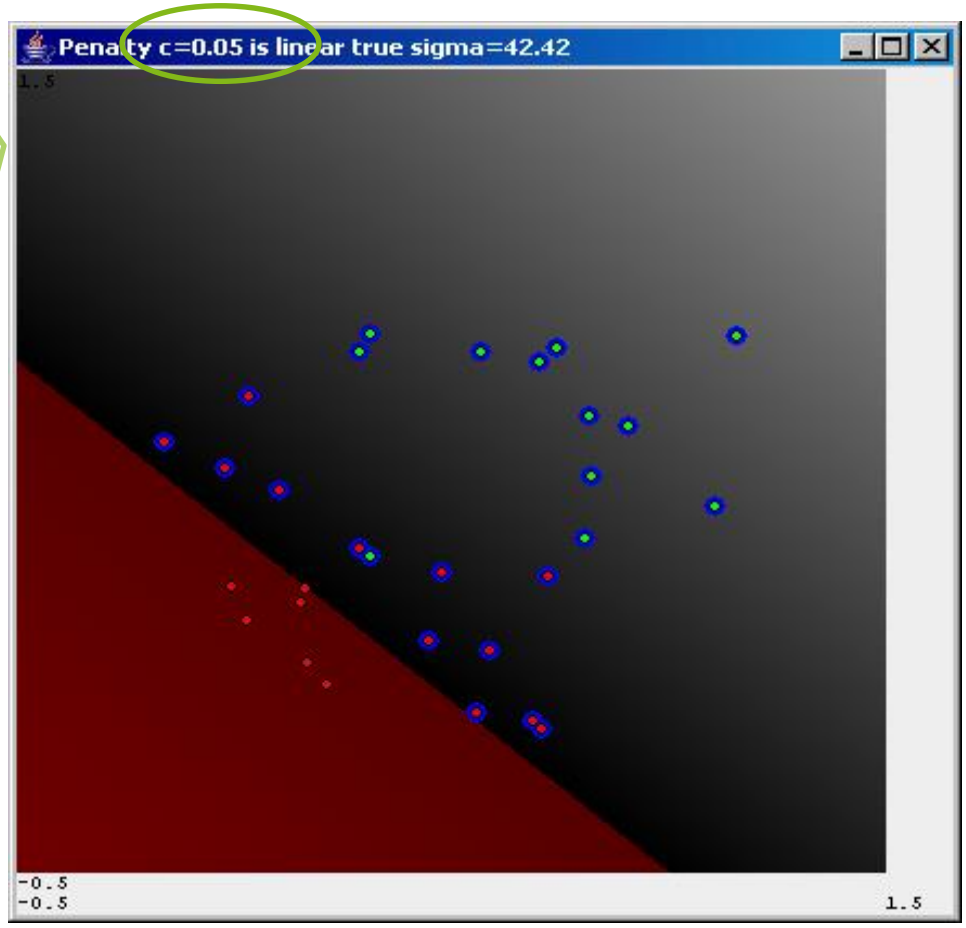
Penalty for misclassifications

White line:
the hyperplane (starts upper right corner)

Black lines:
Margins around hyperplane

Red/green dots:
training instances

Blue circled dots:
support vectors



Visualizing the effect of the C parameter

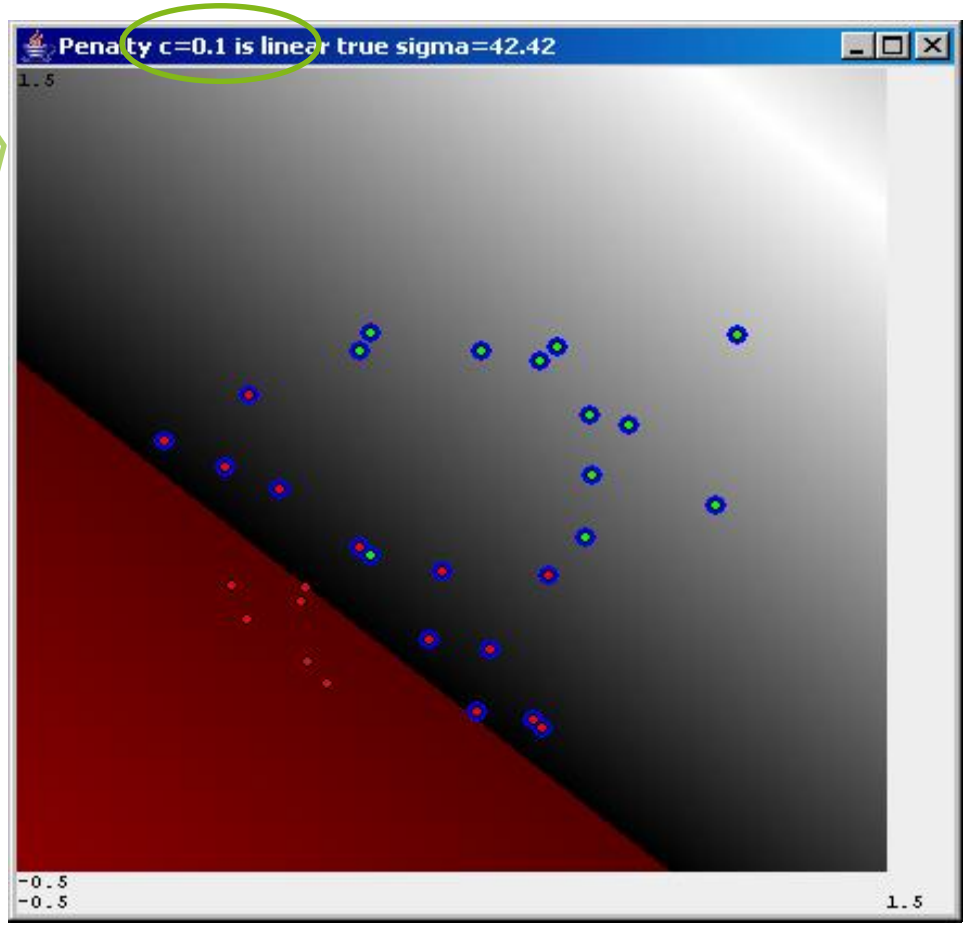
Penalty for misclassifications

White line:
the hyperplane (starts upper right corner)

Black lines:
Margins around hyperplane

Red/green dots:
training instances

Blue circled dots:
support vectors



Visualizing the effect of the C parameter

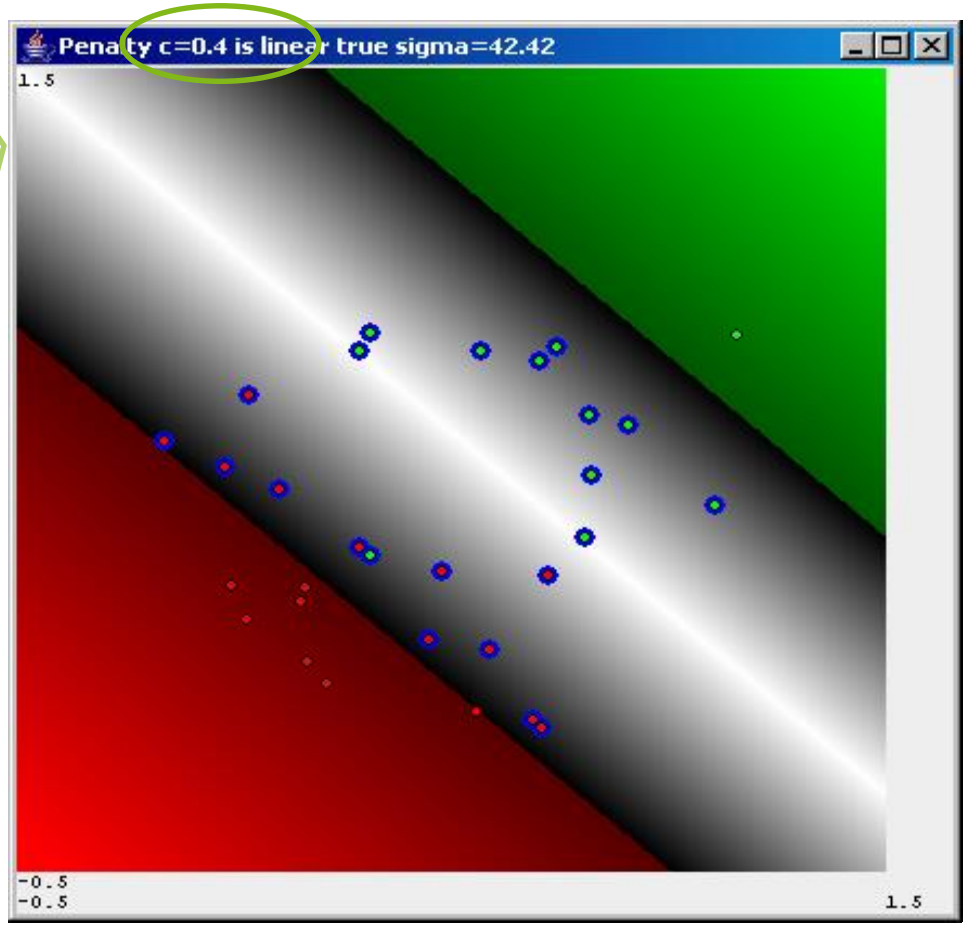
Penalty for misclassifications

White line:
the hyperplane (starts upper right corner)

Black lines:
Margins around hyperplane

Red/green dots:
training instances

Blue circled dots:
support vectors



Visualizing the effect of the C parameter

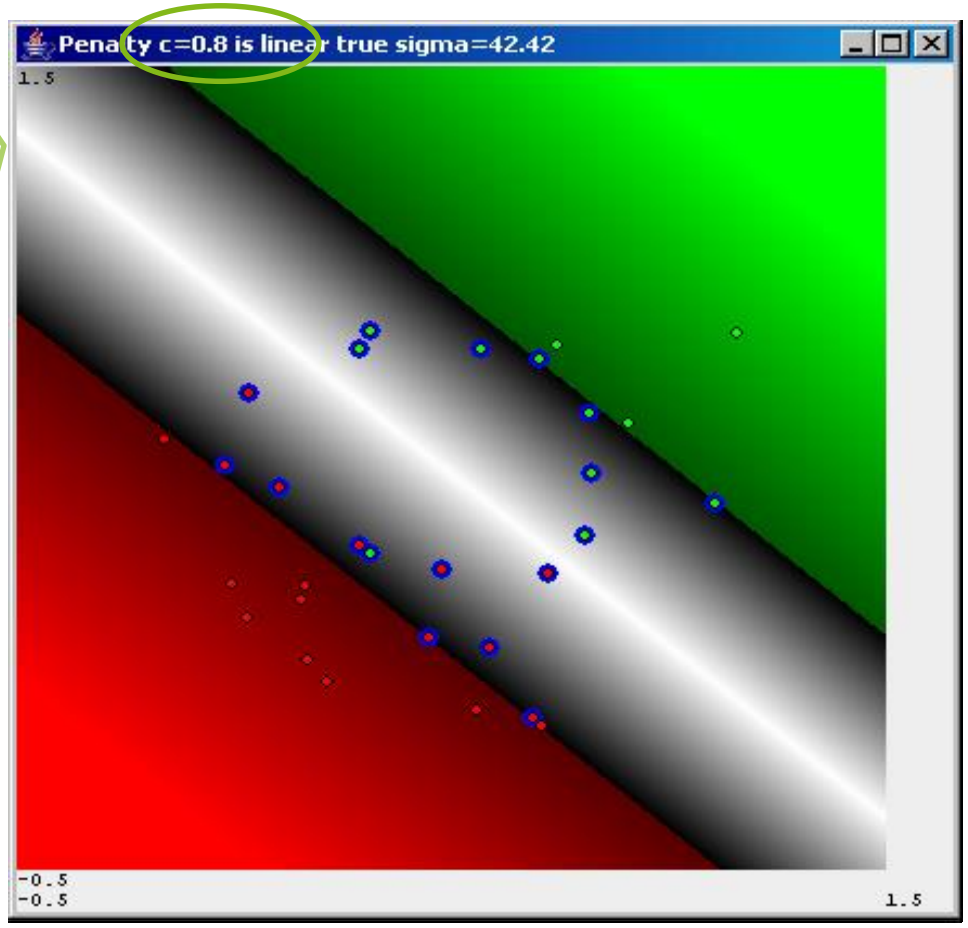
Penalty for misclassifications

White line:
the hyperplane (starts upper right corner)

