Support Vector Machines

Luca Citi
lciti@essex.ac.uk

School of Computer Science and Electronic Engineering
University of Essex (UK)

CE802

Outline

- Linear classifier
  - Linear classifier
  - Fitting the model
- SVM
  - Margin
  - Hard-margin SVM
  - Soft-margin SVM
  - Support vectors
  - Nonlinear SVMs
  - Kernels
  - Regularisation

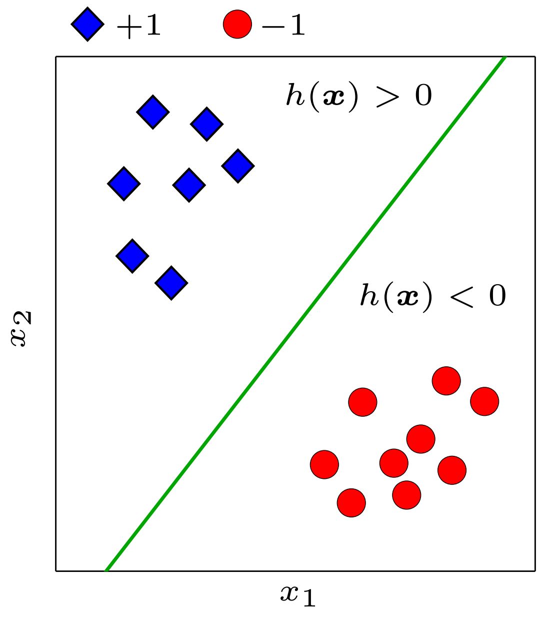
Linear classifier

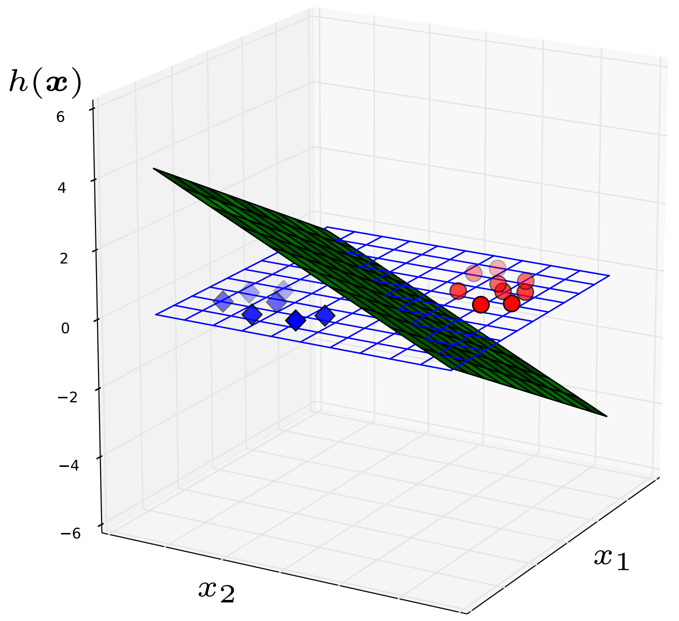
Let's consider a binary classification problem where the two classes are encoded as $\{+1, -1\}$.

A linear classifier tries to identify which class an example belongs to by making a decision based on the value of a linear combination of the features:

$h(\boldsymbol{x}|\boldsymbol{w}, b) = \sum_{i=1}^{P} w_i x_i - b = \langle \boldsymbol{w}, \boldsymbol{x} \rangle - b$ where $[\boldsymbol{w}, b]$ is a vector of parameters ($\boldsymbol{\theta}$)

with the predicted class being simply: $\operatorname{sign}(h(\boldsymbol{x}|\boldsymbol{w}, b))$





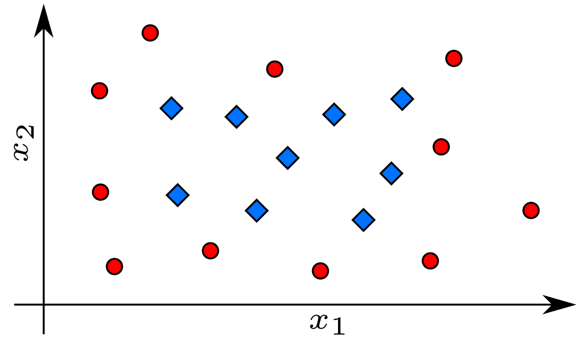
Linear classifier

Pros:

- easy(-ier) to train than other classifiers
- less prone to overfitting
- assuming normalised features[1], the square of the weight w_i^2 indicates the relative importance of the feature x_i;
- the sign of w_i indicates whether the feature x_i has a positive or negative effect on the class $+1$;

Cons:

- prone to underfitting for non linearly separable problems.



[1] When using most classifiers the features should always be normalised to: 0 mean and range included in the interval $[-1, 1]$ or unitary standard deviation

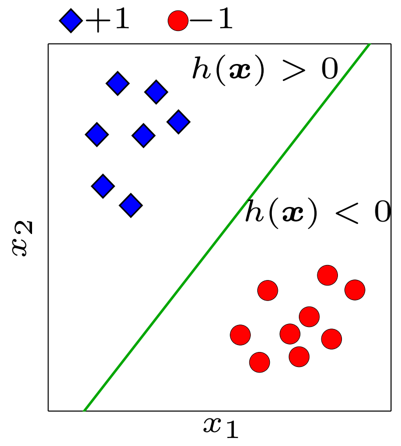
Fitting the model

Given a training set $\mathcal{X} = \{(\boldsymbol{x}^t, r^t)\}_{t=1}^N$ with $r^t \in \{+1, -1\}$ ... we want to find the optimal linear classifier $h(\boldsymbol{x}|\boldsymbol{w}, b)$. How?

Ideally we would like[2]:

- $h(\boldsymbol{x}) > 0$ for all training examples belonging to the class $r = +1$
- $h(\boldsymbol{x}) < 0$ for all training examples belonging to the class $r = -1$

This is equivalent to: $r^t\, h(\boldsymbol{x}^t) > 0$ for all $t = 1..N$



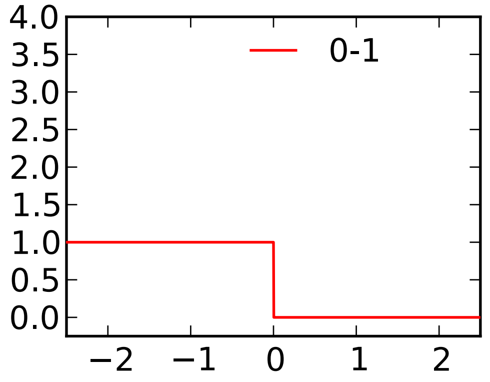
[2] To simplify the notation, we will often write $h(\boldsymbol{x}|\boldsymbol{w}, b)$ as $h(\boldsymbol{x})$ when unambiguous.

Fitting the model: 0-1 loss

So, aiming for: $r^t\, h(\boldsymbol{x}^t) > 0$ for as many examples $t \in \{1..N\}$ as possible seems a good choice.

