Support Vector Machine

CS 534: Machine Learning

Slides adapted from David Sontag, Luke Zettlemoyer, Carlos Guestrin, Vibhav Gogate,
Jason Weston, Trevor Hastie, and Rob Tibshirani

Review: LDA & Logistic Regression

- LDA assumes class conditional densities are multivariate normal with same covariance and different mean
- Logistic regression is generalized linear model with logit link
- Both estimate linear decision boundaries in similar but different ways

Hyperplane

- Hyperplane in p dimensions is a flat affine subspace of dimension p - 1
- General equation for a hyperplane

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = 0$$

- Normal vector $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_p)$ points in the direction orthogonal to the surface of a hyperplane

Hyperplane: Pictorially

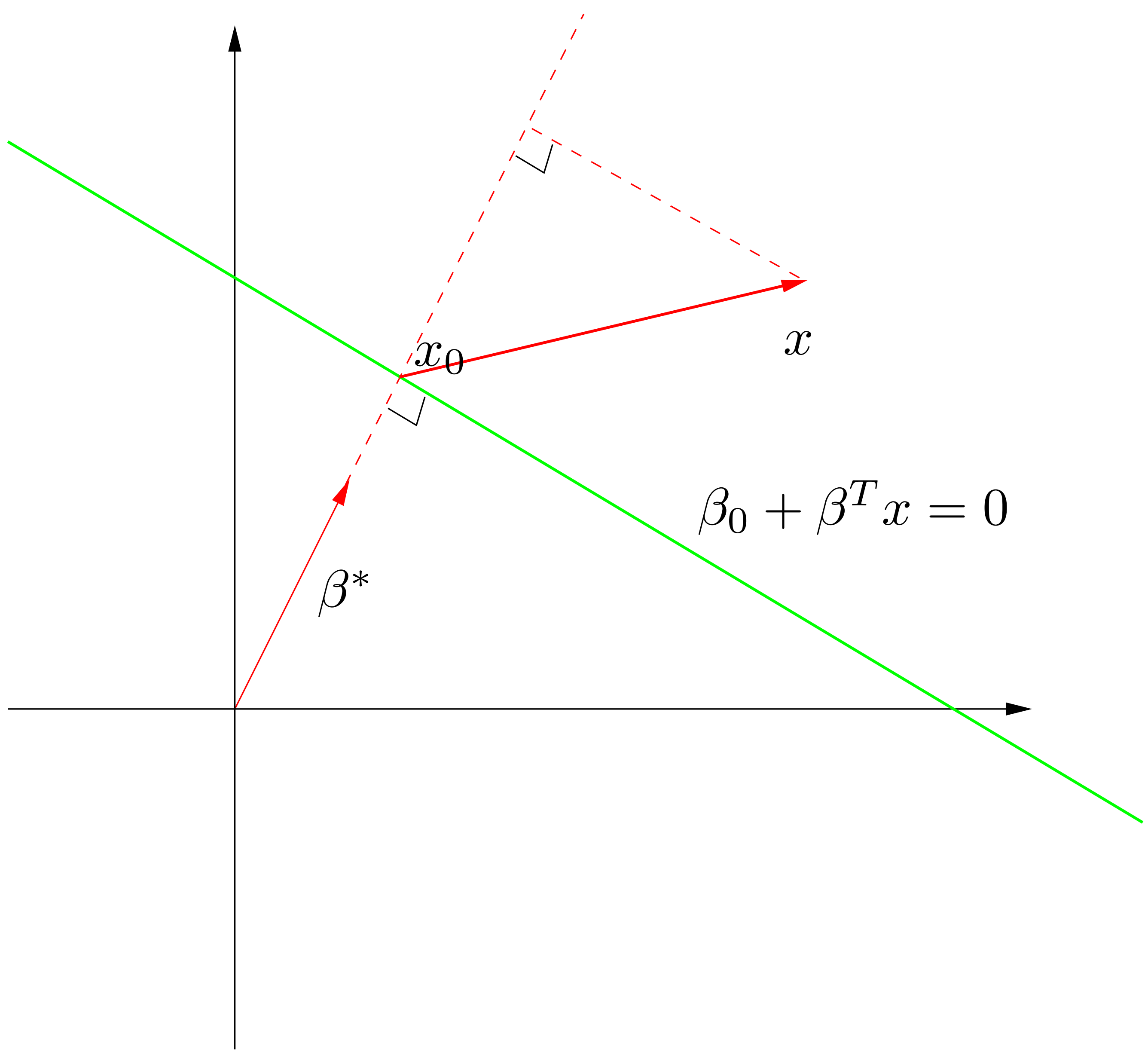
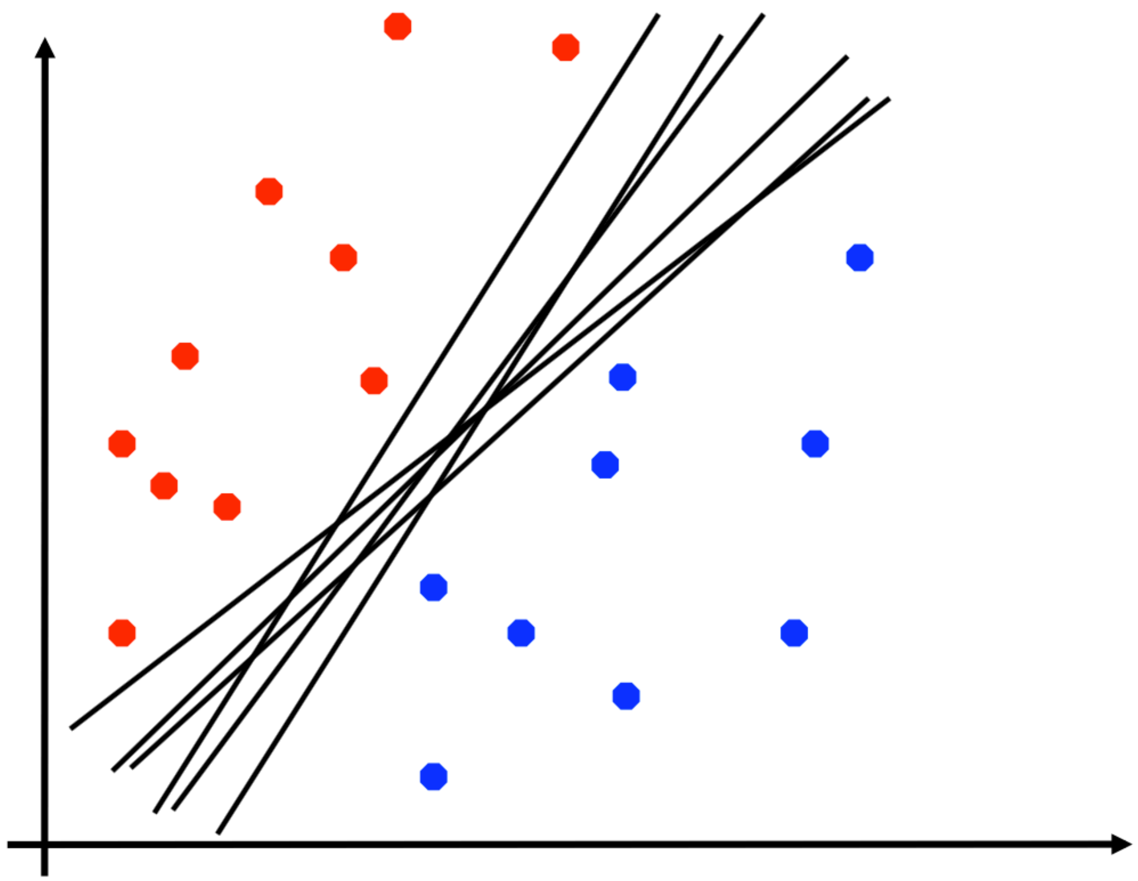


Figure 4.15 (Hastie et al.)

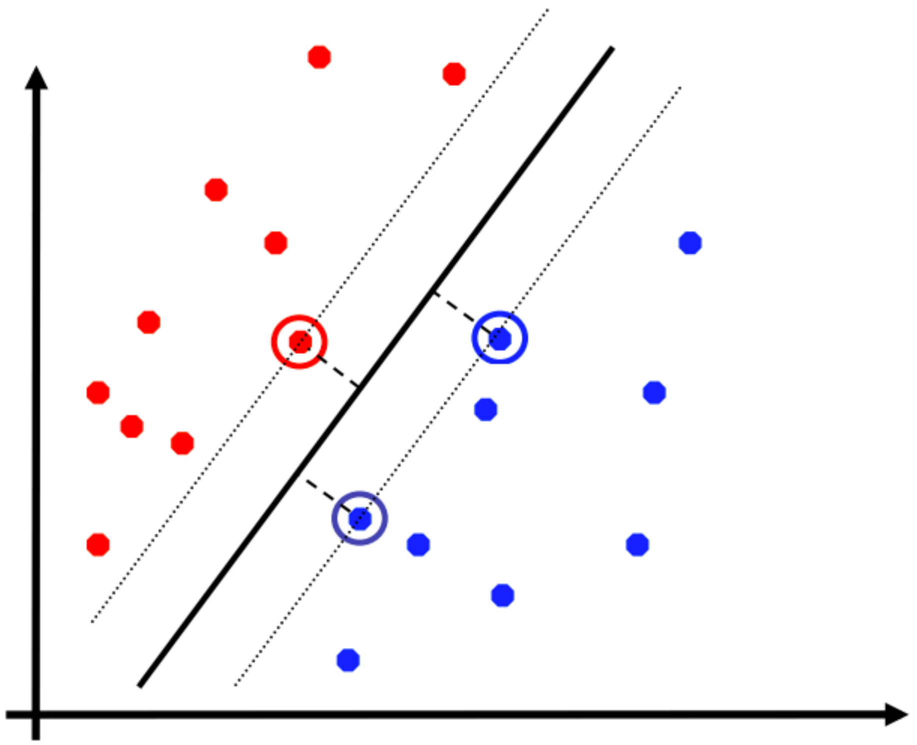
Two-Class Problem

- If the data is truly linearly separable, can we find the hyperplane that separates the classes in our feature space
- Which separator is optimal if there are many options?



Support Vector Machine (SVM)

- Introduced by Boser, Guyon, and Vapnik in 1992
- Chose the linear separator with the largest margin
    - Robust to outliers
    - Good according to intuition, theory, and practice

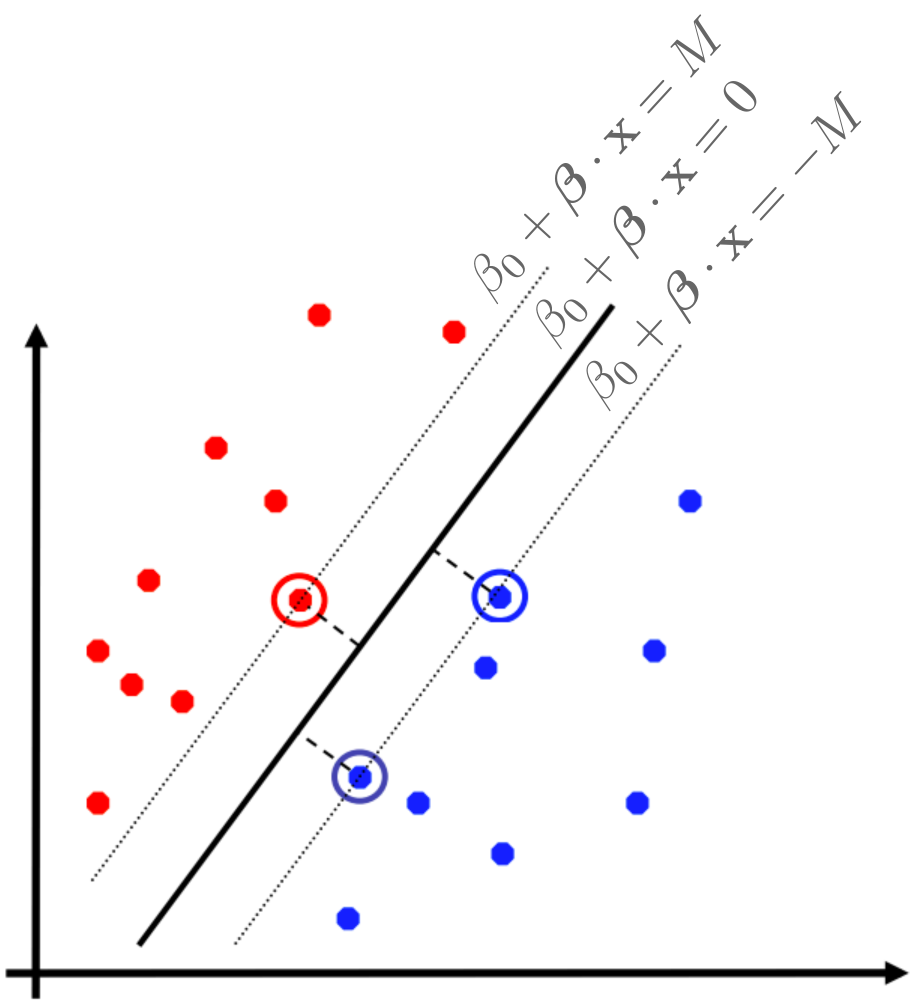


SVM: Key Ideas

- Find large margin separator to improve generalization
- Use optimization to find solution with few errors
- Use kernel trick to make large feature spaces computationally efficient

Empirically good performance in many fields such as text, image recognition, bioinformatics, etc.

