Support Vector Machines

Nonprobabilistic Discriminative Classifiers

Contents

- Support Vector Machines
- SVM with errors in the training data
- Expansion to more than two classes
- Probabilities
- Examples
- Discussion

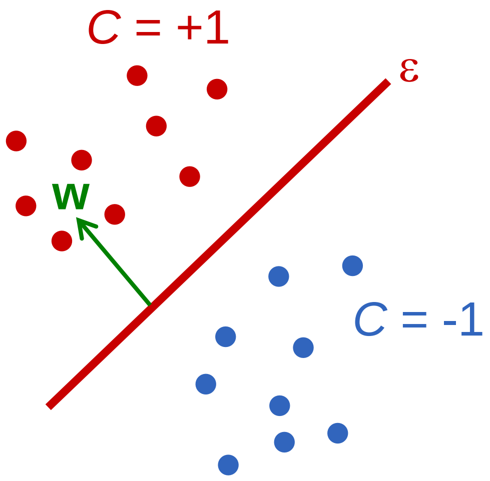
Support Vector Machines: Principle

- Binary classification : $C \in \{-1, +1\}$
- Search for hyperplane ε in feature space that seperates the classes in the training data:

  ε: $\mathbf{w}^T \cdot \mathbf{x} + b = 0$

  w: normal vector

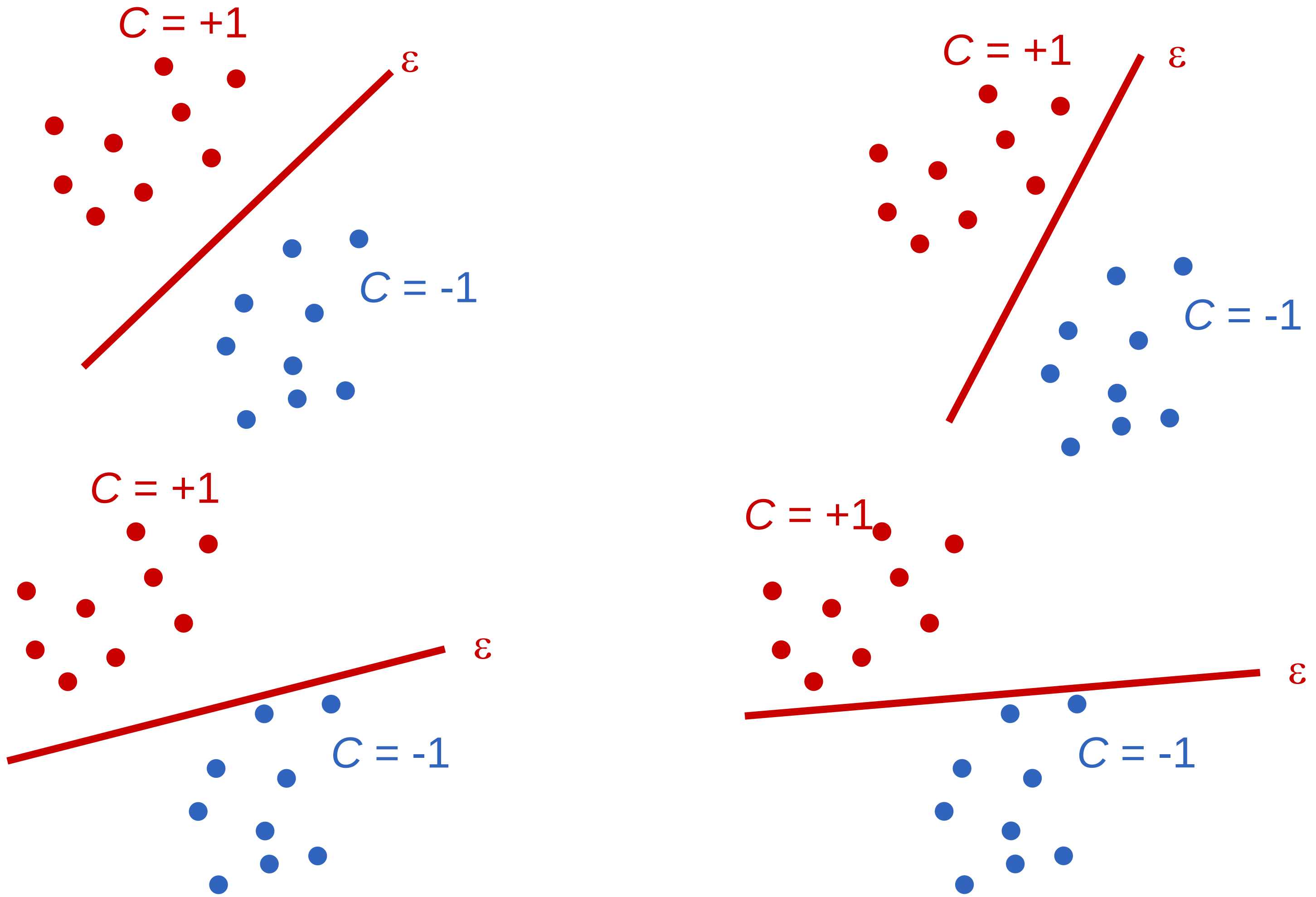
  b: constant term



- **Training:** find **w**, b
- **Classification:** $C = sign\,(\mathbf{w}^T \cdot \mathbf{x} + b)$
- But: Which is the best hyper level for given training data?