Black lines:
Margins around hyperplane

Red/green dots:
training instances

Blue circled dots:
support vectors



Visualizing the effect of the C parameter

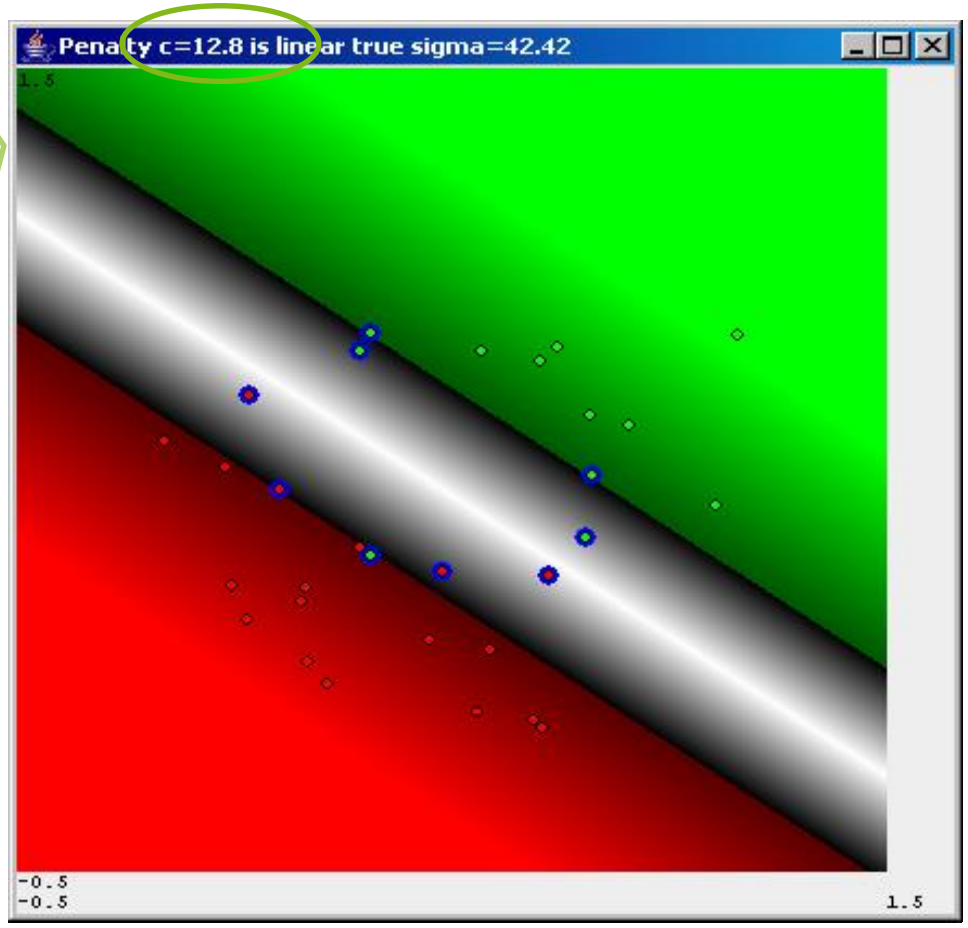
Penalty for misclassifications

White line:
the hyperplane (starts upper right corner)

Black lines:
Margins around hyperplane

Red/green dots:
training instances

Blue circled dots:
support vectors



Visualizing the effect of the C parameter

Penalty for misclassifications

White line:
the hyperplane (starts upper right corner)

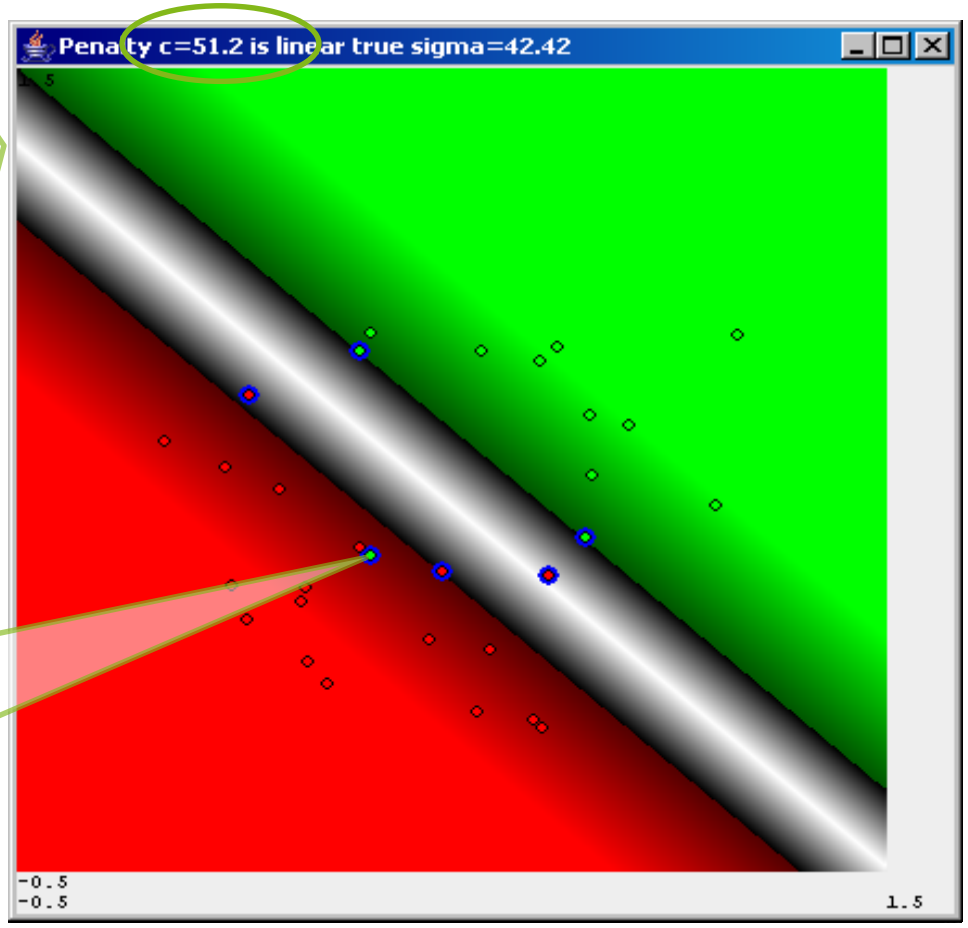
Black lines:
Margins around hyperplane

Red/green dots:
training instances

Blue circled dots:
support vectors

Note: in the non-separable case, all **vectors on- or within the margin** are **support vectors**.

➔ Hence, this one is mislabeled



Visualizing the effect of the C parameter

Penalty for misclassifications

White line:
the hyperplane (starts upper right corner)

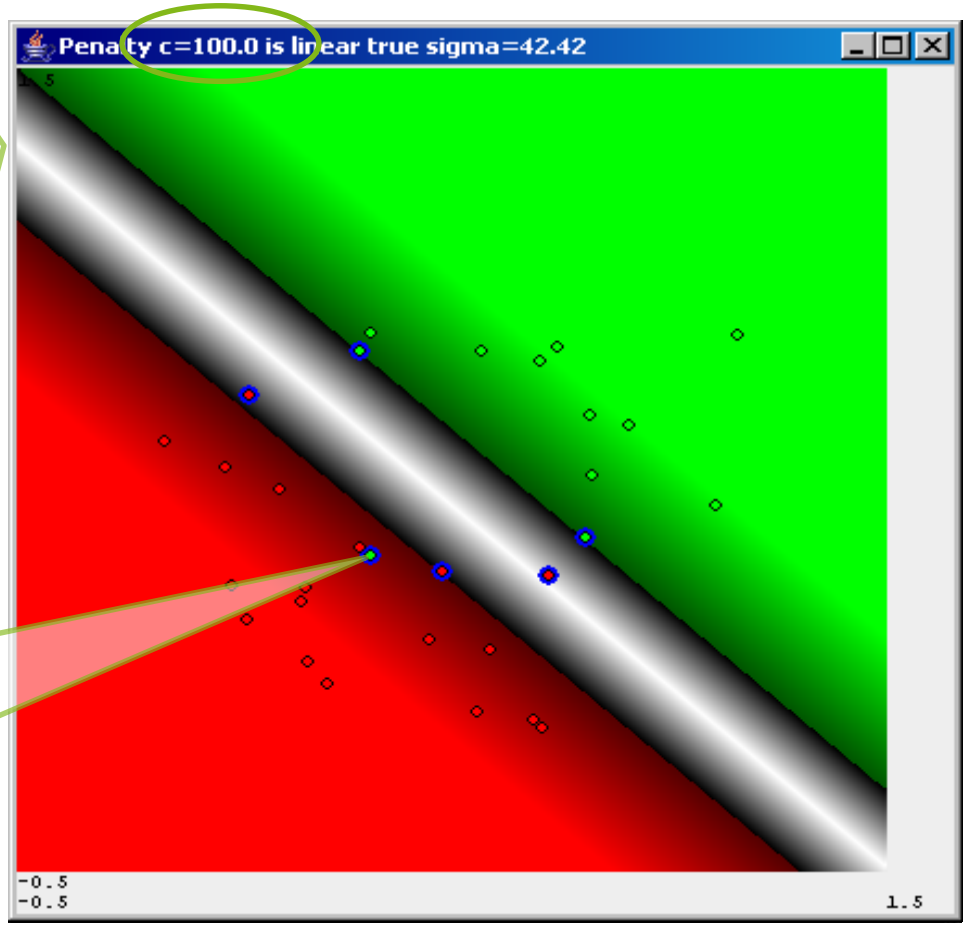
Black lines:
Margins around hyperplane

Red/green dots:
training instances

Blue circled dots:
support vectors

Note: in the non-separable case, all **vectors on- or within the margin** are **support vectors**.

➔ Hence, this one is mislabeled



Visualizing the effect of the C parameter

Penalty for misclassifications

White line:
the hyperplane (starts upper right corner)

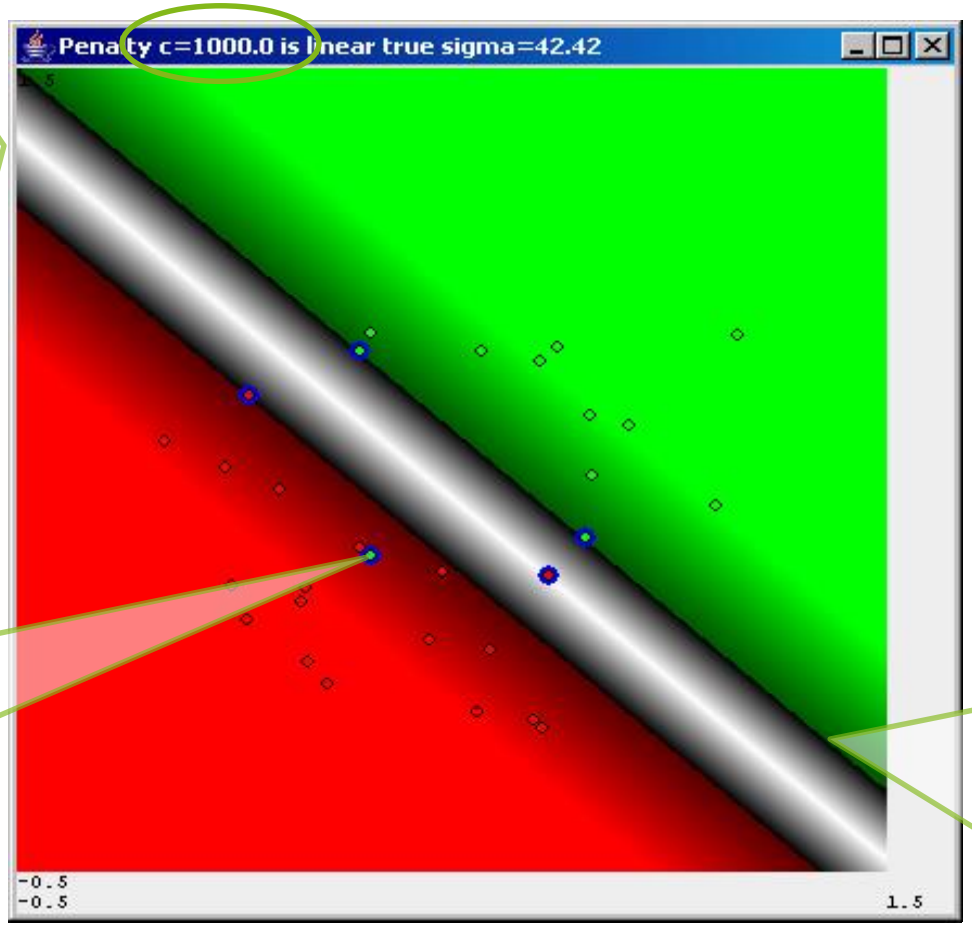
Black lines:
Margins around hyperplane

Red/green dots:
training instances

Blue circled dots:
support vectors

Note: in the non-separable case, all **vectors on- or within the margin** are **support vectors**.