Key Idea #1: Maximal Margin



- Want to enforce the following constraint for every data point

$$\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i \geq +M, \;\; y_i = +1$$
$$\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i \leq -M, \;\; y_i = -1$$

- Equivalent to linear constraints

$$y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i) \geq M$$

Key Idea #1: Maximal Margin

- Pose it as a constrained optimization problem
- Let M denote the distance of the margin

$$\max_{\boldsymbol{\beta}, \beta_0} M$$

$$\text{s.t.} \sum_j \beta_j^2 = 1 \quad \leftarrow \text{controls for complexity}$$

$$y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i) \geq M, \; \forall i$$

Key Idea #1: Maximal Margin

- Simplify the optimization problem by removing the norm constraint

$$\frac{1}{||\boldsymbol{\beta}||}(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_1) \geq M$$

$$\Updownarrow$$

$$(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_1) \geq ||\boldsymbol{\beta}||M$$

- We can arbitrarily scale the norm vector to satisfy these inequalities, so for convenience:

$$M = \frac{1}{||\boldsymbol{\beta}||}$$

Maximizing margin is equivalent to minimizing norm!

Key Idea #1: Maximal Margin

- Equivalent optimization problem

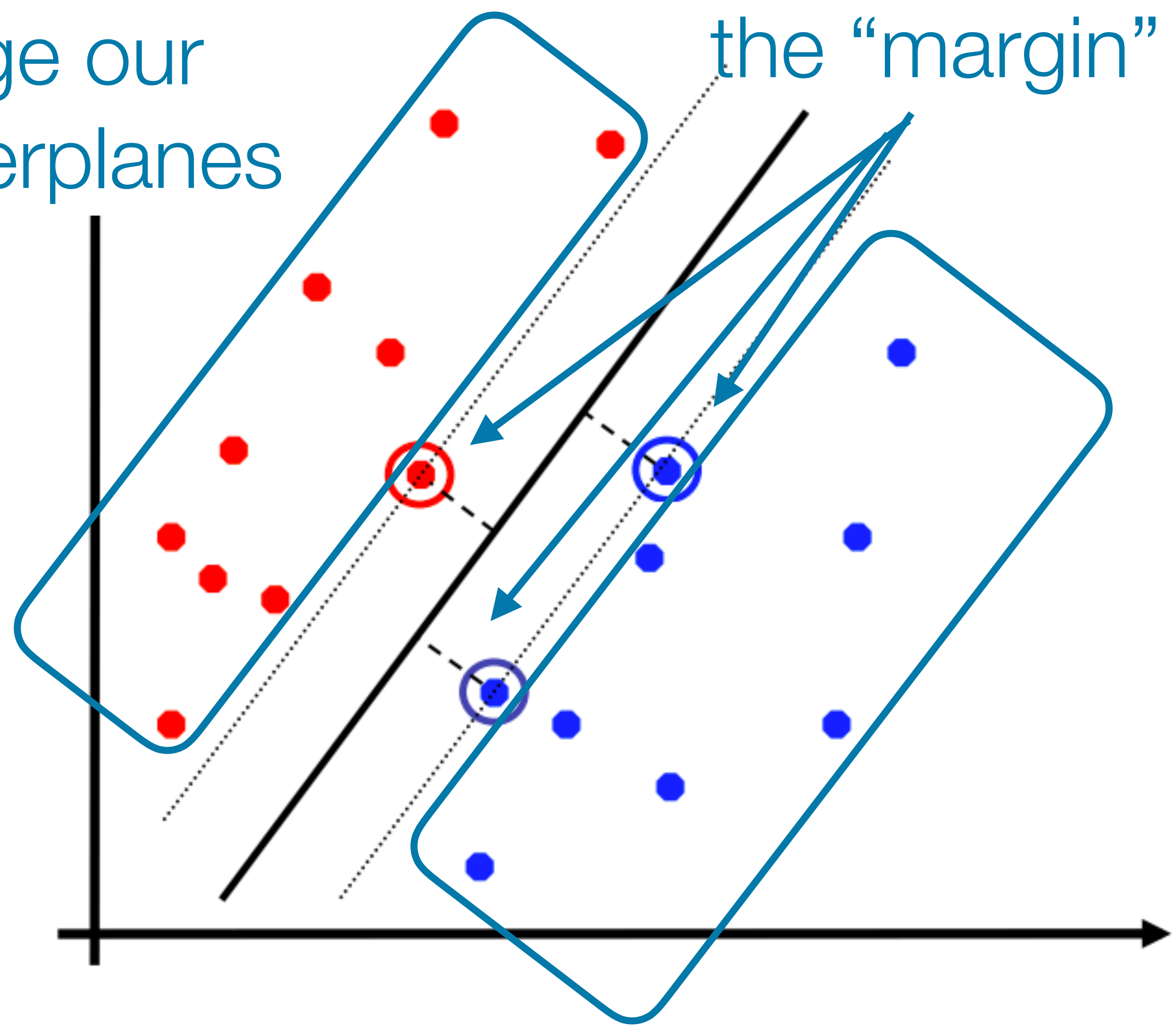
$$\min_{\boldsymbol{\beta},\beta_0} ||\boldsymbol{\beta}||$$
$$\text{s.t. } y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i) \geq 1, \ \forall i$$

- Example of a convex quadratic program
  - Polynomial time algorithms to solve —> very efficient

Key Idea #1: Maximal Margin

Moving these points will not change our separating hyperplanes

Support vectors: data points on the "margin" hyperplanes



What about non-separable case?

Minimize Errors: 0-1 Loss

- Try to find weights that violate as few constraints as possible

$$\mathbb{1}_{\{y \neq \text{sign}(\hat{y})\}}$$

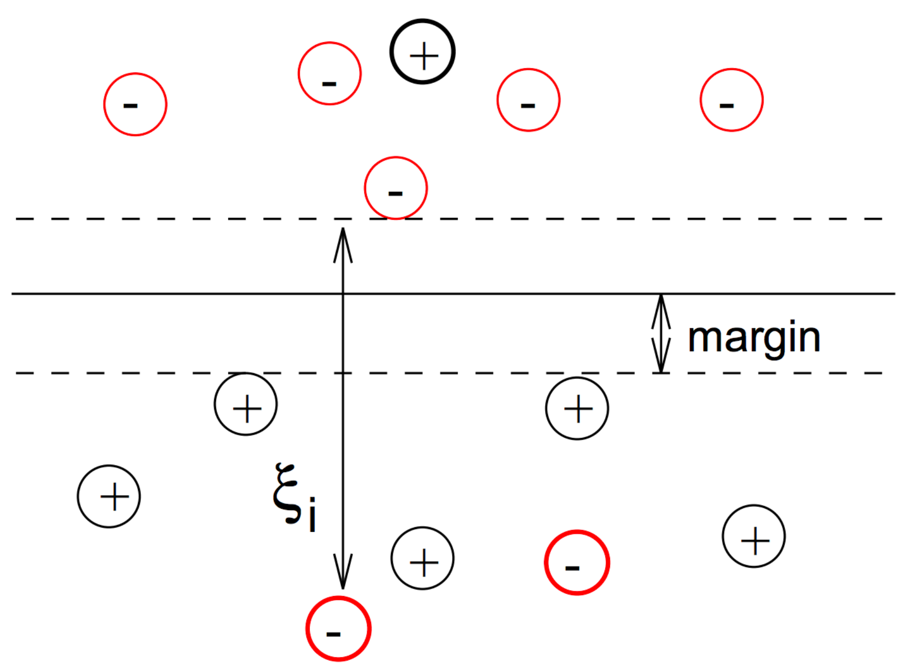
$$\min_{\boldsymbol{\beta}, \beta_0} (||\boldsymbol{\beta}|| + \ell(y_i, \boldsymbol{\beta} \cdot \mathbf{x}_i))$$

$$\text{s.t. } y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i) \geq 1$$

- Minimizing 0-1 loss is NP-hard in the worst case

Key Idea #2: Slack

- Introduce the notion of "slack" variables
  - If functional margin is correct, don't care
  - If functional margin is incorrect (< 1), pay linear penalty



- Modify the constraint:

$$\min ||\boldsymbol{\beta}||$$

$$\text{s.t. } y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i) \geq (1 - \xi_i), \forall i \; \xi_i \geq 0, \sum \xi_i \leq C$$

Key Idea #2: Slack

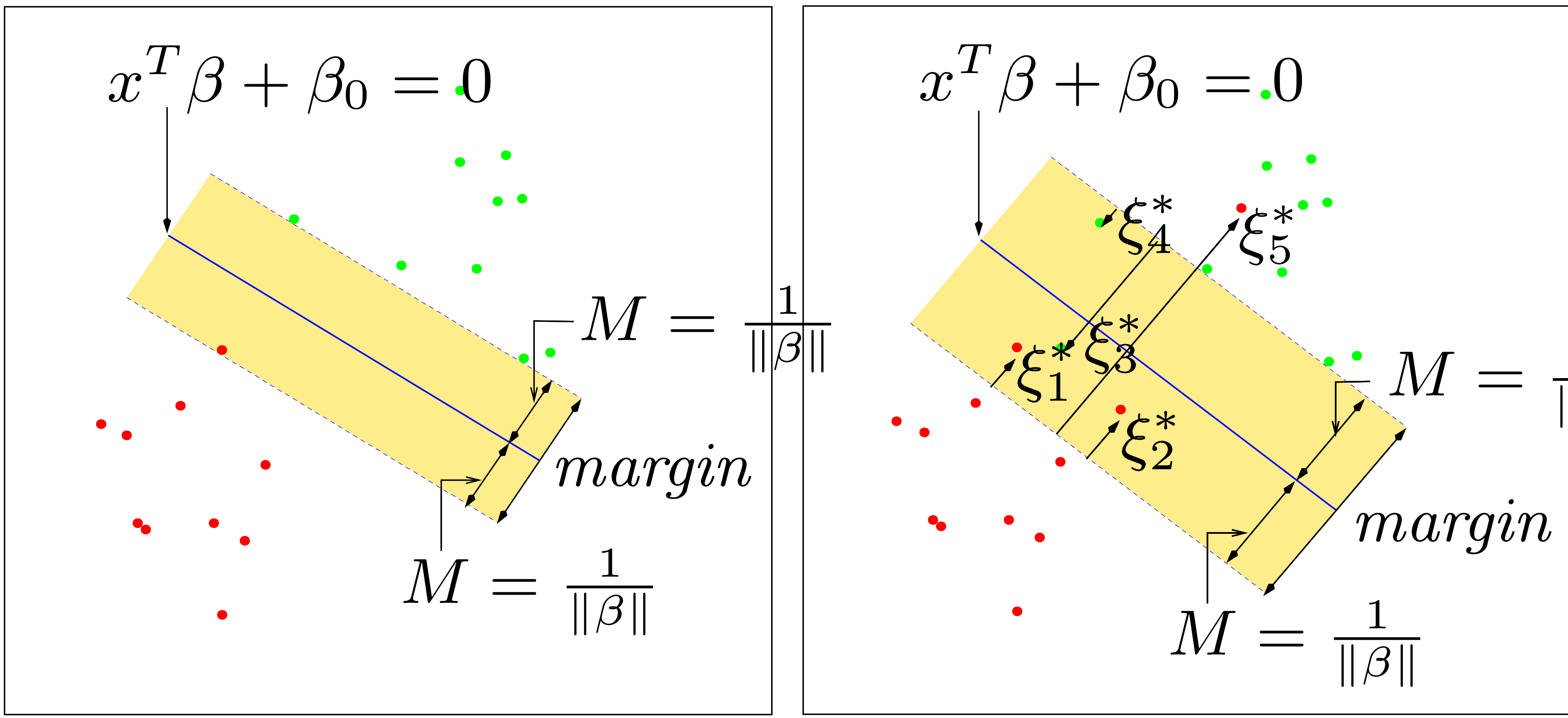


Figure 12.1 (Hastie et al.)

Key Idea #2: Slack

- What is the optimal value of the slack variable as a function of the hyperplane?

  - If functional margin is correct:

$$y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i) \geq 1 \rightarrow \xi_i = 0$$

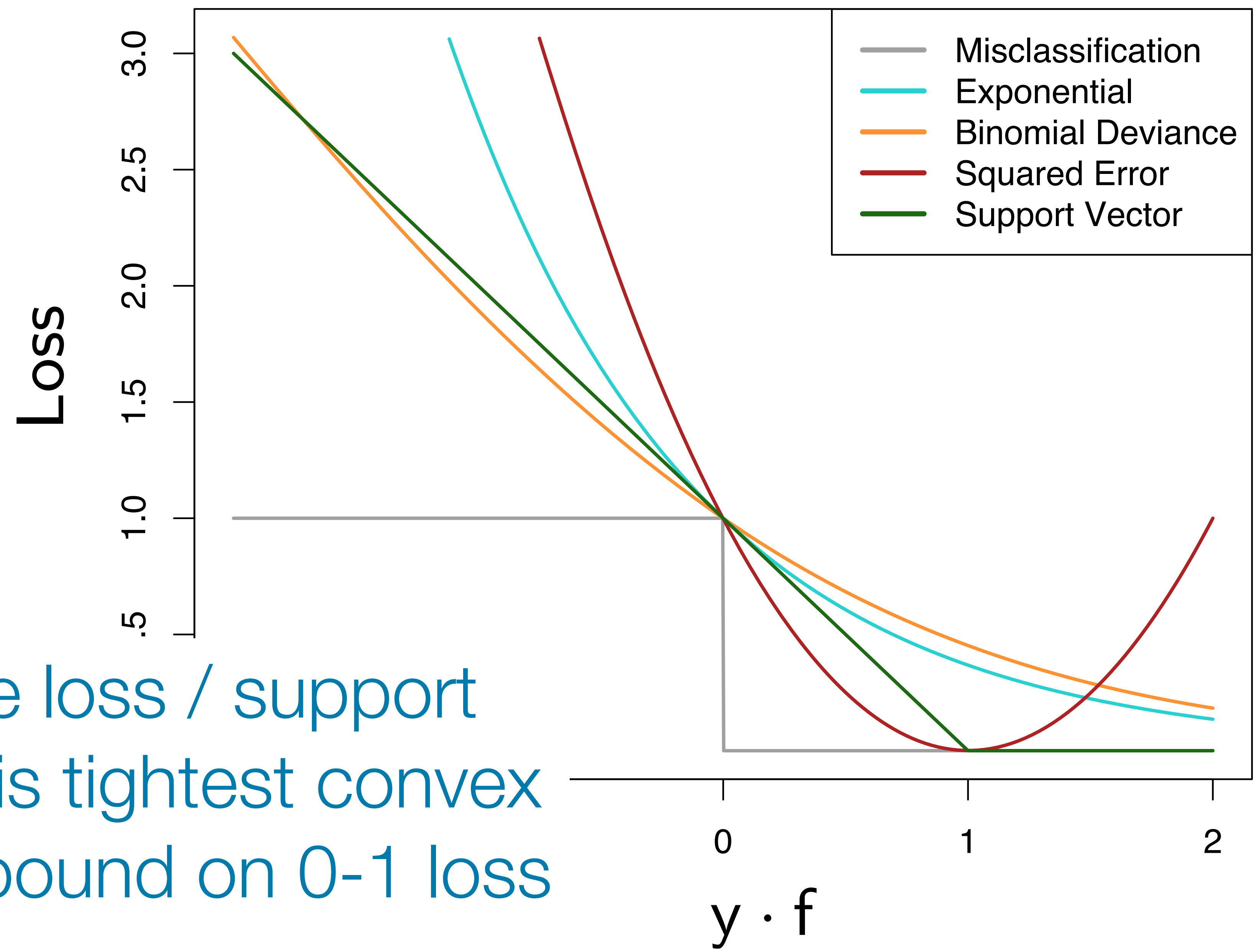
  - If functional margin is incorrect

$$y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i) < 1 \rightarrow \xi_i = 1 - y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i)$$

  - Optimal slack variable

$$\xi_i = \max(0, 1 - y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i))$$

Recap: Classification Loss



Hinge loss / support vector is tightest convex upper bound on 0-1 loss

Figure 10.4 (Hastie et al.)