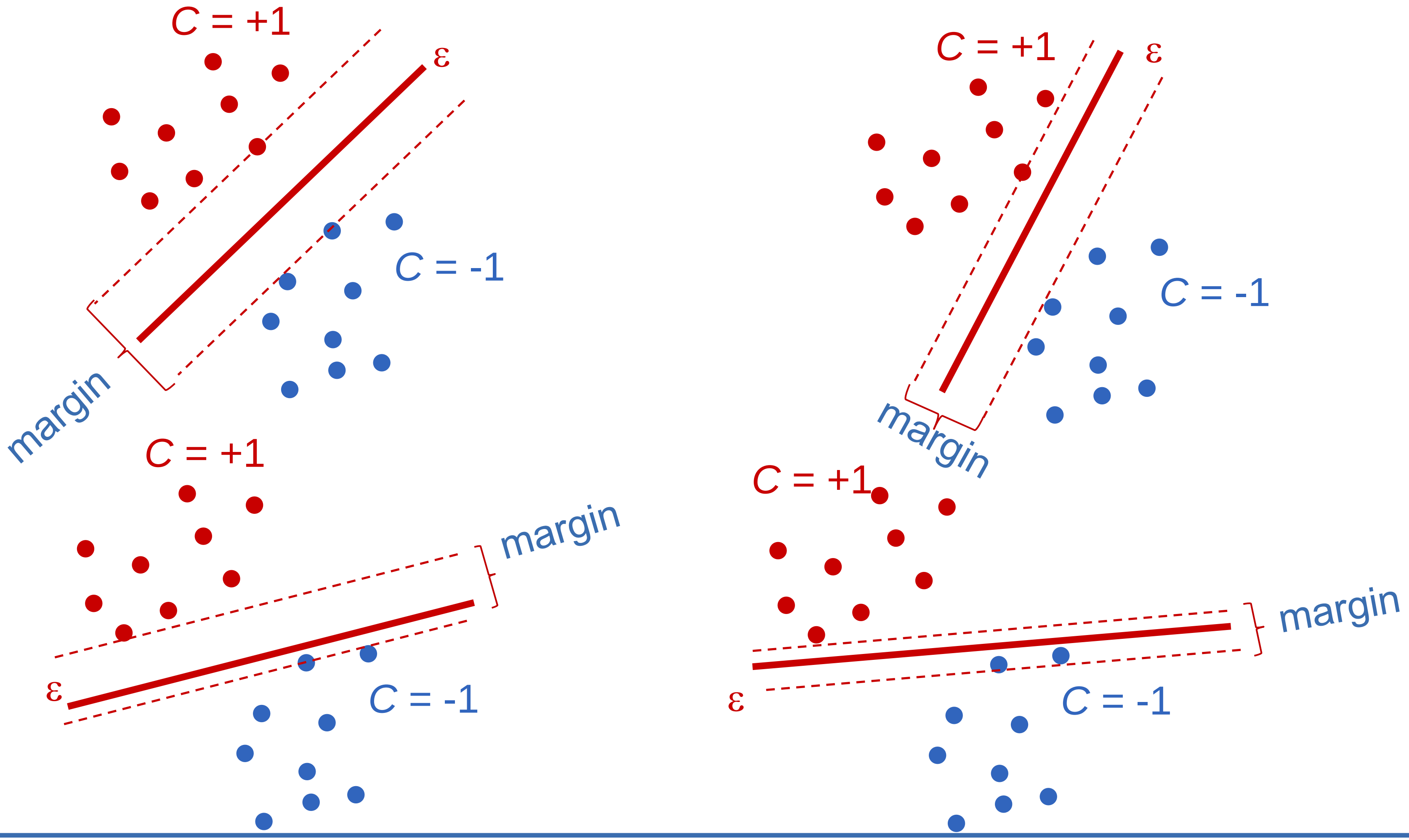
Support Vector Machines: Principle

There are many possible hyperplanes…



Support Vector Machines: Principle

- **Margin:** Region near the hyperplane without training data

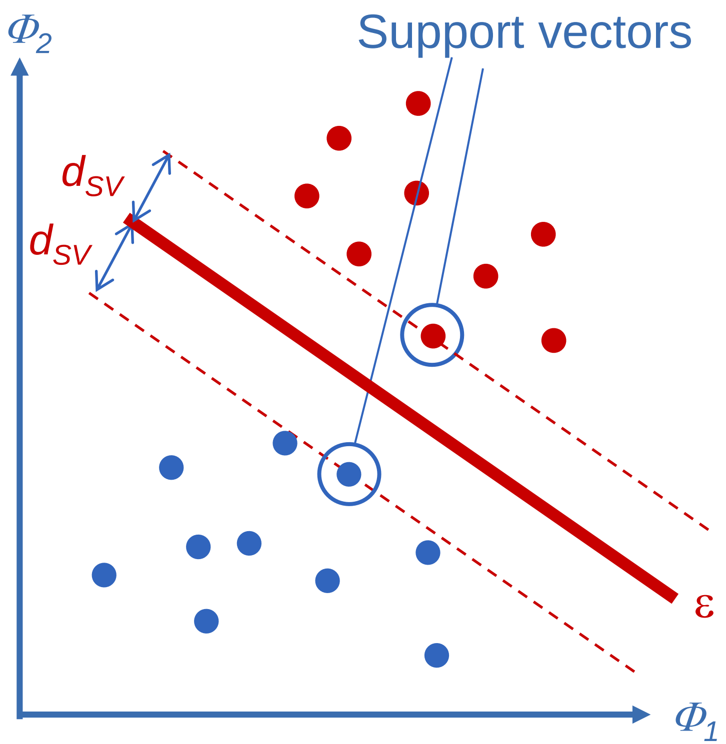


Support Vector Machines: Principle

- Maximum margin principle:
  Determine ε so that the distance d_{SV} of ε to the nearest training sample is maximized [Vapnik, 1998]

→ answer the question: Which is the best hyper level

- The points with distance d_{SV} of ε are called Support Vectors (SV)
- Result depends on SVs only
- But: before training one does not know which samples are SVs



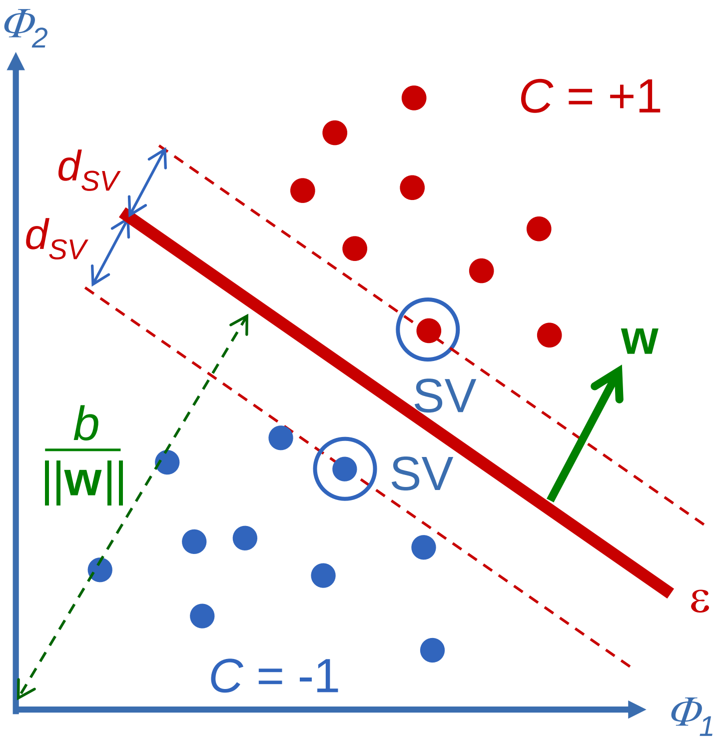
Support Vector Machines: Hyperplane

- Feature Space Mapping $\mathbf{\Phi}(\mathbf{x})$ if the classes are not linearly separable
- Binary classification: class $C \in \{-1, +1\}$
- Hyperplane in the transformed feature space:

$$\varepsilon: \mathbf{w}^T \cdot \mathbf{\Phi}(\mathbf{x}) + b = 0$$

- Distance d_n of a point $\mathbf{\Phi}(\mathbf{x}_n)$ from ε:

$$d_n = \| 1/\|\mathbf{w}\| \cdot [\mathbf{w}^T \cdot \mathbf{\Phi}(\mathbf{x}_n) + b] \|$$



- Distance of the plane from the origin: $b / \|\mathbf{w}\|$
- The length of **w** is undefined → How to scale **w**?

Support Vector Machines: Margin

- Scaling of **w** so that

$$\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b = \pm 1 \text{ for SVs}$$

- Margin is limited by two planes ε_1, ε_2 which are parallel to ε (same normal **w**)