We can do so by defining the function:

$$\ell_{01}(z) = \begin{cases} 1 & \text{if } z < 0 \\ 0 & \text{otherwise} \end{cases}$$



using:

$$L(r, y) = \ell_{01}(ry) \text{ where } y = h(\boldsymbol{x})$$

and then minimizing the empirical risk:

$$R_{\text{em}}(\boldsymbol{w}, b|\mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N} L(r^t, h(\boldsymbol{x}^t|\boldsymbol{w}, b)) = \frac{1}{N}\sum_{t=1}^{N} \ell_{01}(r^t\, h(\boldsymbol{x}^t|\boldsymbol{w}, b))$$

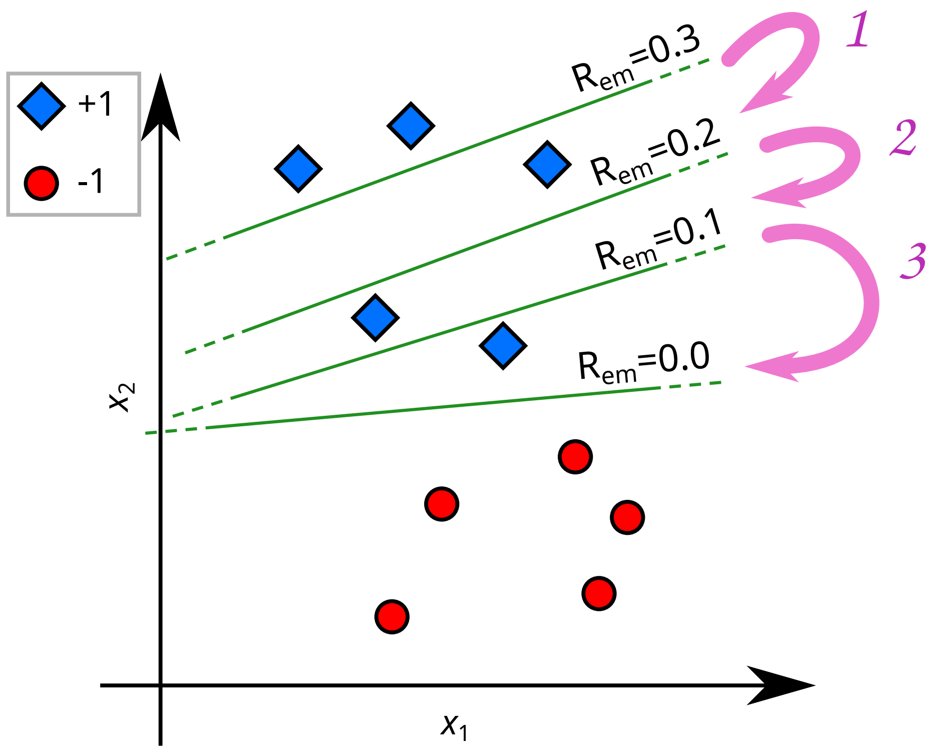
to find the optimal $[\boldsymbol{w}, b]$:

$$[\hat{\boldsymbol{w}}, \hat{b}] = \arg\min_{[\boldsymbol{w},b]} R_{\text{em}}(\boldsymbol{w}, b|\mathcal{X})$$

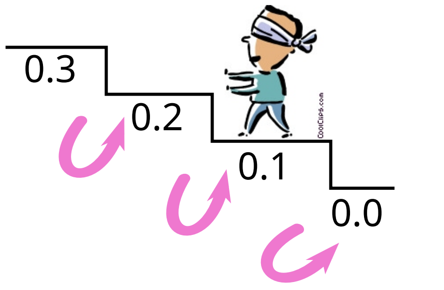
This finds the best separation line according to the 0-1 loss, i.e., by minimizing the fraction of misclassifications in the training set.

Issues of the 0-1 loss (1)

Gradient is zero almost everywhere: finding the optimum in hard (practically unfeasible for most problems)



