CS6375: Machine Learning

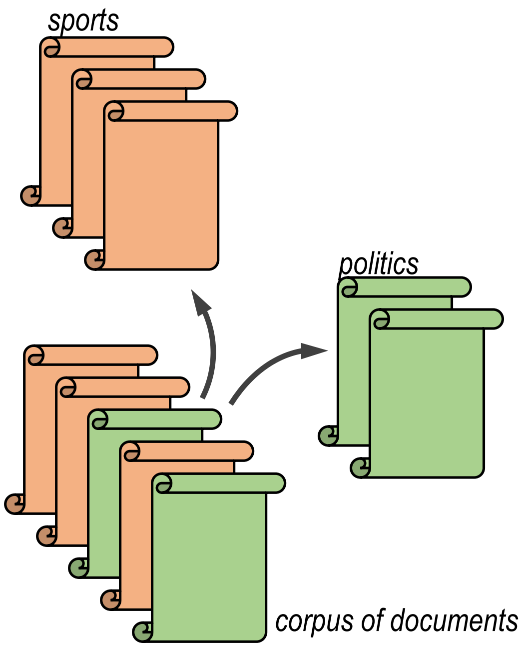
Thanks to Gautam Kunapuli

Support Vector Machines

THE UNIVERSITY OF TEXAS AT DALLAS
Erik Jonsson School of Engineering and Computer Science

Example: Text Categorization

Example: Develop a model to **classify news stories** into various categories based on their **content**.



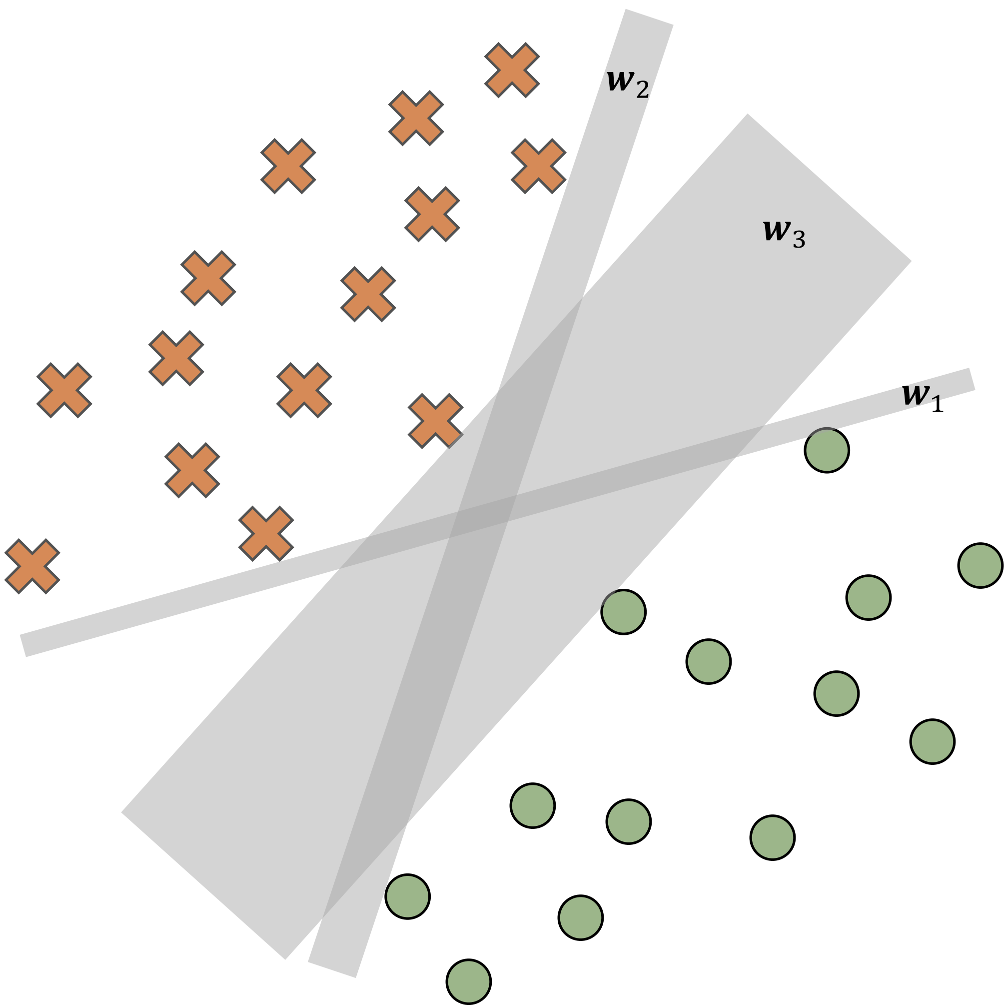
Use the **bag-of-words representation** for this data set

