Support vector machines

MATH 6312
Department of mathematics, UTA

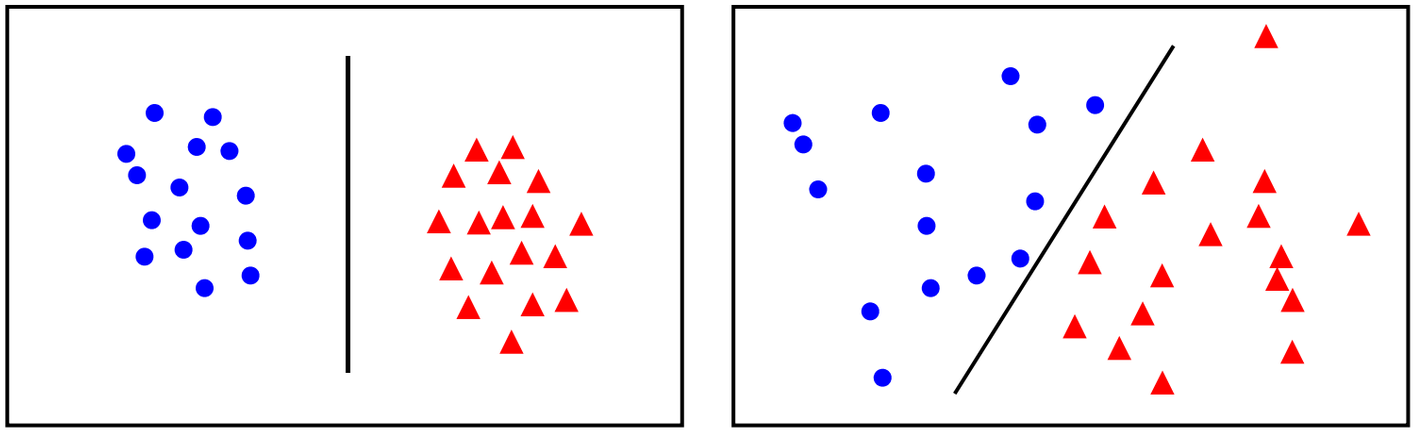
ESL 4.5, 4.5.1-2, 12.1-3, 12.3, 12.3.1-2, 12.3.6-7

Optional: ESL 12.3.3-5

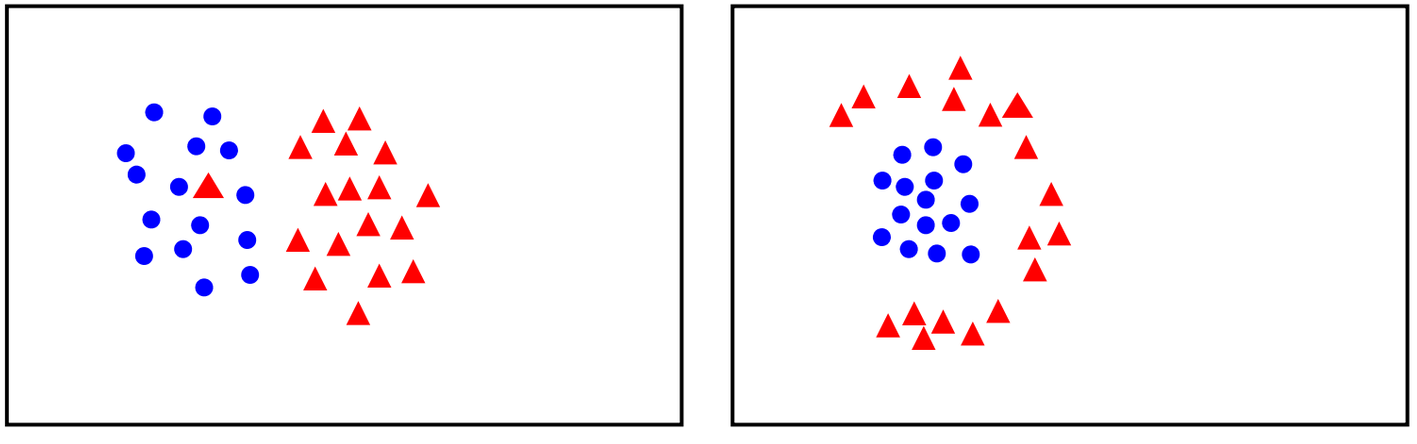
Binary classification

- For $y_i \in \{-1,1\}$, we want to learn a classifier $f(x)$ such that $f(x) \geq 0$ if $y=1$; $f(x)<0$ if $y=-1$.
  - Q. If $y_i \cdot f(x_i)>0$, do you correctly classify the i*th* response?

linearly separable

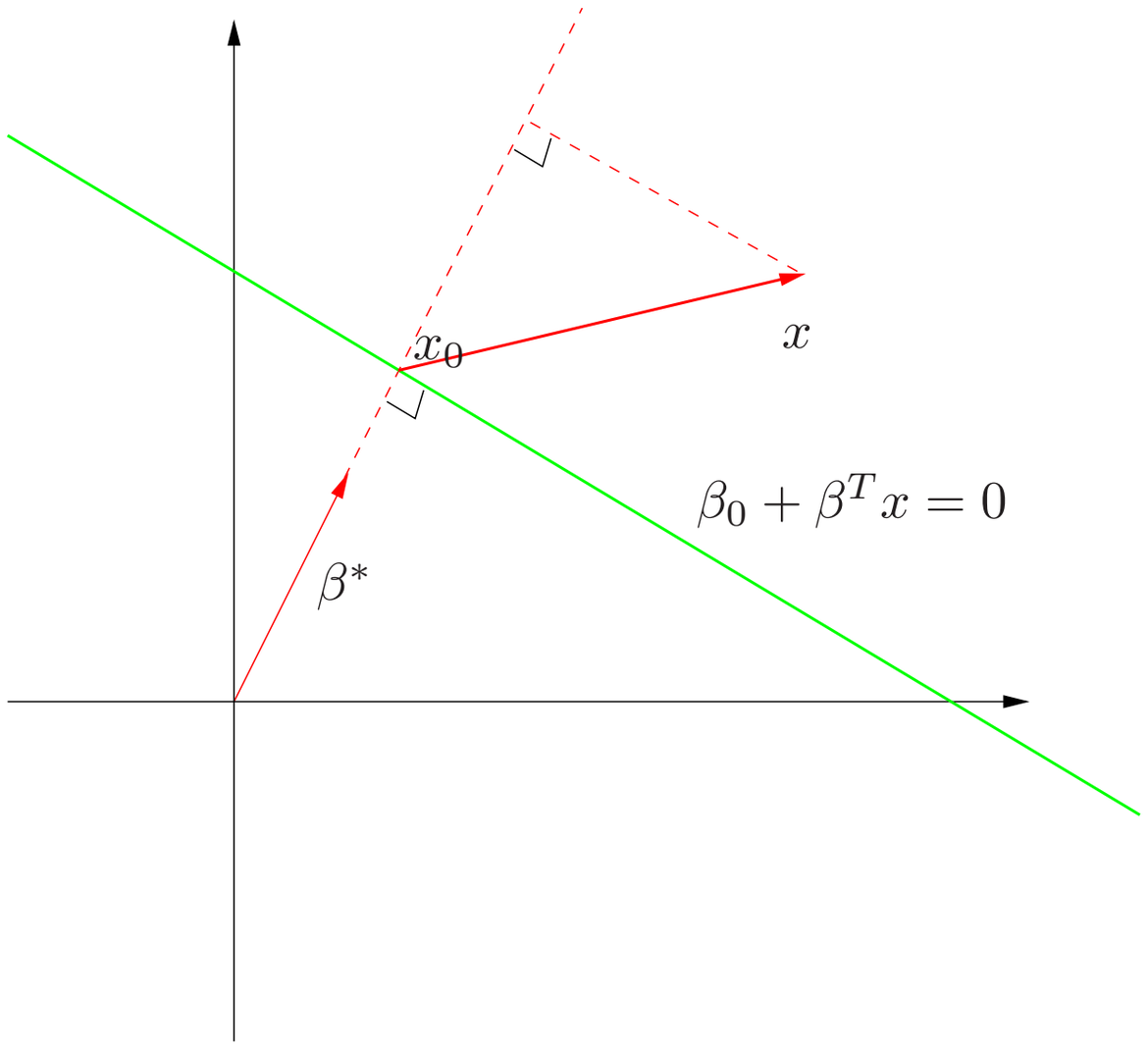


not linearly separable



What is a hyperplane?

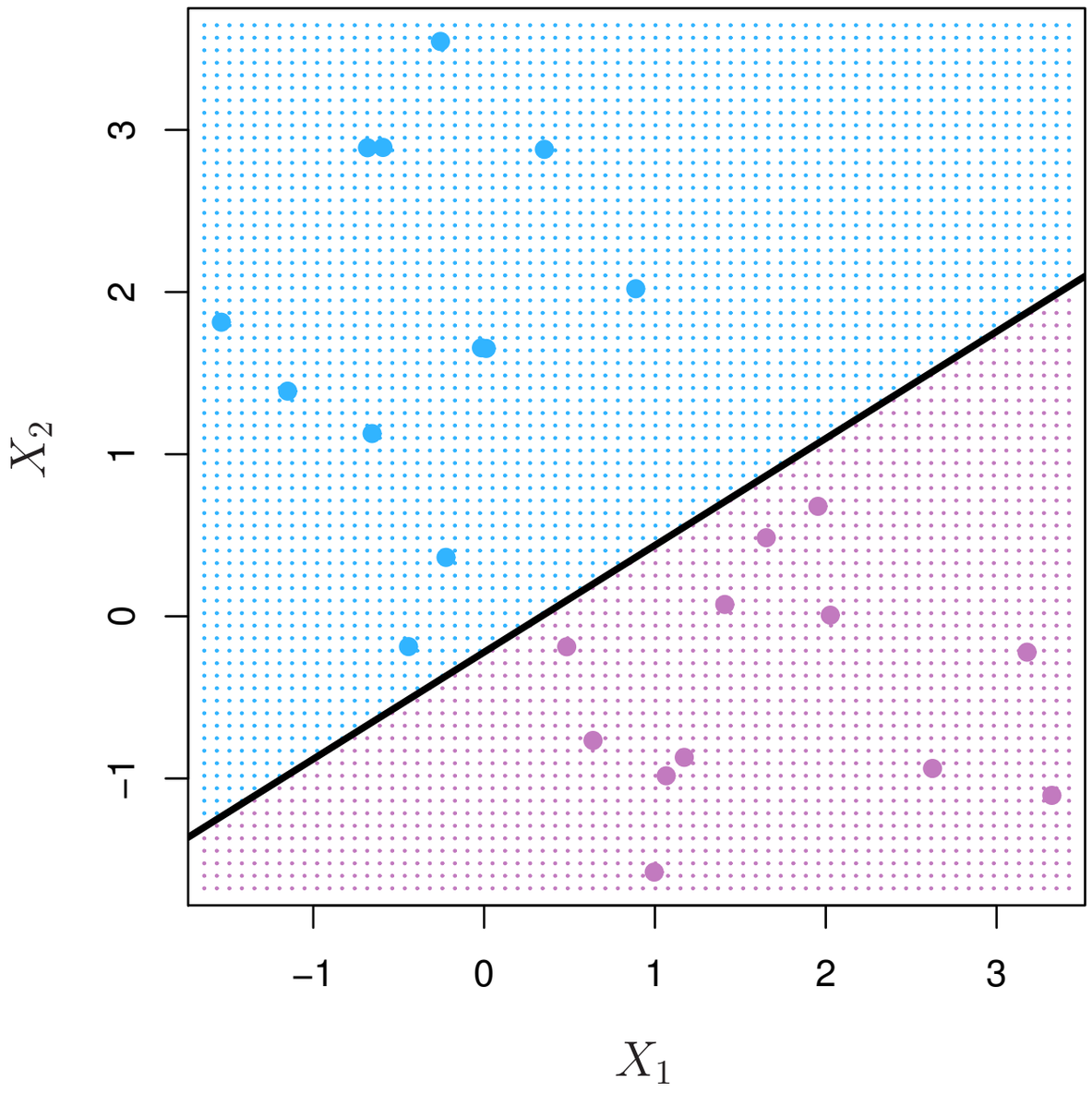
- A hyperplane in p dimensions is a flat affine subspace of dimension p−1. Let $\beta, x \in \mathbb{R}^p$. In general, the equation for a hyperplane has the form $x^T\beta + \beta_0 = 0$.