Empirical risk for the 0-1 Loss

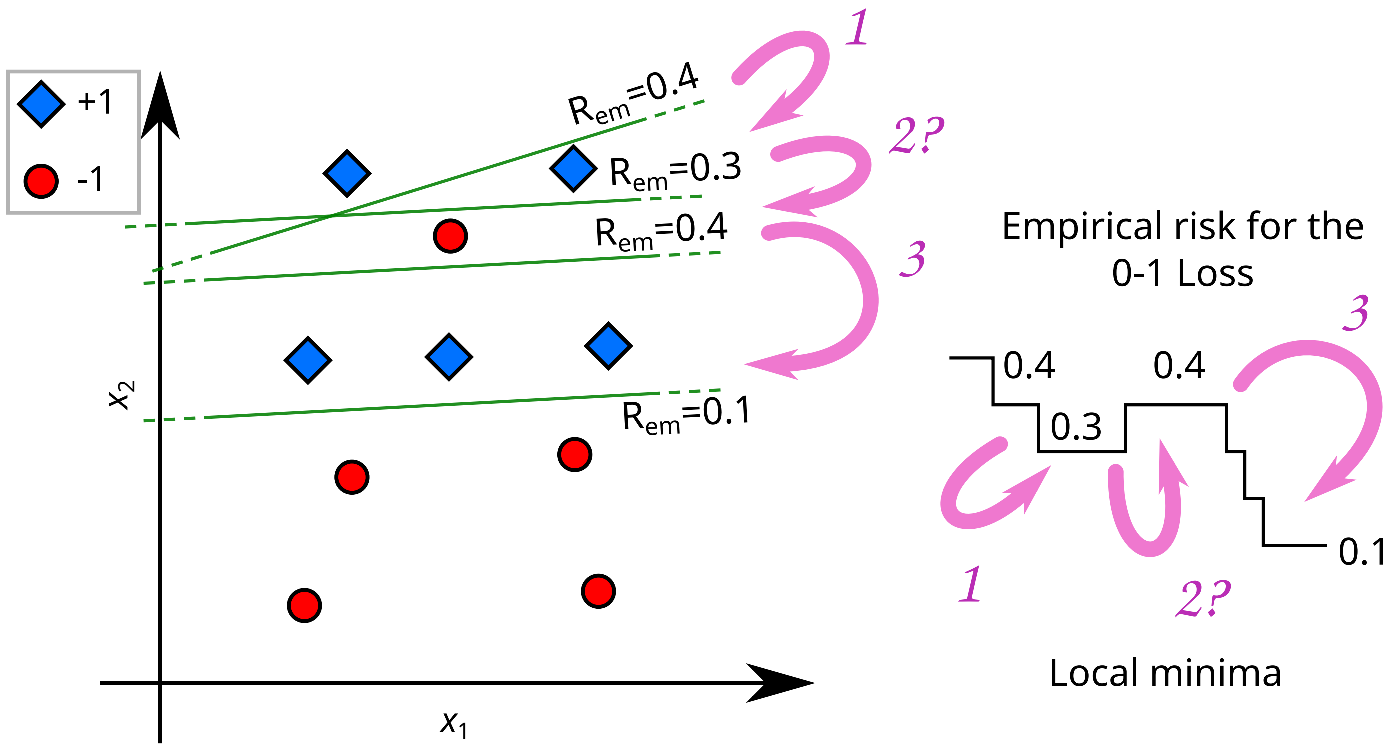


Problem: flat regions

This would be easier

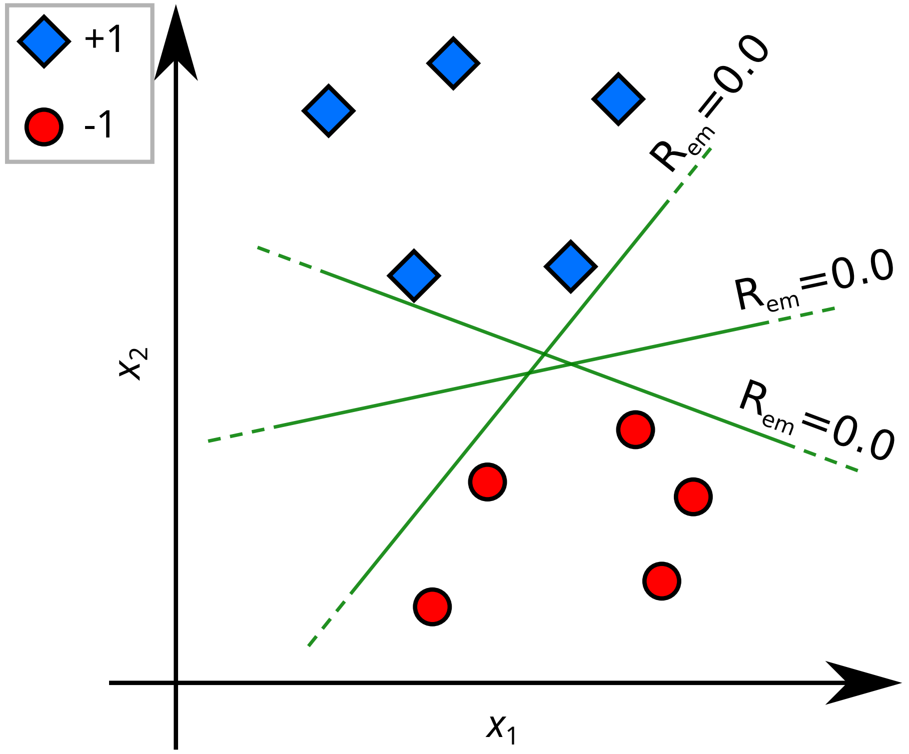
Issues of the 0-1 loss (2)

Presence of (non-global) local minima



Issues of the 0-1 loss (3)

Among all solution minimising the risk, some are clearly better (more robust) than others. Minimizing using the 0-1 loss doesn't do anything to promote those solutions.



Outline

- Linear classifier
  - Linear classifier
  - Fitting the model
- SVM
  - Margin
  - Hard-margin SVM
  - Soft-margin SVM
  - Support vectors
  - Nonlinear SVMs
  - Kernels
  - Regularisation

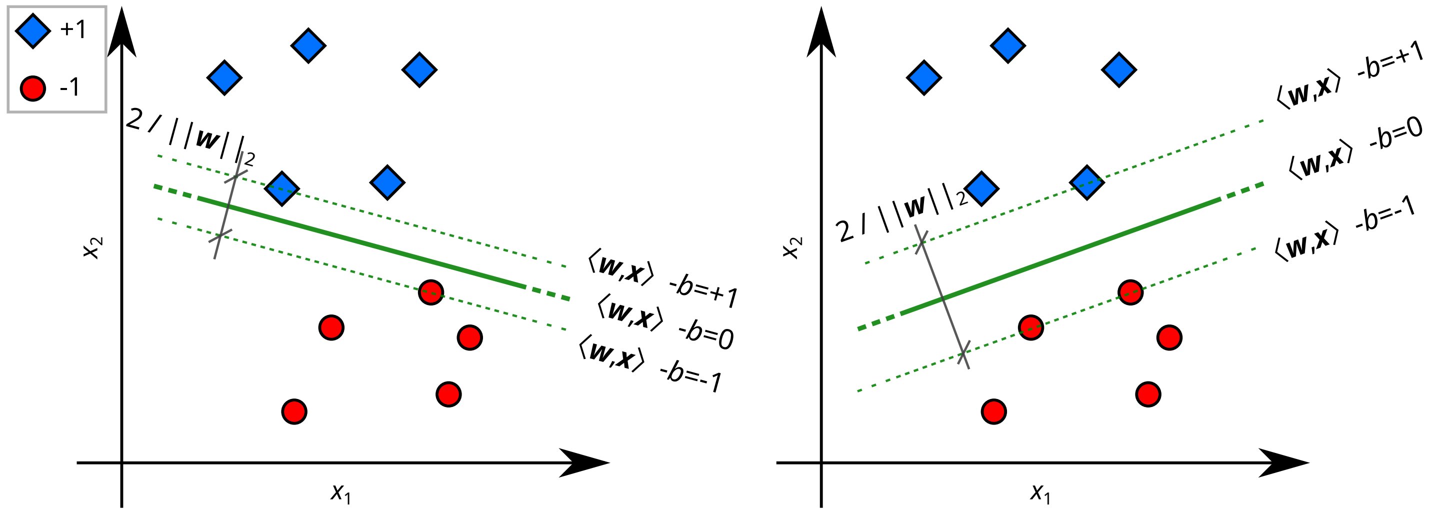
Margin

In the linearly separable case, intuitively, the best solution is the one leading to the largest separation between the two classes.

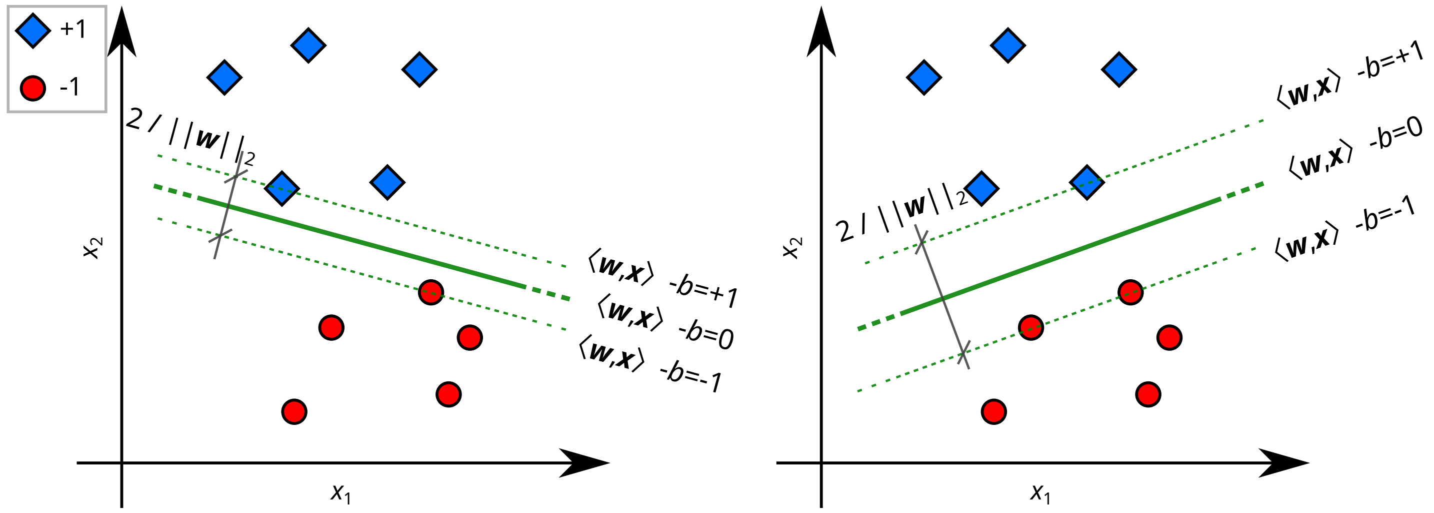
We consider all linear classifiers such that:

$$r^t\, h(\boldsymbol{x}^t) \geq 1 \quad \text{for all } t = 1..N$$

The separation between the two classes, i.e. the distance in the direction orthogonal to the hyperplane $h(\boldsymbol{x}) = 0$ between the closest two points of either class, is at least: $2/\|\boldsymbol{w}\|_2$.