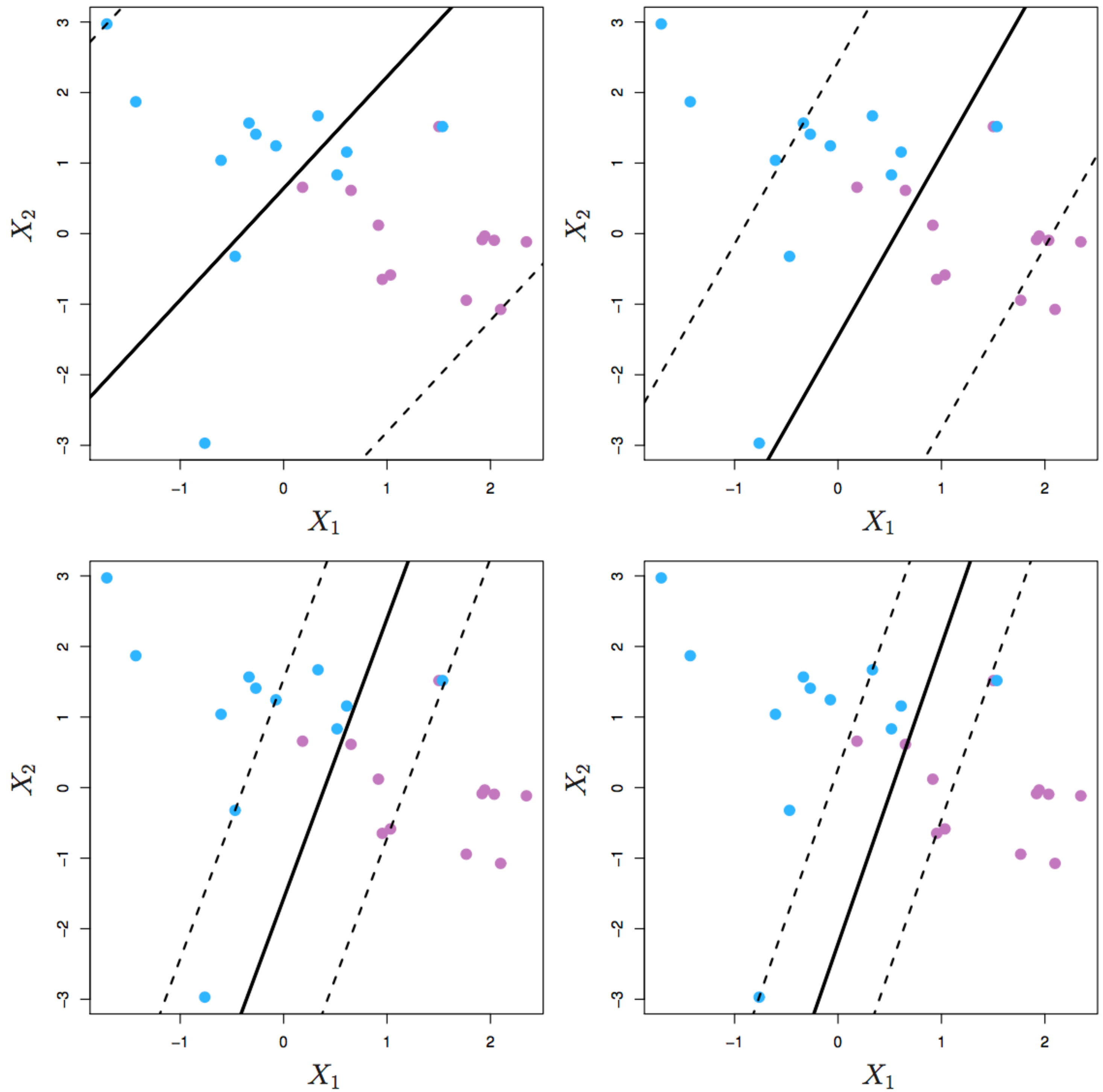
Key Idea #2: Slack

- Convert to an unconstrained optimization problem

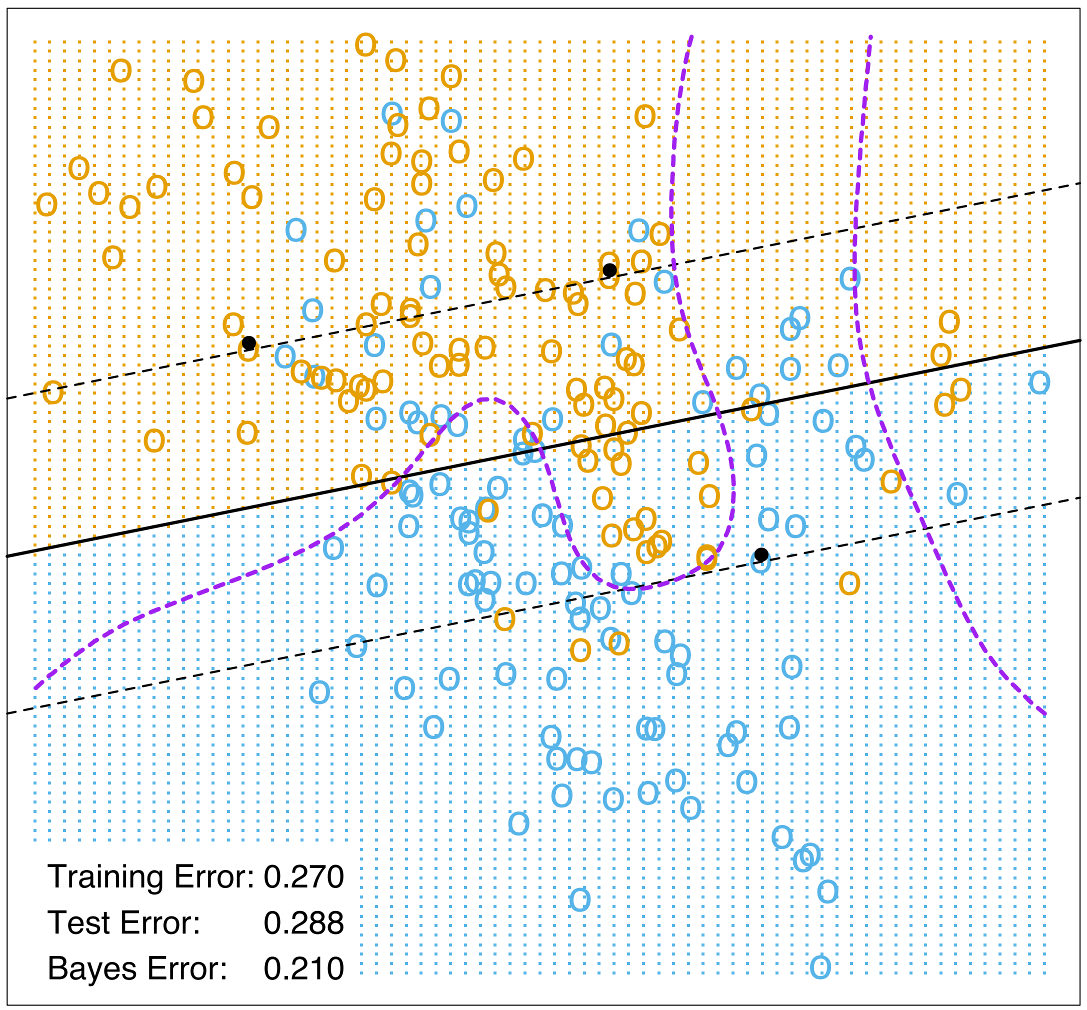
$$\min \left(||\boldsymbol{\beta}||_2^2 + C \sum_i \max(0, 1 - y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i)) \right)$$

- Equivalent form looks like regularization term + hinge loss
- As C gets large, have to separate the data
- As C gets small, ignores the data entirely