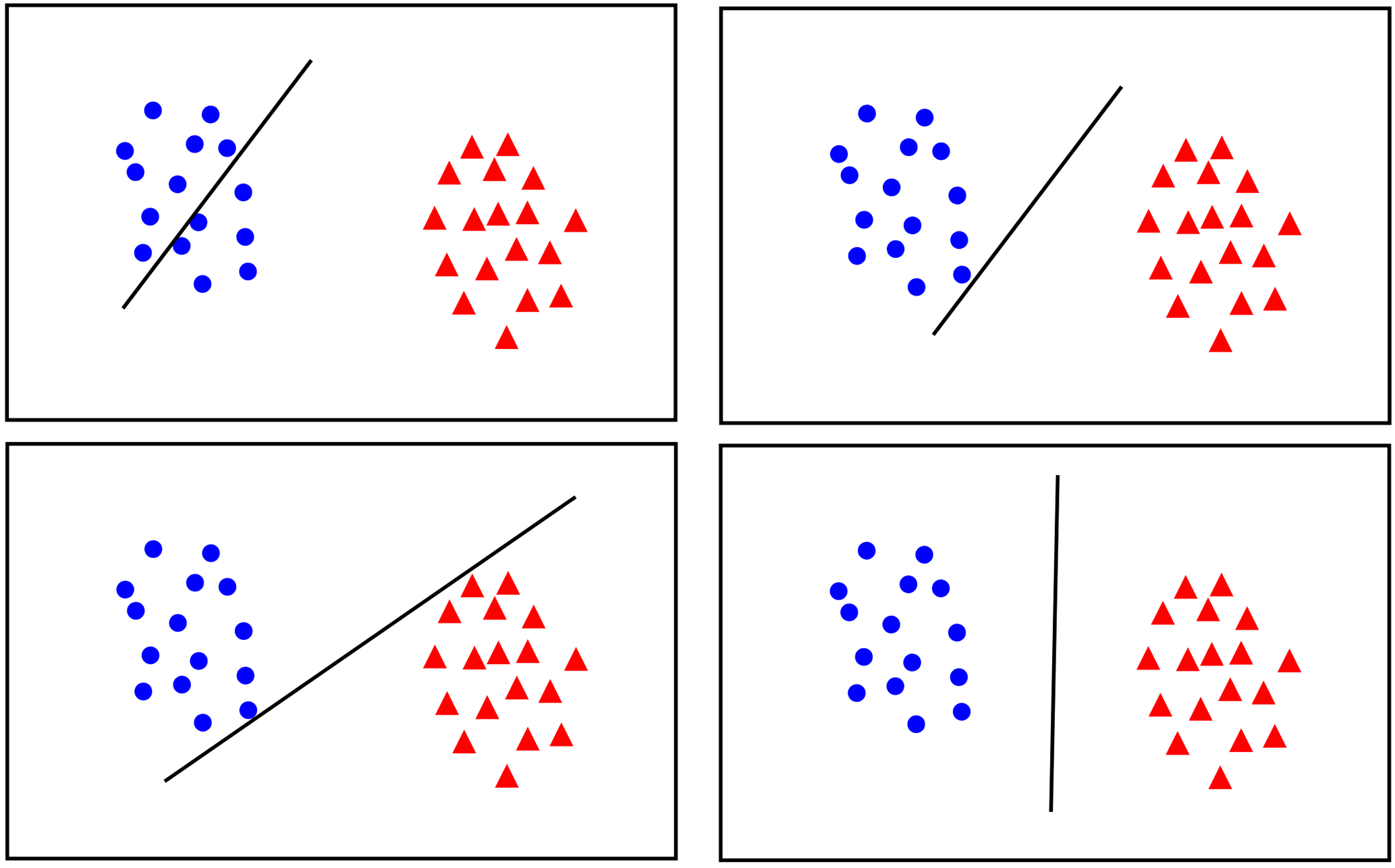
- The vector β is called the normal vector — it points in a direction orthogonal to the surface of a hyperplane. If $\beta_0 = 0$, the hyperplane goes through the origin.

- e.g. a hyperplane in two dimension (line) on the right.

Separating hyperplane

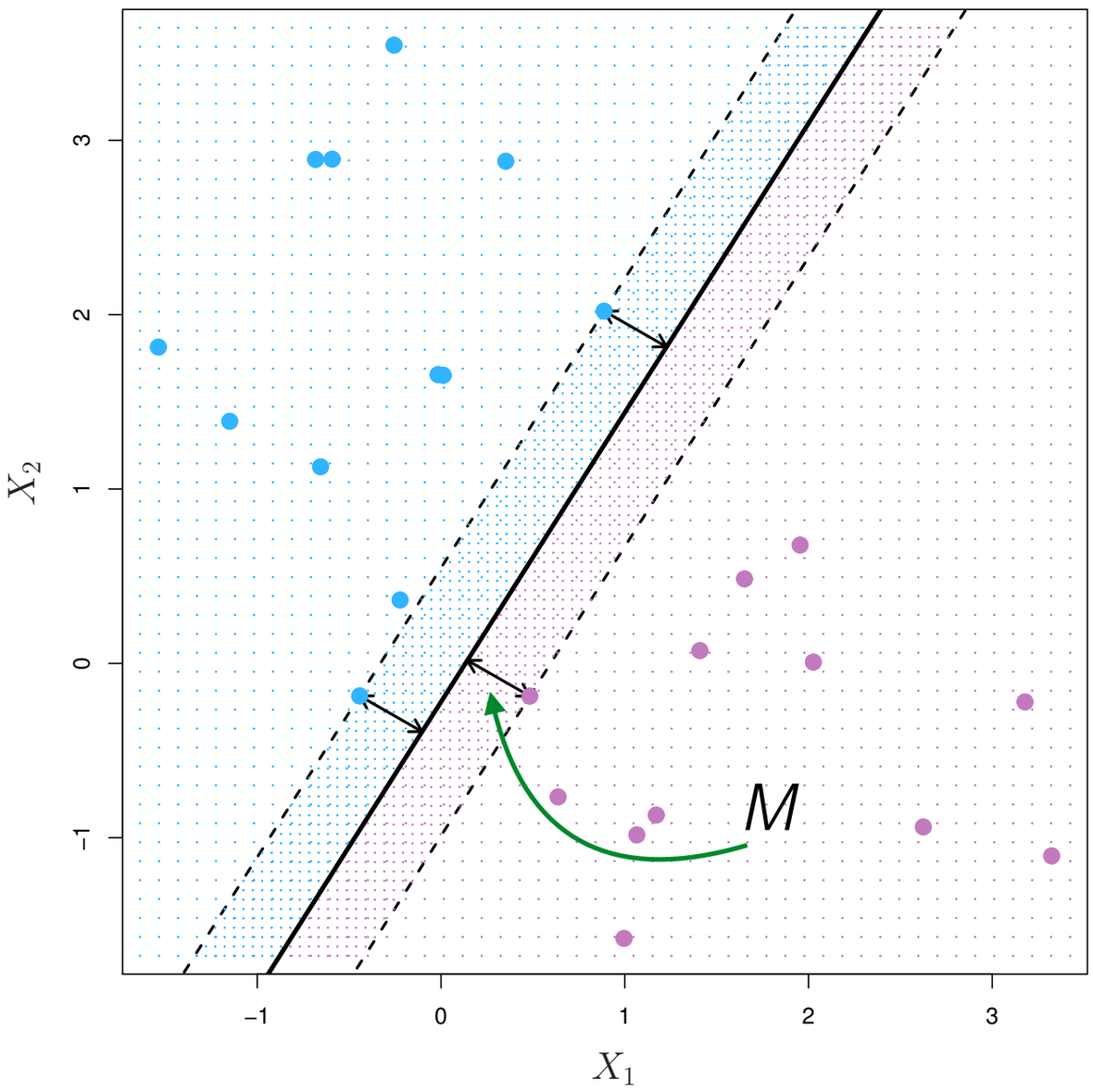


- $f(x)=x^T\beta + \beta_0$, then $f(X)>0$ for points on one side of the hyperplane, and $f(X)<0$ for points on the other.
- If we code the colored points as $y_i=+1$ for blue and $y_i=-1$ for purple, then if $y_i \cdot f(x_i)>0$ for all i, $f(x)=x^T\beta + \beta_0=0$ defines a separating hyperplane.
- Classifiers such as $f(x)=x^T\beta + \beta_0$ that compute a linear combination of features and return the sign is called perceptrons.



- Which one is not a separating hyperplane?
- Which separating hyperplane is the best? (i.e. one that is the most stable under perturbations of the inputs).