Hard-margin Support Vector Machine



Maximizing the margin $2/\|\boldsymbol{w}\|_2$ is equiv. to minimizing $\frac{1}{2}\|\boldsymbol{w}\|_2$ or $\frac{1}{2}\|\boldsymbol{w}\|_2^2$.
The optimization problem:

$$[\hat{\boldsymbol{w}}, \hat{b}] = \arg\min_{[\boldsymbol{w},b]} \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

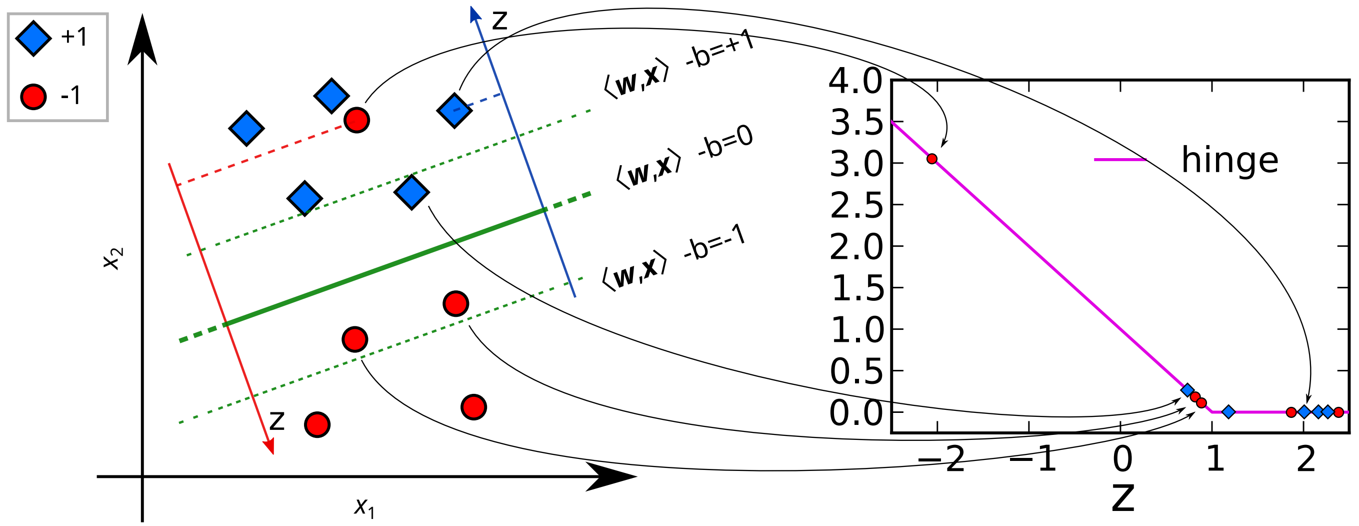
subject to:

$$r^t\left(\langle \boldsymbol{w}, \boldsymbol{x}^t\rangle - b\right) \geq 1 \text{ for all } t = 1..N$$

defines a Hard-margin Linear Support Vector Machine

Non-linearly separable classes

We have seen the optimal solution in the linearly-separable case.



But if the two classes are not linearly separable, there will be no solution satisfying the constraints: $r^t\left(\langle \boldsymbol{w}, \boldsymbol{x}^t\rangle - b\right) \geq 1$ for all $t = 1..N$

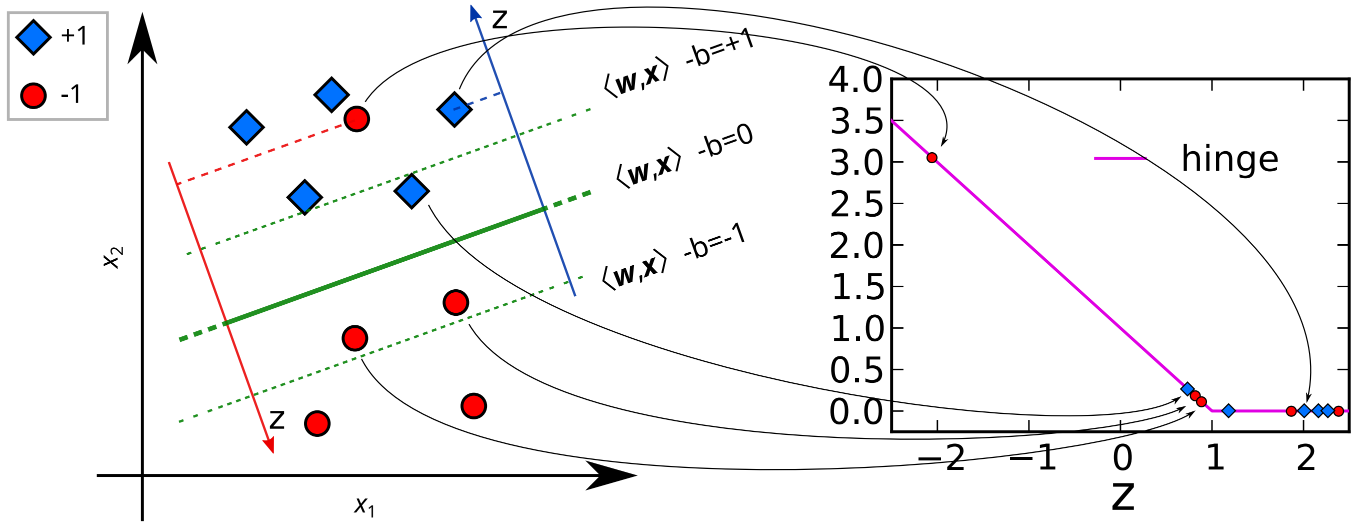
Therefore we introduce a soft margin where points are allowed to violate the constraint but pay a price ("loss") proportional to their distance from the margin boundary: $C\,\ell_{\mathrm{h}}(r\left(\langle \boldsymbol{w}, \boldsymbol{x}\rangle - b\right))$ where ℓ_{h} is the hinge loss:

$$\ell_{\mathrm{h}}(z) = \begin{cases} 1 - z & \text{if } z < 1 \\ 0 & \text{otherwise} \end{cases} = \max\{0, 1 - z\}$$

while C determines the trade-off between margin maximization and error minimization.

Non-linearly separable classes

For each tentative solution $[\boldsymbol{w}, b]$ ...



the hinge loss is zero for points that are correctly classified with sufficient margin, while it is positive for misclassified points or points with insufficient margin.

Intuitively, maximizing the margin (like in the hard-margin case) while minimizing the average loss seems a good strategy.