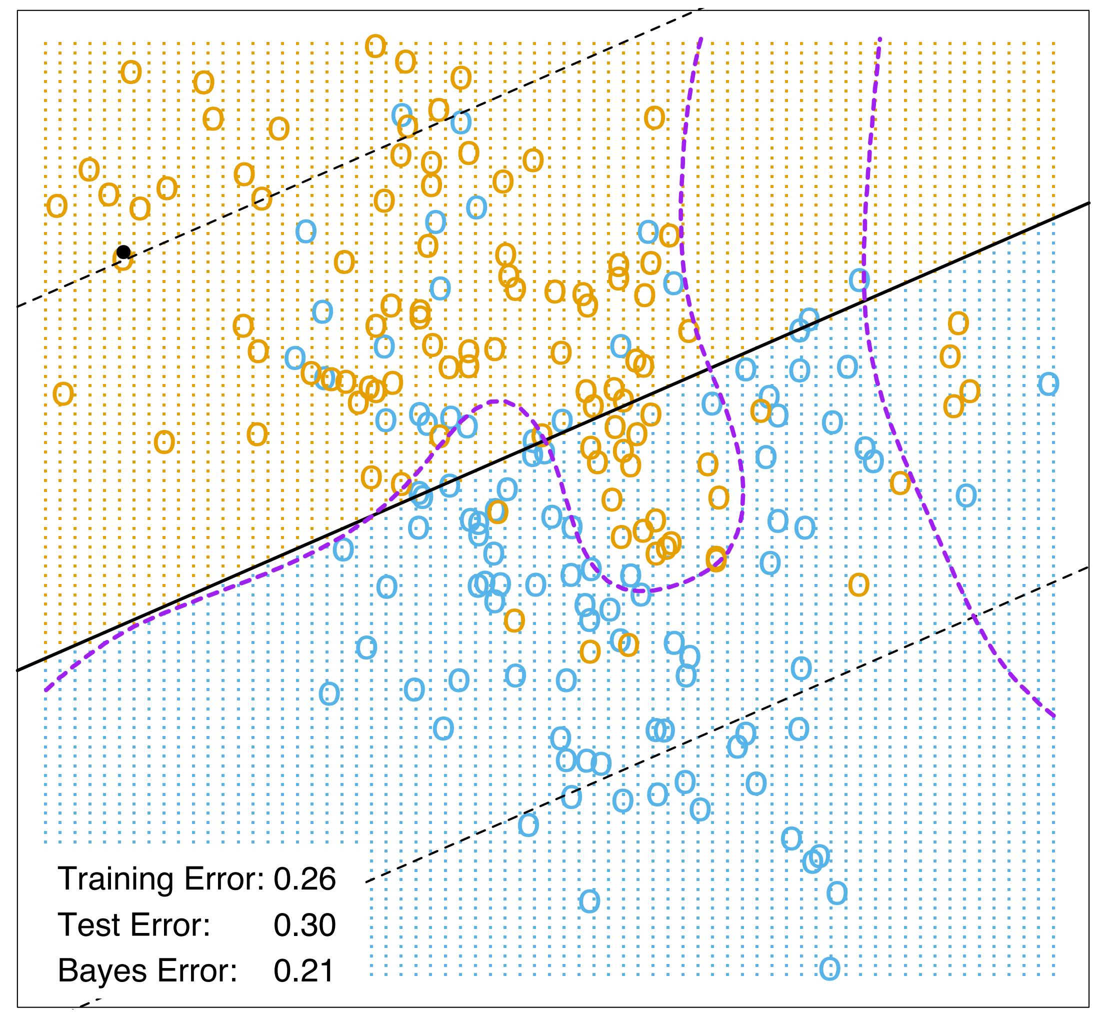
C Regularization Parameter



Example: Linear SVM



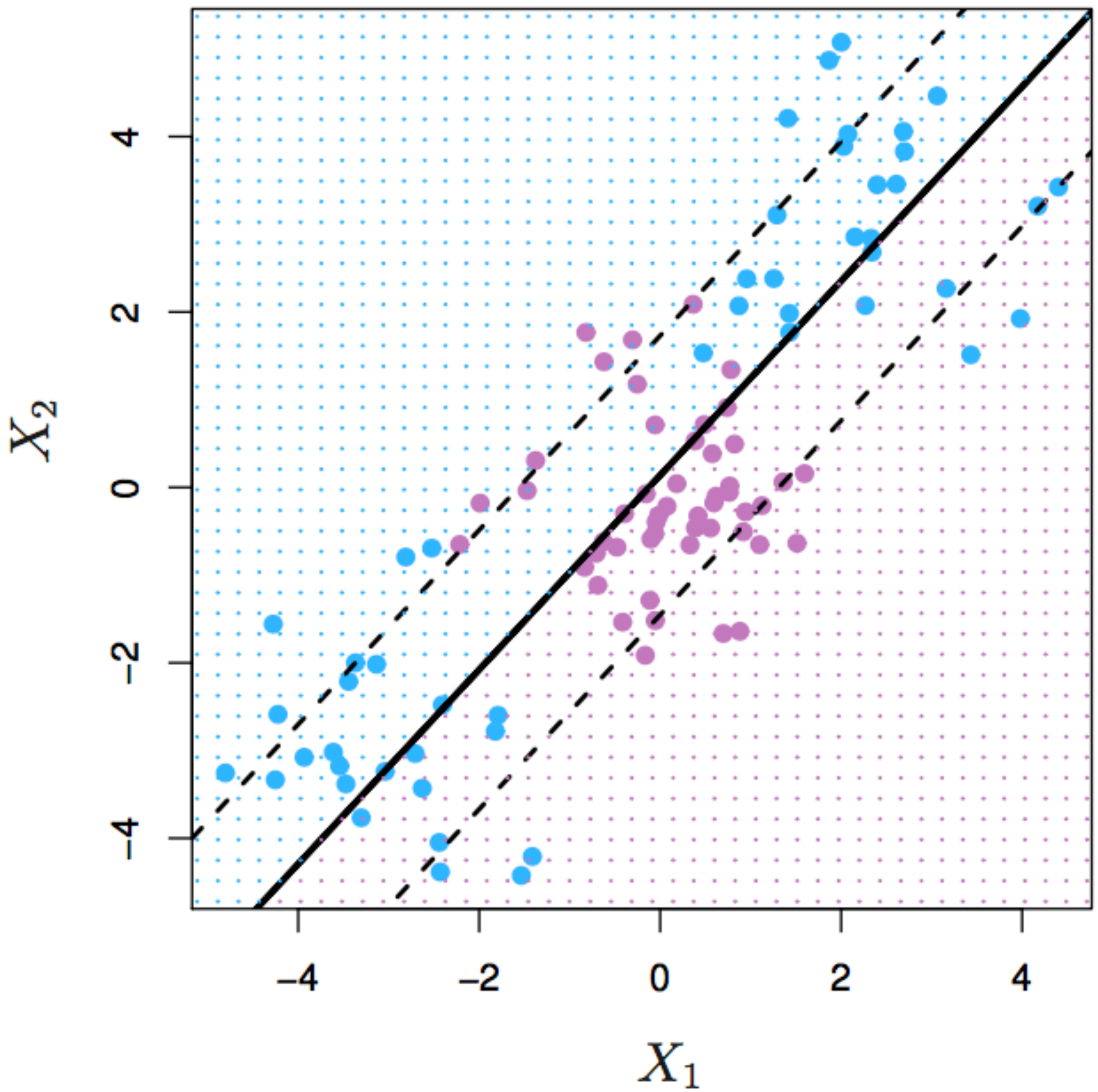
$C = 10000$



$C = 0.01$

Figure 12.2 (Hastie et al.)

Failure of Linear Boundaries



Sometimes a linear boundary won't work. What should we do in this case?

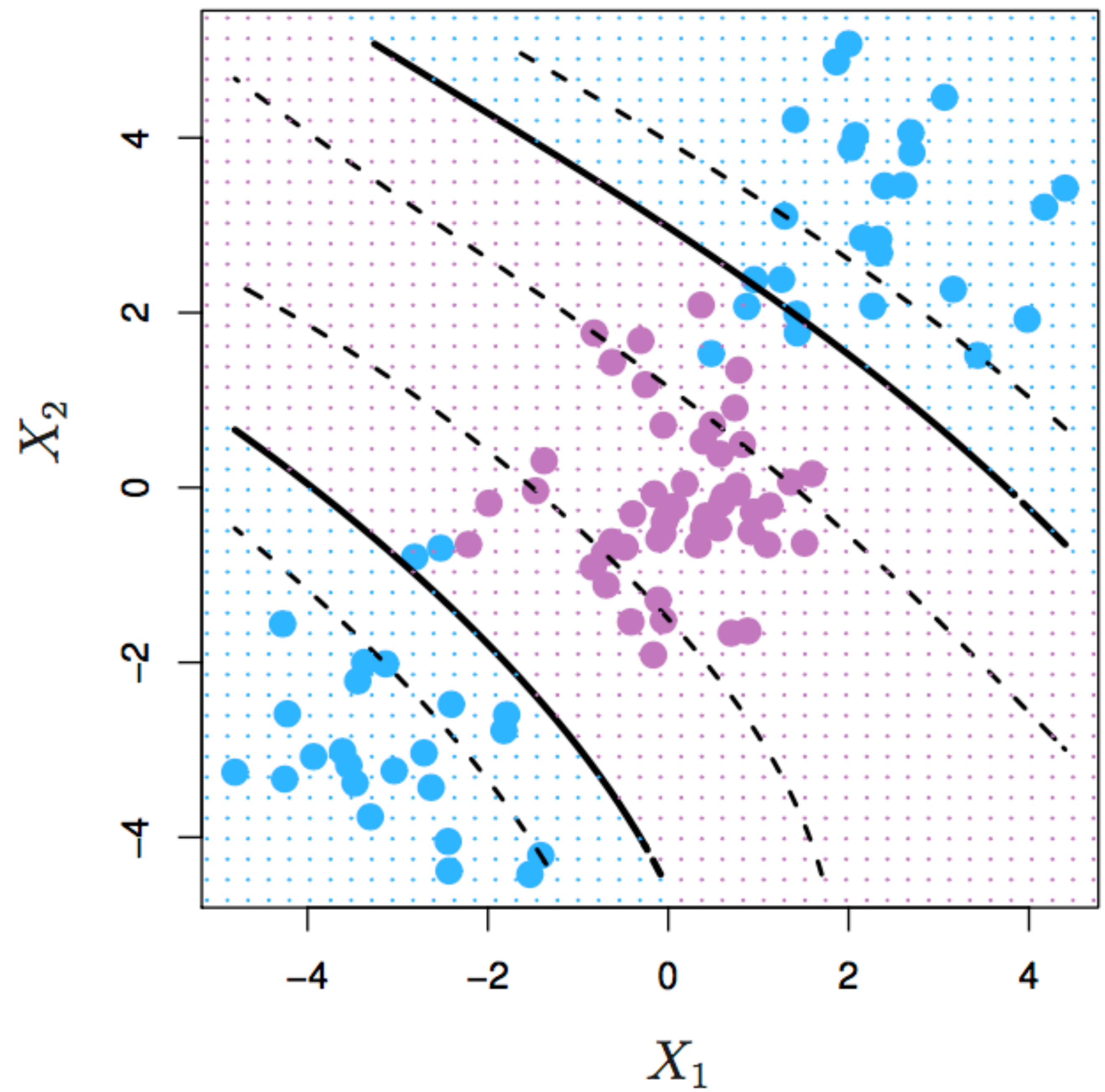
Feature Expansion

- Enlarge the space of features by including transformations
  - Example: $x_1^2, x_1^3, x_1 x_2, x_1 x_2^2$
  - Feature space dimension from p to D where D > p
- Fit SVM on new feature space —> non-linear decision boundaries in original space

Example: Cubic Polynomials

- 2 —> 9 features
- SVM on new feature space solves the problem in the lower-dimensional space

Is there a more elegant and controlled way to introduce nonlinearities?



Key Idea #3: Kernels

- Solve for hyperplane in high dimensional space where data is separable
- High dimensional feature spaces at no extra cost
- If D is very large, many more parameters to learn than in original space. Can we use just the data points to learn the separating hyperplane?

SVM: Optimization Problem

- Primal problem

$$\min \left(||\boldsymbol{\beta}||_2^2 + C \sum_i \max(0, 1 - y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i)) \right)$$

- Computationally convenient to express SVM classifier as

$$\min \frac{1}{2}||\boldsymbol{\beta}||_2^2 + C \sum_i \xi_i$$

$$\text{s.t. } \xi_i \geq 0,\ \ y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i) \geq (1 - \xi_i), \forall i$$

- This is a quadratic program

Review: Lagrange Duality

- Bound or solve an optimization problem via a different optimization problem
- Optimization problems (even non-convex) can be transformed to their dual problems
- Purpose of the dual problem is to determine the lower bounds for the optimal value of the original problem
- Sometimes, solving dual problem is easier

Review: Lagrangian

Original (primal) problem

$$\min_x \; f_0(x)$$

$$\text{s.t. } f_k(x) \le 0, k = 1, 2, \cdots, K$$

$$h_j(x) = 0, j = 1, 2, \cdots, J$$

Positivity constraints

Lagrangian function

$$L(x, \lambda, v) = f_0(x) + \sum_k \lambda_k f_k(x) + \sum_j v_j h_j(x)$$

Lagrange multipliers or dual variables

Review: Dual Problem

Primal problem

$$\min_x \; f_0(x)$$

$$\text{s.t. } f_k(x) \leq 0, k = 1, 2, \cdots, K$$

$$h_j(x) = 0, j = 1, 2, \cdots, J$$

Dual problem

$$\max g(\lambda, v) = \inf_x L(x, \lambda, v)$$

$$\text{subject to } \; \lambda \geq 0$$

$$g(\lambda, v) \leq L(\tilde{x}, \lambda, v) \leq f_0(\tilde{x})$$

Karush-Kuhn-Tucker Conditions

- For general optimization problem, satisfying Karush-Kuhn-Tucker (KKT) conditions means zero duality gap between primal and dual solutions
  - Stationarity: $0 \in \partial f_0(x) + \sum_k \lambda_k \partial f_k(x) + \sum_j v_j \partial h_j(x)$
  - Complementary slackness: $\lambda_k f_k(x) = 0, \ \forall i$
  - Primal feasibility: $f_k(x) \leq 0, \ h_j(x) = 0, \ \forall i, j$
  - Dual feasibility: $\lambda_k \geq 0, \ \forall k$

SVM: Lagrange Function

- Lagrange (primal) function

$$L_p = \frac{1}{2}||\boldsymbol{\beta}||_2^2 + C\sum_i \xi_i$$
$$-\sum_i \alpha_i[y_i(\boldsymbol{\beta}\cdot\mathbf{x}_i + \beta_0) - (1-\xi_i)] - \sum_i \mu_i\xi_i$$

- Dual variables

$$\alpha_i \geq 0,\;\; \mu_i \geq 0$$

SVM: Minimize w.r.t. Primal

- Minimize with respect to primal variables

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_i \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \boldsymbol{\beta} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial \beta_0} = \sum_i y_i \alpha_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \Rightarrow \alpha_i = C - \mu_i$$

SVM: KKT Conditions

- Subset of KKT conditions:

$$
\begin{aligned}
\alpha_i[y_i(\boldsymbol{\beta} \cdot \mathbf{x}_i + \beta_0 - (1 - \xi_i)] = 0 \\
\mu_i \xi_i = 0 \\
y_i(\boldsymbol{\beta} \cdot \mathbf{x}_i + \beta_0) - (1 - \xi_i) \geq 0
\end{aligned}
$$

SVM: Complementary Slackness

- Look at data points and dual variables
  - Non support vectors: points correctly classified

$$\alpha_i = 0 \Rightarrow \mu_i \neq 0 \Rightarrow \xi_i = 0$$

  - Margin support vectors: points on margin correctly classified

$$0 < \alpha_i < C \Rightarrow \mu_i \neq 0 \Rightarrow \xi_i = 0$$

  - Non-margin support vectors: points incorrectly classified

$$\alpha_i = C \Rightarrow \mu_i = 0 \Rightarrow \xi_i > 0$$

SVM: Primal Solution via Dual Variable

- Final solution as linear combination of training data

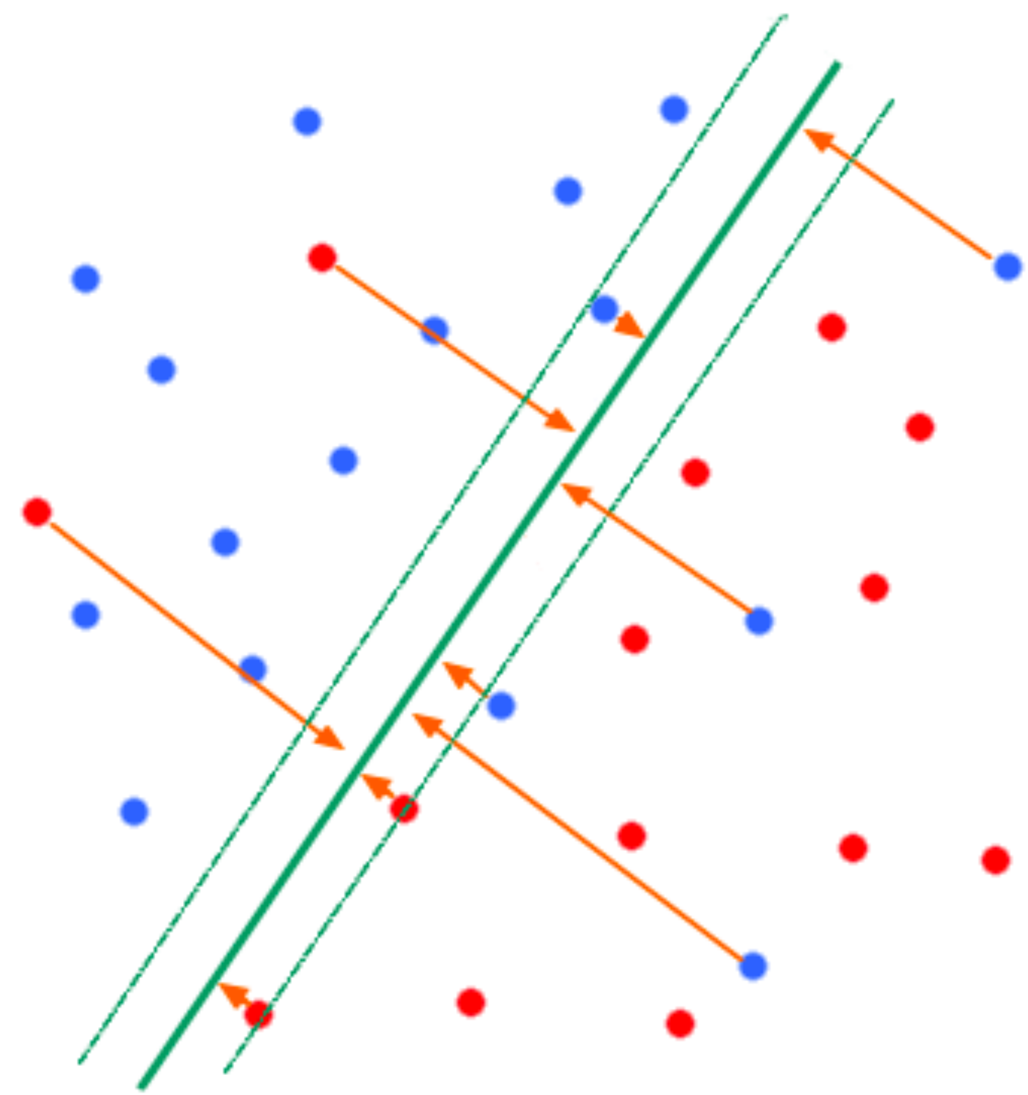
$$\boldsymbol{\beta} = \sum_i \alpha_i y_i \mathbf{x}_i$$