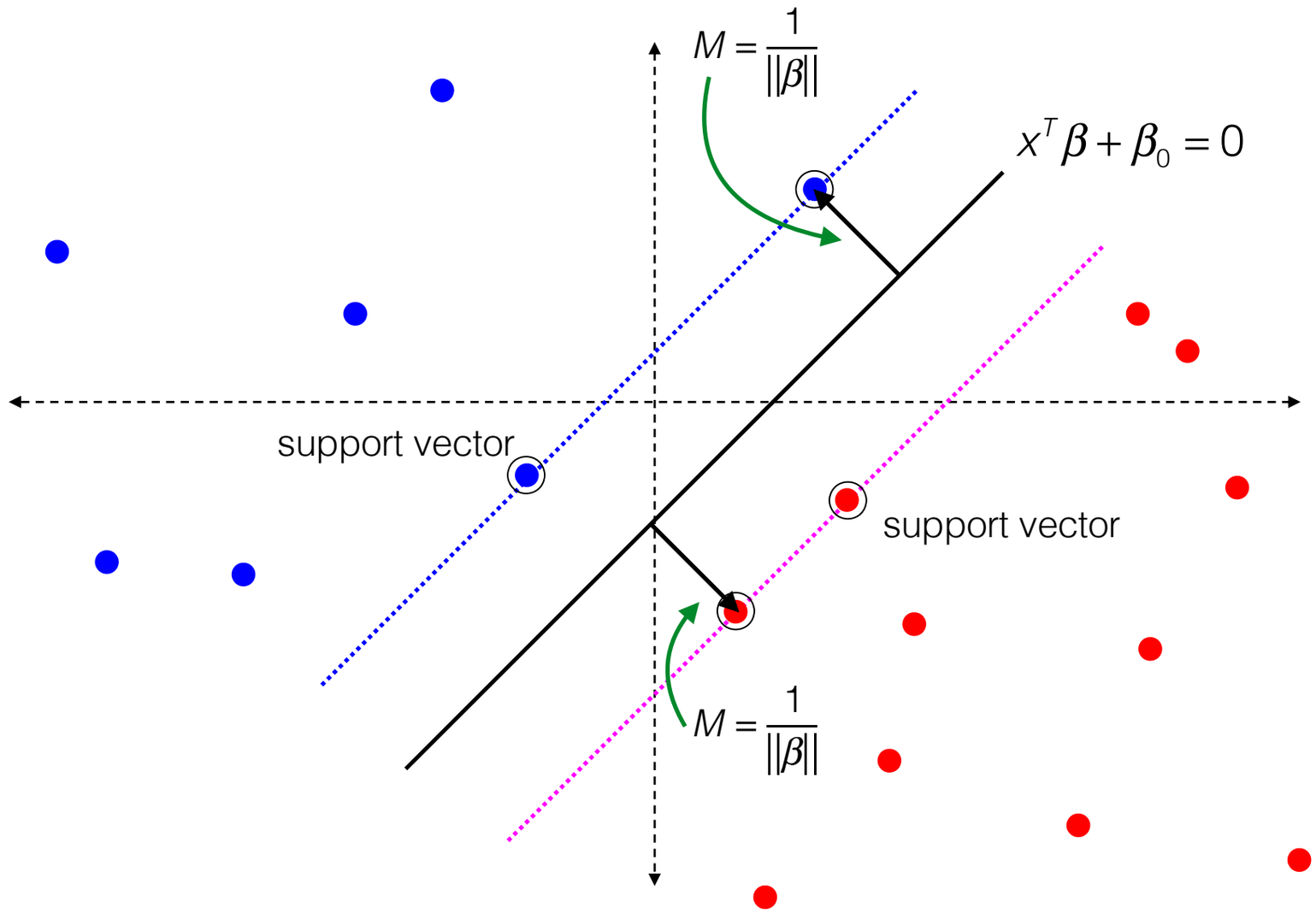
Maximal margin classifier (=(=support vector classifier)



- There may be infinitely many possible separating hyperplanes.
- Among all separating hyperplanes, find the one that makes the biggest margin (gap) between the two classes.

$$\max_{\beta,\beta_0,||\beta||=1} M$$

$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M,\ i = 1,\ldots,N.$$



$$\max_{\beta,\beta_0,||\beta||=1} M$$

subject to $y_i(x_i^T\beta + \beta_0) \geq M,\ i = 1, \ldots, N.$

$$\frac{1}{||\beta||} y_i(x_i^T\beta + \beta_0) \geq M$$

$$||\beta|| = 1/M$$

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2$$

subject to $y_i(x_i^T\beta + \beta_0) \geq 1,\ i = 1, \ldots, N.$

Find the optimal separating hyperplane

- Maximal margin solution:

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2$$

$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1,\ i = 1, \ldots, N.$$

- Lagrangian (primal) function:

$$L_P = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i[y_i(x_i^T\beta + \beta_0) - 1]. \quad \text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1,\ \alpha_i \geq 0\ \forall i$$

- Partial derivatives w.r.t β and β_0 are zeros at the optimum:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i,$$

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

- Dual representation of Lagrangian (primal) function

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

- Kraus-Kuhn-Tucker (KKT) condition: for all i=1,…,N

$$\alpha_i \geq 0$$

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

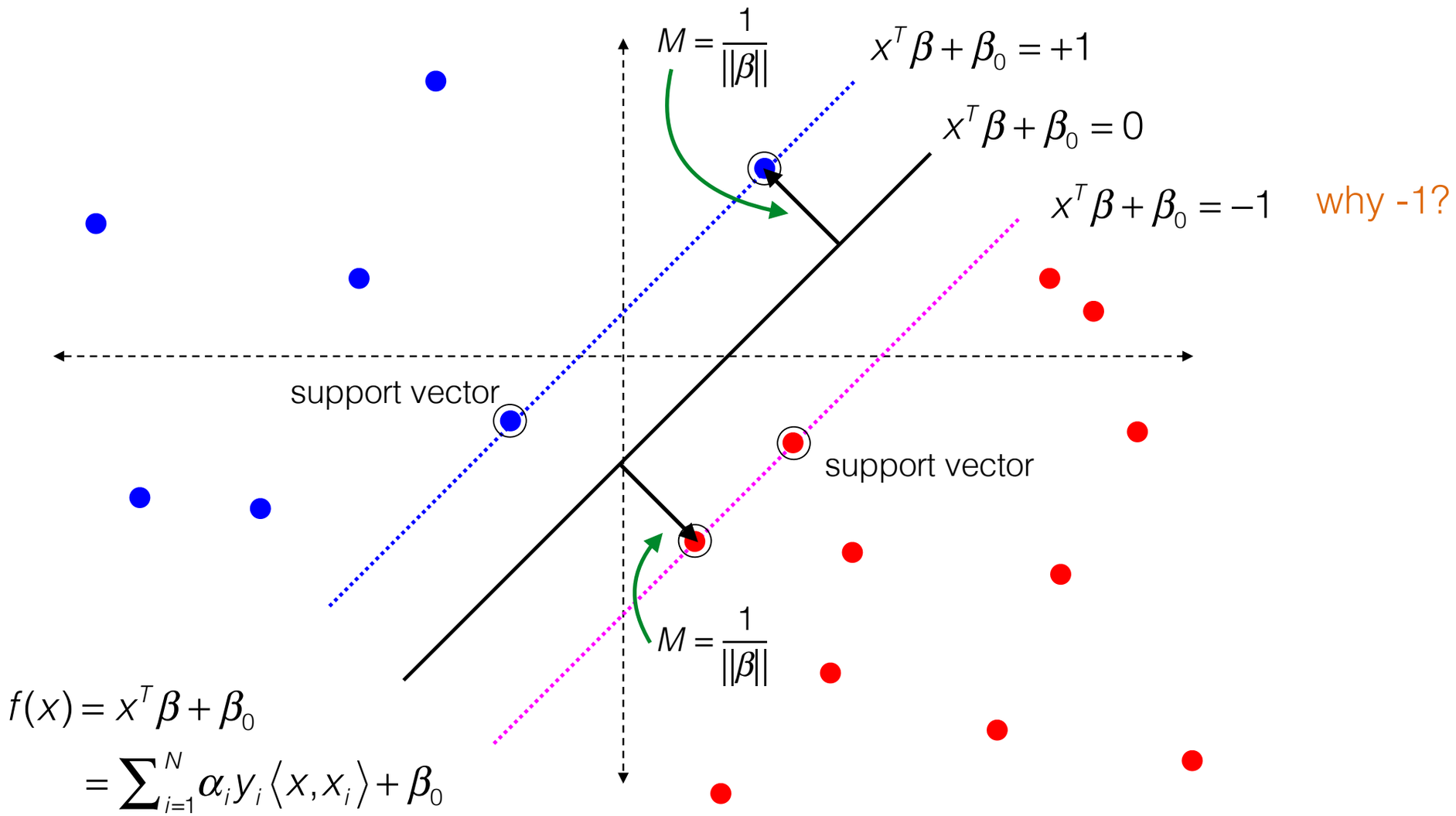
$$\alpha_i [y_i (x_i \beta + \beta_0) - 1] = 0$$

- Using quadratic programming, we can find α_i's minimizing L_D (also satisfying KKT).
- Recall $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$. The support vector classifier is

$$f(x) = x^T \beta + \beta_0$$

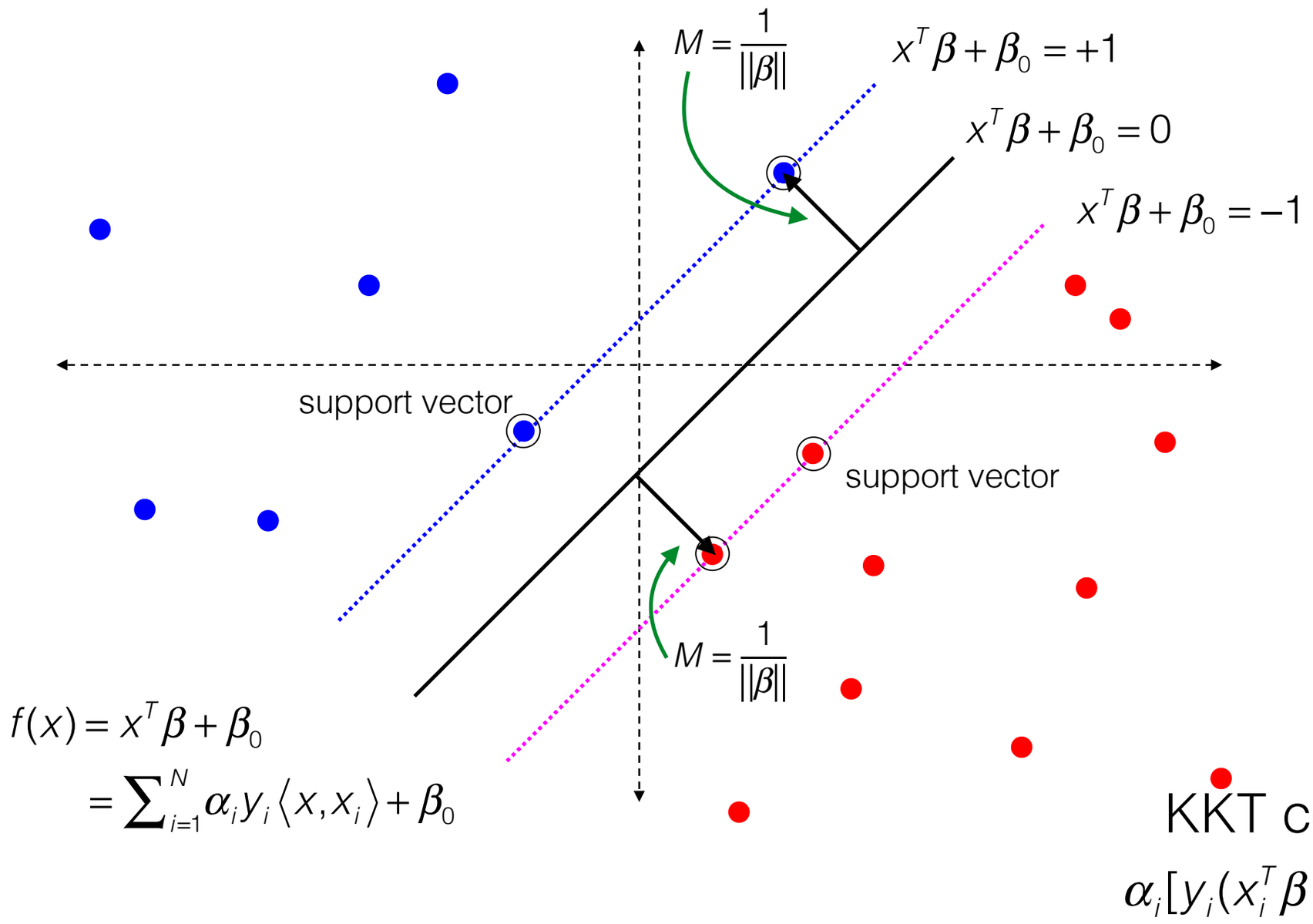
$$= \sum_{i=1}^{N} \alpha_i y_i \langle x, x_i \rangle + \beta_0$$

-



- $x^T\beta + \beta_0 = +1$ or -1 on support vectors. Thus, β_0 can be obtained by

$$\beta_0 = -\frac{\max_{y_i=-1} x_i^T\beta + \min_{y_i=1} x_i^T\beta}{2}$$



- From KKT condition, we can see $\alpha_i > 0$ if x_i is a support vector; otherwise 0. Thus, $\langle x, x_i \rangle$ needs to be computed on support vectors to classify a point x.