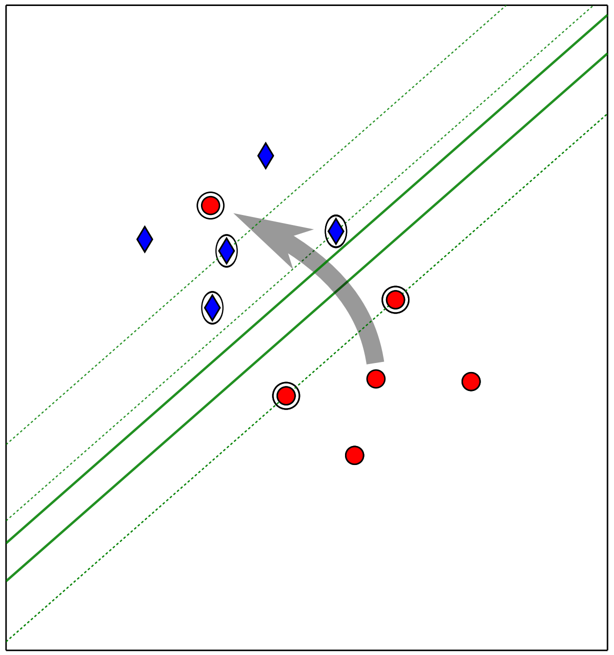
In fact, the optimization problem:

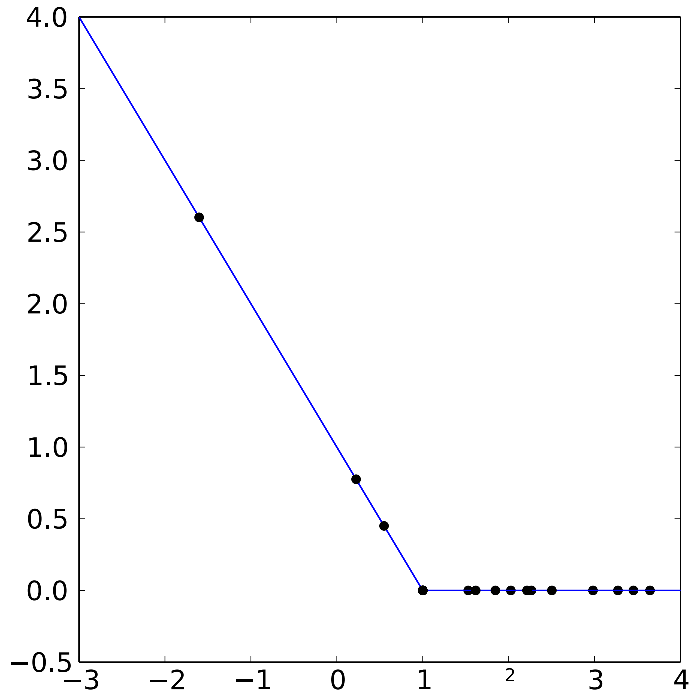
$$[\hat{\boldsymbol{w}}, \hat{b}] = \arg\min_{[\boldsymbol{w}, b]} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C \sum_{t=1}^{N} \ell_{\mathrm{h}}(r^t (\langle \boldsymbol{w}, \boldsymbol{x}^t \rangle - b))$$

defines a Soft-margin Linear Support Vector Machine

Support vectors

Why the name Support Vector Machine?
Let's see what happens when we move some of the training examples (C large):

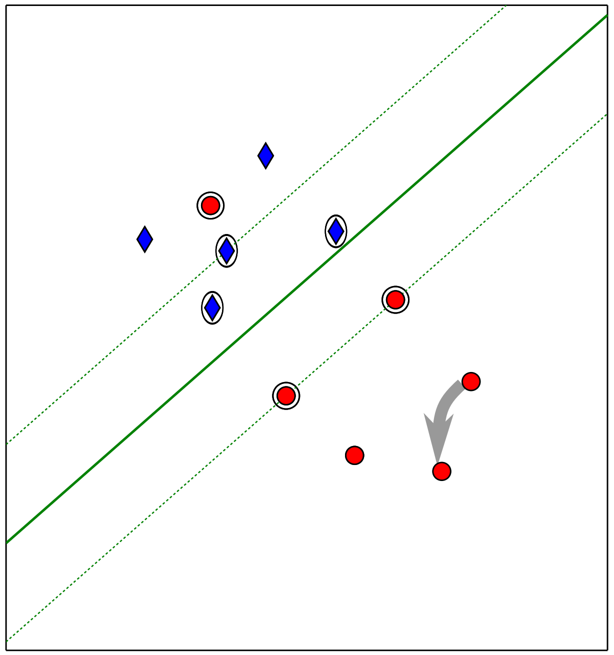


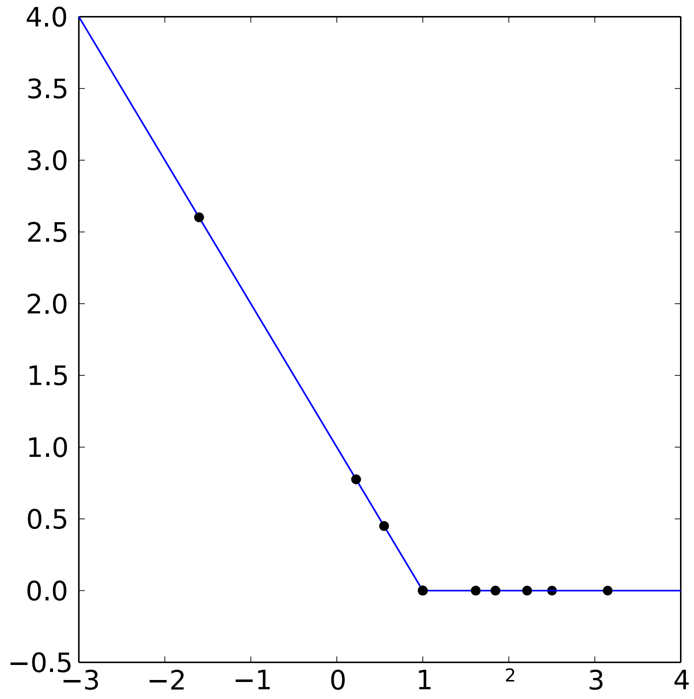


- If the classes are separable all points have 0 loss
- If they become non-separable the margin gets larger and some points have non-zero loss

Support vectors

Why the name Support Vector Machine?
Let's see what happens when we move some of the training examples (C large):