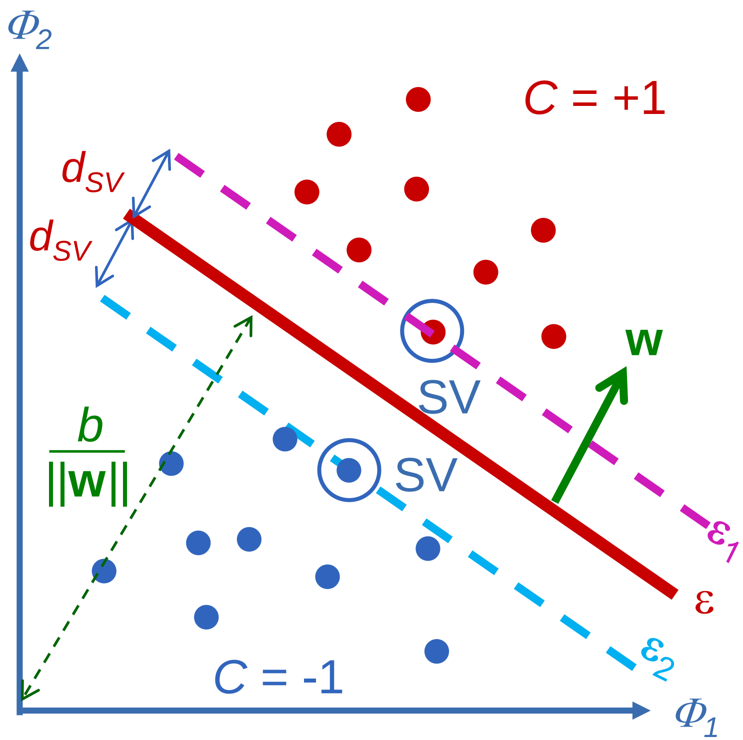
$$\varepsilon_1 : \mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b = +1$$

$$\varepsilon_2 : \mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b = -1$$

or

$$\varepsilon_1 : \mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b - 1 = 0$$

$$\varepsilon_2 : \mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b + 1 = 0$$



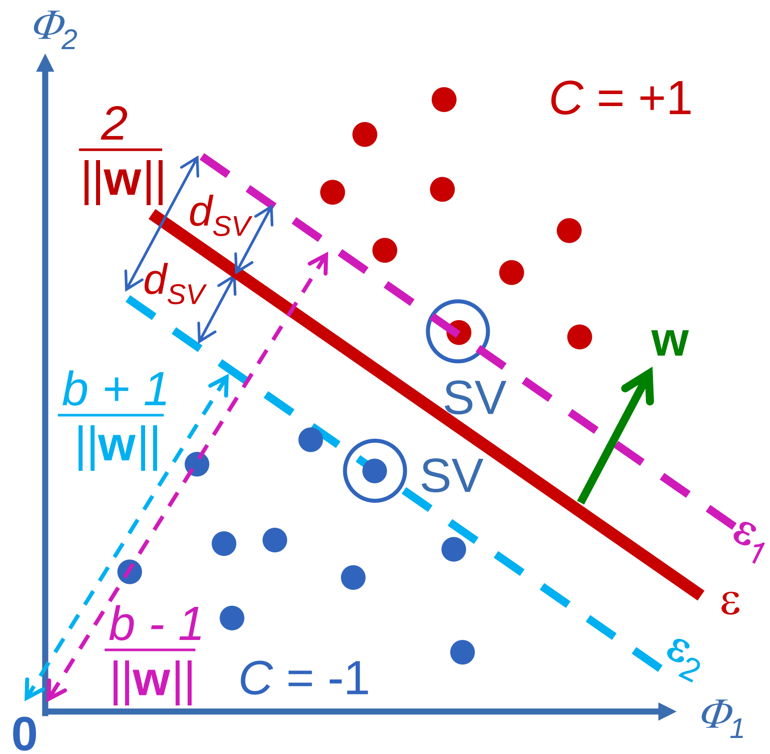
Support Vector Machines: Margin Width

- Distances of the planes ε_1, ε_2 from the origin **0**:
  - $\varepsilon_1 : \mathbf{w}^T \cdot \mathbf{\Phi}(\mathbf{x}_n) + b - 1 = 0$

    $$d\left(\varepsilon_1, \mathbf{0}\right) = \frac{b-1}{\left\|\mathbf{w}\right\|}$$

  - $\varepsilon_2 : \mathbf{w}^T \cdot \mathbf{\Phi}(\mathbf{x}_n) + b + 1 = 0$

    $$d\left(\varepsilon_2, \mathbf{0}\right) = \frac{b+1}{\left\|\mathbf{w}\right\|}$$



- Distance between the two planes: width of the margin $2 \cdot d_{SV}$

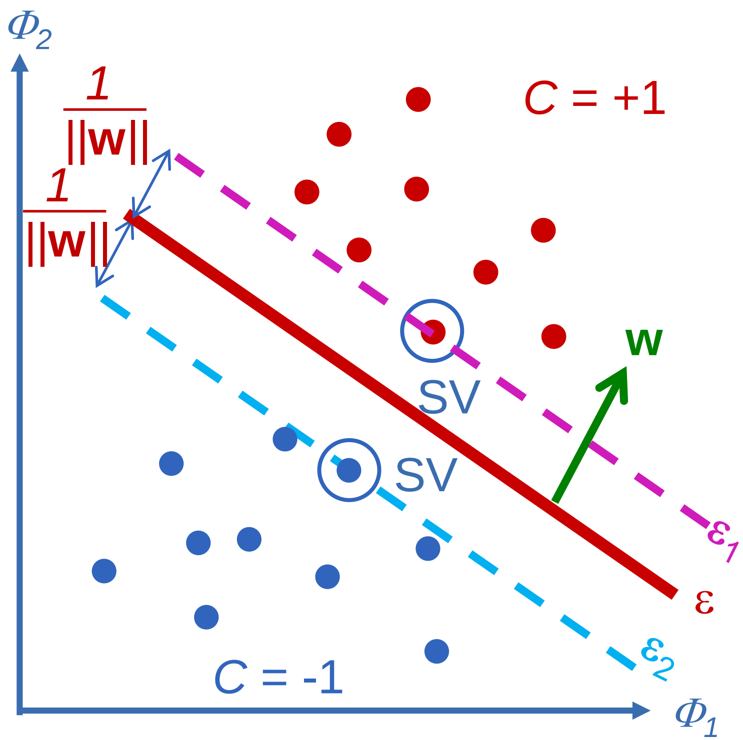
$$2 \cdot d_{SV} = d\left(\varepsilon_2, \mathbf{0}\right) - d\left(\varepsilon_1, \mathbf{0}\right) = \frac{b+1}{\left\|\mathbf{w}\right\|} - \frac{b-1}{\left\|\mathbf{w}\right\|} = \frac{2}{\left\|\mathbf{w}\right\|}$$

Support Vector Machines: Maximum Margin Criterion

- Result: If we scale **w** as defined on the previous slides, maximising the margin is equivalent to

$$\frac{1}{\|\mathbf{w}\|} \rightarrow \max$$

- Without considering the training data, this would result in **w** = **0**
- To obtain a meaningful solution, we have to introduce constraints for the training data!