- Sparse
    - Non-support vectors

    $$\alpha_j = 0$$

    - Support vectors

    $$\alpha_j \geq 0$$



http://rwarloplabs.com/machinelearning/posts/svm.php

SVM: Dual Problem

- Substitute primal optimal solution into dual objective

$$
\begin{aligned}
g(\alpha, \mu) = \frac{1}{2} || \sum_i \alpha_i y_i \mathbf{x}_i ||_2^2 + C \sum_i \xi_i \\
- \sum_i \alpha_i [y_i ((\sum_j \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_i + \beta_0) - (1 - \xi_i)] \\
- \sum_i (C - \alpha_i) \xi_i \\
= \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j
\end{aligned}
$$

SVM Primal vs Dual

- Primal problem: learn p parameters

$$\min \left(||\boldsymbol{\beta}||_2^2 + C \sum_i \max(0, 1 - y_i(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x}_i)) \right)$$

- Dual problem: learn N parameters

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, \ \forall i, \sum_i \alpha_i y_i = 0$$

- Dual form only involves $(\mathbf{x}_i^\top \mathbf{x}_j)$

Inner Products & Support Vectors

- Inner product provide some measure of 'similarity'
- In Euclidean space, inner product is dot product

$$< \mathbf{x}, \mathbf{y} >= \mathbf{x}^\top \mathbf{y} = \sum_i x_i y_i, \text{where } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

- Can rewrite the dual problem to use inner products

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j < \mathbf{x}_i, \mathbf{x}_j >$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, \; \forall i, \sum_i \alpha_i y_i = 0$$

Key Idea #3: Kernels

- Feature map (original feature to new feature space):

$$\Phi : \mathbf{x} \rightarrow \Phi(\mathbf{x}), \mathbb{R}^p \rightarrow \mathbb{R}^D$$

- Hyperplane:

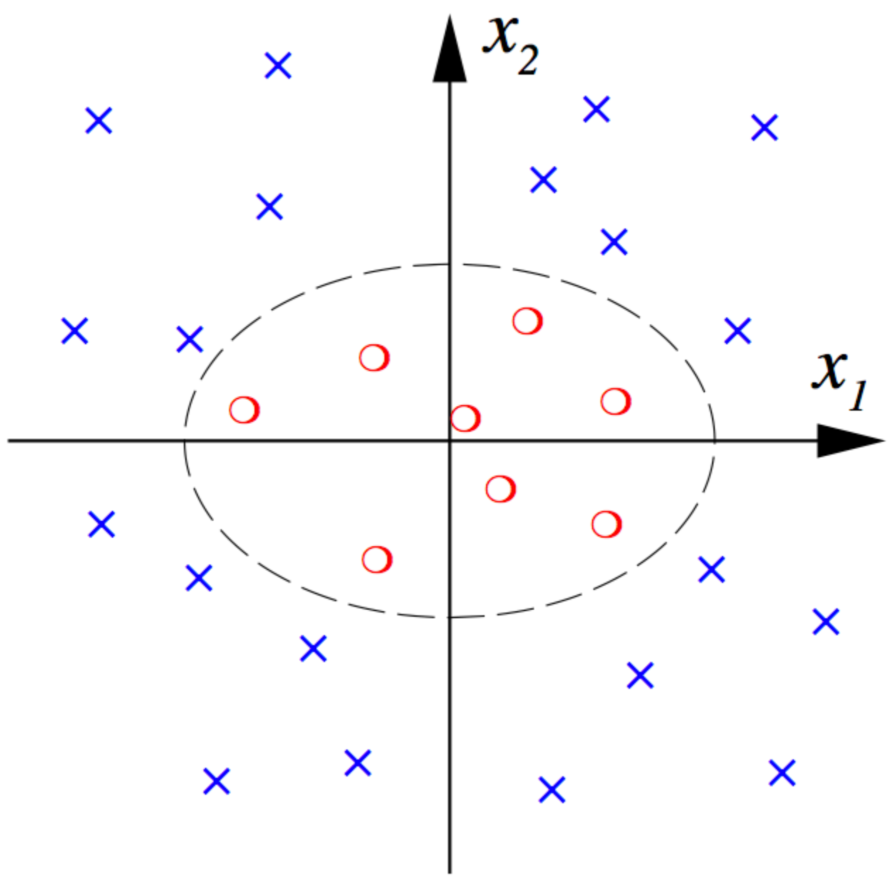
$$f(\mathbf{x}) = \boldsymbol{\beta} \cdot \Phi(\mathbf{x}_i) + \beta_0$$

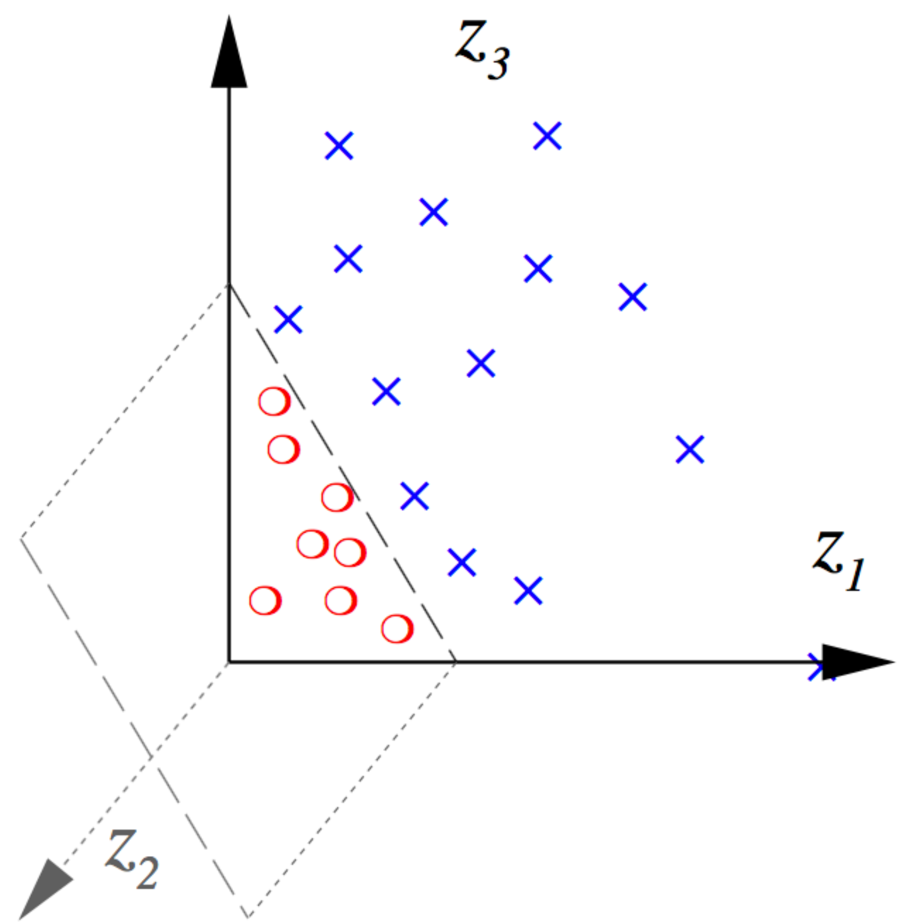
- Primal problem:

$$\min \left(||\boldsymbol{\beta}||_2^2 + C \sum_i \max(0, 1 - y_i(\beta_0 + \boldsymbol{\beta} \cdot \Phi(\mathbf{x}_i))) \right)$$

Example: Quadratic Features

$$\Phi : R^2 \rightarrow R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt(2) x_1 x_2, x_2^2)$$





Key Idea #3: Kernels

- Primal solution with respect to dual:

$$f(\mathbf{x}) = \sum_j \alpha_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}) + \beta_0$$

- Dual problem:

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j < \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, \ \forall i, \sum_i \alpha_i y_i = 0$$

Key Idea #3: Kernels

- Primal space: need to learn in the new D dimension space
- Dual space: only needs to compute the inner product between the pairs to learn N dimensional vector
- Introduce the notion of a kernel (finally):

$$K(\mathbf{x}_i, \mathbf{x}_j) = < \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$$

Key Idea #3: Kernels

- Dual problem

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

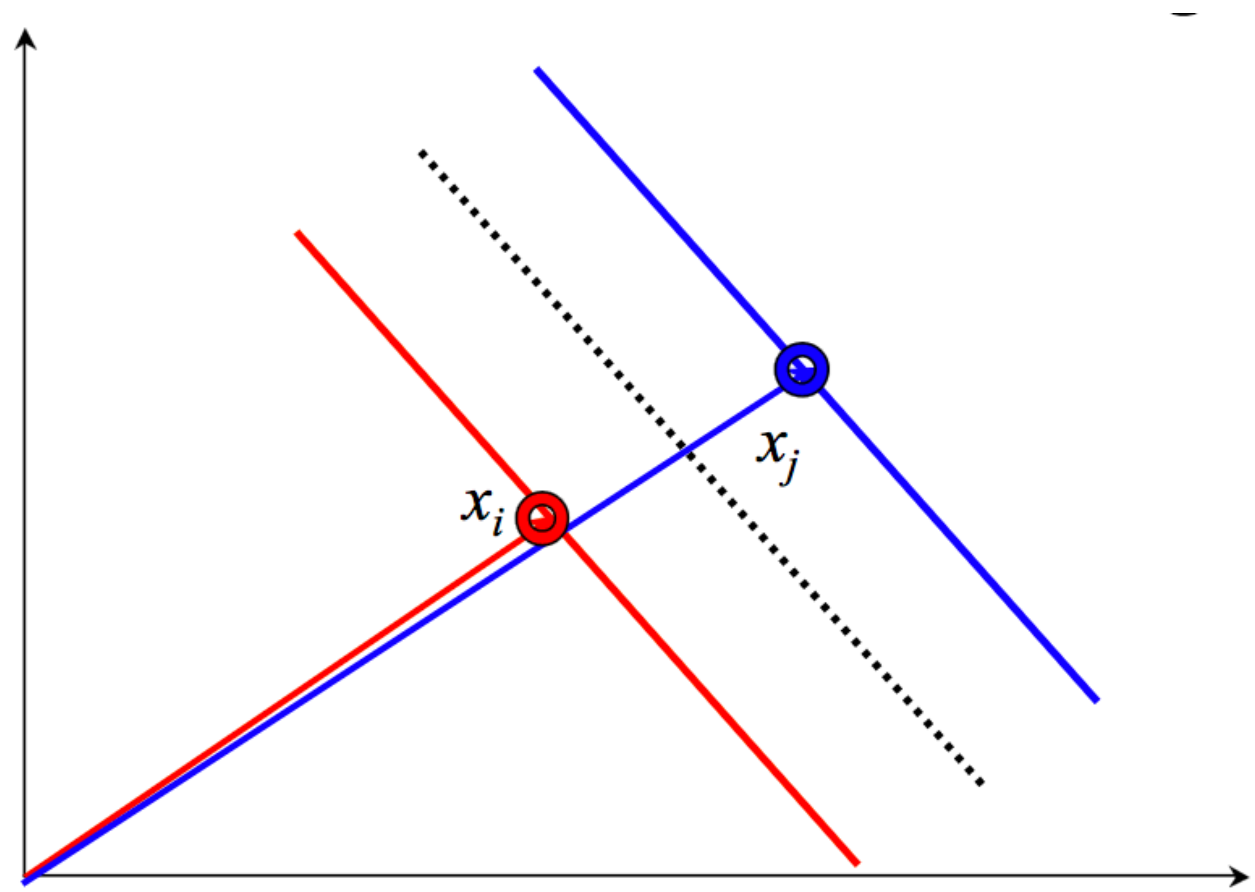
$$\text{s.t. } 0 \leq \alpha_i \leq C, \ \forall i, \sum_i \alpha_i y_i = 0$$