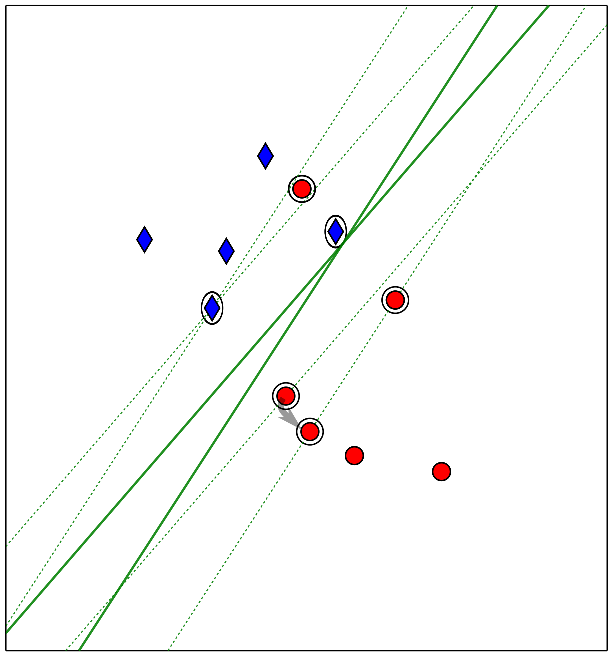


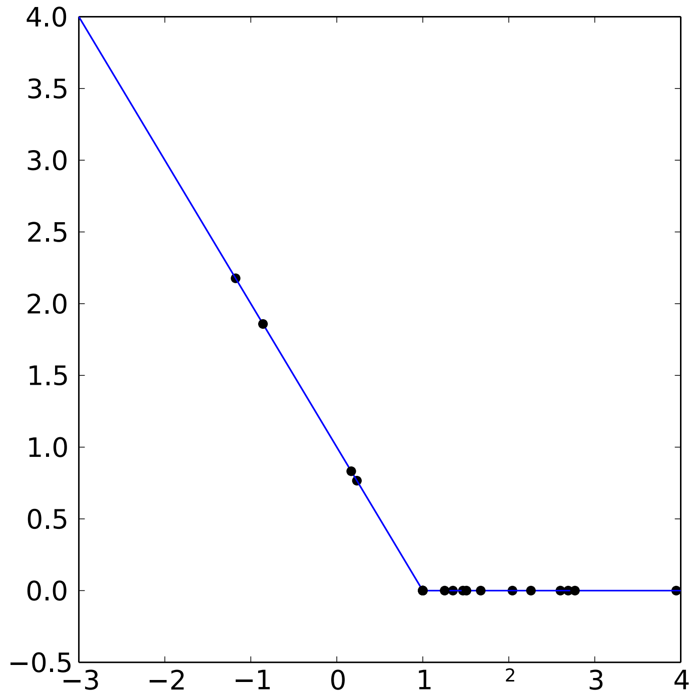


- If point moves within the correct side of the margin (with 0 loss) nothing changes

Support vectors

Why the name Support Vector Machine?
Let's see what happens when we move some of the training examples (C large):

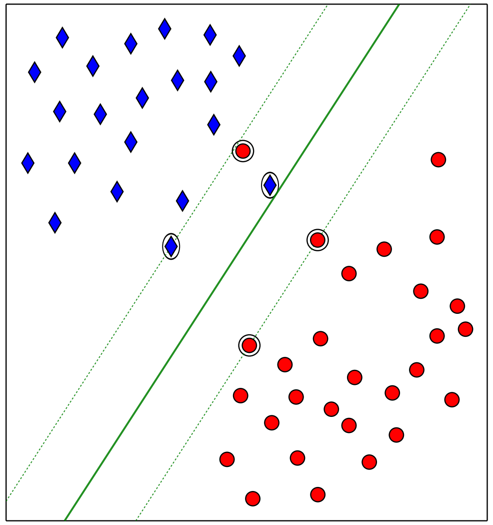


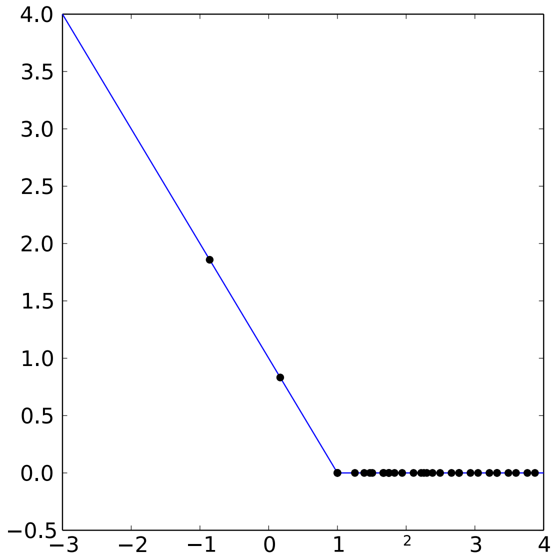


- If a misclassified point moves, the decision hyperplane and the margin change
- If a point on the boundary of the margin zone changes, the decision hyperplane and the margin change

Support vectors

Apparently, the decision hyperplane depends only on a subset of the training points . . .





those for which $r^t\left(\langle \boldsymbol{w}, \boldsymbol{x}^t \rangle - b\right) \leq 1$, i.e.:

- misclassified training points
- training points that are correctly classified with insufficient margin (including points lying exactly on the margin boundary)

These points are called support vectors.

Support vectors

Actually, thanks to the Representer theorem we can say more:

- the optimal $\hat{\boldsymbol{w}}$ is a linear combination of support vectors

$$\hat{\boldsymbol{w}} = \sum_{t=1}^{N} \alpha_t \, r^t \boldsymbol{x}^t$$

  where $\alpha_t = 0$ if $t \notin \mathcal{S}$, the set of indices of support vectors

- and, therefore, the optimal decision function is a weighted sum of the inner product between the instance to be classified and the support vectors

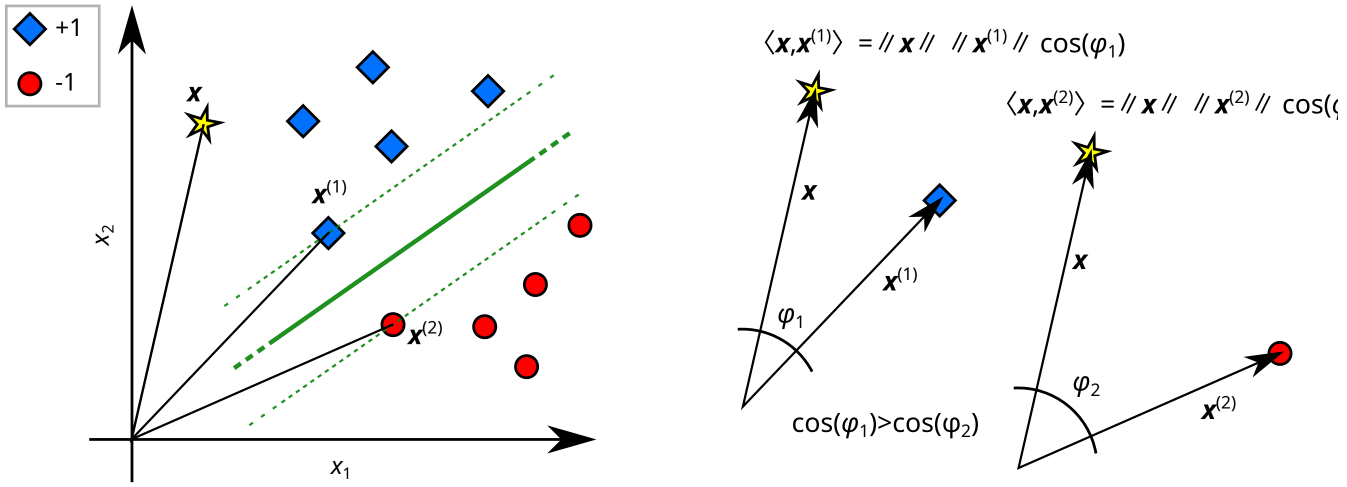
$$\hat{h}(\boldsymbol{x}) = \langle \hat{\boldsymbol{w}}, \boldsymbol{x} \rangle - \hat{b} = \sum_{t=1}^{N} \alpha_t \, r^t \langle \boldsymbol{x}^t, \boldsymbol{x} \rangle - \hat{b} = \sum_{t \in \mathcal{S}} \alpha_t \, r^t \langle \boldsymbol{x}^t, \boldsymbol{x} \rangle - \hat{b}$$

The fact that the decision function only depends on these training instances, explains the name "support vectors".