- This support vector classifier is often called a hard margin SVM.

Why we bother with the dual?

- Sometimes, the dual problem becomes a convex or concave optimization when the primal problem is not.

- However, our primal problem is a convex problem…

- Solving the primal problem gives unique β and β_0, but it may not give unique solution for α_i's.

- Suppose we consider transformed inputs, h(x). Then, the classifier is

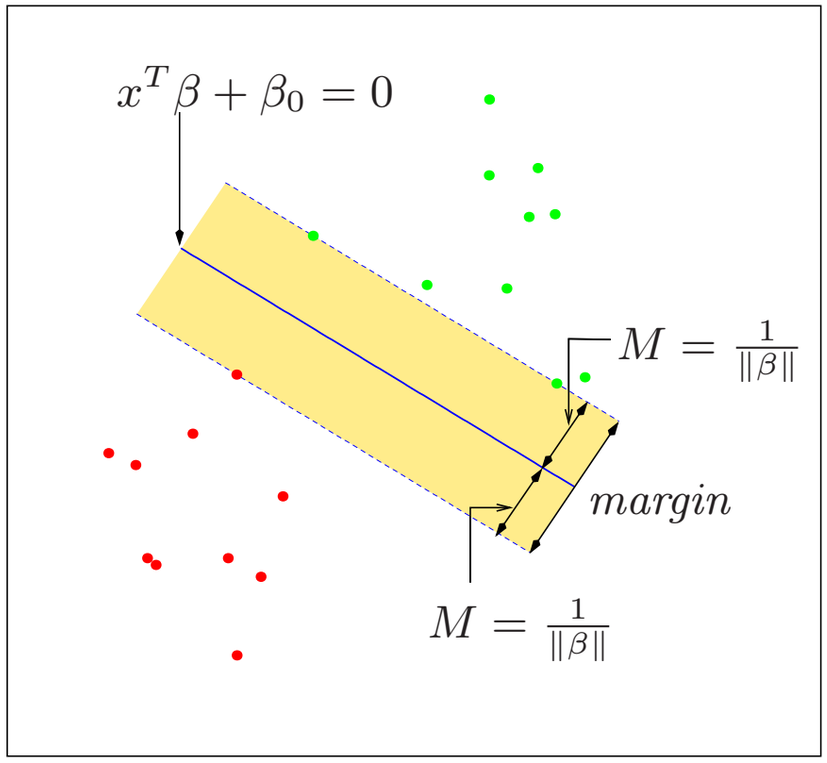
$$f(x) = h(x)^T \beta + \beta_0$$
$$= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0$$

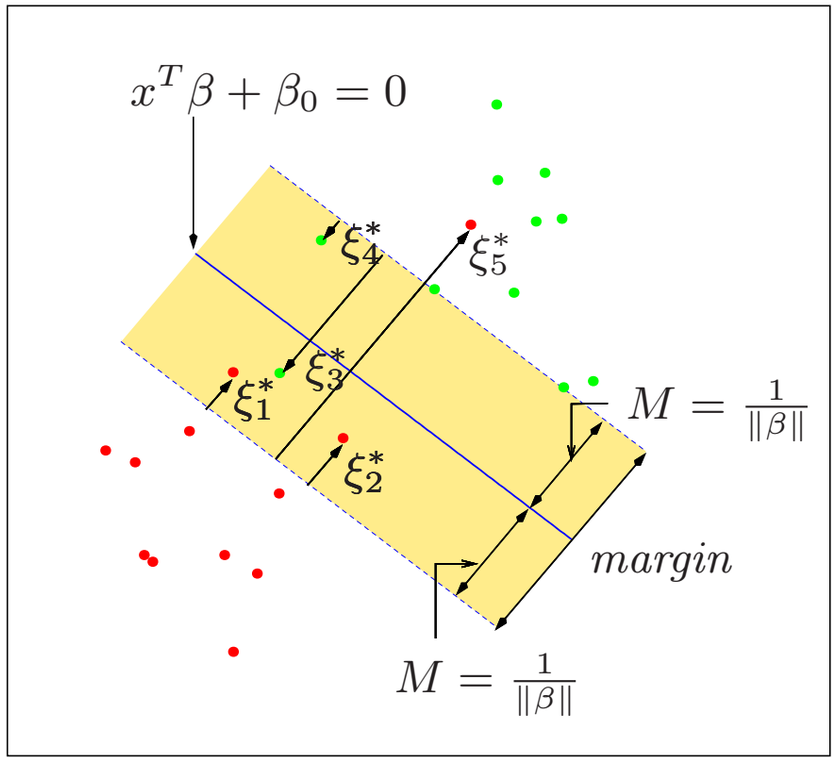
  - The 2nd line can be efficiently calculated if there are only few support vectors.

Support vector machine for non-linearly separable classes

- So far, we have discussed the optimal separating hyperplane. This is a special type of support vector classifier.

- Now, we generalize this idea for non-linearly separable cases. The key idea is that we allow for some points to be on wrong side of the margin.





- We define slack variables $\xi=(\xi_1,\ldots\xi_N)$ where $\xi_i \geq 0$ for all i. The i*th* observation is misclassified if $\xi_i > 1$.

- The cost parameter C: large value of C will discourage positive ξ_i.

  - Larger values of C focus attention more on (correctly classified) points near the decision boundary, while smaller values involve data further away.

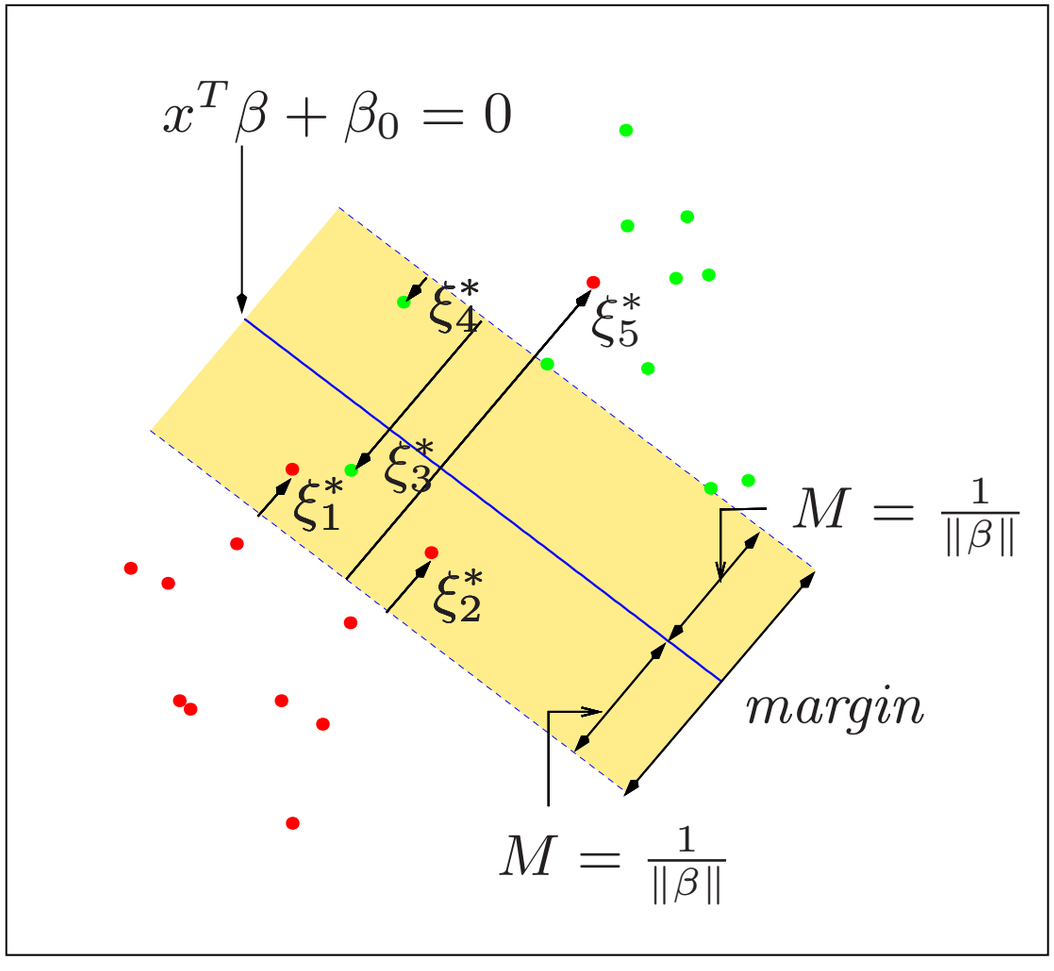
- Soft margin SVM:

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad \xi_i \geq 0,\ y_i(x_i^T\beta+\beta_0) \geq 1-\xi_i\ \forall i,$$

$$f(x) = x^T\beta + \beta_0$$

$$= \sum_{i=1}^{N} \alpha_i y_i \langle x, x_i \rangle + \beta_0$$



- Find the soft margin SVM:

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{subject to} \quad \xi_i \geq 0,\ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i\ \forall i,$$

- Lagrangian primal function

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i,$$

constraints: $\alpha_i,\ \mu_i,\ \xi_i \geq 0\ \forall i.$

- Lagrangian dual function

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\alpha_i\alpha_{i'}y_iy_{i'}x_i^Tx_{i'},$$