- text *across all documents* in the corpus is represented as a bag (multiset) of its words
- captures the multiplicity/**frequency** of words and terms
- does **not** capture semantics, sentiment, grammar, or even word order
- vector space model, where each document is simply represented as a vector of word statistics such as **counts**
- more advanced statistics such as **term-frequency/inverse document frequency** (tf-idf) can be used

	coach	conservative	doping	election	lawmakers	liberal	score	supreme	team	winner	**category**
document 1	13	0	4	0	1	0	7	0	8	6	sports
document 2	0	22	0	8	7	15	0	3	0	7	politics
document 3	2	6	0	1	19	11	2	0	2	0	politics
document n	5	3	4	0	1	2	12	2	8	1	sports

SVM for Linearly Separable Data

Problem: Find a **linear classifier** $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ such that
$$\mathbf{sign}\big(f(\boldsymbol{x})\big) = +1, \text{ when positive example}$$
$$\mathbf{sign}\big(f(\boldsymbol{x})\big) = -1, \text{ when negative example}$$



The data set is **linearly separable**, that is separable by a linear classifier (hyperplane). There exist many different classifiers! **Which one is the best?**

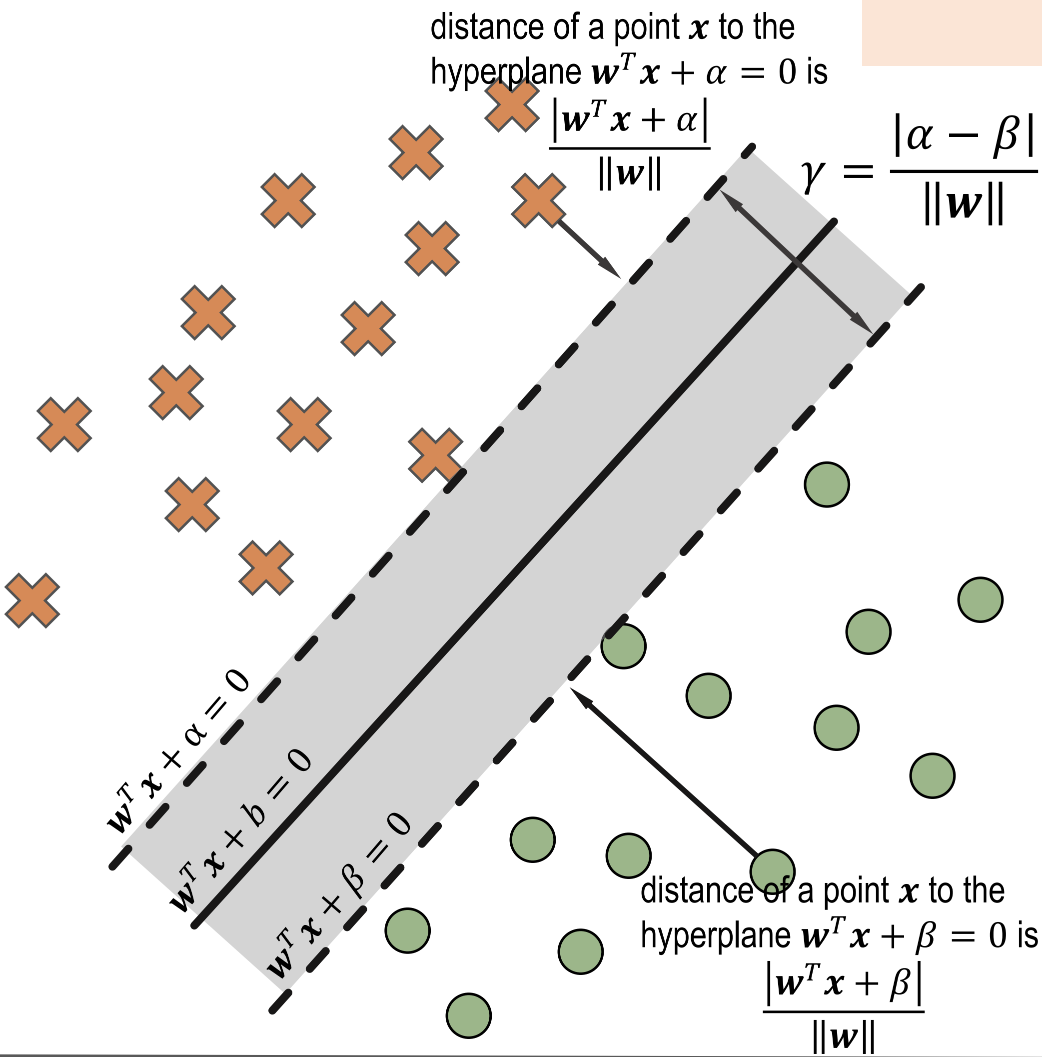
- Prefer hyperplanes that achieve **maximum separation between the two data sets**
- the separation between the two data sets achieved by a classifier is called the **margin of the classifier**
- **Bias**: select a classifier with the **largest margin**