➔ Hence, this one is mislabeled



Varying C leads to convergence to a stable solution.

Generally: C controls the bias-variance trade-off in SVM:

➔ Bigger C result in smaller margins,

➔ hence less support vectors,

➔ hence more variance

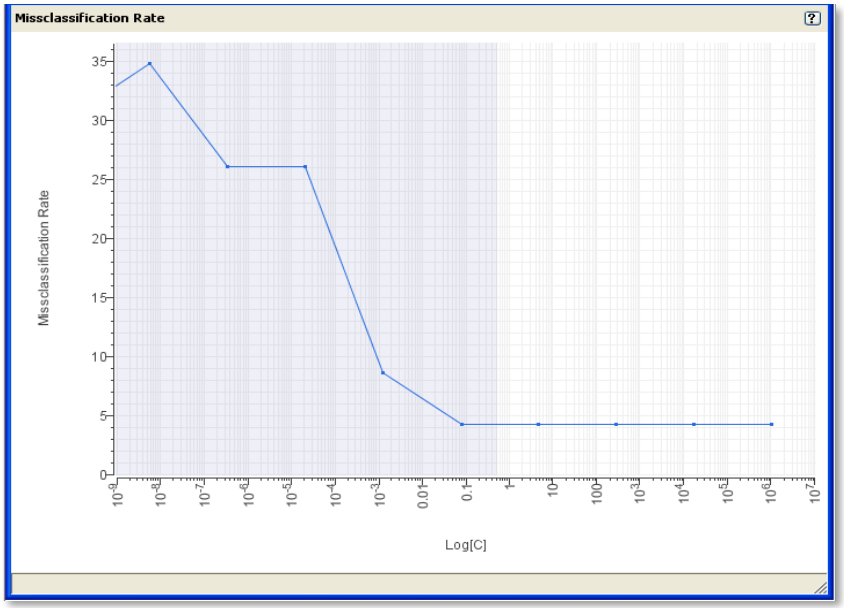
Experimental observations
…for SVMs on gene expression data

Gene expression characteristic: $p \gg N$

- C **too low**: nearly no penalty for misclassification
  ➔ Overgeneralization (“I don’t care for data”, underfitting)
- C **larger**
  ➔ Converging to a stable solution

Figure

- Typical curve for gene expression:
  Misclassification rate as a function of $\log(C)$
- In general, C is a hyper-parameter
  which can be optimized
  (beware of overfitting → see V06)



Zurich University
of Applied Sciences

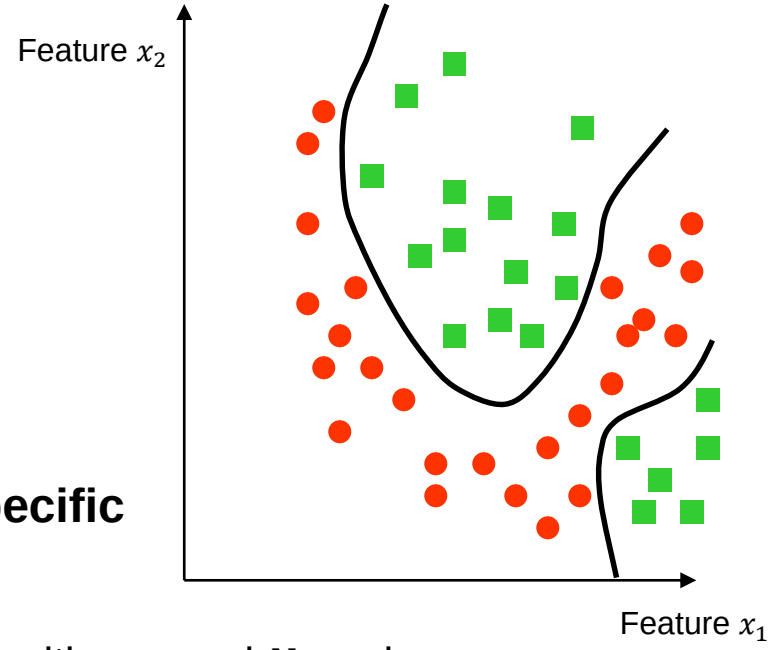
3. KERNELS

(3) Support vector machines

Towards linearly *completely* non-separable data

Solution

- Map the data into a **higher-dimensional space**
- Define a separating hyperplane there



If this is a **typical** case is very much **domain specific**

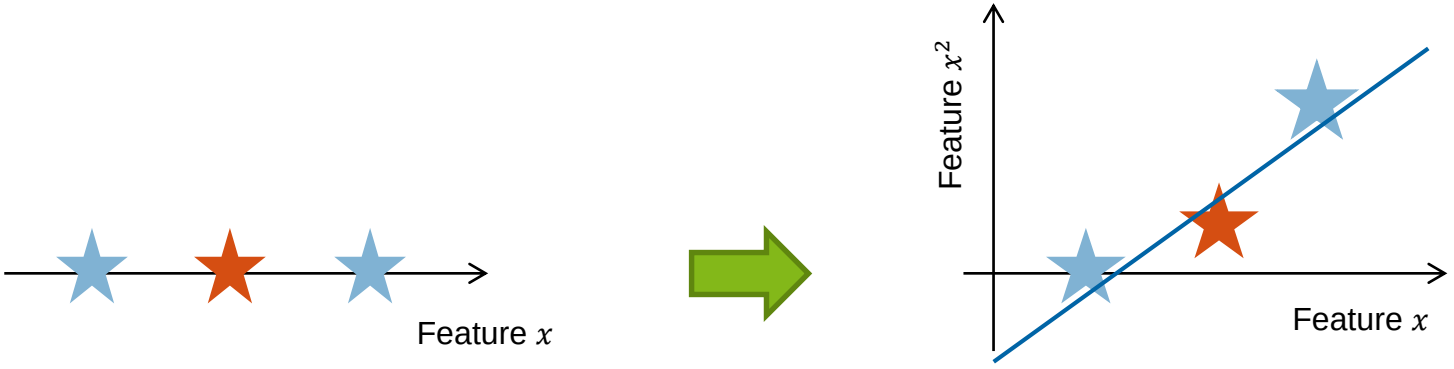
- E.g., **not** in gene expression data, $p \gg N$
  ➔ Practically always linearly separable
- E.g., **often** in multimedia analysis and pattern recognition, p and N are large
  ➔ Practically always difficult non-linear relationships

Variable transformation

Making non-separable cases separable

Only a single variable x
➔ Not separable by a point (hyperplane in 1D)

Take single variable x and x^2
➔ Separable by a line (hyperplane in 2D)



Optimization view: Given features $x_1, {x_1}^2, x_2, {x_2}^2, \dots, x_p, {x_p}^2$ ➔ maximize M, subject to…

- $y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} {x_{ij}}^2\right) \geq M(1-\xi_i)$, $\sum_{i=1}^{N} \xi_i \leq \tilde{C}$, $\xi_i \geq 0$, $\sum_{j=1}^{p} \sum_{k=1}^{2} {\beta_{jk}}^2 = 1$.

This is now a **quadratic equation**, in which a linear decision boundary may exist
➔ Enlarging the feature space allows for non-linear decision boundaries in input space

SVM – Feature space

Power of feature space transformation

- With an **appropriately chosen** feature space of sufficient dimensionality…
- …**any** consistent training **set** can be made **separable**
- (see last slide)

Equivalent technical optimization («dual form»)

- Maximize $L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \boldsymbol{x_i}^T \boldsymbol{x_k}$
  $= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \boldsymbol{K(x_i, x_k)}$

Kernel Trick

- Remember: **only scalar products** <.,.> enter SVM optimization
- Let's rewrite the scalar product after transformation to feature space:

$$\left\langle \begin{pmatrix} x_i \\ {x_i}^2 \end{pmatrix} \cdot \begin{pmatrix} x_{i'} \\ {x_{i'}}^2 \end{pmatrix} \right\rangle = x_i x_{i'} + {x_i}^2 {x_{i'}}^2 := K(x_i, x_{i'})$$

K is known as a **kernel function**. It can be used to **efficiently compute feature space** transformations (i.e., we don't really have to work/optimize in higher dimensions).

➔ Instead of calculating the inner product, we calculate the kernel $K(.,.)$

➔ As soon as we use some K, the resulting **support vector classifier** becomes a **SVM**

Other kernel functions

Different distance measures between instances

The following Kernels are commonly used

- Identity (inner product): $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$
  - this yields the standard support vector classifier

- **Polynomial** of degree d: $K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$
  - the polynomial kernel of degree $d = 1$ is just the identity kernel

- **Radial basis (Gaussian)**: $K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right)$
  - the Gaussian kernel computes a very local neighborhood
  - γ is a hyper parameter controlling the width of that neighborhood, i.e., the degree of non-linearity: **low γ → great width**
  - the feature space spanned by the Gaussian kernel is implicit and infinite-dimensional

i.e., only locally near instances x_i and $x_{i'}$ have kernel function values substantially larger than zero.

because it is inversely related to the variance of the Gaussian → see below

Kernels can also be used for non-vector data

- E.g., string kernels for text
- Many research has been devoted to develop non-standard kernels for specialized tasks (→ google "kernel methods" by the authors *Christianini* and *Shawe-Taylor*)

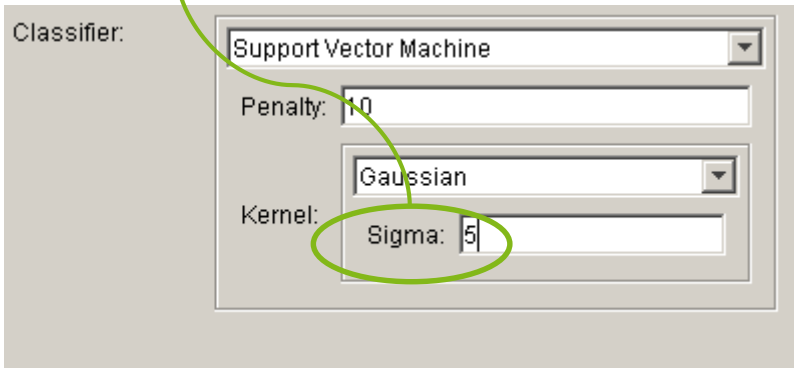
More intuition for the Gaussian kernel

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right)$$

Consider a space with one or more clusters per class

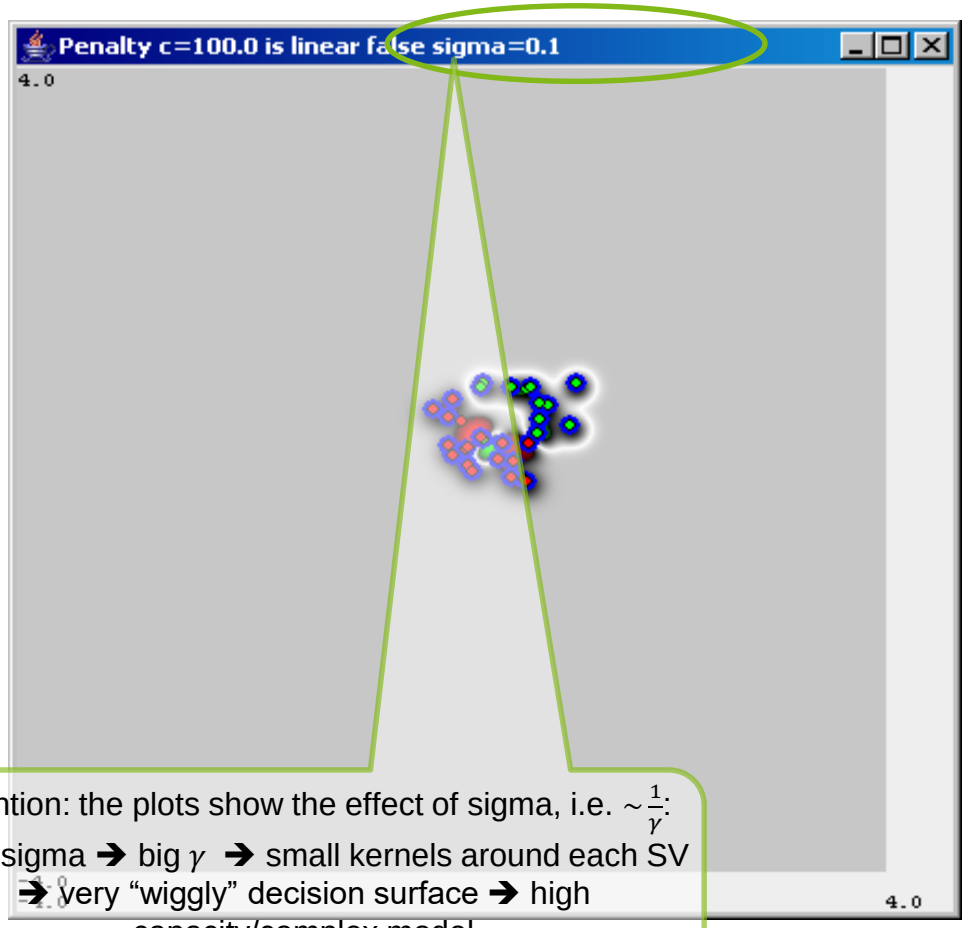
- A classifier might place a Gaussian around each cluster
- This separates the clusters from the remaining space of non-class members
- Can be accomplished with the Gaussian kernel:

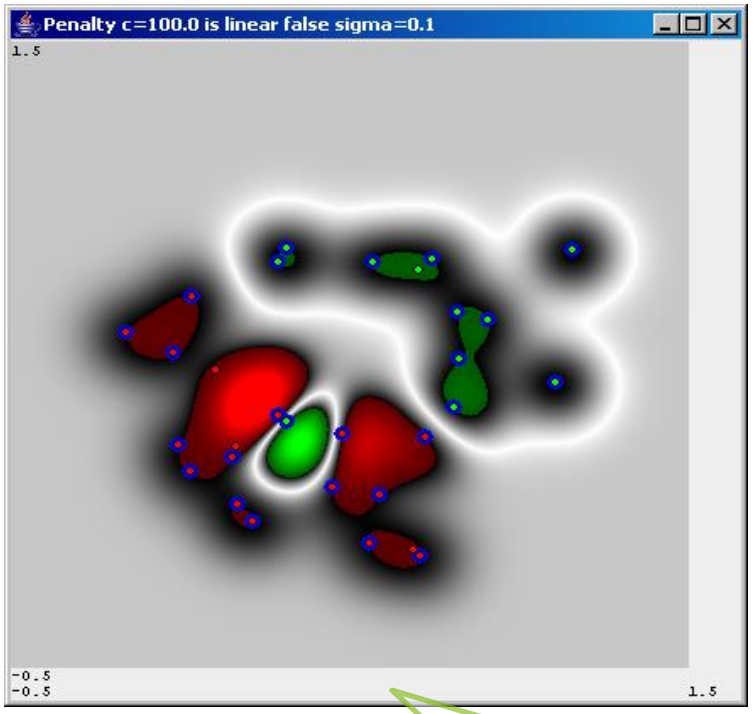
  Place a Gaussian with a width $\gamma = \frac{1}{2\sigma^2}$ over each support vector in the training set



Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



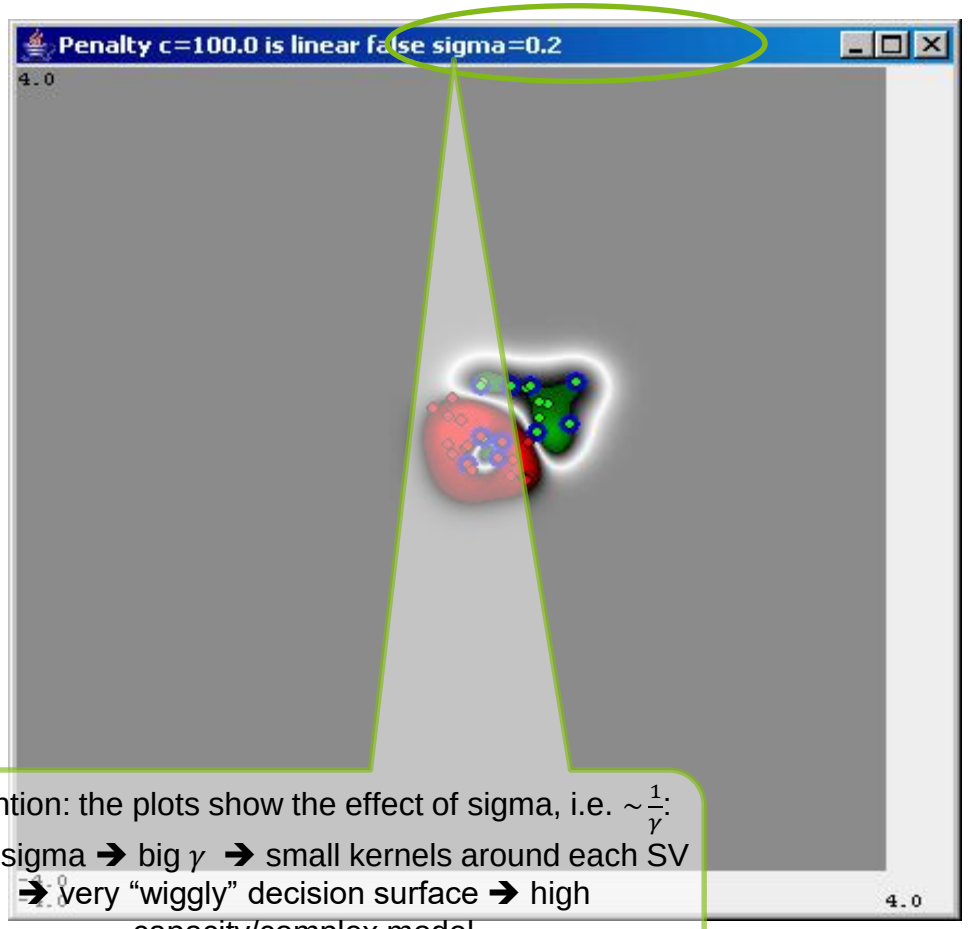


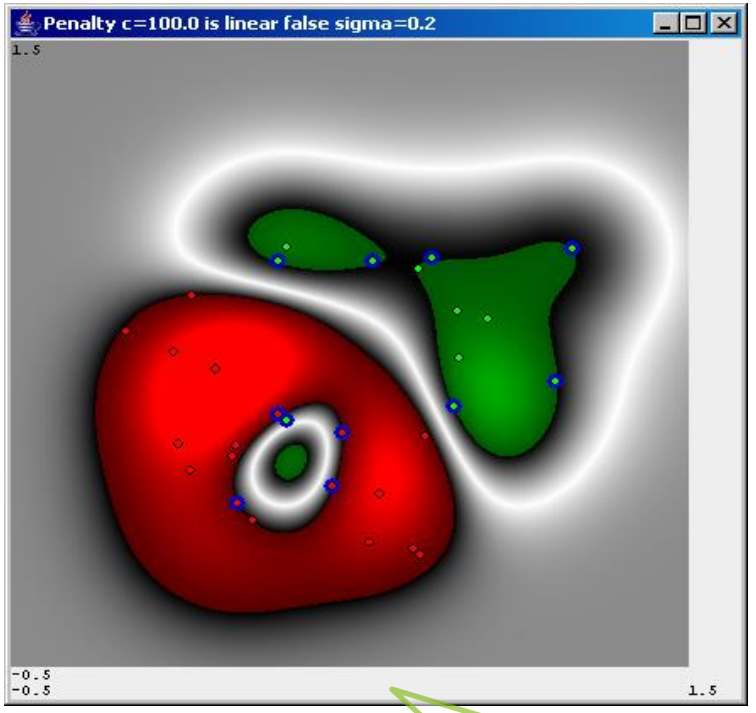
Zoom of plot on the right

Attention: the plots show the effect of sigma, i.e. $\sim\frac{1}{\gamma}$:
small sigma ➔ big γ ➔ small kernels around each SV ➔ very "wiggly" decision surface ➔ high capacity/complex model

Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



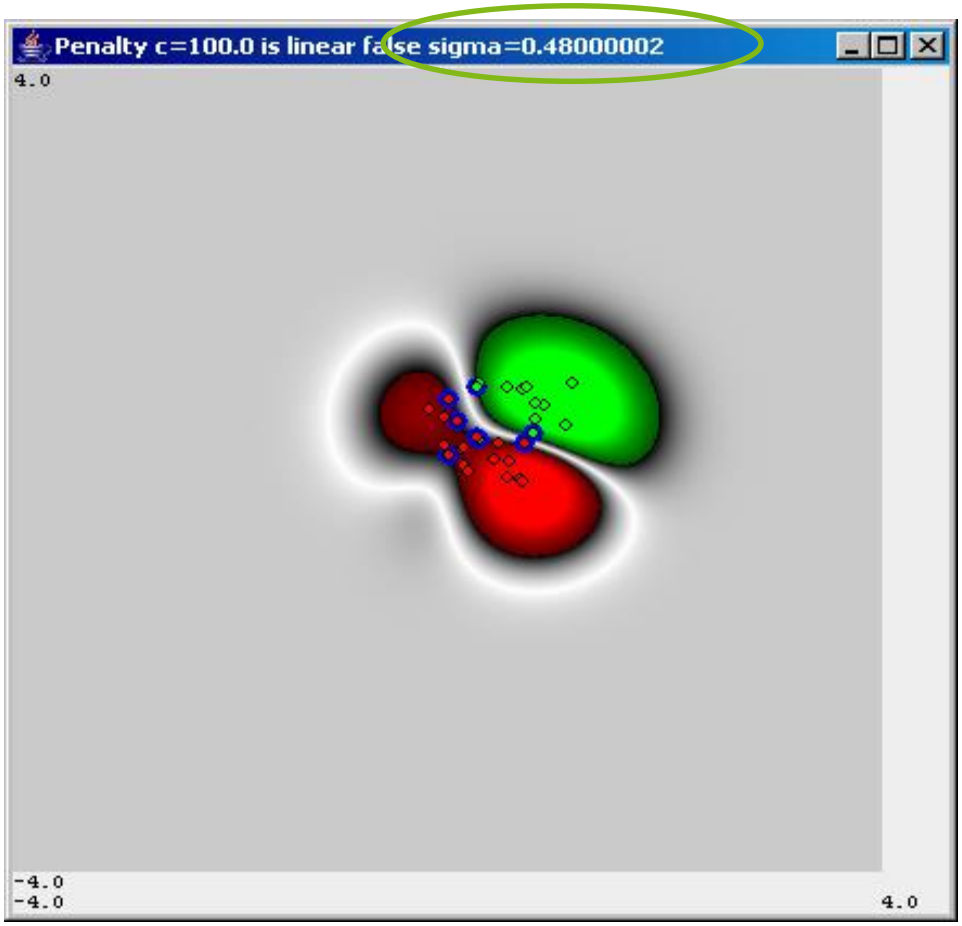


Zoom of plot on the right

Attention: the plots show the effect of sigma, i.e. $\sim\frac{1}{\gamma}$:
small sigma ➔ big γ ➔ small kernels around each SV ➔ very "wiggly" decision surface ➔ high capacity/complex model

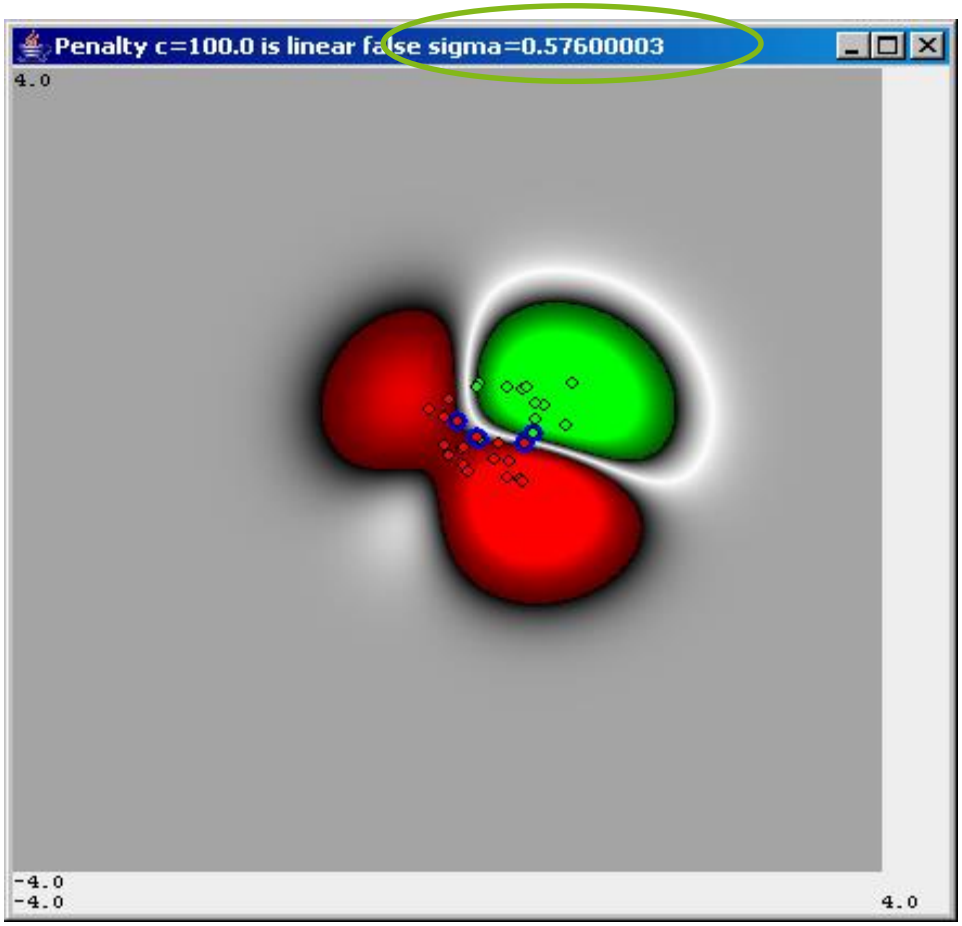
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