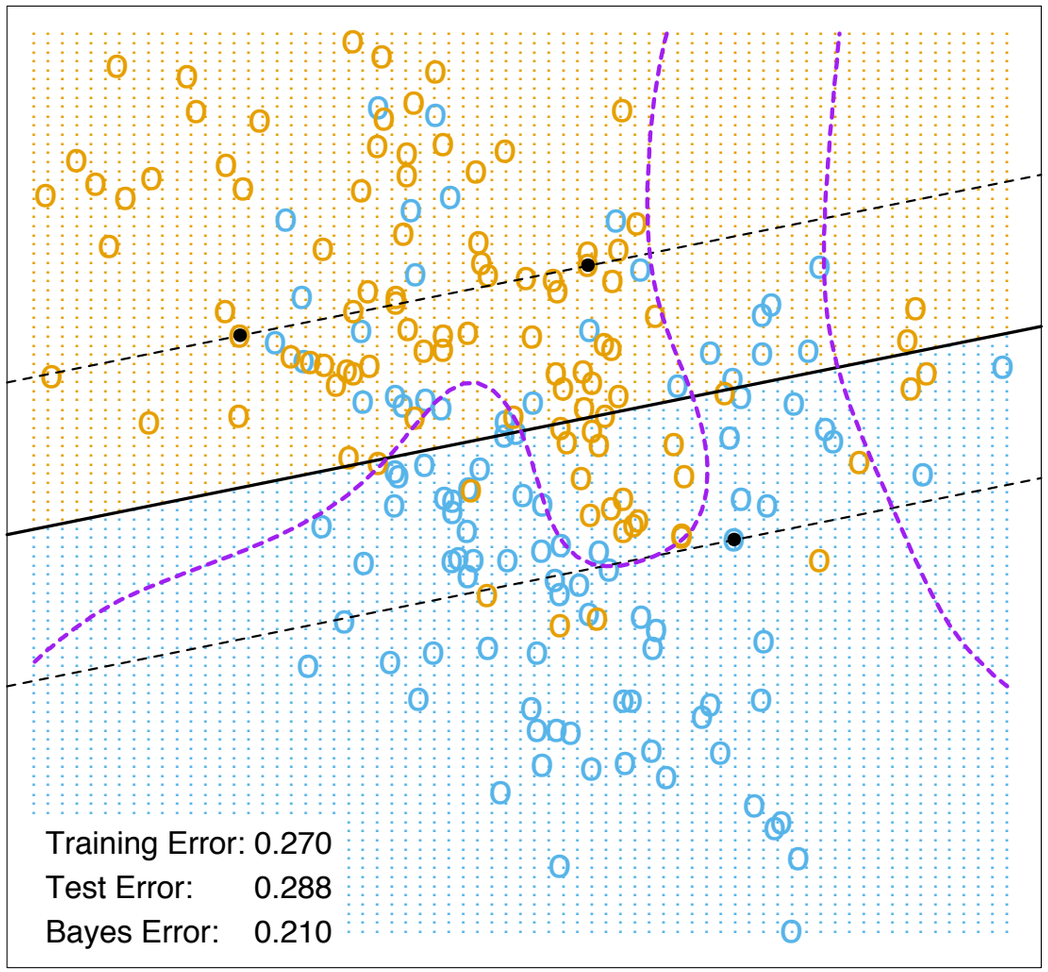
constraints: $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N}\alpha_iy_i = 0.$

KKT

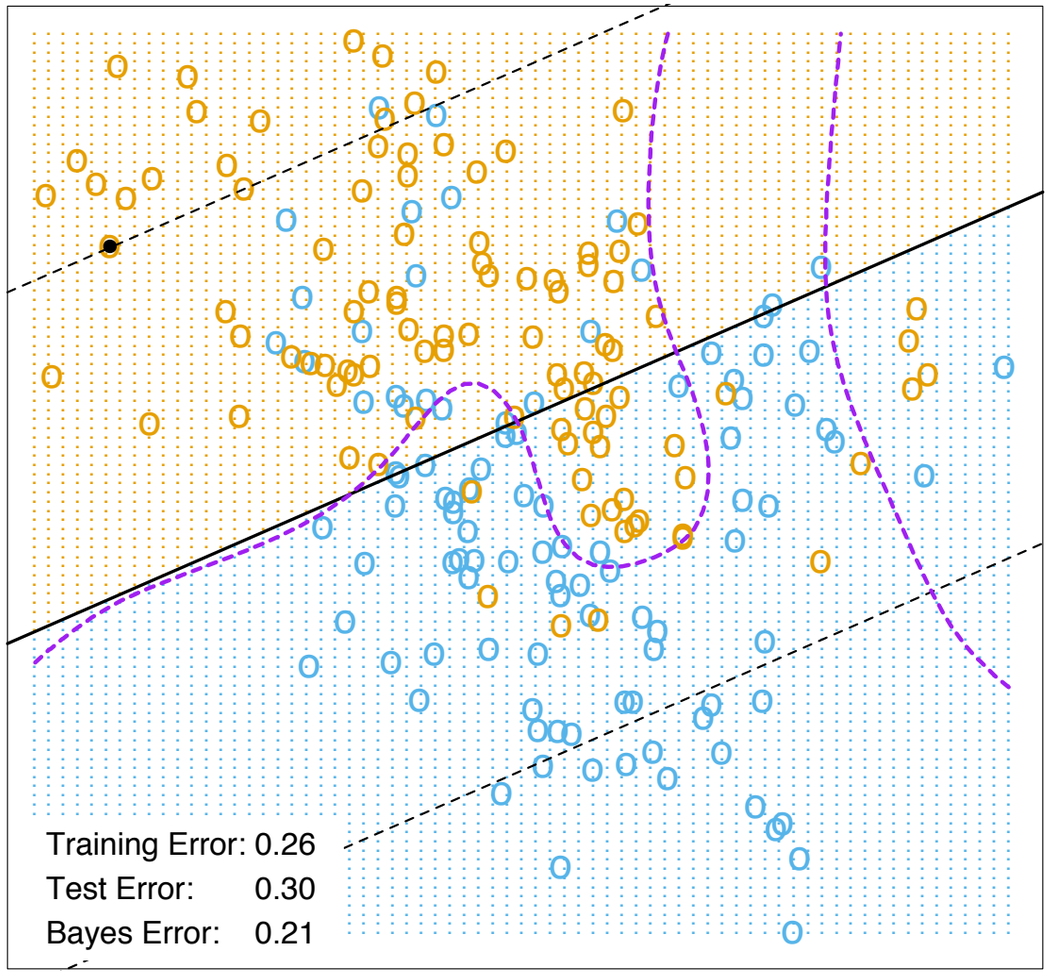
$$\begin{aligned}
\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\xi_i)] &= 0,\\
\mu_i\xi_i &= 0,\\
y_i(x_i^T\beta + \beta_0) - (1-\xi_i) &\geq 0,
\end{aligned}$$

$$\begin{aligned}
\beta &= \sum_{i=1}^{N}\alpha_iy_ix_i,\\
0 &= \sum_{i=1}^{N}\alpha_iy_i,\\
\alpha_i &= C - \mu_i,\ \forall i,
\end{aligned}$$

$$f(x) = x^T\beta + \beta_0$$
$$= \sum_{i=1}^{N}\alpha_iy_i\langle x, x_i\rangle + \beta_0$$



$C = 10000$

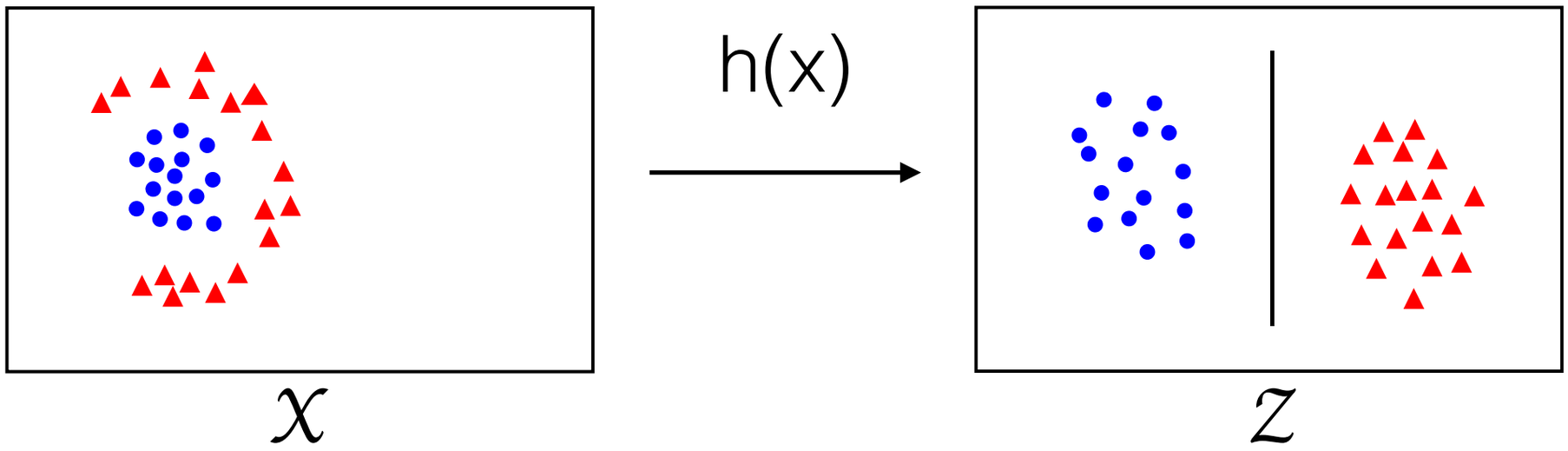


$C = 0.01$

- ESL 12.2: The linear support vector boundary for the mixture data example with two overlapping classes. The broken lines indicate the margins, where f(x)=±1.
- The support vectors ($\alpha_i>0$) are all the points on the wrong side of their margin. The black solid dots are those support points falling exactly on the margin ($\xi_i=0$, $\alpha_i>0$).
- In the left panel 62% of the observations are support vectors, while in the right panel 85% are.

Feature map

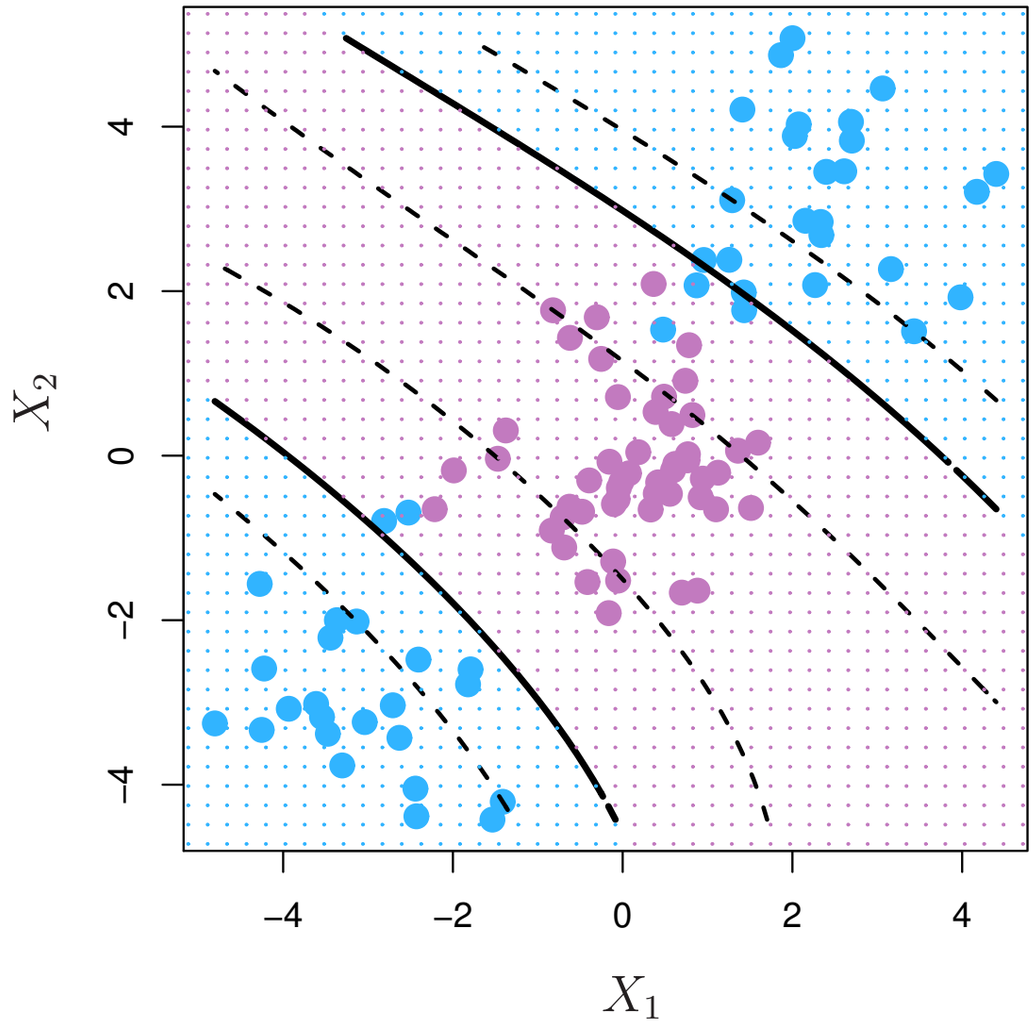
- Consider (non)linear transformation of inputs: $h(x) = (h_1(x), h_2(x), \ldots, h_M(x))$.
- We do feature mapping from the original feature space $\mathcal{X}$ onto an enlarged feature space $\mathcal{Z}$. The feature map h(x) is basis in $\mathcal{Z}$ (basis expansion).