Support Vector Machines: Constraints

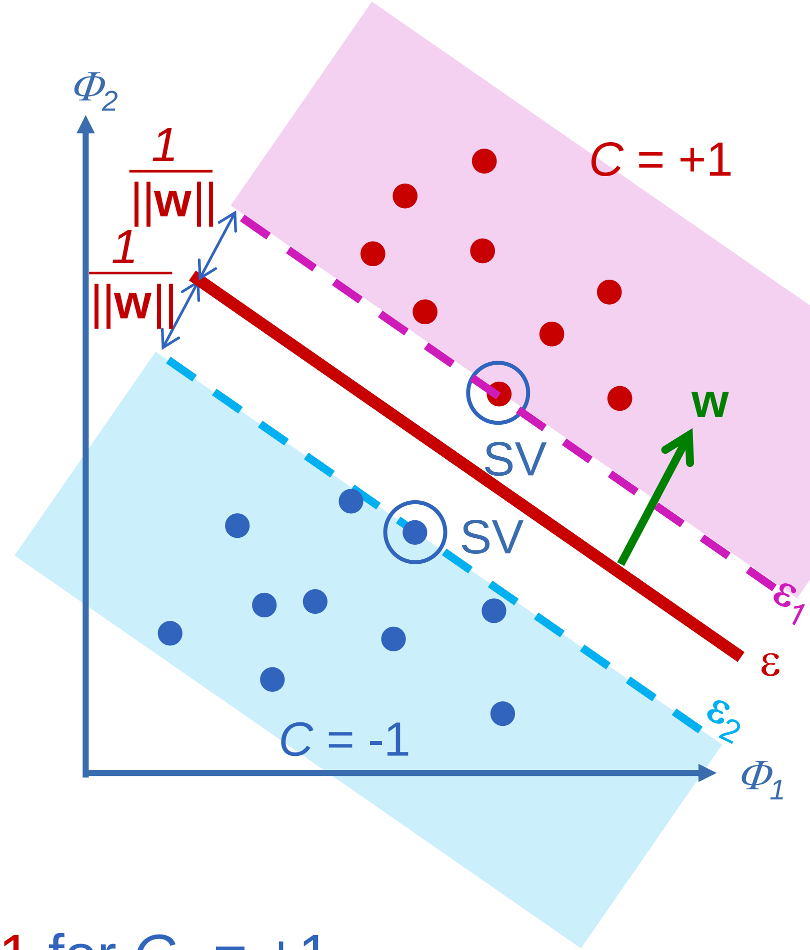
- Constraints for feature vectors $\mathbf{x}_n$ with class $C_n = +1$:
  - SVs are on plane ε_1

    → $\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b = +1$

  - All other points have to be on the side of ε_1 indicated by the direction of w (because they have to be classified correctly!)

    → $\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b > +1$

  - Consequently: $\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b \geq +1$ for $C_n = +1$
- Similarly: $\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b \leq -1$ for $C_n = -1$



Support Vector Machines: Constraints

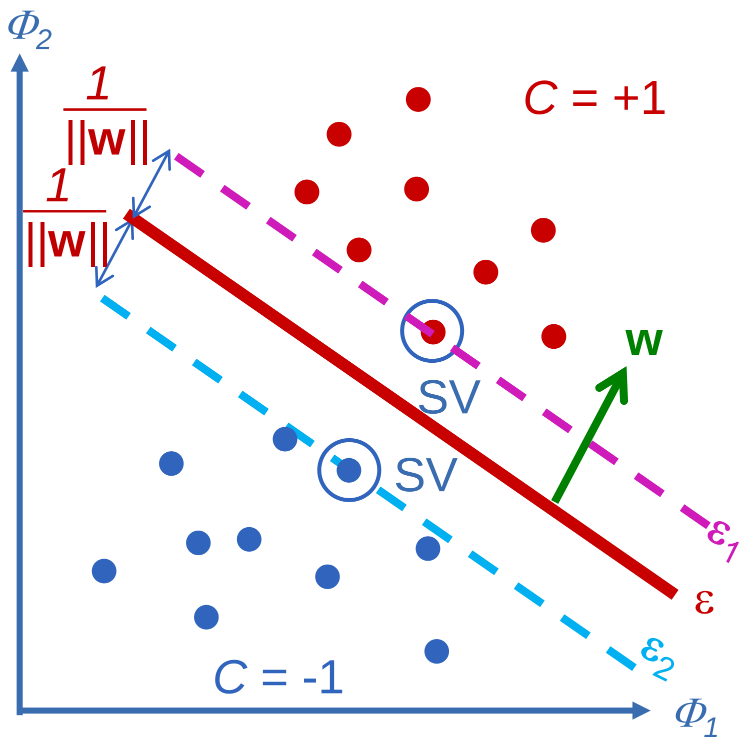
- Constraints:

$$\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b \geq +1 \text{ for } C_n = +1$$

$$\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b \leq -1 \text{ for } C_n = -1$$

- Multiplication of these inequalities by C_n yields a uniform representation for the constraints:

$$C_n \cdot [\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b] \geq 1 \quad \forall\ \mathbf{x}_n$$



- The identity applies to the support vectors

Support Vector Machines: Training

- We want to maximize the margin separating the training data given the constraints introduced by the training samples, thus

$$\frac{1}{\|\mathbf{w}\|} \rightarrow \max$$

subject to $C_n \cdot [\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b] \geq 1 \;\; \forall \; \mathbf{x}_n$

- This is mathematically difficult

- We solve the equivalent problem: $½ \cdot ||\mathbf{w}||^2 = ½ \cdot \mathbf{w}^T \cdot \mathbf{w} \rightarrow \min$ subject to the same constraints

- Optimization with inequalities as constraints
→ Lagrange multipliers $\alpha_n \geq 0$ (one for each training sample), training data comes to play in the process of searching for best hyperplane via Lagrange multipliers

Support Vector Machines: Training

- New objective function to be minimized subject to $\alpha_n \geq 0 \;\; \forall\, \mathbf{x}_\mathrm{n}$:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = ½ \cdot \mathbf{w}^\mathrm{T} \cdot \mathbf{w} - \Sigma\, \alpha_n \cdot \{C_n \cdot [\mathbf{w}^\mathrm{T} \cdot \boldsymbol{\Phi}(\mathbf{x}_\mathrm{n}) + b] - 1\}$$

- Derivatives: $dL / d\mathbf{w} = \mathbf{w} - \Sigma\, \alpha_n \cdot C_n \cdot \boldsymbol{\Phi}(\mathbf{x}_\mathrm{n})$
  $dL / db = -\Sigma\, \alpha_n \cdot C_n$

$dL / d\mathbf{w} = 0 \quad \rightarrow \quad \mathbf{w} = \Sigma\, \alpha_n \cdot C_n \cdot \boldsymbol{\Phi}(\mathbf{x}_\mathrm{n})$

$dL / db = 0 \quad \rightarrow \quad \Sigma\, \alpha_n \cdot C_n = 0$

- Substituting this result in L leads to a new objective function :

$$Z(\boldsymbol{\alpha}) = \Sigma\, \alpha_n - ½ \cdot \Sigma\,\Sigma\, \alpha_n \cdot \alpha_m \cdot C_n \cdot C_m \cdot \boldsymbol{\Phi}(\mathbf{x}_\mathrm{n})^\mathrm{T} \cdot \boldsymbol{\Phi}(\mathbf{x}_\mathrm{m}) \rightarrow \mathrm{min}$$

with additional constraints : $\alpha_n \geq 0$ and $\Sigma\, \alpha_n \cdot C_n = 0$

Support Vector Machines: Kernel Trick

- **Kernel trick:** Substitution of $\boldsymbol{\Phi}(\mathbf{x}_n)^T \cdot \boldsymbol{\Phi}(\mathbf{x}_m)$ by **Kernel function**

$$K(\mathbf{x}_n, \mathbf{x}_m) = \boldsymbol{\Phi}(\mathbf{x}_n)^T \cdot \boldsymbol{\Phi}(\mathbf{x}_m)$$

- Examples:
  - Gaussian Kernel (also called “Radial Basis Function“, RBF):

$$K_{RBF}(\mathbf{x}_n, \mathbf{x}_m) = e^{-\frac{(\mathbf{x}_n - \mathbf{x}_m)^2}{2 \cdot \sigma^2}}$$

  - Polynomial Kernel:

$$K_{poly}(\mathbf{x}_n, \mathbf{x}_m) = \left(s \cdot \mathbf{x}_n^T \cdot \mathbf{x}_m + r\right)^d$$

Support Vector Machines: Kernel Trick

- **Advantages of applying the Kernel trick**:
  - No need to define a Feature Space Mapping $\Phi(\mathbf{x})$
  - No need to explicitly calculate $\Phi(\mathbf{x})$ or $\Phi(\mathbf{x}_n)^T \cdot \Phi(\mathbf{x}_m)$
  - The feature space mapping is carried out implicitly by applying the kernel function to substitute for the inner product
  - Implicitly, one can work in very high-dimensional feature spaces without the additional computational burden

→ SVM can (in principle) deal with an arbitrary number of clusters per class in feature space!