*Linearly separability is a simplifying assumption we make in order to derive a maximum-margin model; it **assumes that there is no noise** in the data set, and hence, the resulting model does not require a loss function.*
This is not a realistic assumption for real-world data sets.

Maximizing the Margin of a Classifier

Problem: Find a **linear classifier** $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ with the largest margin such that

$$\mathbf{sign}\big(f(\boldsymbol{x})\big) = +1, \text{ when positive example}$$
$$\mathbf{sign}\big(f(\boldsymbol{x})\big) = -1, \text{ when negative example}$$



Let the margin be defined by two (parallel) hyperplanes $\boldsymbol{w}^T\boldsymbol{x} + \alpha = 0$ and $\boldsymbol{w}^T\boldsymbol{x} + \beta = 0$.

The **margin** (γ) of the classifier is the **distance** between the two hyperplanes that form the **boundaries of the separation**

$$\gamma = \frac{|\alpha - \beta|}{||\boldsymbol{w}||}$$

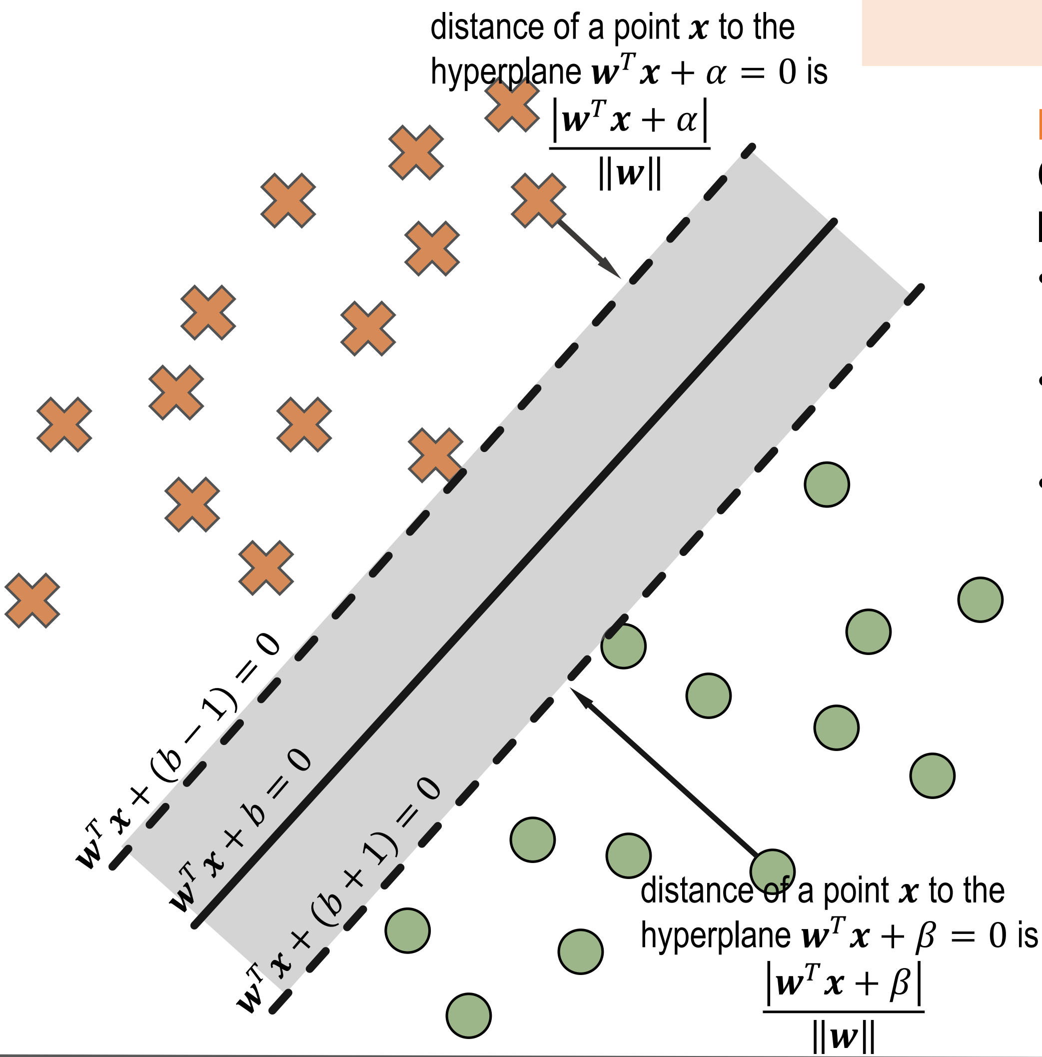
Without loss of generality, we can set $\alpha = b - 1$ and $\beta = b + 1$ (why?) and the margin is