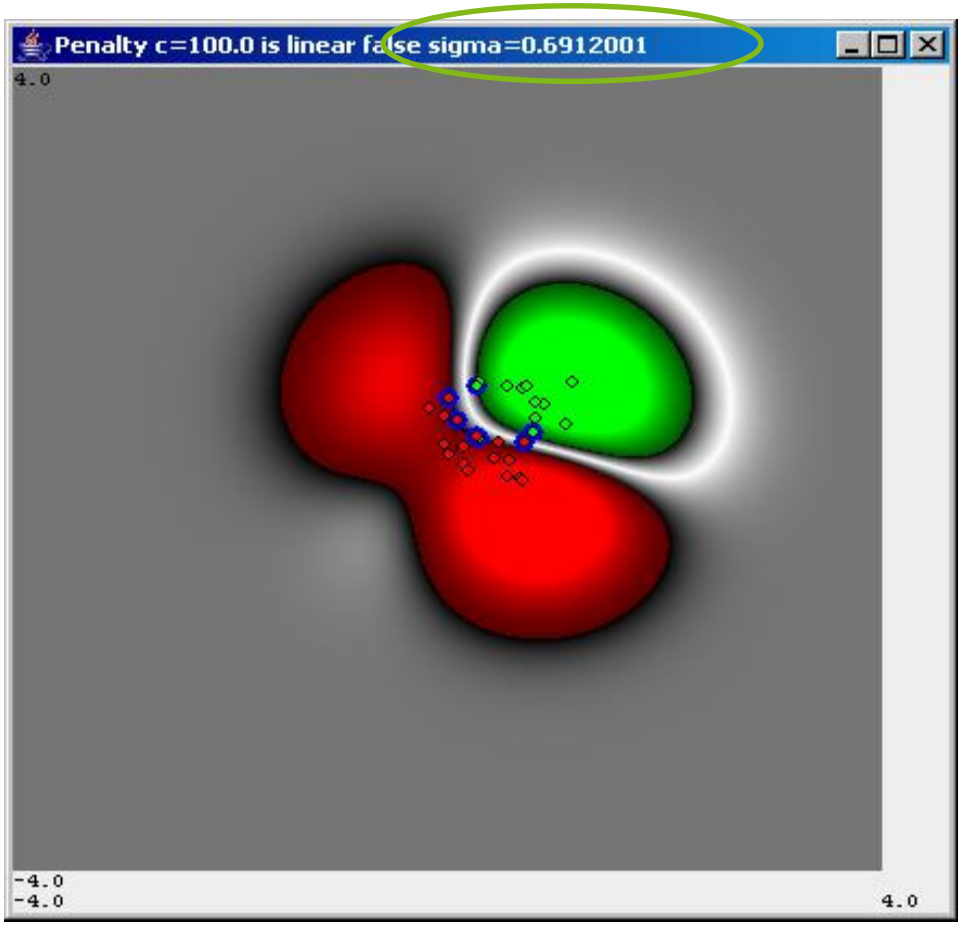
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



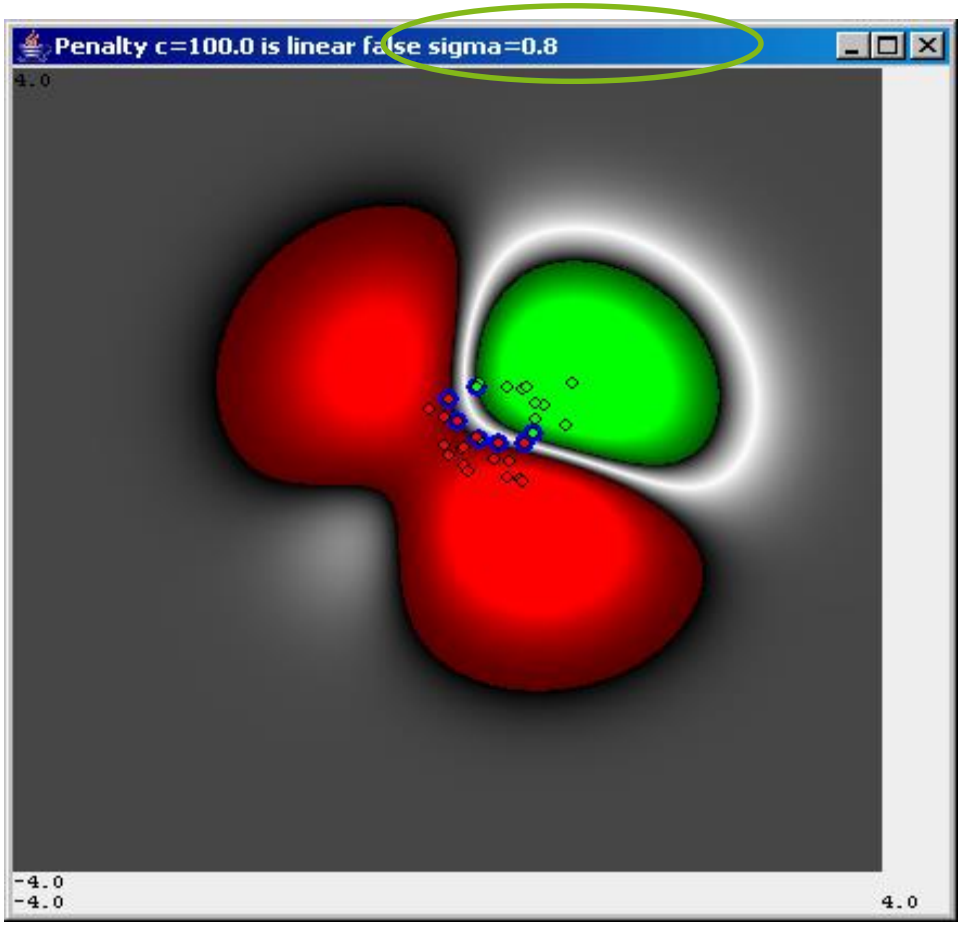
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



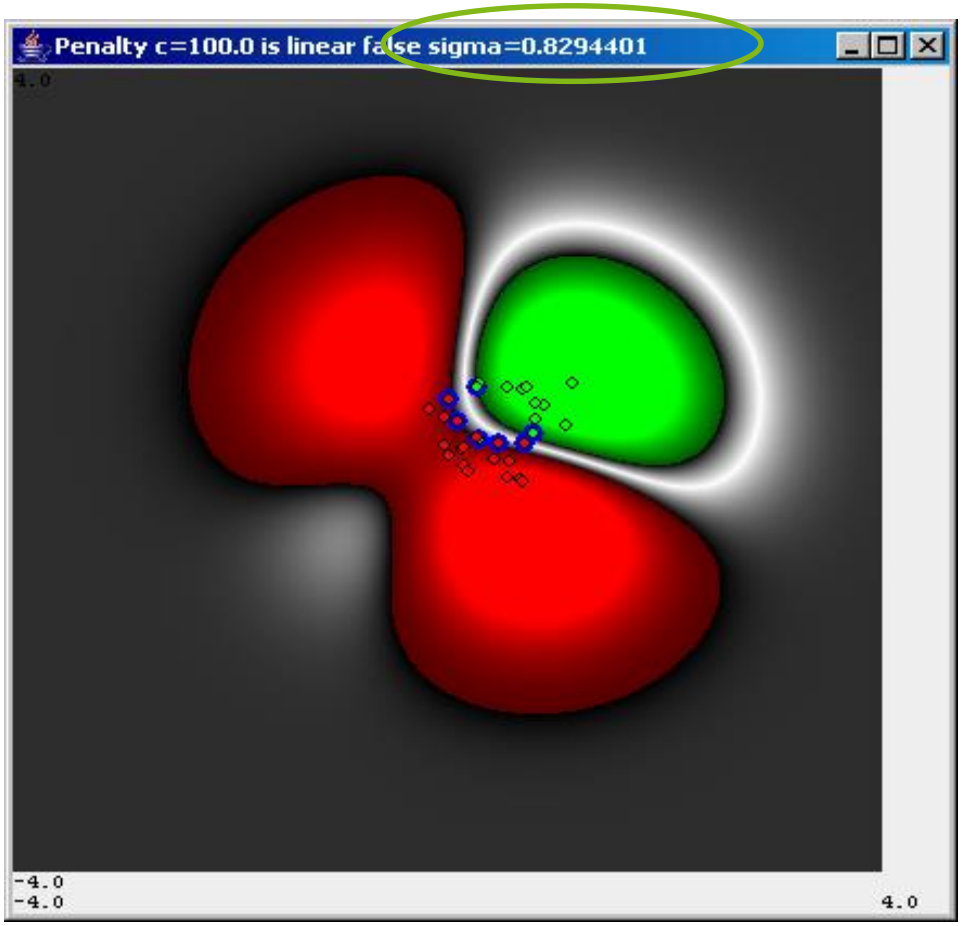
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



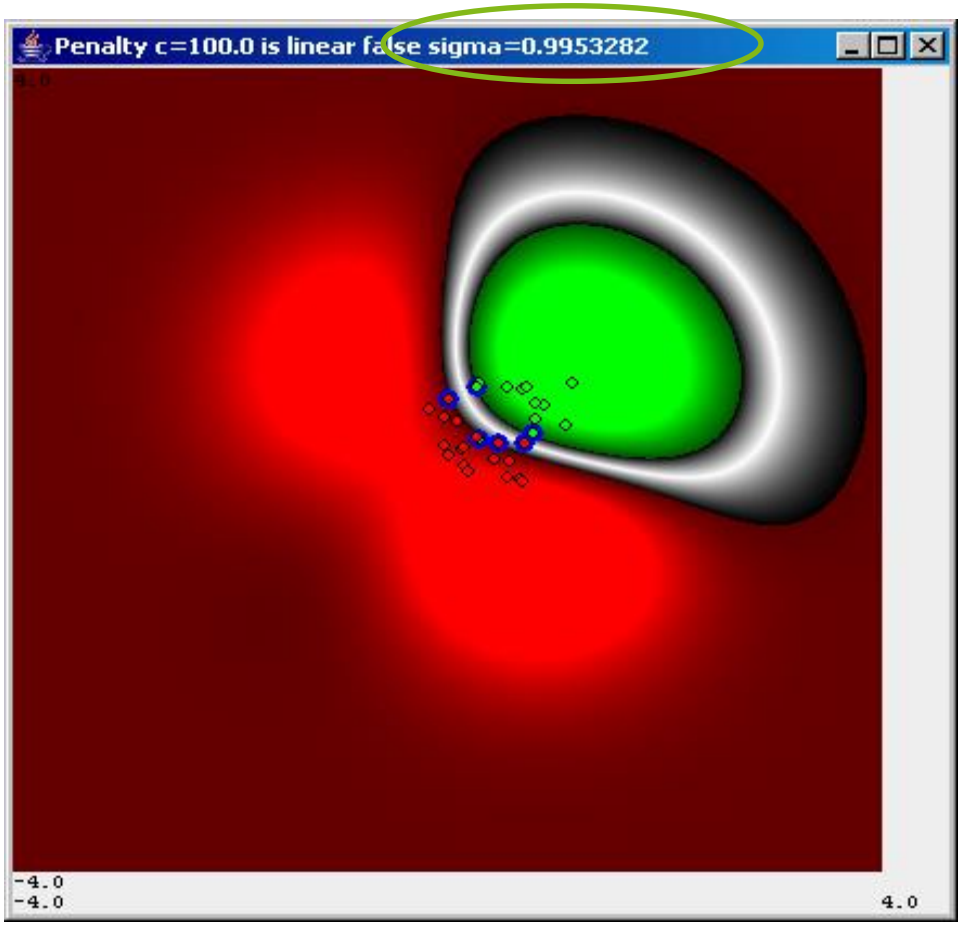
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