- Kernel function is used to make non-linear feature map

- Think of kernel measure as similarity

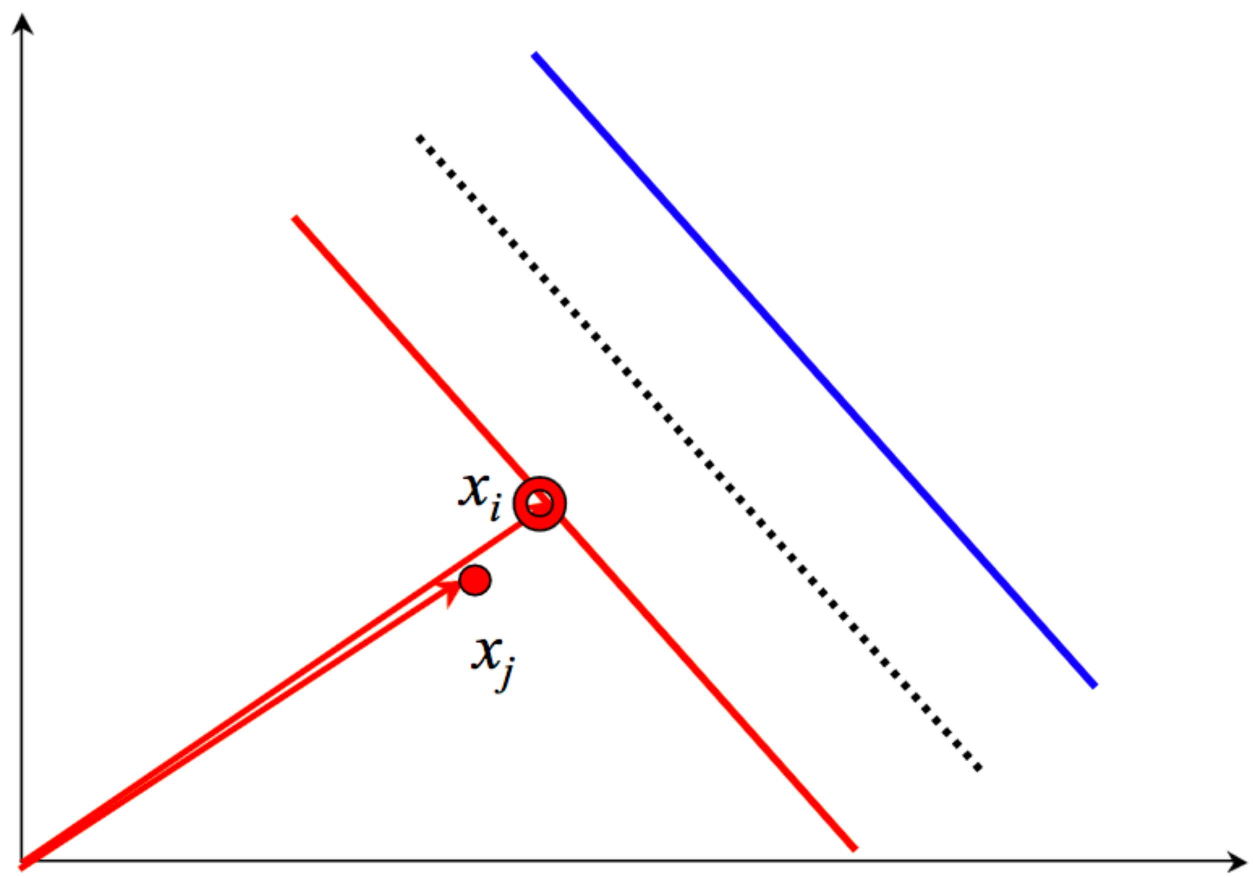
Intuition of Kernels

- 2 very similar vectors that predict different classes tend to maximize the margin width



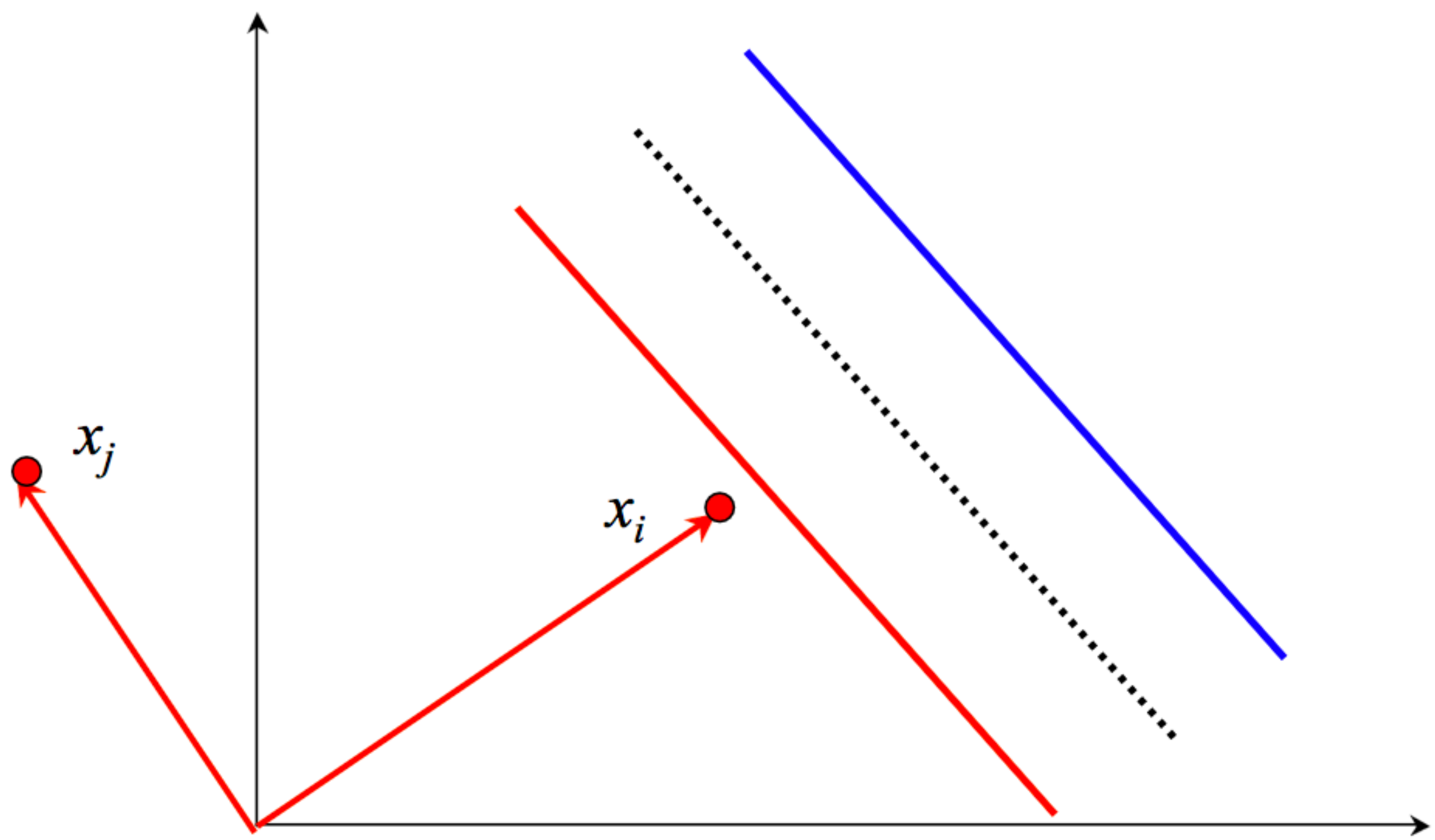
Intuition of Kernels

- 2 similar vectors that predict same class are redundant, keep the one closer to the margin



Intuition of Kernels

- 2 dissimilar vectors that predict same class don't count at all



Polynomial SVM

- Kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$$

- Example: polynomial of degree 2

  - Explicit computation:

$$\begin{aligned}\Phi((x_1, x_2)) \cdot \Phi((\tilde{x}_1, \tilde{x}_2)) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (\tilde{x}_1^2, \sqrt{2}\tilde{x}_1\tilde{x}_2, \tilde{x}_2^2) \\ &= x_1^2\tilde{x}_1^2 + 2x_1\tilde{x}_1x_2\tilde{x}_2 + x_2^2\tilde{x}_2^2\end{aligned}$$

  - Kernel:

$$\begin{aligned}K(\mathbf{x}, \tilde{\mathbf{x}}) &= (\mathbf{x} \cdot \tilde{\mathbf{x}})^2 \\ &= x_1^2\tilde{x}_1^2 + 2x_1\tilde{x}_1x_2\tilde{x}_2 + x_2^2\tilde{x}_2^2\end{aligned}$$

Benefits of Kernels

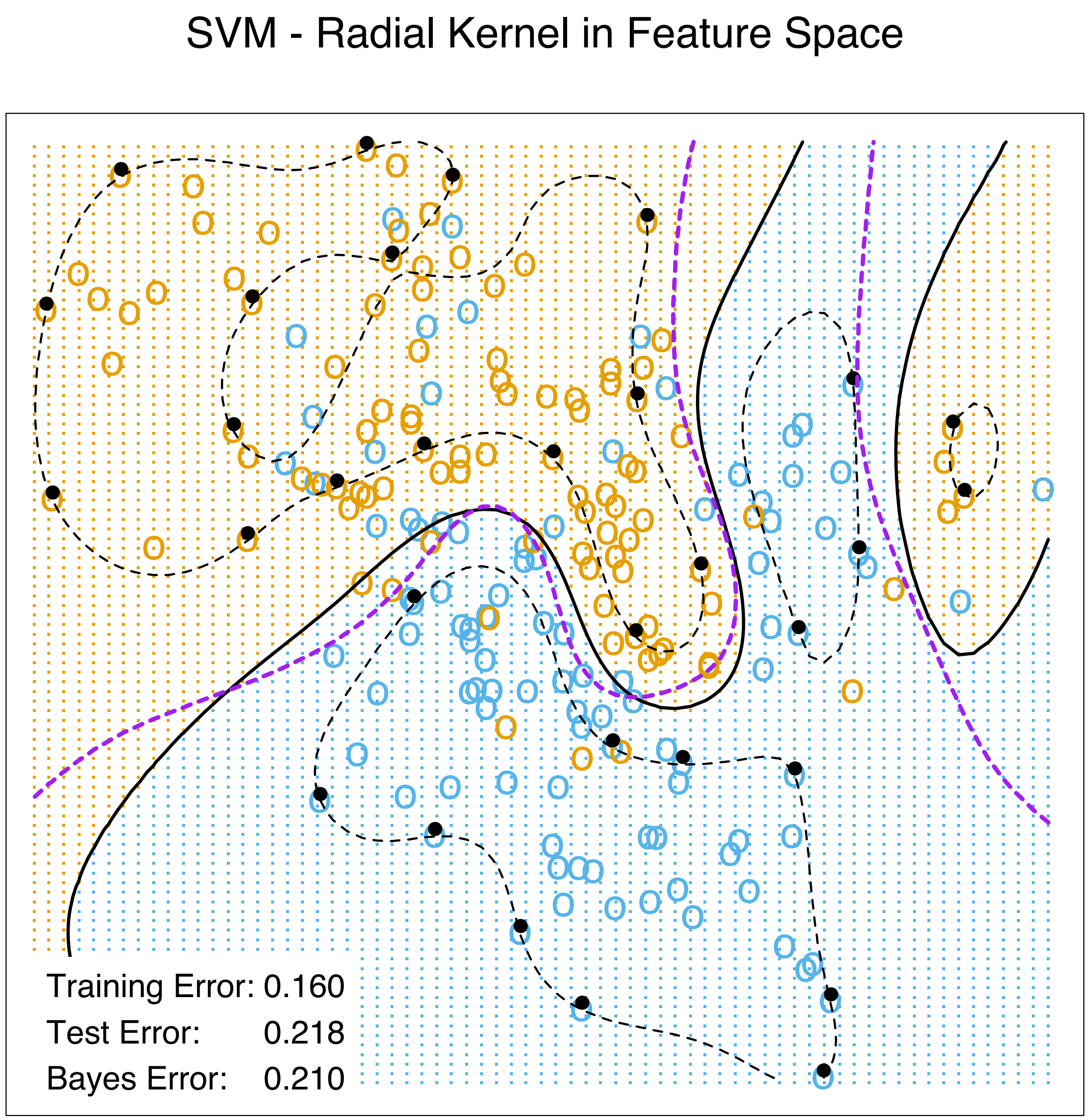
- Efficient: often times easier than computing feature map and then dot product
  - Especially from memory perspective — need to store less
- Flexibility: function chosen arbitrary so long as existence of feature map is guaranteed

Common Kernels

Name	Kernel Function
Polynomials of degree exactly d	$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$
Polynomials of degree up to d	$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$
Gaussian / Radial	$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\gamma \|\|\mathbf{u} - \mathbf{v}\|\|_2^2\right)$
Sigmoid (neural network)	$K(\mathbf{u}, \mathbf{v}) = \tanh(\eta \mathbf{u} \cdot \mathbf{v} + \nu)$

Active area of research!