$$\gamma = \frac{2}{||\boldsymbol{w}||}$$

Maximizing the Margin of a Classifier

Problem: Find a **linear classifier** $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ with the largest margin such that

$$\mathbf{sign}\big(f(\boldsymbol{x})\big) = +1, \text{ when positive example}$$
$$\mathbf{sign}\big(f(\boldsymbol{x})\big) = -1, \text{ when negative example}$$



Problem Formulation

Given a linearly-separable data set $(x_i, y_i)_{i=1}^n$, learn a linear classifier $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ such that

- all the training examples with $y_i = +1$ lie **above the margin**, that is $\boldsymbol{w}^T\boldsymbol{x} + b \geq 1$
- all the training examples with $y_i = -1$ lie **below the margin** $\boldsymbol{w}^T\boldsymbol{x} + b \leq -1$
- the **margin** is maximized

$$\gamma = \frac{2}{\|\boldsymbol{w}\|}$$

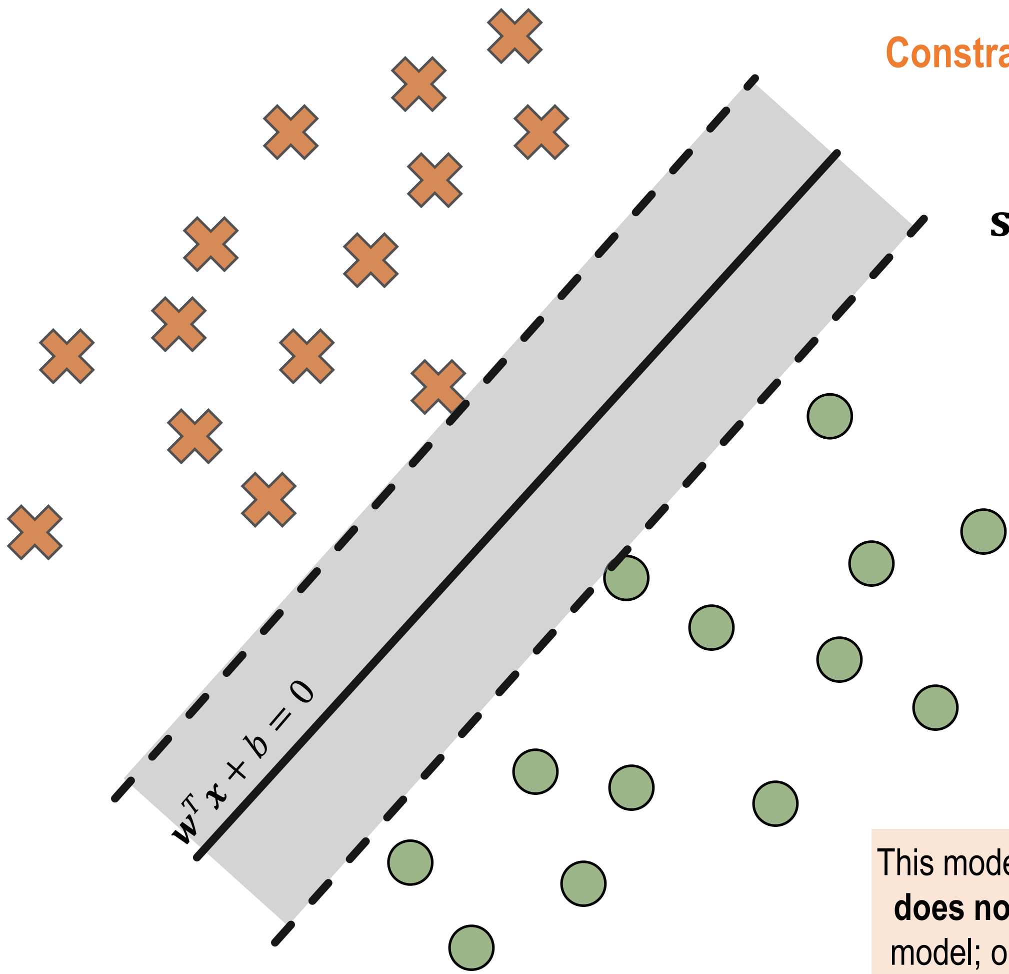
Note that $\displaystyle \max_{\boldsymbol{w}} \frac{2}{\|\boldsymbol{w}\|}$

$$\equiv \min_{\boldsymbol{w}} \frac{\|\boldsymbol{w}\|}{2}$$
$$\equiv \min_{\boldsymbol{w}} \frac{\|\boldsymbol{w}\|^2}{2}$$
$$\equiv \min_{\boldsymbol{w}} \frac{\boldsymbol{w}^T\boldsymbol{w}}{2}$$

Hard-Margin Support Vector Machine

Problem: Find a **linear classifier** $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ with the largest margin such that

$$\mathbf{sign}\big(f(\boldsymbol{x})\big) = +1, \text{ when positive example}$$
$$\mathbf{sign}\big(f(\boldsymbol{x})\big) = -1, \text{ when negative example}$$



Constrained optimization problem

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$
$$\textbf{subject to} \quad y_i \cdot \left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \geq 1, i = 1, \dots, n$$

- Convex, quadratic **minimization problem** called the **primal problem.** Guaranteed to have a **global minimum**