Support Vector Machines: Training

- Substituting the kernel function for the inner product we get:

$$Z(\boldsymbol{\alpha}) = \Sigma\ \alpha_n - ½ \cdot \Sigma\ \Sigma\ \alpha_n \cdot \alpha_m \cdot C_n \cdot C_m \cdot K(\mathbf{x}_\mathrm{n}, \mathbf{x}_\mathrm{m}) \rightarrow \mathrm{min}$$

with the constraints: $\alpha_n \geq 0$ and $\Sigma\ \alpha_n \cdot C_n = 0$

- Minimizing of $Z(\boldsymbol{\alpha})$ with consideration of constraints leads to a quadratic optimization problem [Vapnik, 1998]
- The parameters to be determined are the Lagrange factors α_n
- Support Vectors: training samples $\mathbf{x}_\mathrm{n}$ with $\alpha_n > 0$!!
- The result of training is the Lagrange factors and the parameter b!

Support Vector Machines: Training

- Determination of b from Support Vectors and α_n :
  - Constraints for SV: $C_{SV} \cdot [\mathbf{w}^T \cdot \mathbf{\Phi}(\mathbf{x}_{SV}) + b\,] = 1$
  - Using $\mathbf{w} = \Sigma\, \alpha_n \cdot C_n \cdot \mathbf{\Phi}(\mathbf{x}_\mathrm{n})$

    → $C_{SV} \cdot [\Sigma\, \alpha_n \cdot C_n \cdot \mathbf{\Phi}(\mathbf{x}_\mathrm{n})^T \cdot \mathbf{\Phi}(\mathbf{x}_{SV}) + b\,] = 1$
  - Again: apply the Kernel function $K(\mathbf{x}_\mathrm{n}, \mathbf{x}_\mathrm{m})$

    → $C_{SV} \cdot [\Sigma\, \alpha_n \cdot C_n \cdot K(\mathbf{x}_\mathrm{n}, \mathbf{x}_{SV}) + b\,] = 1$
  - There is one such equation for b per support vector

    → b is calculated from each support vector once

    → Final value for b by averaging

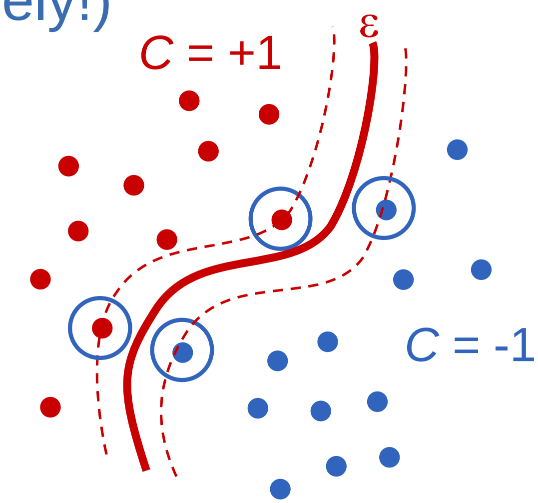
Support Vector Machines: Classification

- Classification: The class C for a feature vector $\mathbf{x}$ results from the sign of $\mathbf{w}^T \cdot \Phi(\mathbf{x}) + b$:

$$C = \text{sign}[\mathbf{w}^T \cdot \Phi(\mathbf{x}) + b] = \text{sign}[\Sigma\, \alpha_n \cdot C_n \cdot \Phi(\mathbf{x}_\text{n})^T \cdot \Phi(\mathbf{x}) + b]$$

- Again, the Kernel trick works: $C = \text{sign}[\Sigma\, \alpha_n \cdot C_n \cdot K(\mathbf{x}_\text{n}, \mathbf{x}) + b]$
- The sum only needs to be taken over SVs (because for all other training data α_n is 0, can be used inversely!)
- Transition to high dimensional feature space

  → Can deliver non-linear boundaries in the original feature space

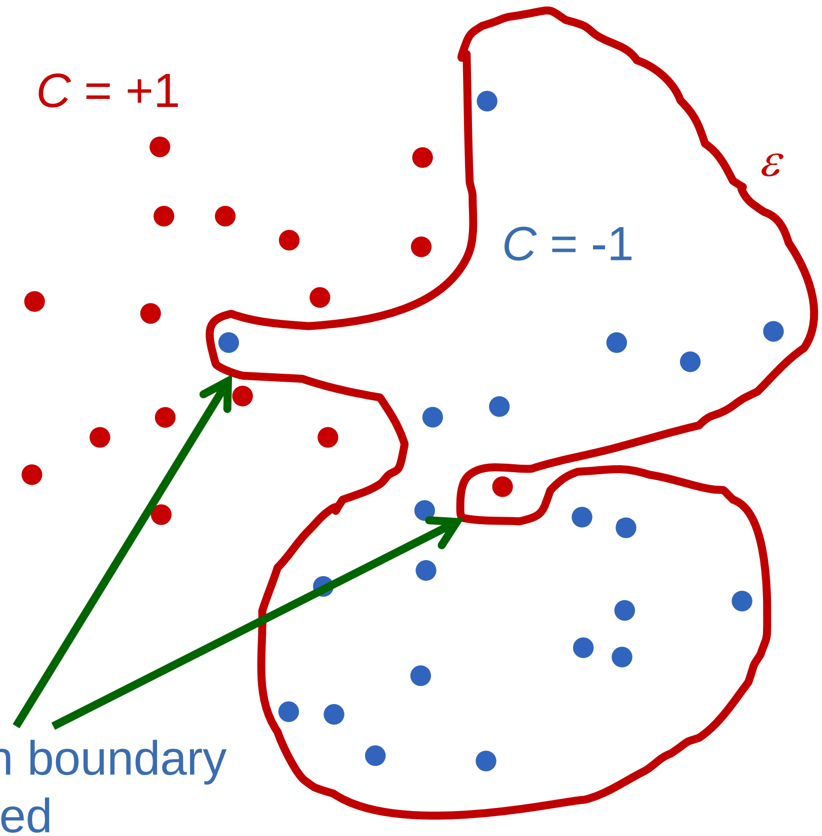


Support Vector Machines: Overfitting

- SVM can potentially separate all possible configurations of points

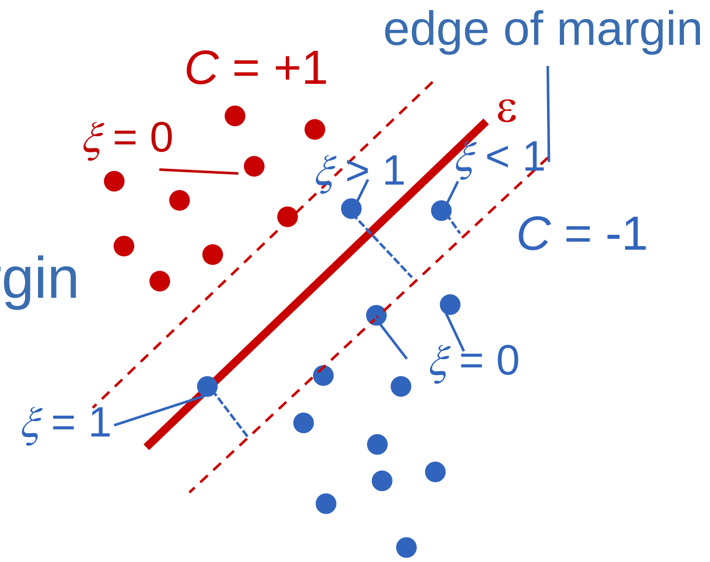
- **Danger:**
  - **Overfitting**
  - **Complex models requiring too many parameters (→ many SVs)**

→ Expansion of the model!