Example: Nonlinear Kernels



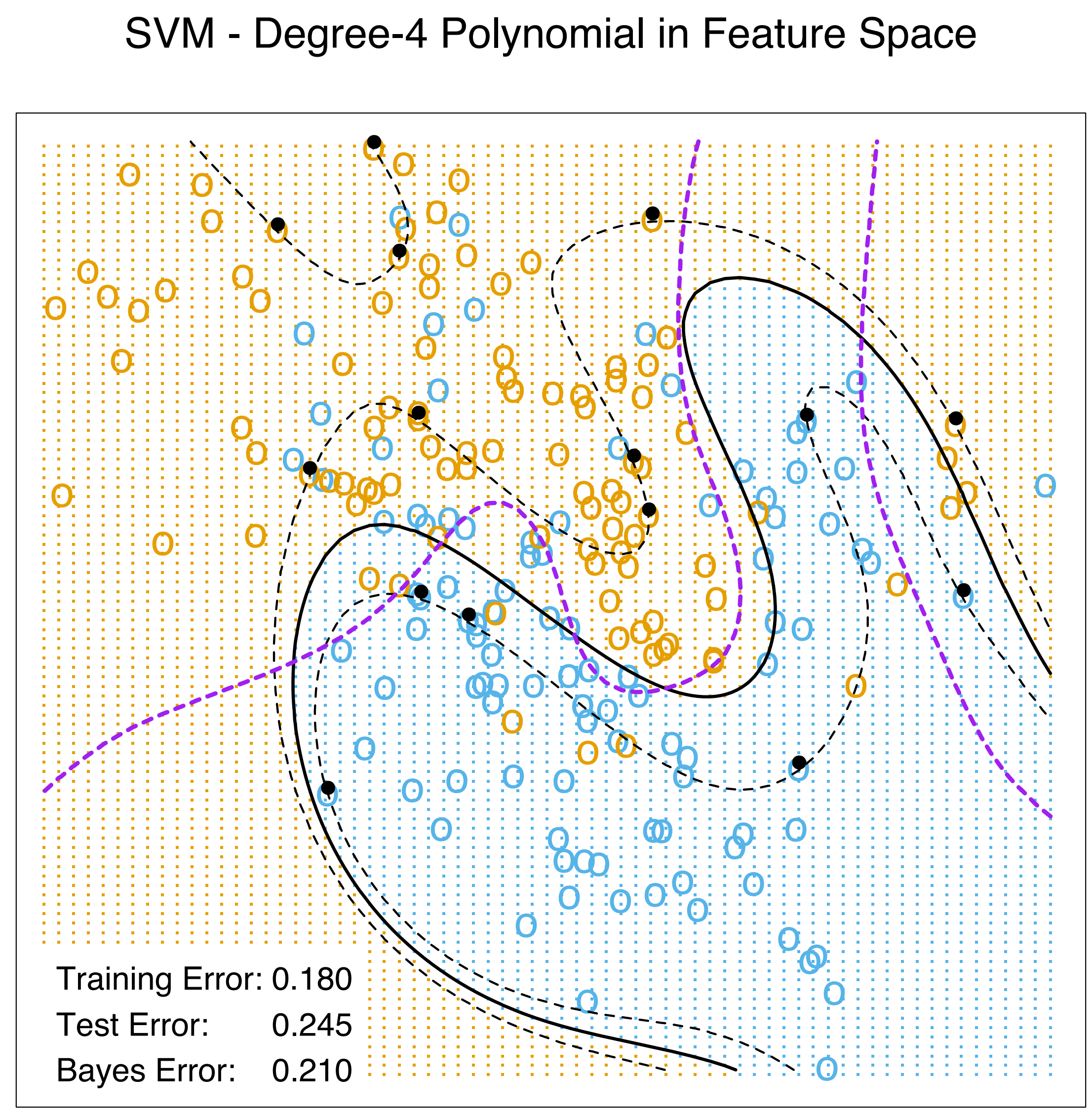


Figure 12.3 (Hastie et al.)

Example: Radial Kernel Curves

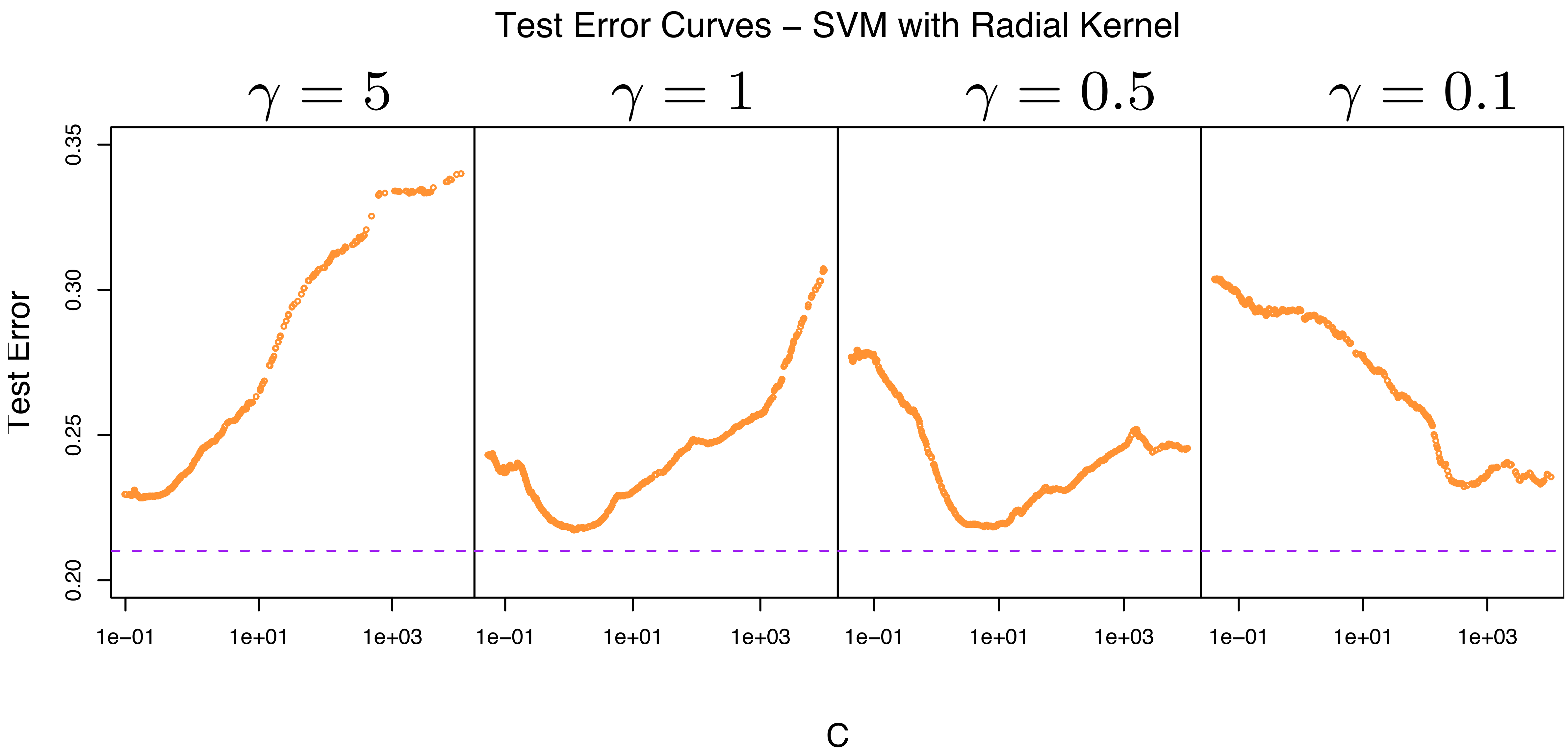


Figure 12.6 (Hastie et al.)

Kernel SVM: Overfitting

- Huge feature space with kernels — what about overfitting?
    - SVM theory says that a solution with a large margin leads to good generalization
    - But overfitting is always likely at some point in time

Example: Skin of Orange

- 2 class problem
- First class as 4 standard normal independent features
- Second class has 4 standard normal independent features but conditioned on the 2-norm being between 9 and 16
- Augment features with 6 additional standard Gaussian noise features

Example: Kernel SVM

TABLE 12.2. *Skin of the orange: Shown are mean (standard error of the mean) of the test error over 50 simulations. BRUTO fits an additive spline model adaptively, while MARS fits a low-order interaction model adaptively.*

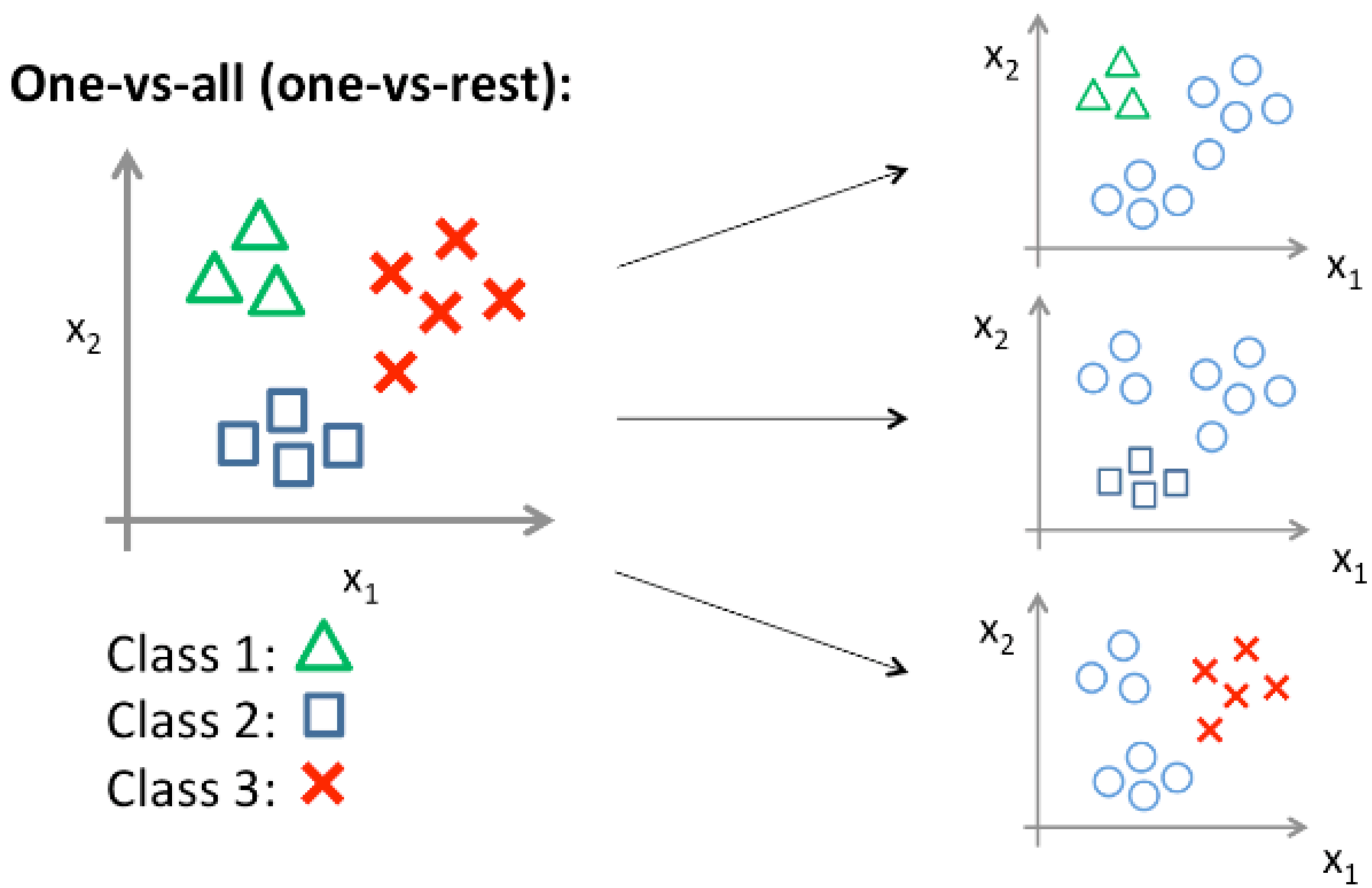
		Test Error (SE)	
	Method	No Noise Features	Six Noise Features
1	SV Classifier	0.450 (0.003)	0.472 (0.003)
2	SVM/poly 2	0.078 (0.003)	0.152 (0.004)
3	SVM/poly 5	0.180 (0.004)	0.370 (0.004)
4	SVM/poly 10	0.230 (0.003)	0.434 (0.002)
5	BRUTO	0.084 (0.003)	0.090 (0.003)
6	MARS	0.156 (0.004)	0.173 (0.005)
	Bayes	0.029	0.029

Kernel SVM: Overfitting

- Control overfitting by
  - Setting C via cross-validation
  - Choose better kernel
  - Vary parameters of the kernel (i.e., width of Gaussian, etc.)

Multi-class SVM

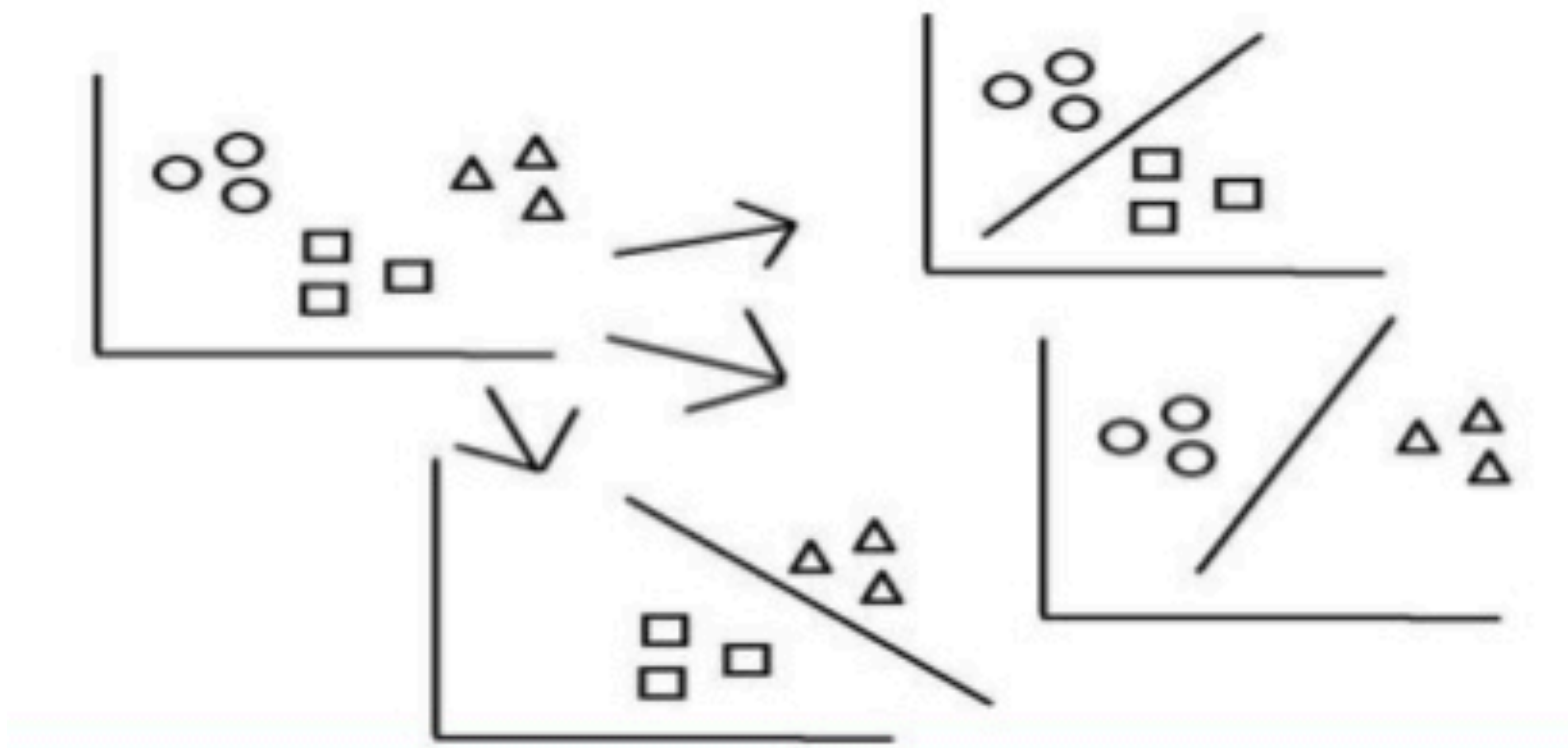
- One versus all: Fit K different 2-class SVM classifiers, each class versus the rest. Classify for the largest value



https://houxianxu.github.io/2015/04/25/support-vector-machine/

Multi-class SVM

- One versus one: Fit all pairwise classifiers. Classify using the class that wins the most pairwise competitions



https://www.slideshare.net/Paxcel/binary-and-multi-class-strategies-for-machine-learning