- $$\begin{aligned} f(x) &= h(x)^T\beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0. \end{aligned}$$

Example: cubic polynomials

- We use a basis expansion of cubic polynomials based on two predictors
  - From 2 predictors to 9 basis functions
- The SV classifier in the enlarged space ($\mathcal{Z}$) solves the problem in the lower-dimensional space.



$$\hat{\beta}_0+\beta_1 X_1+\beta_2 X_2+\beta_3 X_1^2+\beta_4 X_2^2+\beta_5 X_1 X_2+\beta_6 X_1^3+\beta_7 X_2^3+\beta_8 X_1 X_2^2+\beta_9 X_1^2 X_2 = 0$$

Non-linear SVM and kernels

- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers — through the use of kernels.

- Note that h(x) is involved in the support vector classifier only through the inner product!

- Also, most of the α_i are zero, and we only need to evaluate the kernel on S (a set of support vectors).

$$f(x) = h(x)^T \beta + \beta_0$$

kernel: $K(x,x_i)$

$$= \sum_{i \in S} \alpha_i y_i \boxed{< h(x), h(x_i) >} + \beta_0$$

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

- Define the kernel function: $K(x,x') = \langle h(x), h(x') \rangle$
  - $K(\cdot,\cdot)$ is a function satisfying
    1. $K(\alpha x+\beta y,z)=\alpha\cdot K(x,z)+\beta\cdot K(y,z)$: Bilinearity
    2. $K(x,y)= K(y,x)$: Symmetry
    3. $K(x,x)\geq 0$ with equality if and only if $x=0$: Positivity
- Examples

dth-Degree polynomial: $K(x, x') = (1 + \langle x, x' \rangle)^d$,

Radial basis: $K(x, x') = \exp(-\gamma \|x - x'\|^2)$, =Gaussian kernel

Neural network: $K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$.

Kernels vs basis functions

- 2nd degree polynomial kernel with two predictors:

$$\begin{aligned} K(X, X') &= (1 + \langle X, X' \rangle)^2 \\ &= (1 + X_1 X_1' + X_2 X_2')^2 \\ &= 1 + 2X_1 X_1' + 2X_2 X_2' + (X_1 X_1')^2 + (X_2 X_2')^2 + 2X_1 X_1' X_2 X_2'. \end{aligned}$$

- Corresponding h(x) = (h_1(x),h_2(x),…,h_6(x)) is

$$h_1(X) = 1,\ h_2(X) = \sqrt{2}X_1,\ h_3(X) = \sqrt{2}X_2$$
$$h_4(X) = X_1^2,\ h_5(X) = X_2^2,\ h_6(X) = \sqrt{2}X_1X_2$$

- Observe that

$$f(x) = \sum_{i \in S} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0$$
$$= \sum_{i \in S} \alpha_i y_i K(x, x_i) + \beta_0$$

- Which one can be computed more efficiently?

- Polynomials (especially high-dimensional ones) get wild fast. We may a large number of basis functions. However, computing polynomial kernels can be done very cheaply.

- Example: d*th* degree polynomial basis of $x \in \mathbb{R}^p$

  - We can find the SVM solution using the kernel (e.g. $(1+\langle x,x' \rangle)^d$) without transforming predictors.

  - It is possible to use explicit basis functions to transform predictors. (i.e. we map predictors onto the space $\mathcal{Z}$). However, it is not computationally efficient.

    - There are ${}_{d+p}C_p$ basis functions to compute at a given data point x.

Kernel tricks

- Suppose you have a learning algorithm that can be written in terms of inner products between input vectors.

- By replacing these inner products with a kernel, you can magically allow your algorithm to work efficiently in the high dimensional feature space corresponding to the kernel.

- Kernel tricks is not unique to SVM. They can be applied

- Not all kernels are valid kernels. Kernels need to satisfy technical condition called Mercer's theorem (our definition of kernels satisfies the condition); otherwise the SVM optimization may diverge.

- Mercer's theorem provides technical condition for the existence of the (possibly finite) expansion of a kernel K.

$$K(x, x') = \sum_{m=1}^{\infty} \phi_m(x)\phi_m(x')\delta_m = \langle h(x), h(x') \rangle$$

- Then, the basis function can be obtained by

$$h_m(x) = \sqrt{\delta_m}\phi_m(x)$$

- In practice, K is a valid kernel if its kernel matrix is positive semi-definite.

  - Kernel matrix $[K]_{ij}=K(x_i,x_j)$ for any i,j in the training sample.

- This allows us to select kernels such that the feature map could be an infinite vector! (e.g.: Gaussian kernels)

- Considering the infinite dimensional feature map may not guarantee separation of points (that's why we have soft margin SVMs).

- However, we can run SVM on an infinite dimensional feature space, and the finite dimensional solution is obtained.

$$f(X) = \sum_{i \in S} \alpha_i y_i K(x, x_i) + \beta_0$$

- This is quite powerful as it covers a vast range of functions in reproducing kernel Hilbert space (RKHS).

Infinite dimensional feature space

- Consider the Gaussian kernel with a 1-dimensional predictor x.

$$K(x,x') = \exp(-(x-x')^2)$$

$$= \exp(-x^2)\exp(-x'^2)\sum_{k=0}^{\infty}\frac{2^k x^k x'^k}{k!}$$

$$= \sum_{k=0}^{\infty}\phi(x)\phi(x')\frac{2^k}{k!}$$

- There are infinitely many basis functions, and indeed the space $\mathcal{Z}$ is infinite dimensional!

- We only need to the basis function exists. If so, we can find the finite dimensional solution (soft or hard margin SVM) in $\mathcal{X}$.

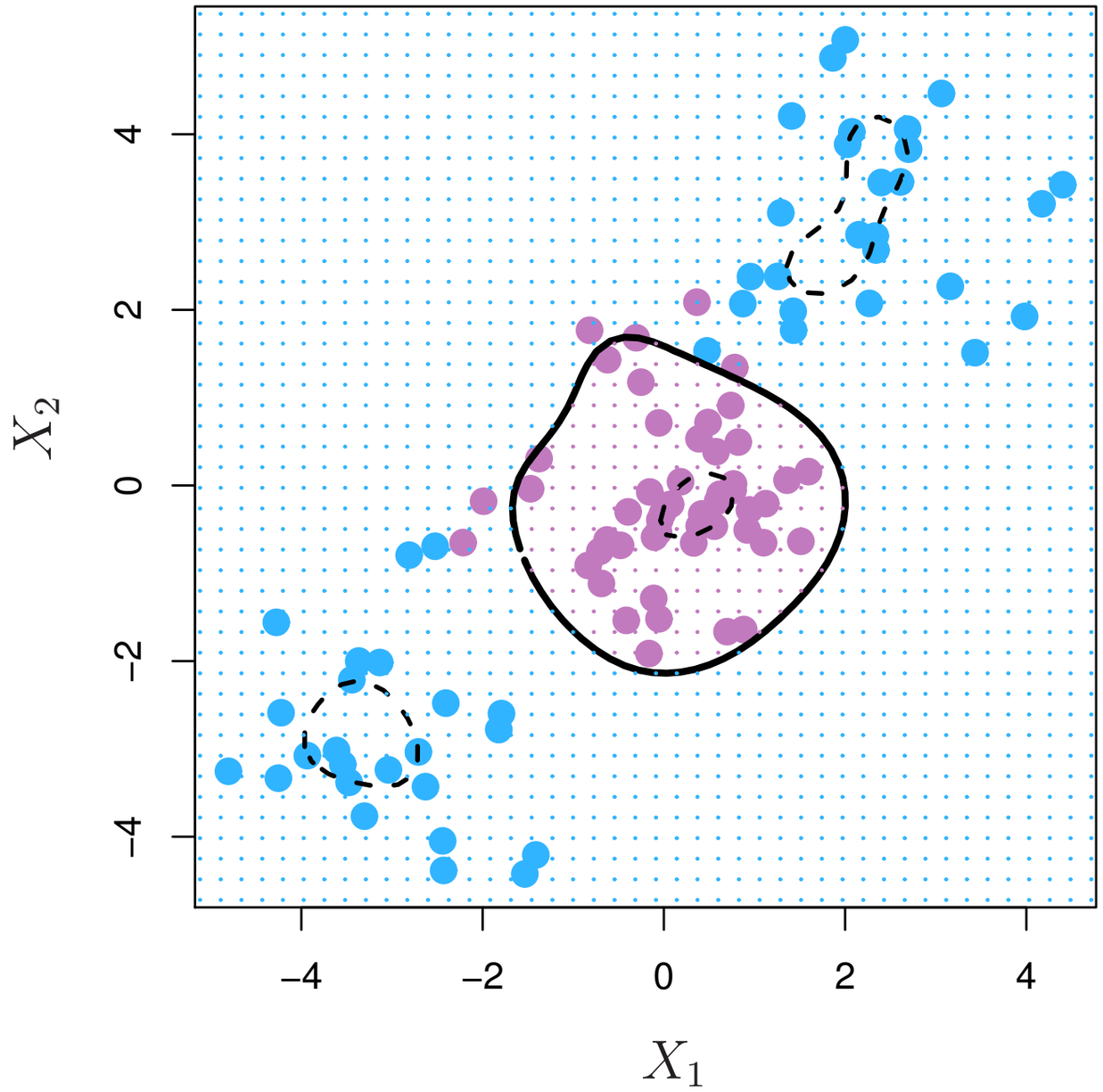
Gaussian kernel SVM

Radial basis: $K(x, x') = \exp(-\gamma\|x - x'\|^2)$,

- Implicit feature space: the feature space is infinite dimensional!
- Controls variance by squashing down most dimensions severely

$$f(x) = h(x)^T \beta + \beta_0$$
$$= \sum_{i \in S} \alpha_i y_i K(x, x_i) + \beta_0$$

- Using the Gaussian kernel, we classify data points based on similarities. (sounds like nearest methods?)