Very unlikely shape of the decision boundary
→ Stronger generalisation is desired

SVM with Errors in the Training Data



- Introduction of a slack variable $\xi_n \geq 0$ for every training sample with:
  - $\xi_n = 0$: Samples at the edge of the margin or in the region assigned to C_n
  - $\xi_n = 1$: Samples on ε
  - $\xi_n < 1$: Sample in the margin but on the correct side of ε
  - $\xi_n > 1$: Samples on the wrong side of ε, i.e. training samples having a wrong class label

  – For points inside the margin or on the wrong side of ε, this definition implies $\xi_n = |C_n - (\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b)|$

SVM with Errors in the Training Data

- Using these slack variables, the constraints become

$$C_n \cdot [\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b\,] \geq 1 - \xi_n$$

- Cost $P > 0$ for samples with $\xi_n > 0$ → penalise occurrence of too many outliers

- New objective function: $P \cdot \Sigma\, \xi_n + ½ \cdot ||\mathbf{w}||^2$ → min

  with constraints $\xi_n \geq 0$
  $C_n \cdot [\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b\,] \geq 1 - \xi_n$

- Lagrange multipliers $\alpha_n \geq 0$ and $\mu_n \geq 0$

- Objective function to be minimized:

$$L(\mathbf{w}, b, \alpha, \mu) = ½ \cdot \mathbf{w}^T \cdot \mathbf{w} + P \cdot \Sigma\, \xi_n - \Sigma\, \alpha_n \cdot \{C_n \cdot [\mathbf{w}^T \cdot \Phi(\mathbf{x}_n) + b] - 1 + \xi_n\} - \Sigma\, \mu_n \cdot \xi_n$$

SVM with Errors in the Training Data

- Again, we determine the first derivatives of the objective function by **w**, *b* and ξ_n and set them to zero, which leads to:

$$\mathbf{w} = \Sigma\ \alpha_n \cdot C_n \cdot \Phi(\mathbf{x}_\mathrm{n})$$

$$\Sigma\ \alpha_n \cdot C_n = 0$$

$$\mu_n = P - \alpha_n$$

- Substitution of these results in $L(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\mu})$:

$$Z(\boldsymbol{\alpha}) = \Sigma\ \alpha_n - ½ \cdot \Sigma\ \Sigma\ \alpha_n \cdot \alpha_m \cdot C_n \cdot C_m \cdot K(\mathbf{x}_\mathrm{n}, \mathbf{x}_\mathrm{m}) \rightarrow \mathrm{min}$$

with constraints: $0 \leq \alpha_n \leq P$ and $\Sigma\ \alpha_n \cdot C_n = 0$

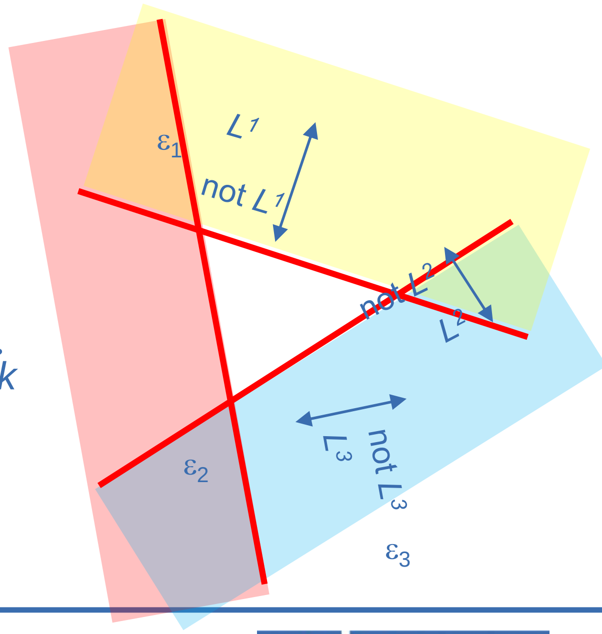
- Only difference to the case without slack variables:
  - Lagrange factors α_n also have to be $\leq P$

SVM with Errors in the Training Data

- Solution for the factors α_n by quadratic optimization
- Interpretation of α_n:
  - Samples with $\alpha_n = 0$ do not contribute to the classification
  - Samples with $\alpha_n > 0$: Support Vectors
    - $\alpha_n < P$: these samples are situated exactly at the edge of the margin, i.e. $\xi_n = 0$
    - $\alpha_n = P$ are located inside the margin or outside of the margin on the wrong side of ε.
- Calculation of b: only from support vectors with $0 < \alpha_n < P$
- **Classification :** analogous to the case without error in training data

SVM: Expansion to more than Two Classes

- SVM solves a binary problem
- There is no straight-forward expansion to the multi-class case
- **Transition to more than two classes: "one against the rest"**
  - For all classes L^k: Determine ε_k so that all classes but L^k provide the negative examples
  - Leads to N_c binary classifiers
  - Classification :
    - ➢ Classify on the basis of all hyperplanes ε_k
    - ➢ Problem: ambiguities!



SVM: Expansion to More than Two Classes

- **Transition to more than two classes: “one against one“**
- For all pairs of classes L^j, L^k : Determine ε_{jk} from the training data of both classes
  - For N_c classes → $N_c \cdot (N_c - 1) / 2$ SVM classifiers!
  - Classification:
    - ➢ Classify on the basis of all hyperplanes ε_{jk}
    - ➢ Count the votes for each class L^i and select the class receiving the largest number of votes
  - Takes longer in classification and training, can be parallelized
  - Ambiguities still exist, but they occur less frequently

Parameters

- For SVM, there are two groups of parameters that are not determined in the training procedure:

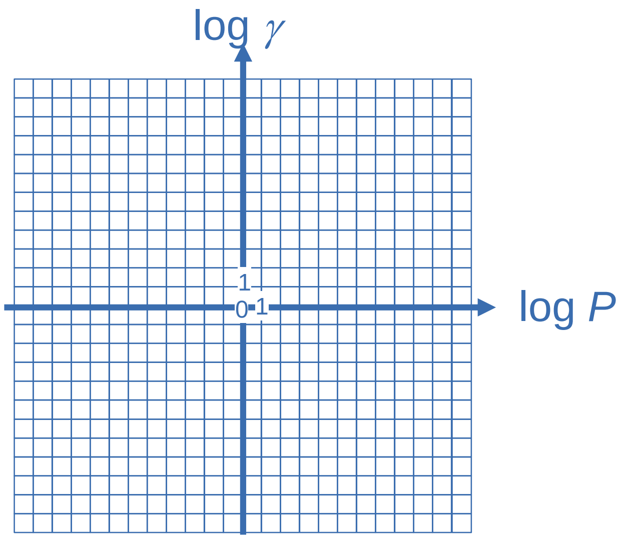
  1) Parameters of the kernel function (Gaussian kernel: $\gamma = ½ \cdot \sigma^{-2}$)

  2) Penalty term *P*

- These parameters are often specified by the user
- They should also be determined from the training data
- Approach : Grid search with cross-validation