SVMs as instance-based learners

Let's have a second look at: $\hat{h}(\boldsymbol{x}) = \sum_{t \in \mathcal{S}} \alpha_t \, r^t \langle \boldsymbol{x}^t, \boldsymbol{x} \rangle - \hat{b}$

What is the intuition behind it? Why is the inner product useful for classification?



+1
-1
x
x(1)
x(2)
x2
x1
⟨x,x(1)⟩ = // x // // x(1) // cos(φ1)
⟨x,x(2)⟩ = // x // // x(2) // cos(
x
x(1)
φ1
cos(φ1)>cos(φ2)
x
φ2
x(2)

Let's assume a toy example where $b = 0$, C large, and the two support vectors are at the same distance from the origin.

Under these conditions $\hat{h}$ simplifies to: $\hat{h}(\boldsymbol{x}) = \alpha[\langle \boldsymbol{x}^{(1)}, \boldsymbol{x} \rangle - \langle \boldsymbol{x}^{(2)}, \boldsymbol{x} \rangle]$

which means that a new example $\boldsymbol{x}$ will be classified as:

- $+1$ if $\langle \boldsymbol{x}^{(1)}, \boldsymbol{x} \rangle > \langle \boldsymbol{x}^{(2)}, \boldsymbol{x} \rangle$, i.e. if $\varphi_1 < \varphi_2$
- -1 if $\langle \boldsymbol{x}^{(1)}, \boldsymbol{x} \rangle < \langle \boldsymbol{x}^{(2)}, \boldsymbol{x} \rangle$, i.e. if $\varphi_1 > \varphi_2$

We can observe that a SVM behaves like a weighted k-NN using the inner product as similarity measure and a more principled way to select the training examples retained.

Nonlinear SVMs

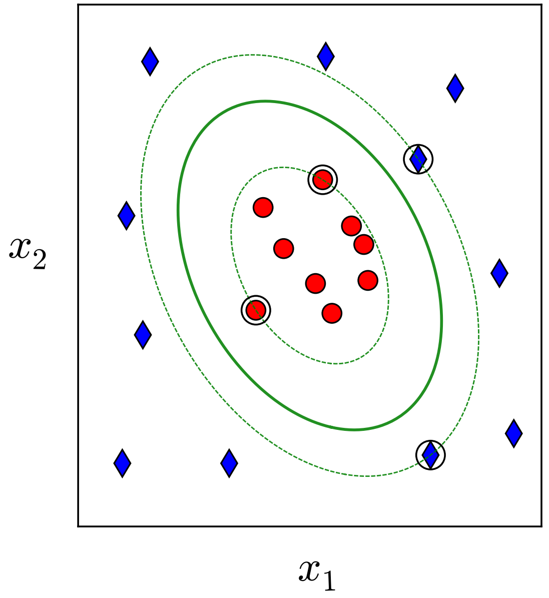
In the case of linear regression, we saw that we can "create new features" by computing nonlinear functions of the original features (e.g., powers in polynomial regression)

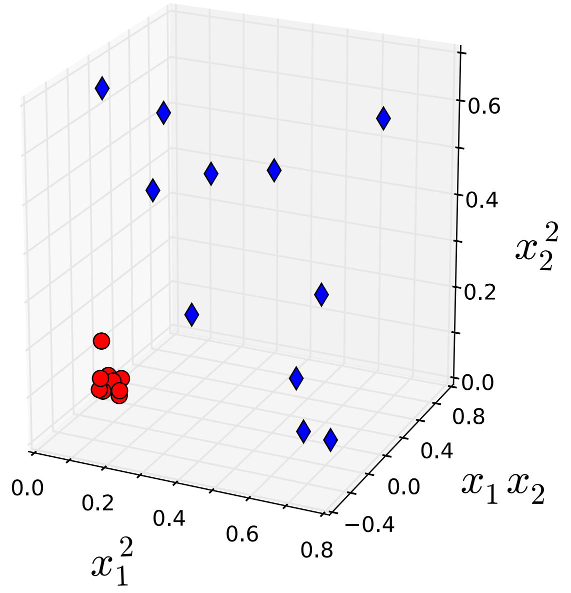
In general, we can introduce a function $\phi(\boldsymbol{x})$ that maps the original features into a new set of mapped features.

We can then train a SVM using the elements of the new feature vector $\phi(\boldsymbol{x})$ as features and build a linear classifier in the mapped feature space.

This would correspond to a non-linear classifier in the original feature space.

$$\phi(\boldsymbol{x}) = \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_2 x_2 \end{bmatrix}$$





Nonlinear SVMs (maths given as reference)

We can now take our original Soft-margin Linear SVM

$$[\hat{\boldsymbol{w}}, \hat{b}] = \arg\min_{[\boldsymbol{w},b]} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{t=1}^{N} \ell_{\mathrm{h}}(r^t\, h(\boldsymbol{x}^t|\boldsymbol{w}, b))$$

and replace the original feature vector $\boldsymbol{x}$ with the nonlinear mapping $\phi(\boldsymbol{x})$, using:

$$\boldsymbol{w} = \sum_{t=1}^{N} \alpha_t\, r^t \phi(\boldsymbol{x}^t) \quad \begin{matrix} \rightarrow \\ \searrow \end{matrix} \quad \|\boldsymbol{w}\|_2^2 = \langle \boldsymbol{w}, \boldsymbol{w}\rangle = \sum_{t=1}^{N}\sum_{s=1}^{N} \alpha_t\, \alpha_s\, r^t r^s \langle \phi(\boldsymbol{x}^t), \phi(\boldsymbol{x}^s)\rangle$$

$$h(\boldsymbol{x}|\boldsymbol{w}, b) = \langle \boldsymbol{w}, \phi(\boldsymbol{x})\rangle - b = \sum_{t=1}^{N} \alpha_t\, r^t \langle \phi(\boldsymbol{x}^t), \phi(\boldsymbol{x})\rangle - b$$

Notice that $\phi(\boldsymbol{x})$ is never required explicitly but only as argument of the inner product. Therefore, given a feature mapping $\phi(\boldsymbol{x})$, we define the corresponding Kernel as:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z})\rangle$$

Finally, we can restate the optimization problem as finding the optimal $[\hat{\boldsymbol{\alpha}}, \hat{b}]$ such that:

$$[\hat{\boldsymbol{\alpha}}, \hat{b}] = \arg\min_{[\boldsymbol{\alpha},b]} \frac{1}{2}\sum_{t=1}^{N}\sum_{s=1}^{N} \alpha_t\, \alpha_s\, r^t r^s K(\boldsymbol{x}^t, \boldsymbol{x}^s) + C\sum_{t=1}^{N} \ell_{\mathrm{h}}\left(r^t\left(\sum_{s=1}^{N} \alpha_s\, r^s K(\boldsymbol{x}^s, \boldsymbol{x}^t) - b\right)\right)$$

Although one could in principle solve this problem directly, there are more efficient approaches that recast it as a constrained optimization problem that can efficiently solved in the dual space.