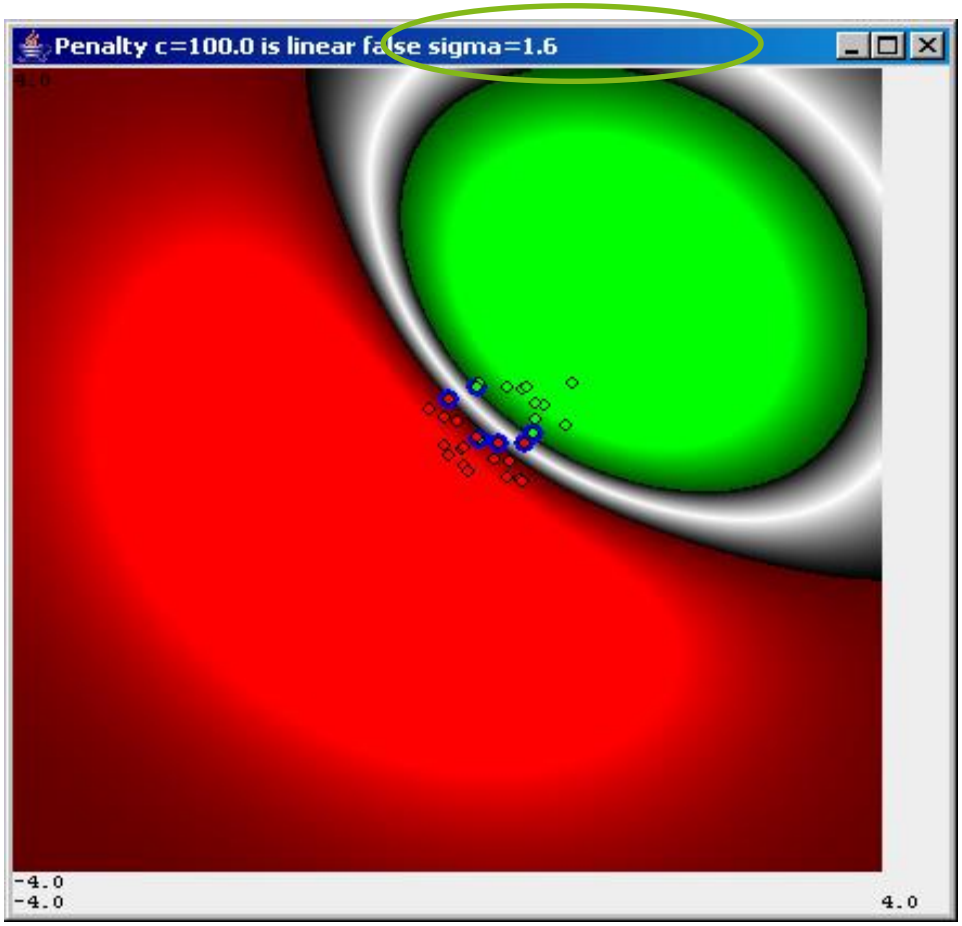
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



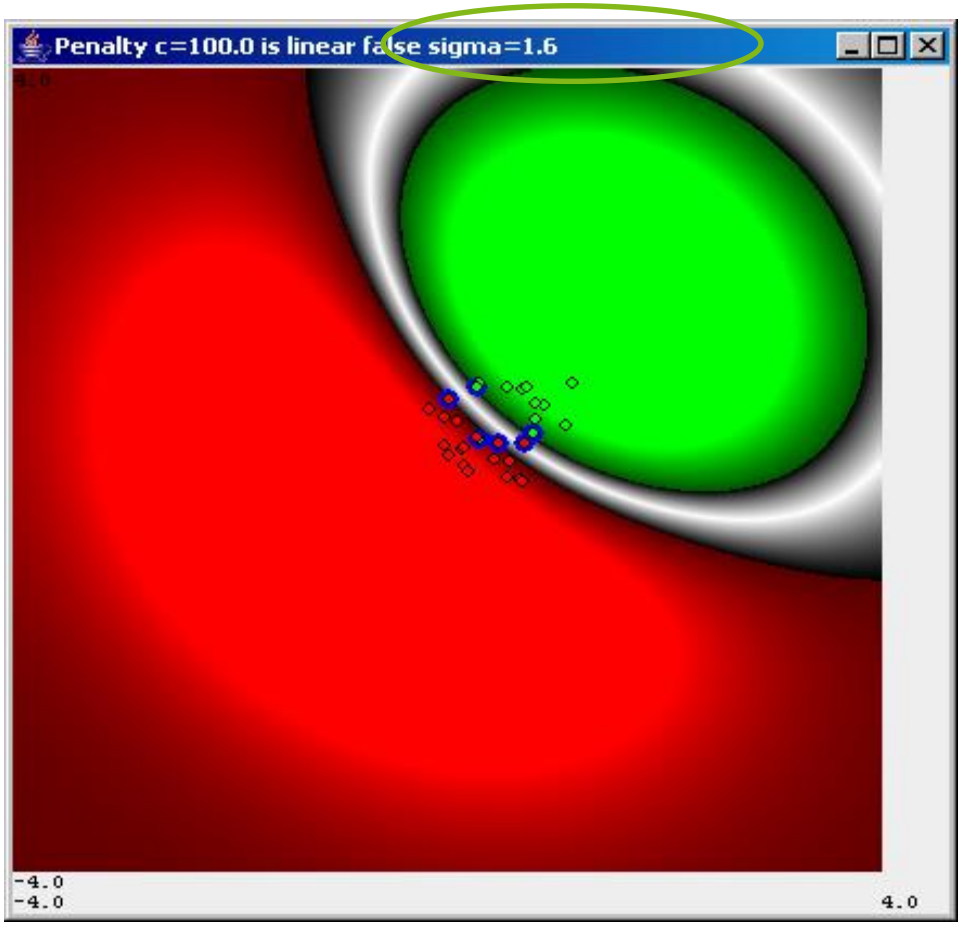
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



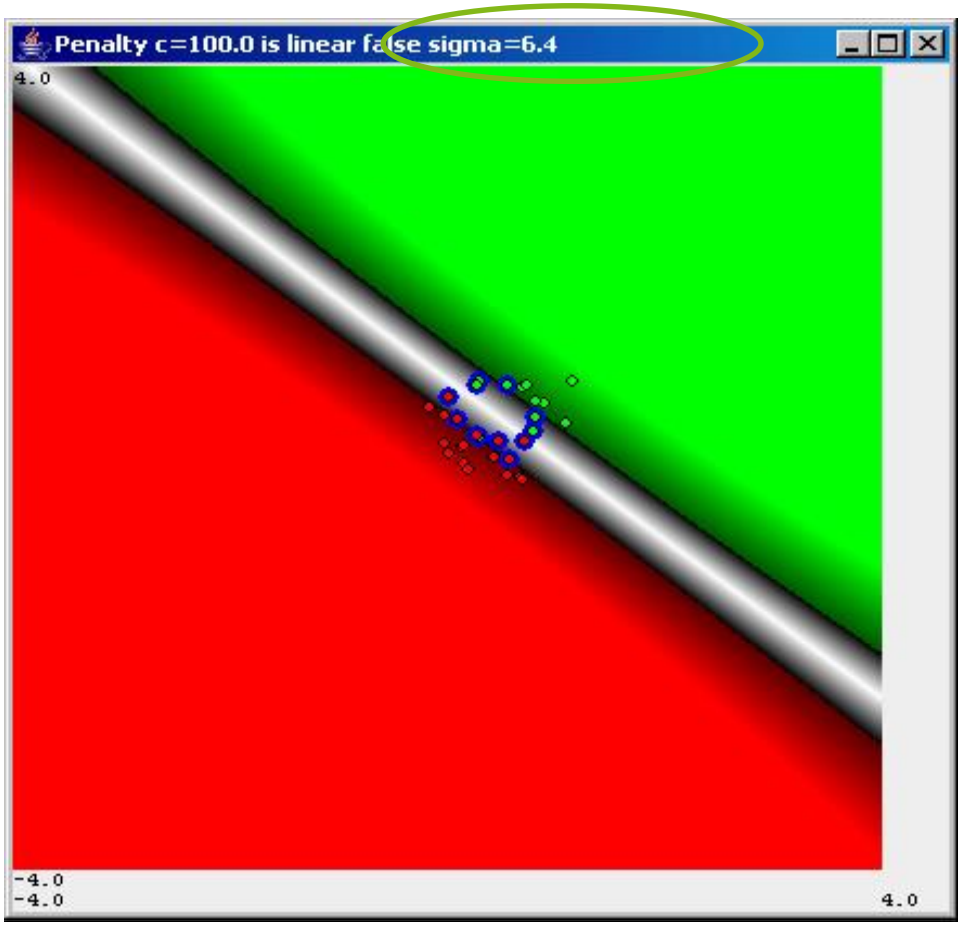
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



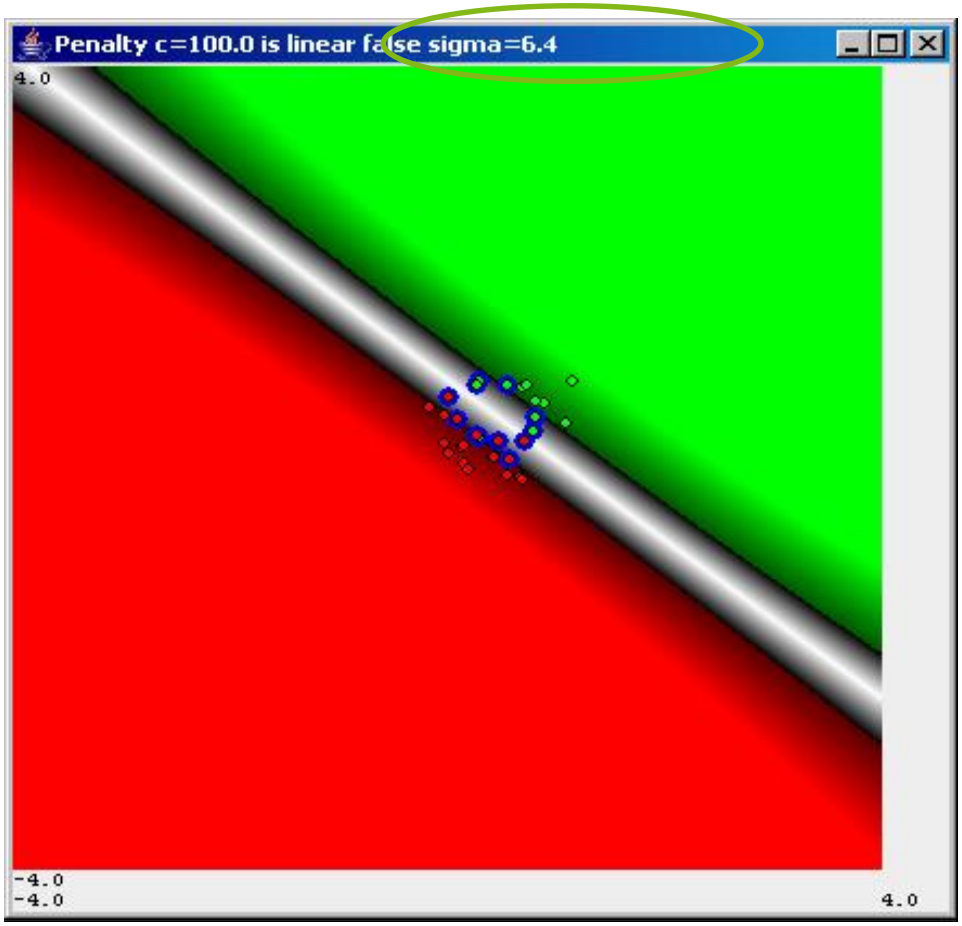
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