- The problem is no longer unconstrained; need additional optimization tools to ensure **feasibility of solutions** (that solutions satisfy the constraints)
- Further properties of the formulation can be studied by deriving the **Lagrangian** and the **dual problem**

This model is called the hard-margin SVM as it is rigid and **does not allow flexibility for misclassifications** by the model; only feasible when data set is **linearly separable**.

Hard-Margin SVM: Primal Problem

Primal problem for hard-margin SVM

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\textbf{subject to} \quad y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, i = 1, \dots, n$$

for each constraint, which corresponds to **each training example** ($i = 1, \dots, n$), we introduce **new variables** called **dual variables** or **Lagrange multipliers**, $\alpha_i \geq 0$ *(the Lagrange multipliers will give us a mechanism to ensure feasibility, that is, ensure that the optimal solutions ($\boldsymbol{w}$ and b) indeed achieve linear separation of the two classes)*

Lagrangian function for hard-margin SVM

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \cdot [y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1]$$

the **Lagrangian function** is a function of the primal variables ($\boldsymbol{w}$ and b) **and** the dual variables ($\alpha_i \geq 0$); *(the Lagrangian function converts a constrained optimization problem into an unconstrained optimization problem)*

If we can find a minimization of the Lagrangian function with $\sum_{i=1}^{n} \alpha_i \cdot [y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1] = 0$ the resulting solution will **also** be the solution to our original constrained problem.

for **each training example** ($i = 1, \dots, n$), the following must hold for the optimal solution:

$y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 \geq 0$ (primal feasibility)

$\alpha_i \geq 0$ (dual feasibility)

$\alpha_i \cdot [y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1] = 0$ (complementarity)

Hard-Margin SVM: First-Order Conditions

Lagrangian function of a support vector machine

$$L(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\mathbf{w}'\mathbf{w} - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}'\mathbf{x}_i - b) - 1 \right]$$

Differentiate the Lagrangian with respect to the primal variables ($\boldsymbol{w}$ and b)

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b, \alpha_i) = 0 : \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\nabla_b L(\mathbf{w}, b, \alpha_i) = 0 : \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

*the classifier is **a linear combination of training examples** ($i = 1, \ldots, n$)*

These are the **first-order optimality conditions**. We can now **eliminate the primal variables** by substituting the optimality conditions into the Lagrangian.

Hard-Margin SVM: Dual Problem

support vector machine **dual problem**

$$\max \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \quad \forall i = 1 \ldots n$$

*the dual problem depends **only** on the **dual variables** (α_i) and the **inner products** ($\boldsymbol{x}_i^T \boldsymbol{x}_j$) between **each** pair of training examples*

Why bother with the dual?

- both primal and dual problems are **convex (quadratic) optimization problems**. ***No duality gap*** *(that is, the primal solution and dual solution will be exactly the same)*
- dual has **fewer constraints**. ***Easier to solve***
- dual solution is **sparse**. ***Easier to represent***

support vector machine **primal problem**

$$\min \quad \frac{1}{2}||\mathbf{w}||_2^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad \forall i = 1 \ldots n$$

Hard-Margin SVM: Support Vectors

Recall that for **each training example** $(i = 1, \dots, n)$, the following must hold for the final classifier (optimal solution)

$$y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 \geq 0 \quad \text{(primal feasibility)}$$

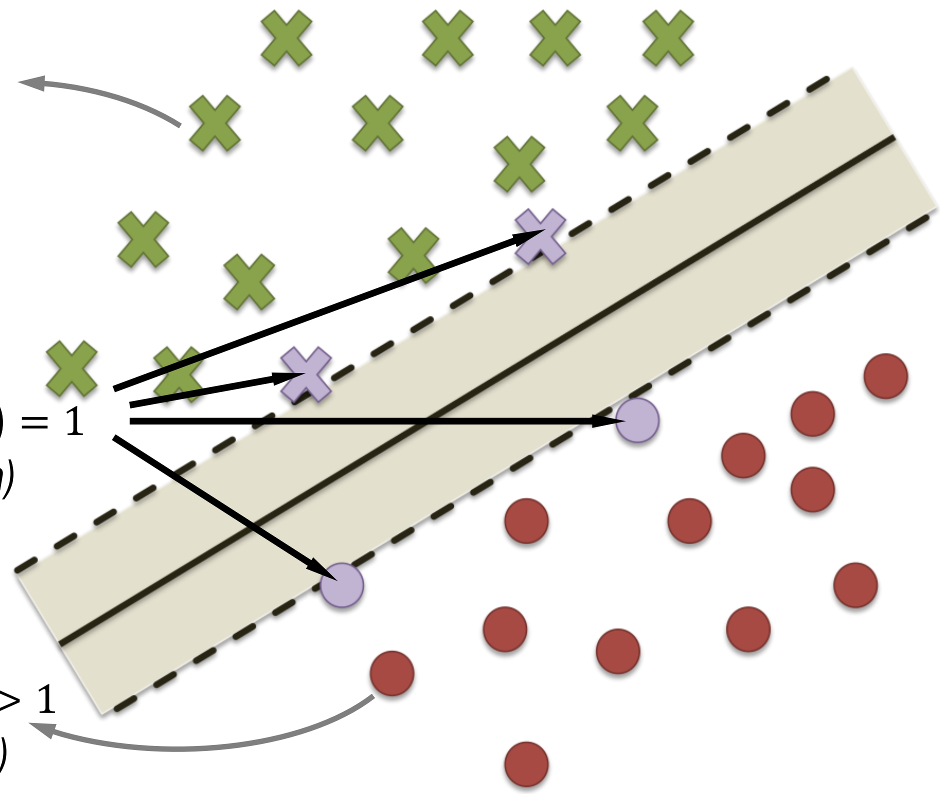
$$\alpha_i \geq 0 \quad \text{(dual feasibility)}$$

$$\alpha_i \cdot [y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1] = 0 \quad \text{(complementarity)}$$