SVM vs Logistic Regression

- For linear kernel, rephrase SVM optimization problem:

$$\min\left(\sum_i \max(0, 1 - y_i f(\mathbf{x}_i)) + \lambda||\boldsymbol{\beta}||_2^2\right)$$

- Loss + penalty
- Similar to logistic regression
- Hinge loss is slightly different

SVM vs Logistic Regression

- For (nearly) separable classes, SVM and LDA is better than logistic regression
- When not, logistic regression with ridge penalty and SVM are very similar
- Logistic regression provides probabilities while SVM does not
- For nonlinear boundaries, kernel SVM is popular and computationally efficient

Support Vector Regression

- Adapt SVM for regression
- Linear SVR model using "ϵ -insensitive" error measure
  - Loss measure:

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

  - Minimization objective (separable case):

$$\sum_i V_\epsilon(y_i - (\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0)) + \frac{\lambda}{2} ||\boldsymbol{\beta}||_2^2$$

SVR Error Measure

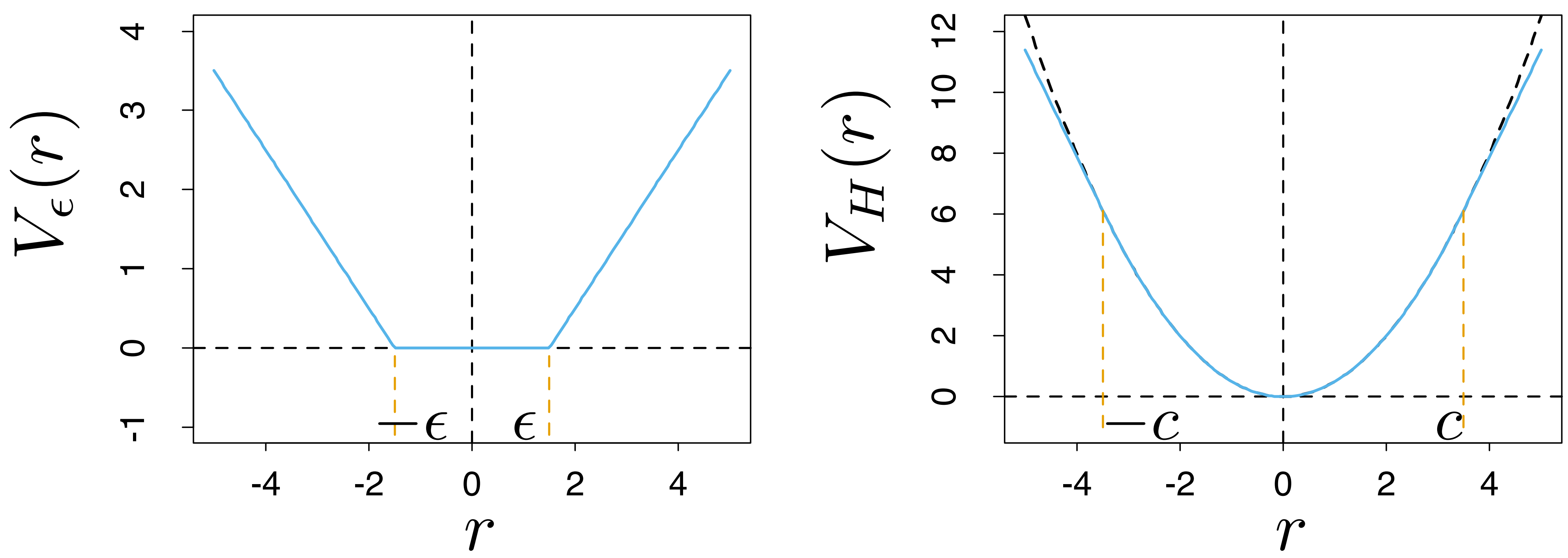


Figure 12.8 (Hastie et al.)

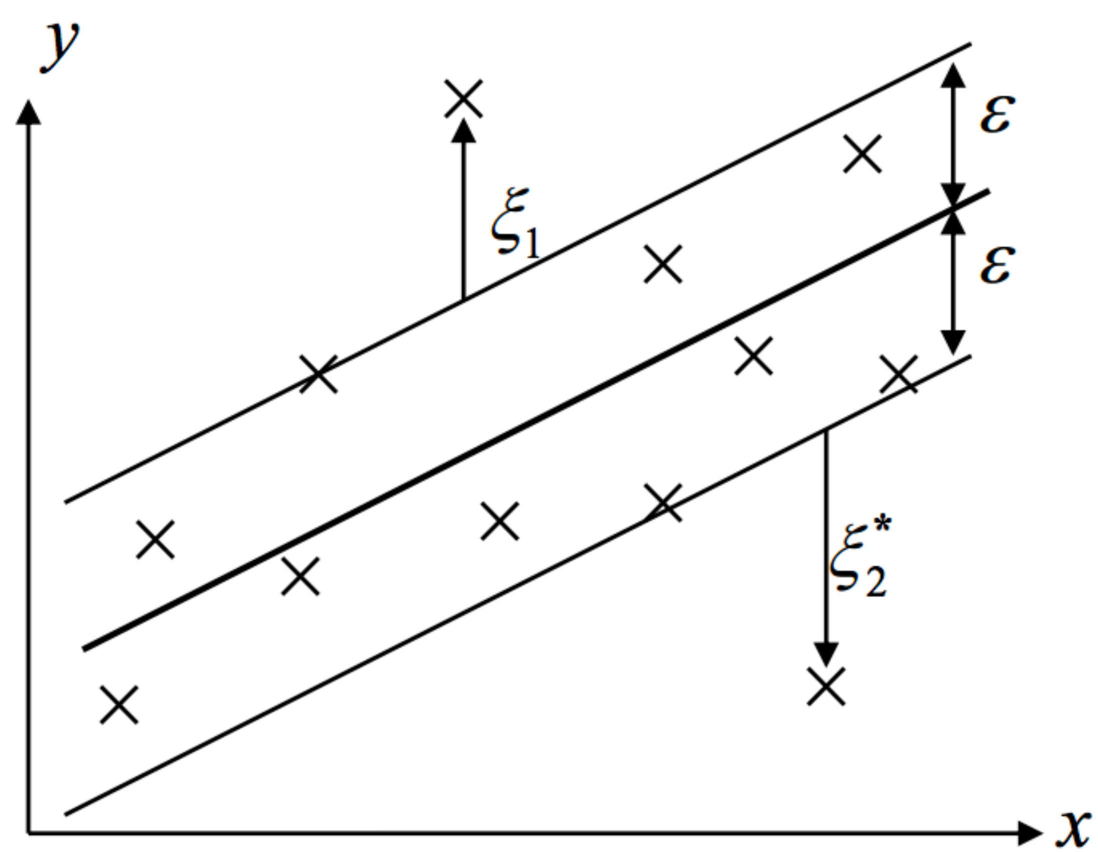
Linear SVR Problem

$$\min \frac{1}{2}||\boldsymbol{\beta}||_2^2 + C\sum_i(\xi_i + \xi_i^*)$$

$$\text{s.t. } y_i - \mathbf{x}_i^\top \boldsymbol{\beta} - \beta_0 \leq \epsilon + \xi_i$$

$$\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0 - y_i \leq \epsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0$$



Linear SVR Problem

- Solution function has the form:

$$\hat{\beta} = \sum_i (\hat{\alpha}_i^* - \hat{\alpha}_i)\mathbf{x}_i$$

$$\hat{f}(\mathbf{x}) = \sum_i (\hat{\alpha}_i^* - \hat{\alpha}_i) < \mathbf{x}, \mathbf{x}_i > + \beta_0$$

kernel trick!

- Dual:

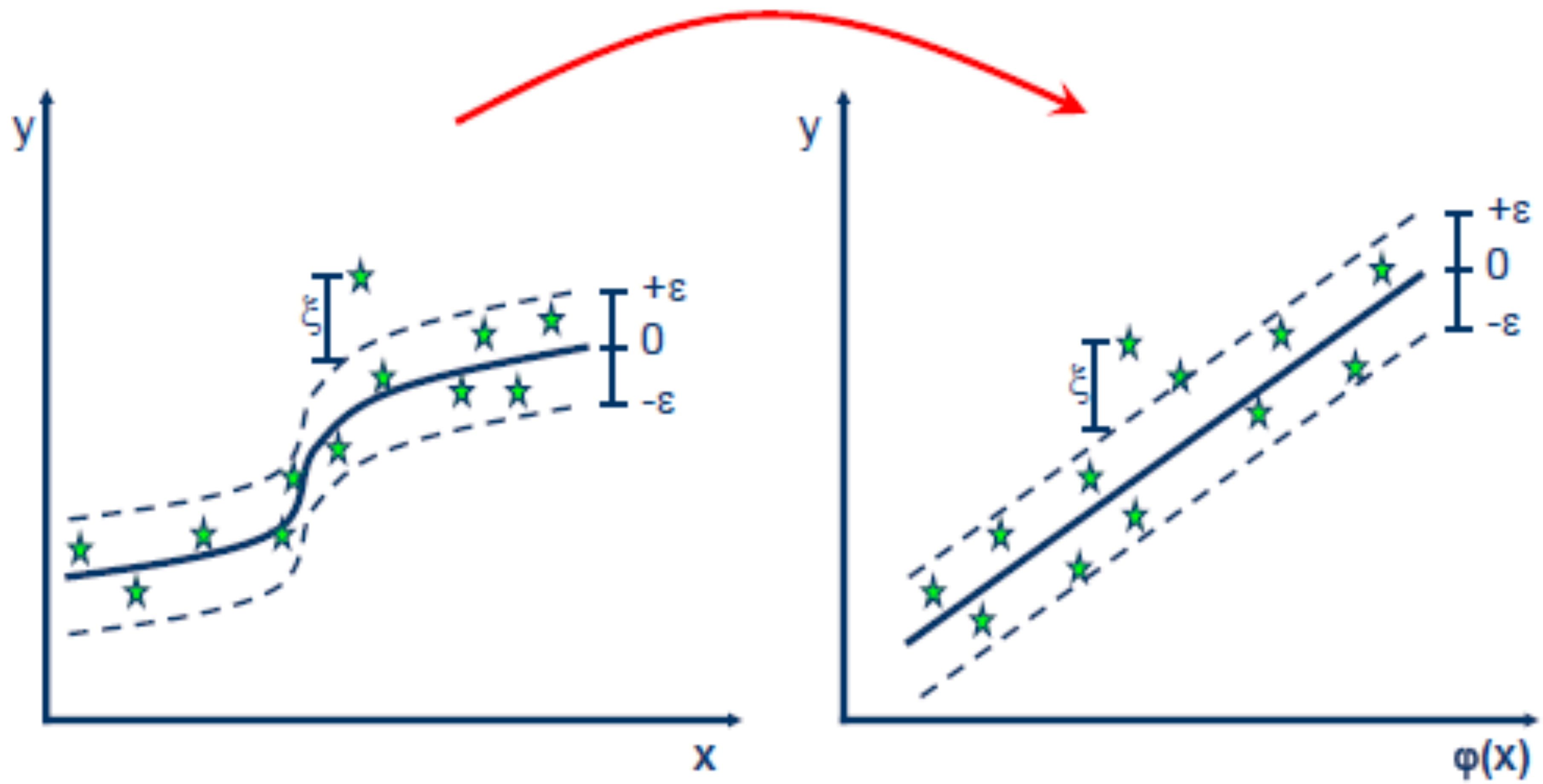
$$\min \frac{1}{2} \sum_i \sum_j (\hat{\alpha}_i^* - \hat{\alpha}_i)(\hat{\alpha}_j^* - \hat{\alpha}_j) \boxed{< \mathbf{x}_i, \mathbf{x}_j >}$$

-

$$+ \epsilon \sum_i (\alpha_i^* - \alpha_i) - \sum_i y_i(\alpha_i^* - \alpha_i)$$

$$\text{s.t.} \sum_i (\alpha_i - \alpha_i^*) = 0, 0 \leq \alpha_i, \alpha_i^* \leq \frac{1}{\lambda}, \;\; \alpha_i \alpha_i^* = 0$$

Example: Kernel SVR



http://webgol.dinfo.unifi.it/wordpress/wp-content/uploads/2016/02/pic.png