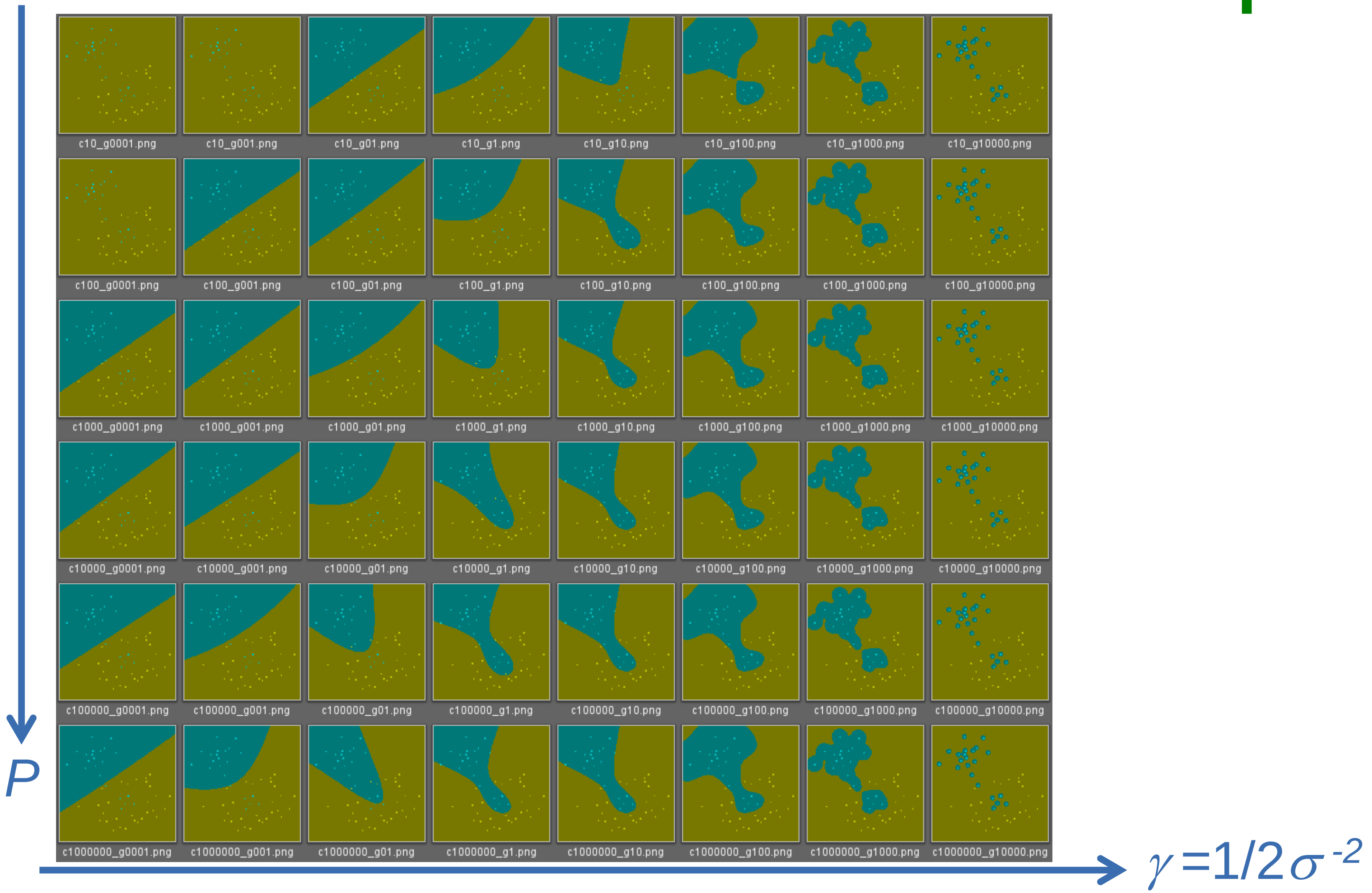
Grid Search with Cross-Validation

- In the parameter space all integer values for log P or log γ are investigated, e.g. between -15 and15



- For every value pair, a SVM is learned from the training data
- Cross-validation:
  - SVM is only trained using a part of the training data
  - From the rest of the training data, the training error is determined (number of training samples assigned to the wrong class)
- Result: The value pair for P, γ for which the training error is minimal
  - Can be refined locally

Grid Search with Cross-Validation: Example



Created using the tool on http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Probabilities

- SVM provides no probabilities
- Classification for SVM:

$$C = \mathrm{sign}[\Sigma\ \alpha_n \cdot C_n \cdot K(\mathbf{x}_\mathrm{n}, \mathbf{x}) + b] = \mathrm{sign}[f(\mathbf{x})]$$

- The function $f(\mathbf{x})$ depends on the distance d of $\Phi(\mathbf{x})$ from the decision boundary: $f(\mathbf{x}) = d \cdot \| \mathbf{w} \|$
- Note that d ís a signed distance: $C = \mathrm{sign}(d)$
- Remember: For logistic regression, the posterior probablity was determined as $\sigma(d \cdot \| \mathbf{w} \|)$

(σ: logistic Sigmoid function)

Probabilities

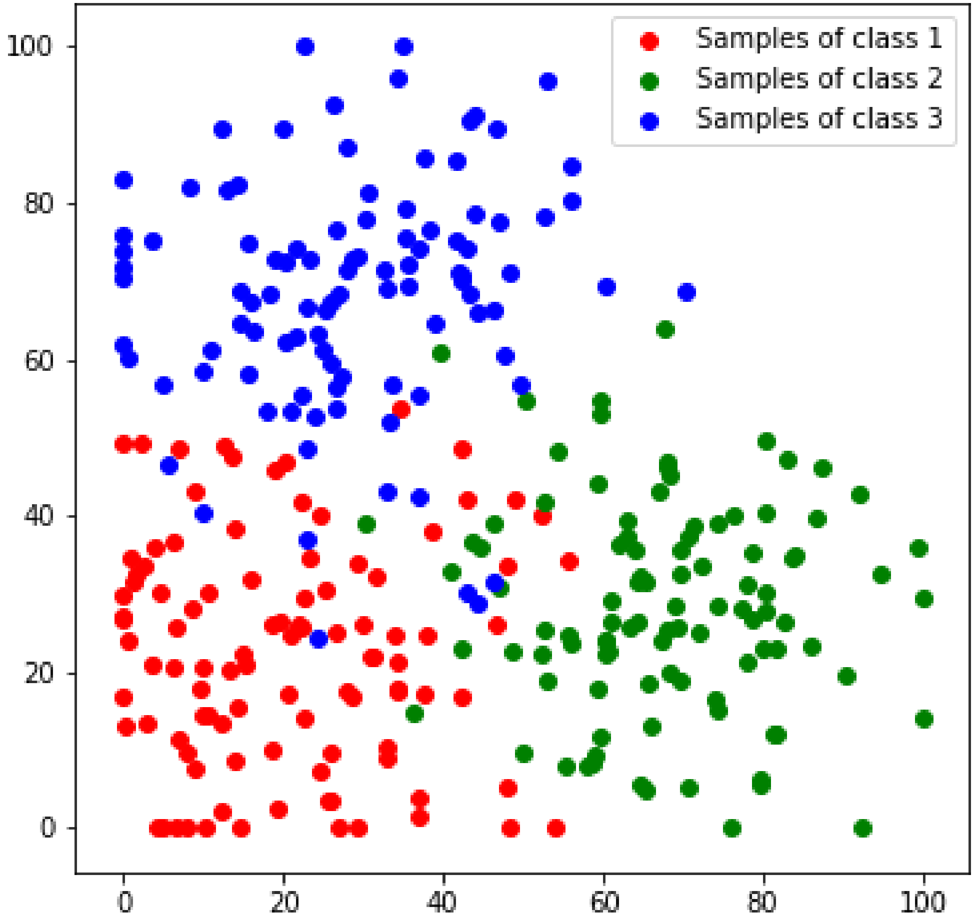
- $f(\mathbf{x})$ is a new feature and becomes an argument for the sigmoid function
- But we have to perform a linear transformation (because ||**w**|| is not known)
- Thus $P(C=1|\mathbf{x}) = \sigma(A \cdot f(\mathbf{x}) + B) = \dfrac{1}{1+e^{-(A \cdot f(\mathbf{x})+B)}}$
- The parameters A, B are learned from training data
- For that purpose, one must use training samples that are **not** used to train the SVM