Case 3: $\alpha_i = 0$ and $y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b) = 1$
(degenerate case; not shown in figure)

Case 1a: $\alpha_i = 0$ and $y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b) > 1$
*(training example **is not** on the margin)*

Case 2: $\alpha_i > 0$ and $y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b) = 1$
*(training example **is** on the margin)*



Case 1b: $\alpha_i = 0$ and $y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b) > 1$
*(training example **is not** on the margin)*

Hard-Margin SVM: Support Vectors

Recall that for **each training example** $(i = 1, \ldots, n)$, the following must hold

$$y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 \geq 0 \quad \text{(primal feasibility)}$$

$$\alpha_i \geq 0 \quad \text{(dual feasibility)}$$

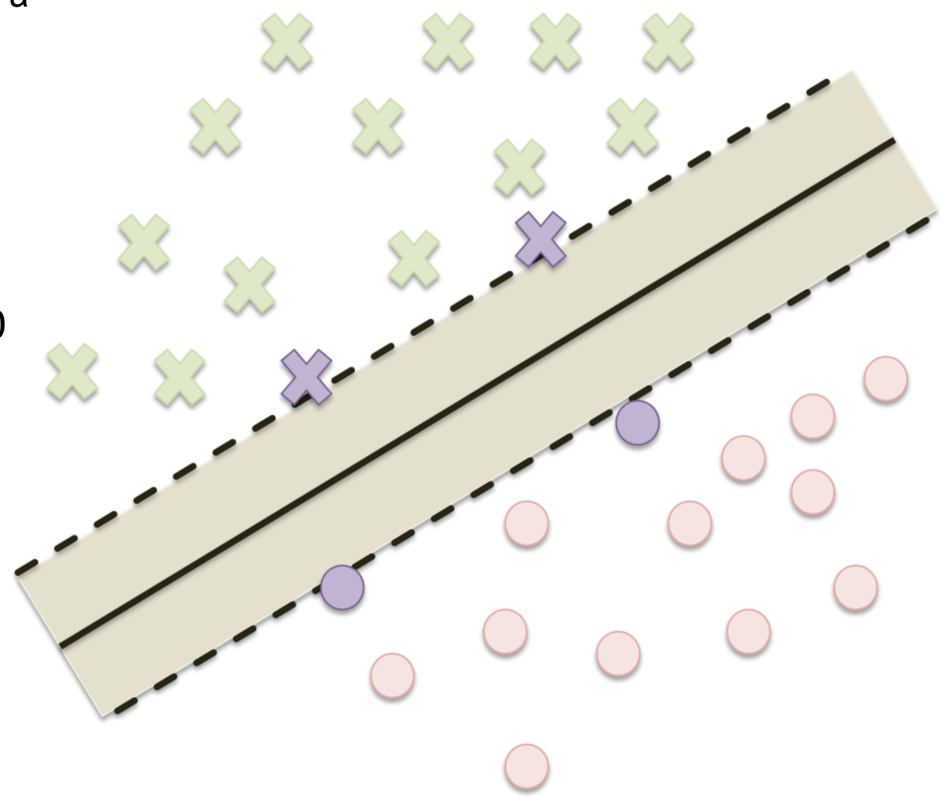
$$\alpha_i \cdot [y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1] = 0 \quad \text{(complementarity)}$$

*the training examples with $\alpha_i > 0$ (non-zero) are called the **support vectors** because they support the classifier. All other training examples have $\alpha_i = 0$, and this makes the solution **sparse**.*

The first order condition that showed that the classifier was a linear combination of the training data:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Only a small number of training examples will have $\alpha_i > 0$ and a large number of training examples will have $\alpha_i = 0$.