Cost parameter C

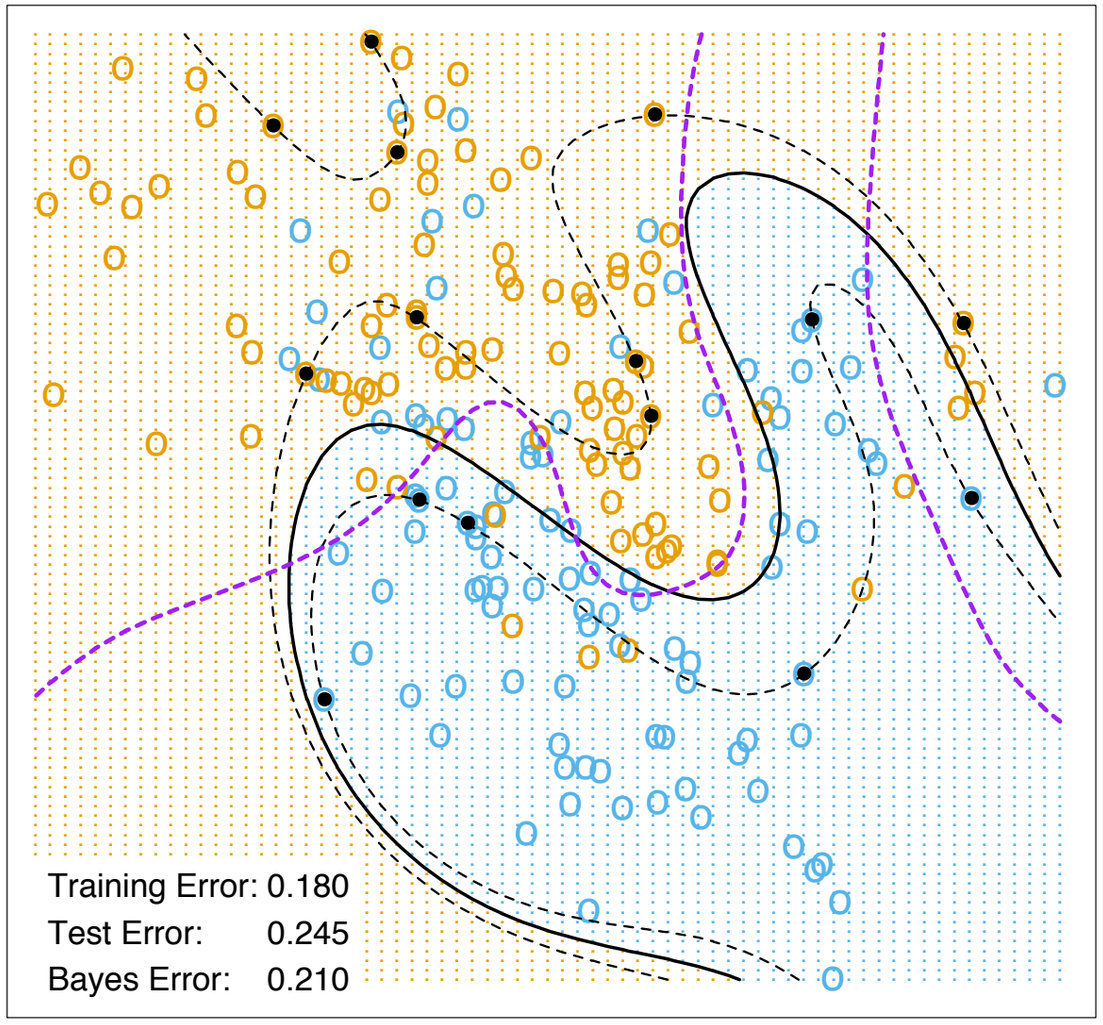
- The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there.

- A large value of C will discourage any positive ξ_i and lead to an overfit wiggly boundary in the original feature space;

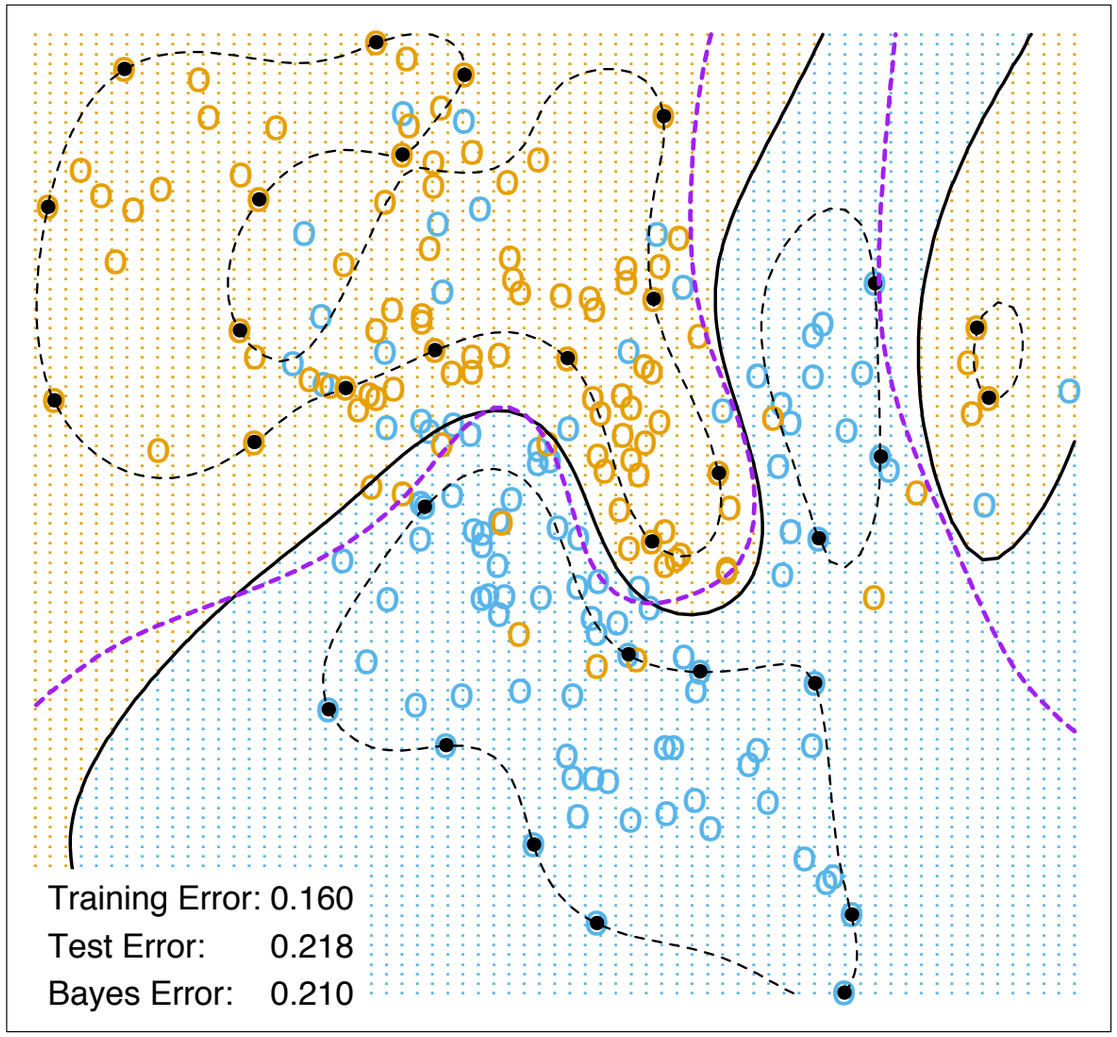
- A small value of C will encourage a small value of $\|\beta\|$, which in turn causes f(x) and hence the boundary to be smoother.

- R package svmpath utilizes a path algorithm to efficiently fit the entire sequencing of SVM with varying C.

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



- ESL 12.3: In each case C is tuned to approximately achieve the best test error performance, and C = 1 worked well in both cases.

- Solid lines are decision boundaries. Broken lines are margins. They are linear in $\mathcal{Z}$. But these lines become non-linear in $\mathcal{X}$.

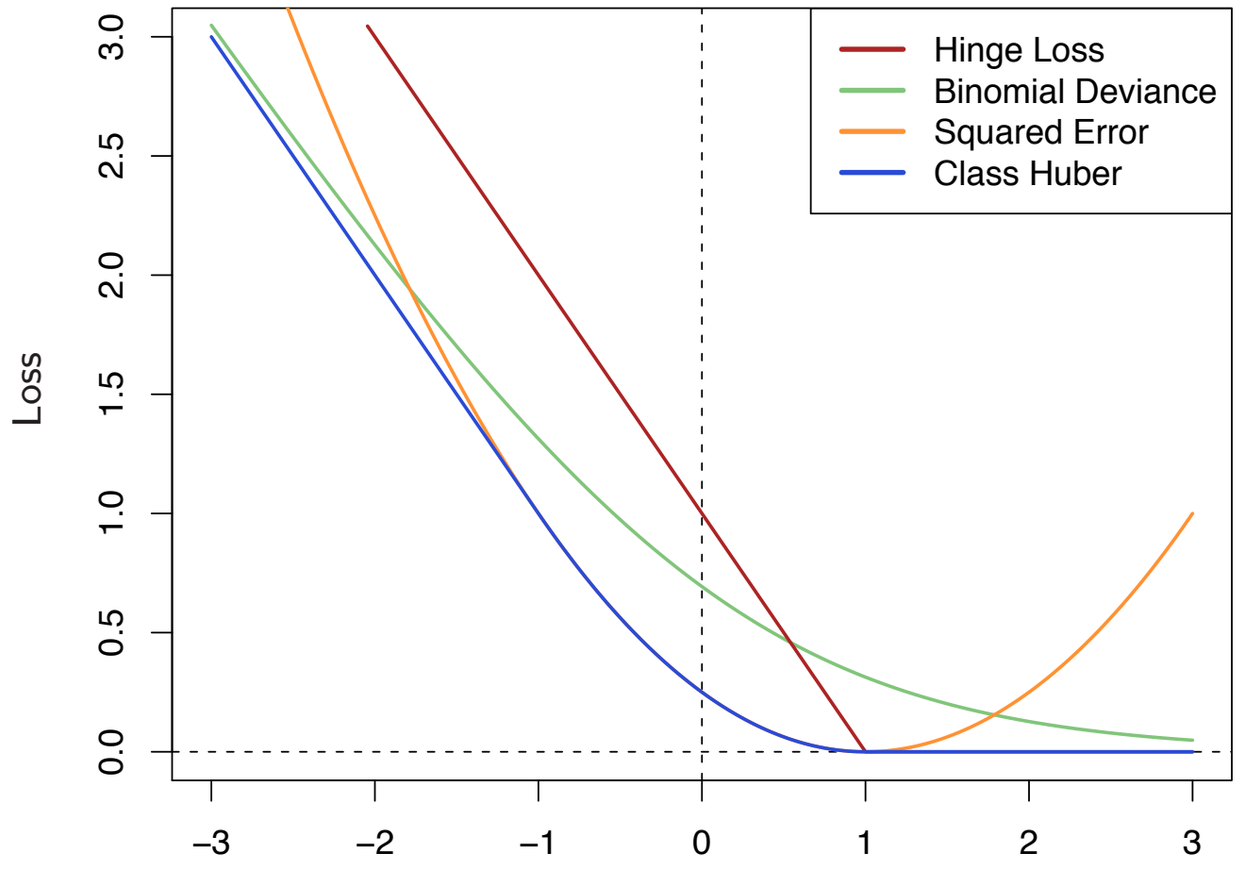
Another look on SVM

- Consider a penalized optimization problem:

$$\min_{\beta_0,\,\beta} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\|\beta\|^2 \qquad\qquad f(x) = h(x)^T\beta + \beta_0$$

- $L(y,f)=[1-y\cdot f]_+$ is called hinge loss function. The subscript + indicate positive part.
  - Q. When $y_i \cdot f(x_i) < 1$ happen in SVM?
- Choosing $\lambda=1/C>0$, the solution for the optimization is equivalent to the soft-margin SVM with feature map h(x).
- When the data are separable, β defines the optimal separating hyperplane (hard-margin SVM) as $\lambda \to 0$.