Kernels

In the previous slides we introduced the Kernel function:

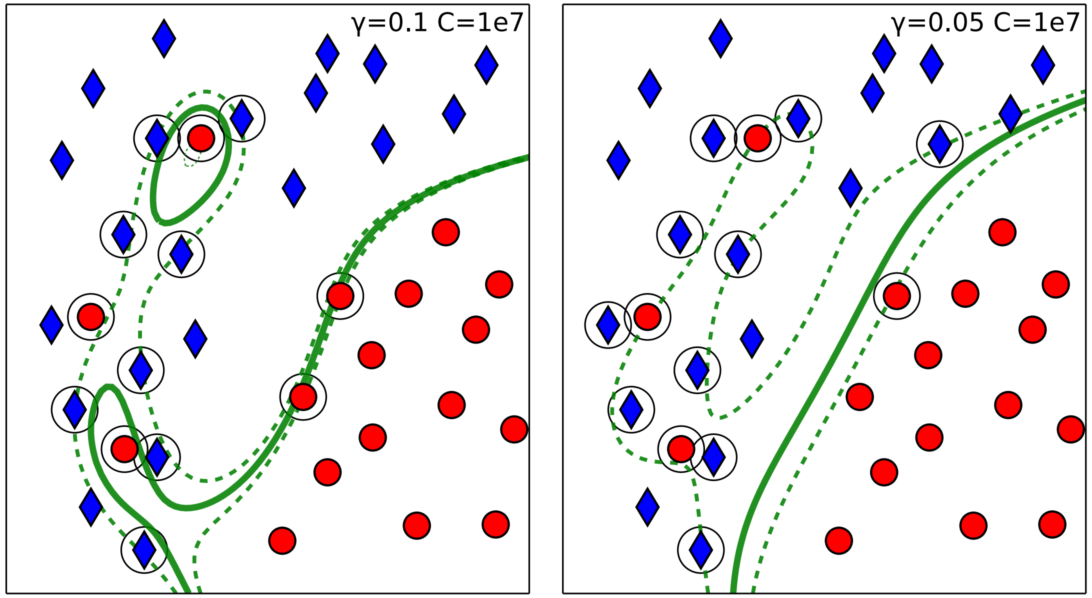
- The kernel trick allowed the use of SVMs as nonlinear classifiers.
- Like the inner product, the kernel represents a kind of similarity function.
- Given ϕ, the corresponding kernel $K(\boldsymbol{x}, \boldsymbol{z})$ can be easily computed as the inner product between $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{z})$.
- Often $K(\boldsymbol{x}, \boldsymbol{z})$ is cheaper to compute than ϕ; as the algorithm depends on $K(\boldsymbol{x}, \boldsymbol{z})$ (and only implicitly on ϕ), we can train and use a SVM in the high dimensional feature space given by ϕ without having to explicitly compute it.
  In fact, using kernels, the decision function is simply:

$$h(\boldsymbol{x}) = \sum_{s \in \mathcal{S}} \alpha_s \, r^s K(\boldsymbol{x}^s, \boldsymbol{x}) - b$$

Kernels

Examples of widely used kernels:

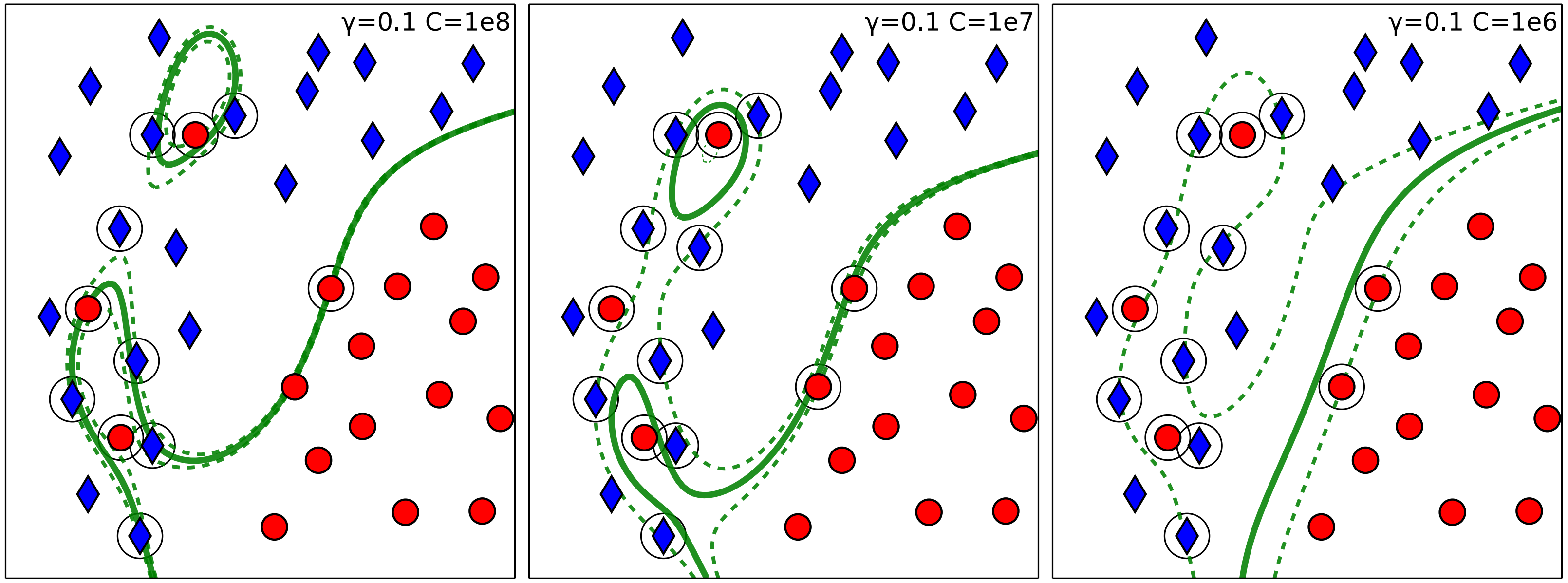
- Linear: $K(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{x}, \boldsymbol{z} \rangle$
- Polynomial (homogeneous)[3]: $K(\boldsymbol{x}, \boldsymbol{z}) = (\langle \boldsymbol{x}, \boldsymbol{z} \rangle)^d$
- Polynomial (inhomogeneous): $K(\boldsymbol{x}, \boldsymbol{z}) = (\langle \boldsymbol{x}, \boldsymbol{z} \rangle + \gamma)^d$
- Gaussian radial basis function: $K(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\gamma \left\| \boldsymbol{x} - \boldsymbol{z} \right\|^2)$, $\gamma > 0$.



[3] Our example of slide 19 corresponds to a quadratic homogeneous polynomial kernel

Regularisation

The parameter C (and also the kernel's parameters γ, d, $\ldots$) tunes the bias (underfitting) VS variance (overfitting) trade-off.
Example:



Often their optimal values need to be estimated through cross-validation.

Outline

- Linear classifier
  - Linear classifier
  - Fitting the model
- SVM
  - Margin
  - Hard-margin SVM
  - Soft-margin SVM
  - Support vectors
  - Nonlinear SVMs
  - Kernels
  - Regularisation

References

Course material reading:

Alpaydin 2010/2014 13.1!,13.2,13.3,13.4,13.5,13.6