→ Again, the training data are divided into two groups

Training: see logistic regression

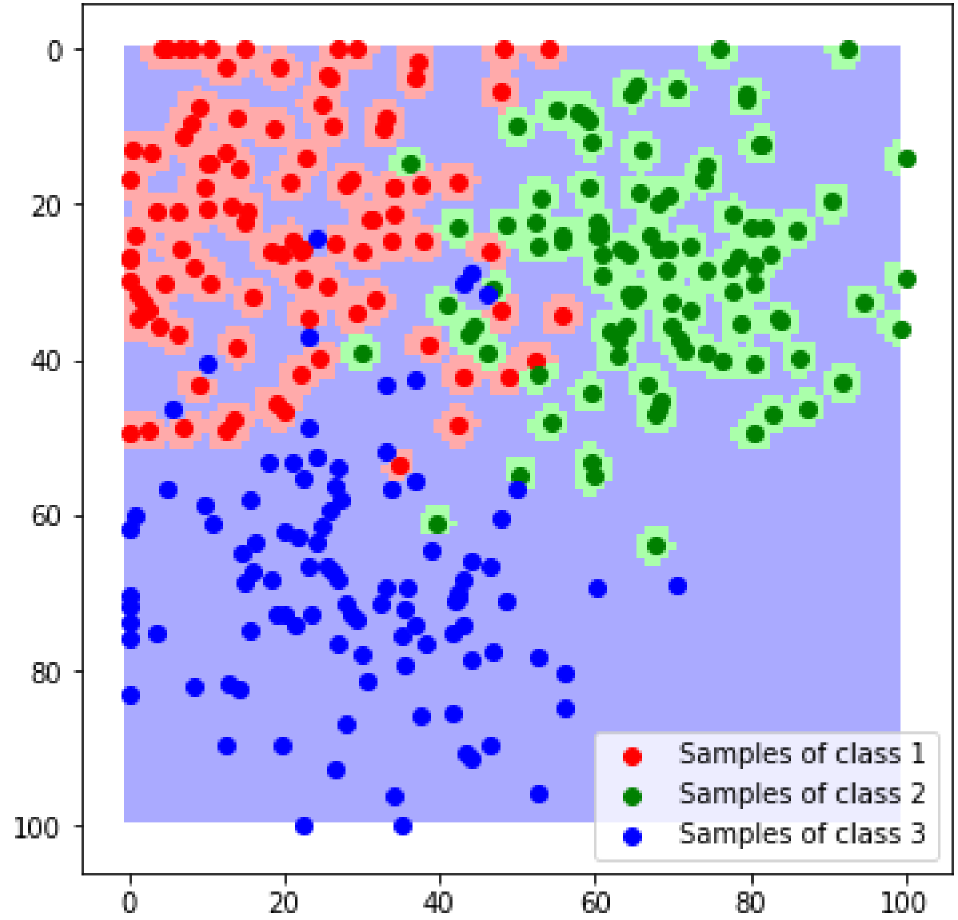
Examples

- Data set with 3 classes



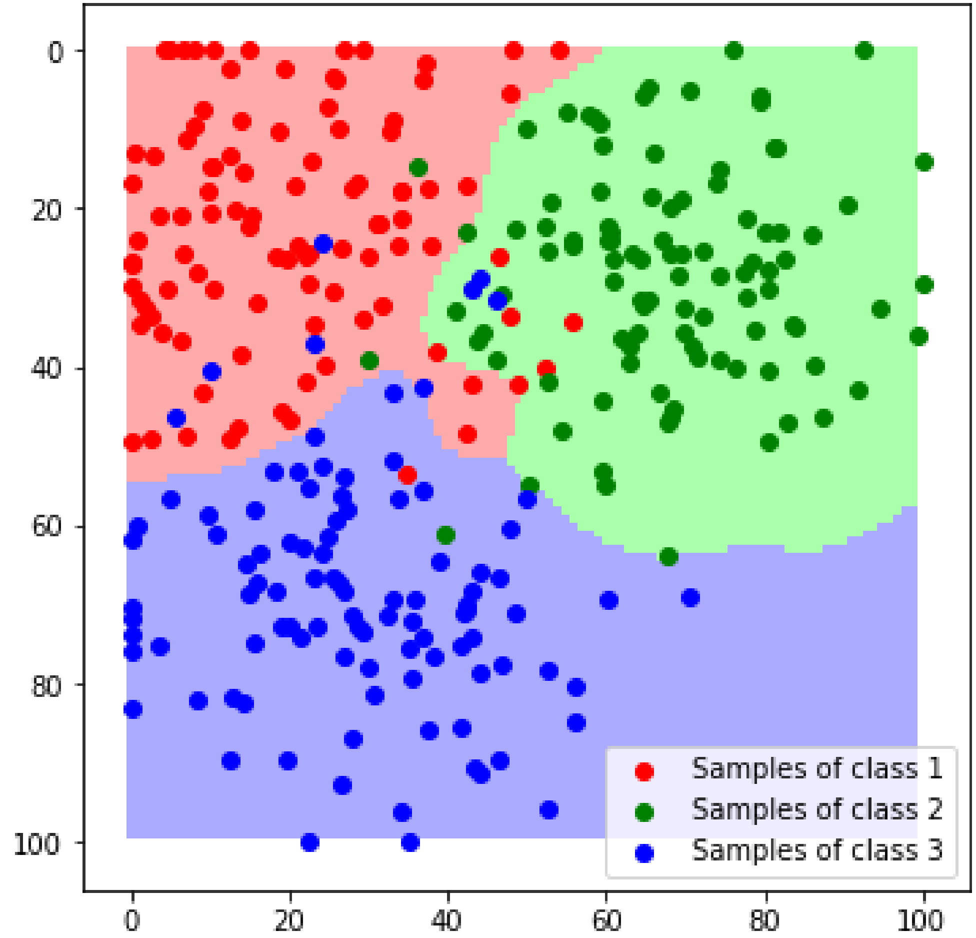
Examples

- SVM trained with **gamma = 1.0**
- Weak regularization
- Results in a strongly overfitted model



Examples

- SVM trained with **gamma = 0.01**
- A lower coefficient leads to a stronger regularization
- Here this leads to a much better model
- In general the hyperparameters should be optimized e.g. in a grid search



Support Vector Machines: Discussion

- SVM with Gauss-Kernel and slack variables provide good results
- SVM often serve as a baseline for comparison with other procedures
- Parameters of the kernel function and *P* must be determined
- Both of these parameters affect the smoothing of the decision boundary
- **Problems of SVM:**
  - The transition to more than two classes not obvious
  - Derivation of a quality indicator for the result
  - SVM is slow compared to Random Forests, especially during training