Loss Function	$L[y, f(x)]$	Minimizing Function
Binomial Deviance	$\log[1 + e^{-yf(x)}]$	$f(x) = \log \frac{\Pr(Y = +1\|x)}{\Pr(Y = -1\|x)}$
SVM Hinge Loss	$[1 - yf(x)]_+$	$f(x) = \text{sign}[\Pr(Y = +1\|x) - \frac{1}{2}]$
Squared Error	$[y - f(x)]^2 = [1 - yf(x)]^2$	$f(x) = 2\Pr(Y = +1\|x) - 1$
"Huberised" Square Hinge Loss	$-4yf(x)$, $yf(x) < -1$ $[1 - yf(x)]_+^2$ otherwise	$f(x) = 2\Pr(Y = +1\|x) - 1$



- ESL 12.4: The support vector loss function (hinge loss), compared to the negative log-likelihood loss (binomial deviance) for logistic regression, squared-error loss, and a Huberized version of the squared hinge loss.

- The Huberized square hinge loss shares attractive properties of logistic regression (smooth loss function, estimates probabilities), as well as the SVM hinge loss (support vectors).

More than 2 classes?

- The SVM as defined works for K = 2 classes. What do we do if we have K > 2 classes?
- OVA (One versus All).
  - Fit K different 2-class SVM classifiers $f_k(x)$, k = 1, . . . , K; each class versus the rest. Classify x to the class k for which $f_k(x)$ is largest.
  - Multiclass logistic model does this by fitting (K-1) log odds ratios.
- OVO (One versus One).
  - Fit all ${}_KC_2$ pairwise classifiers $f_{k,l}(x)$. Classify x to the class that wins the most pairwise competitions.
- Which to choose? If K is not too large, use OVO.

SVM vs logistic regression

- SVM optimization:

$$\min_{\beta_0,\,\beta} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\|\beta\|^2$$

loss + penalty

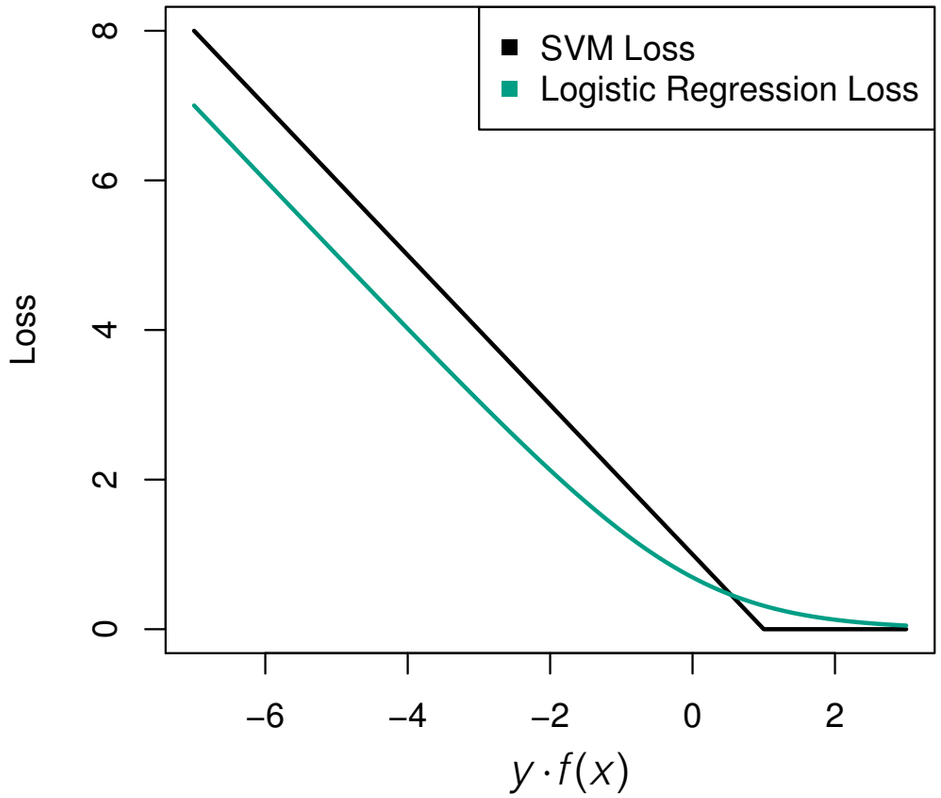
- The hinge loss is similar to loss in logistic regression (binomial deviance).

- When classes are (nearly) separable, SVM does better than logistic regression (LR). So does LDA.

- When not, LR with ridge penalty and SVM very similar.

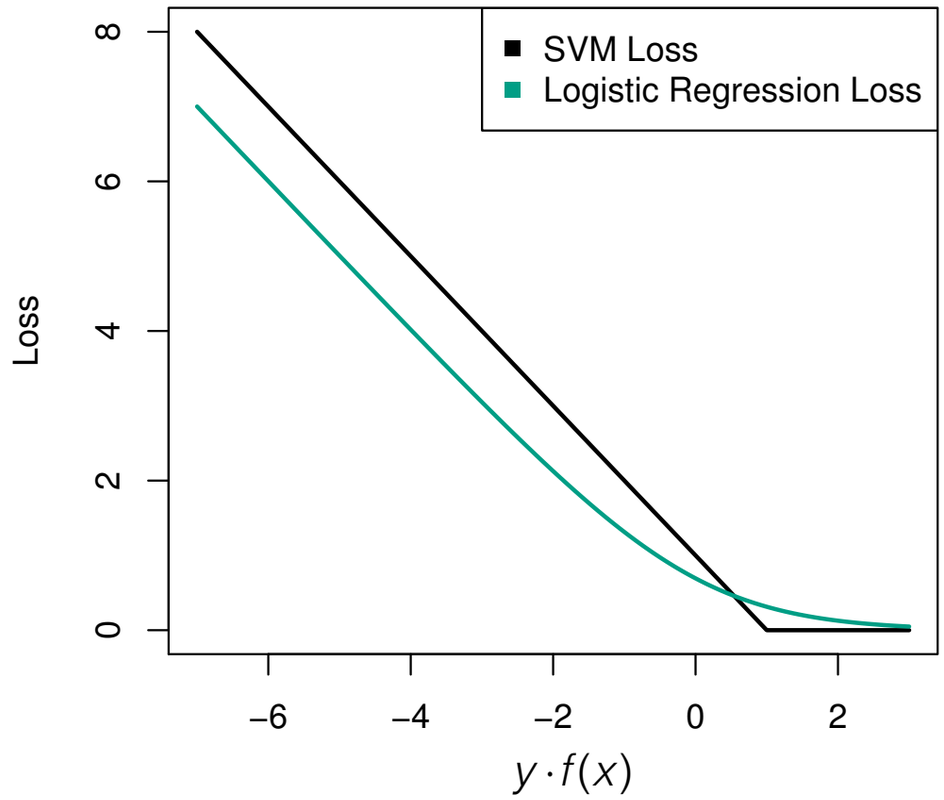
- If you wish to estimate probabilities, LR is the choice.

- For nonlinear boundaries, kernel SVMs are popular. We can use kernels with LR and LDA as well, but computations are more expensive.



SVM vs logistic regression

- When p>>N: use LR or SVM without a kernel (linear kernel)
- When p is small, N is intermediate:
  - Use SVM with Gaussian kernel
- When p is small, N is large:
  - Consider some feature mapping h(x), then use logistic regression or SVM without a kernel
- Neural network is likely to work well for most of these settings, but may be slower to train.



SVM for regression

- SVM can be adapted for regression with quantitative response variables (support vector regression machine). Let's start with the linear regression:

$$f(x) = x^T \beta + \beta_0$$

- SVM optimization:

residual

$$H(\beta, \beta_0) = \sum_{i=1}^{N} V(y_i - f(x_i)) + \frac{\lambda}{2}\|\beta\|^2$$

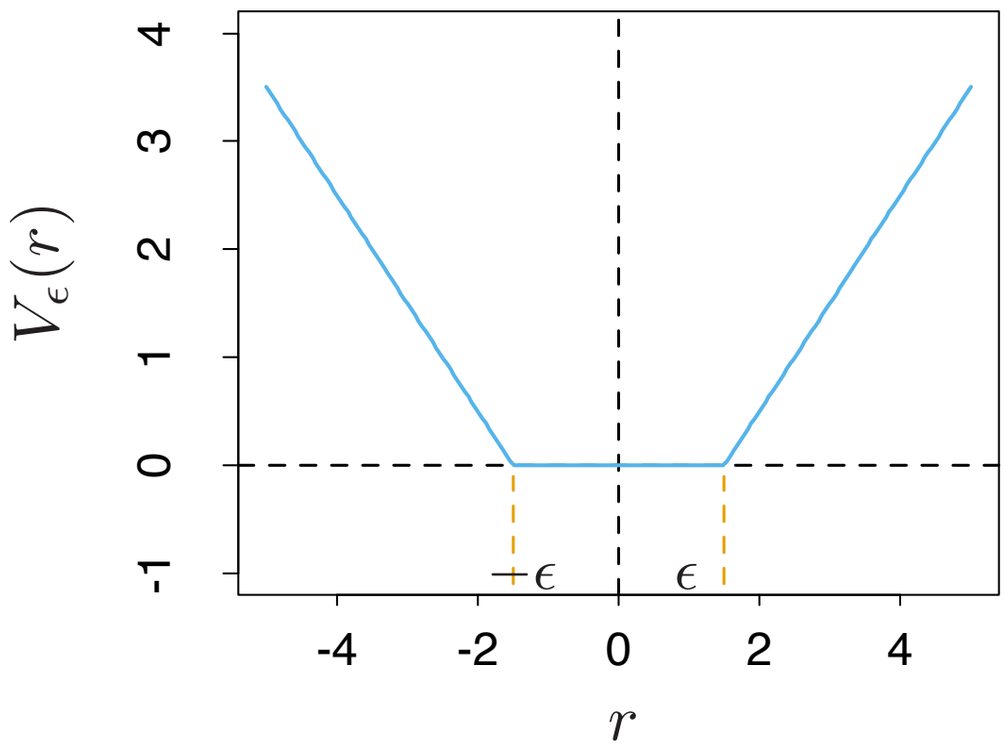
- Again, this has form of loss + penalty. V is the error function.
  - Q. If V(r) = r^2 (squared error), what does this SVM optimization do?

Support vector error measure

- ε-sensitive error: ignoring errors less than ε

$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon, \\ |r| - \epsilon, & \text{otherwise.} \end{cases}$$

- SVM classifier ignores points that are far way from the decision boundary.
- Similarly, in regression, points with small residuals receive low errors.
- Linear tails make fitting less sensitive to outliers.



ESL 12.8: ε-sensitive error

SV regression machine and kernels

- For regression, we can also consider some feature maps.

$$f(x) = \sum_{m=1}^{M} \beta_m h_m(x) + \beta_0.$$

- Penalized optimization give us the solution has the form

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{a}_i K(x, x_i)$$

- The response vector y is involved in coefficient above.

Take home messages

- SVM classifier maximizes the margin between classes.

- The SVM classifier can be obtained by solving the dual problem using the quadratic programming.

- Using kernel tricks, one can find the SVM solution in very high dimensional feature space without explicitly knowing complex feature maps.

Large scale SVM solver

- libSVM:
- http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- The library is available in Python, R, MATLAB, Perl, CUDA, and many other programming languages.
- libSVM is adapted in R though the R package e1071.