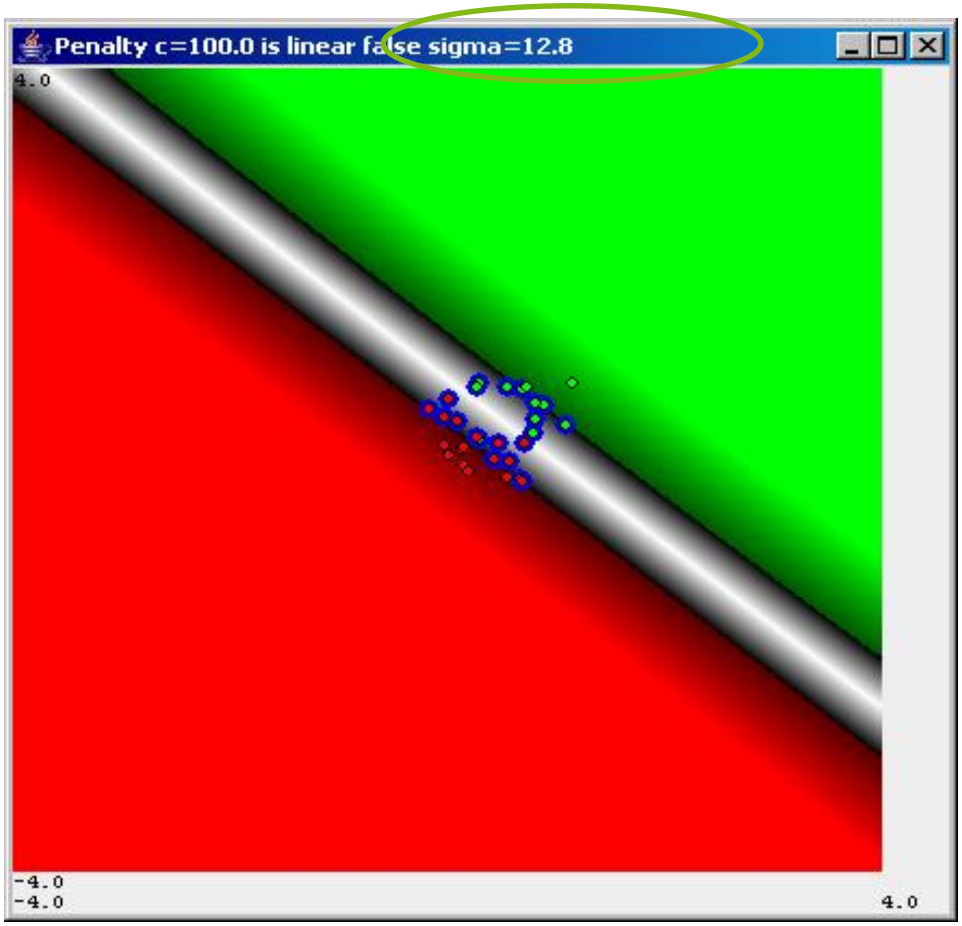
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



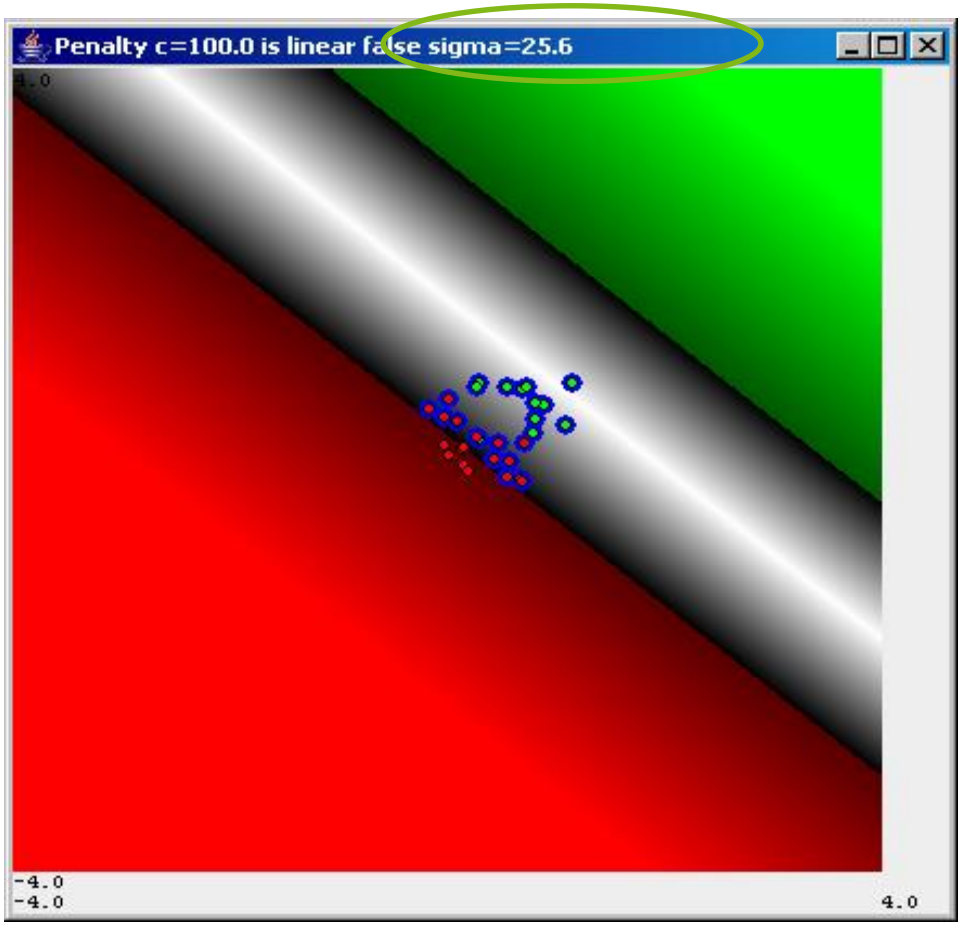
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



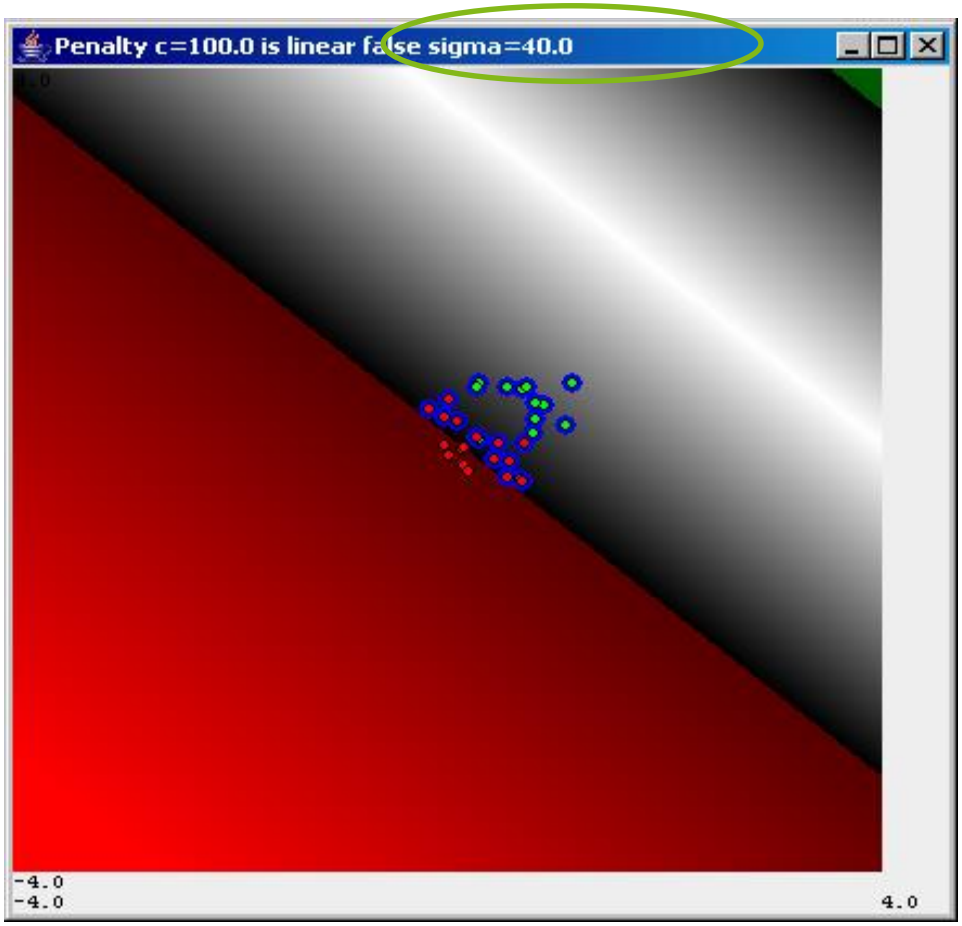
Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



Visualizing the effect of the γ parameter

Width of the Gaussian kernel $\gamma = \frac{1}{2\sigma^2}$



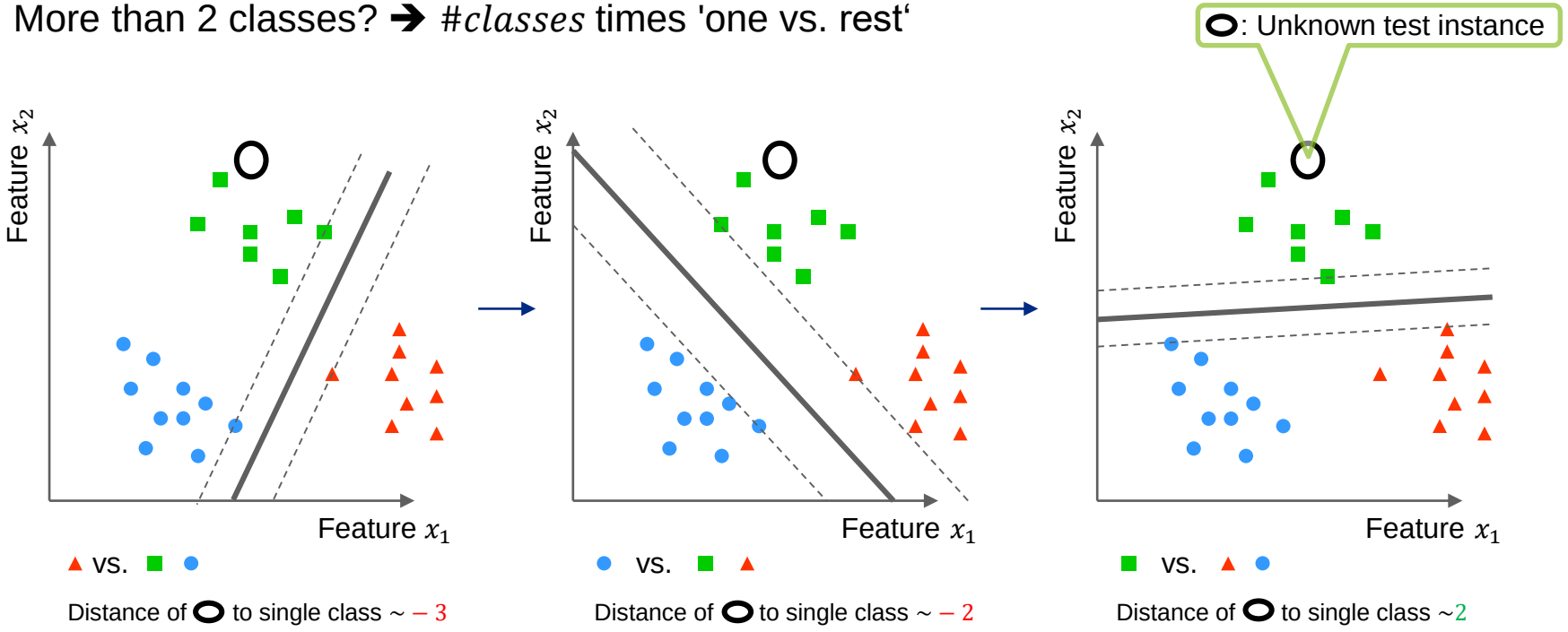
Zurich University
of Applied Sciences

4. MORE THAN TWO CLASSES

Multiclass classification with SVM

One vs. rest (or one vs. all) classification

- SVM is a binary classifier ➔ can only separate two classes
- More than 2 classes? ➔ $\#classes$ times 'one vs. rest'



➔ O has **highest positive distance** (confidence) in **green** case ➔ will be classified as green

Multiclass classification with SVM (contd.)

One vs. one classification

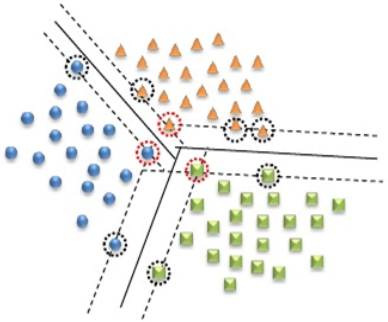
- Assume k classes
- Learn **all** $\frac{k(k-1)}{2}$ **pairwise comparisons**
- Classify unknown instance by **majority vote** among all pairwise comparisons

Other extensions

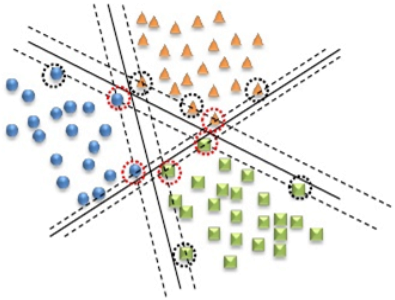
- …from the binary case exist
- Most commonly, “one vs. one” or “one vs. rest” is used

Which method to use?

- One vs. one has fatter margins, but longer training time
  (see S. Herrero, “Multiclass Classification using Massively Threaded Multiprocessors” on SlideShare)



One vs. one



One vs. all

Review

- Support vector machine (SVM) is **one of the best off-the-shelf classifiers**
- SVM can be used for **classification** (seen here), **regression** and **novelty detection** (→ see appendix for a picture)

- SVM **optimization** is mathematically very **complex**, the **application** and basic ideas are remarkably **simple** (→ see appendix for h formula)

- SVM builds upon the **ideas** of **support vector classifiers** (no kernels) and **maximal margin classifiers** (no C), all using **separating hyperplanes**

- **Feature space transformations** via kernels allow **non-linear decision boundaries**
- Parameters $\boldsymbol{C}$ and $\boldsymbol{\gamma}$ (with Gaussian kernel) allow to **adjust** the **flexibility** of SVM (bias-variance trade-off)

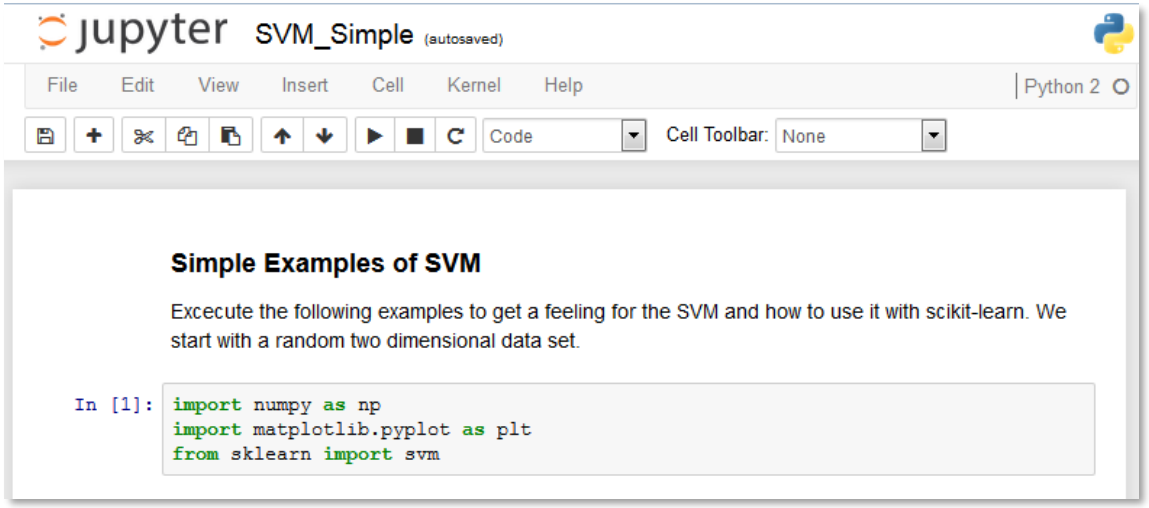
More $\boldsymbol{C}$ ➔ less capacity for misclassification ➔ more variance, less margin
More $\boldsymbol{\gamma}$ ➔ tighter Gaussian kernels ➔ more non-linear decision surface

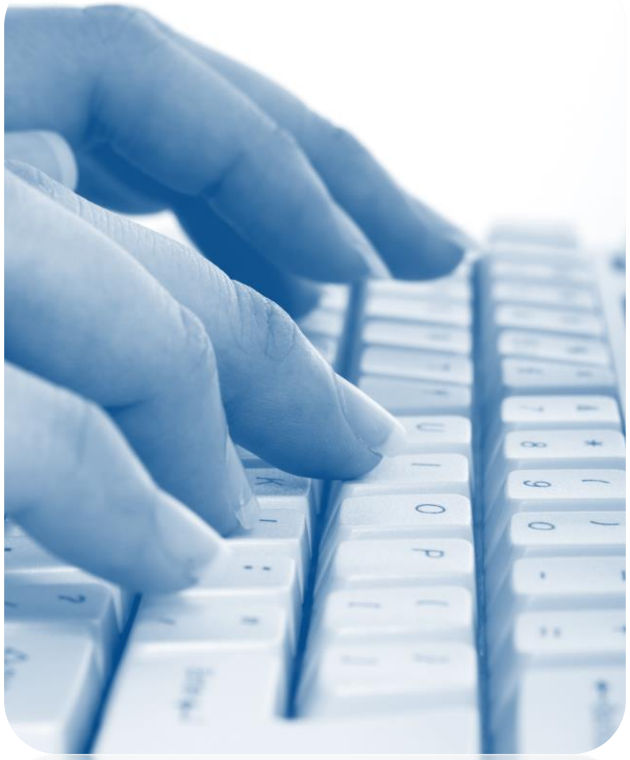
P04.2: Simple SVM example

Execute the IPython notebook **`SVM_Simple.ipynb`**

- Further explore scikit-learn syntax
- Play around with SVM parameters

➔ Additional notebook source: https://github.com/oduerr/ml-playground/blob/master/python/SVM/SVM_Simple.ipynb





Zurich University
of Applied Sciences

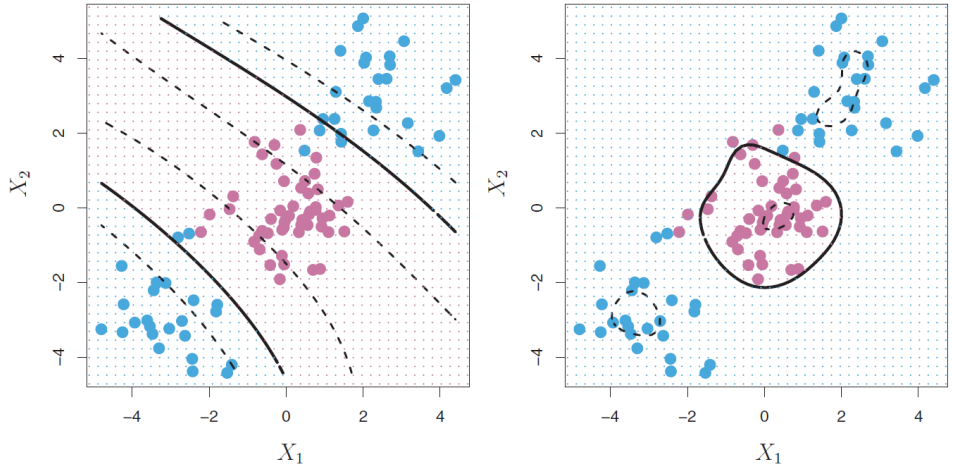
APPENDIX

SVM classification function

$$h(x) = \beta_0 + \sum_{i \in S} \alpha_i \cdot K(x, x_i)$$

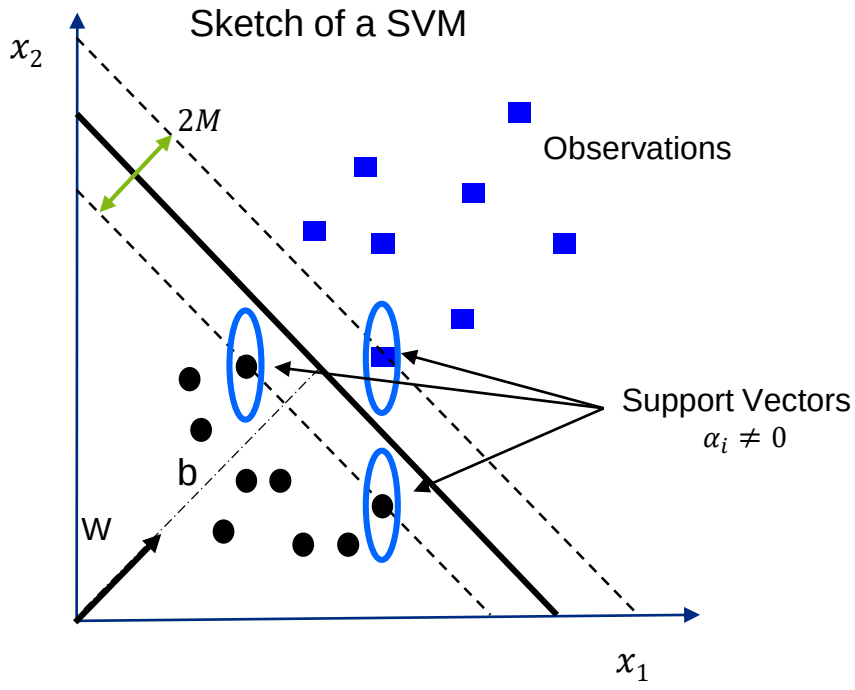
Where

- S is the set of support vectors x_i from training
- β_0 and α_i are the optimized parameters from training
- x is a test instance
- $K(.)$ is a kernel function



A polynomial ($d = 3$) kernel (left) and a Gaussian (radial) kernel (right) capturing the non-linear decision boundary in different ways (figure 9.9 of [ISL, 2014, p. 353]).

Geometry of SVM



Formulation

- $< \boldsymbol{w}, \boldsymbol{x} > + b = \pm 1$ for SV
  ➔ Objective: minimize $||\boldsymbol{W}||$
- After minimization:

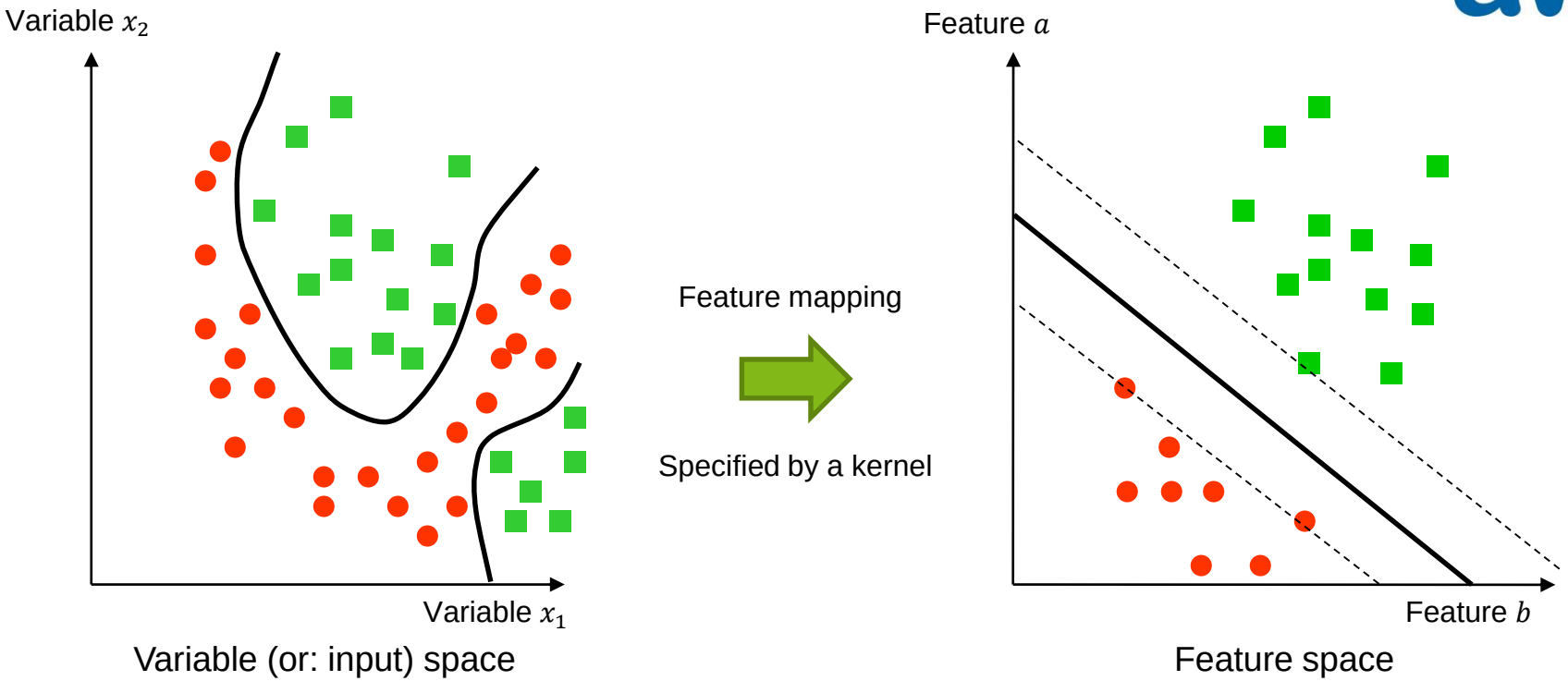
$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Interesting quantities

- Components of W (importance of feature)
- Fraction of support vectors (how much SVs depends on examples)
- Which observations are SVs (i.e. have an $\alpha \neq 0$)

How the kernel works



Note on the **number** of **tunable parameters**

- Non-linear SVM has several parameters (through the kernel), linear one has (almost) none

➔ If $N \ll p$, linear SVM is sufficient due to high dimensionality (**low point density** → see V06)

Advanced Topics

Possible extensions

- Handling of categorical variables
- Outlier Detection with one-class SVM (→ see plot)
- SVM Regression
- ROC-Analysis with SVM