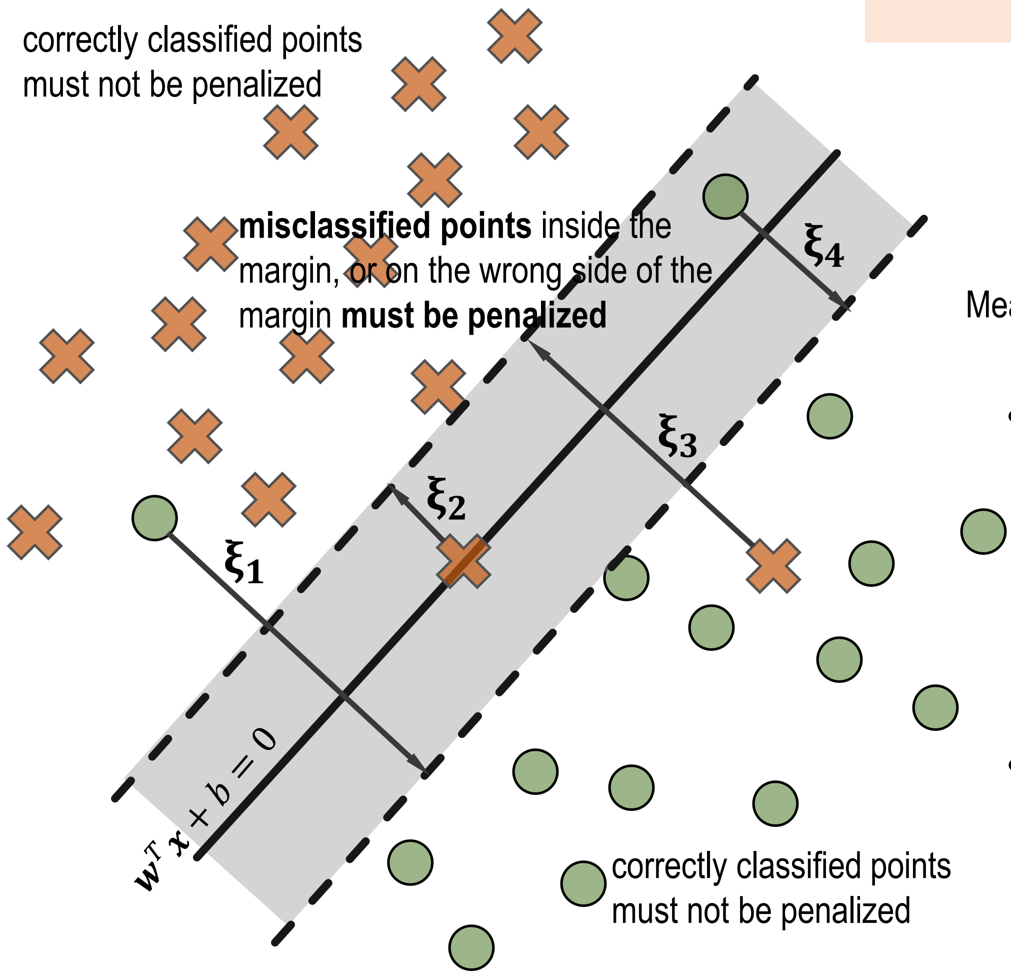


The optimal classifier is a **sparse linear combination of the training examples**, that is, the classifier depends **only on the support vectors**. *This means that if we removed all other training examples except the support vectors, the solution would remain unchanged.*

Soft-Margin SVM: Loss Function

Problem: Find a **linear classifier** $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ with the largest margin such that
$\mathbf{sign}(f(\boldsymbol{x})) = +1$, when positive example
$\mathbf{sign}(f(\boldsymbol{x})) = -1$, when negative example
and **misclassifications are minimized**.

So far, assumed that the data is **linearly separable**, which is not valid in **real-world applications**.



Measure the **misclassification error** for each training example

$$\xi_i = \begin{cases} 0, & y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \\ y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i + b) & y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i + b) < 1 \end{cases}$$

*Penalize each **misclassification by the size of the violation**, using the **hinge loss** (contrast with the loss function of the Perceptron)*

$$\xi_i = L(f(x_i), y_i) = \max\{0, 1 - y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i + b)\}$$

Soft-Margin SVM: Formulation

Problem: Find a **linear classifier** $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ <u>with the largest margin</u> such that

$\mathbf{sign}\big(f(\boldsymbol{x})\big) = +1$, when positive example

$\mathbf{sign}\big(f(\boldsymbol{x})\big) = -1$, when negative example

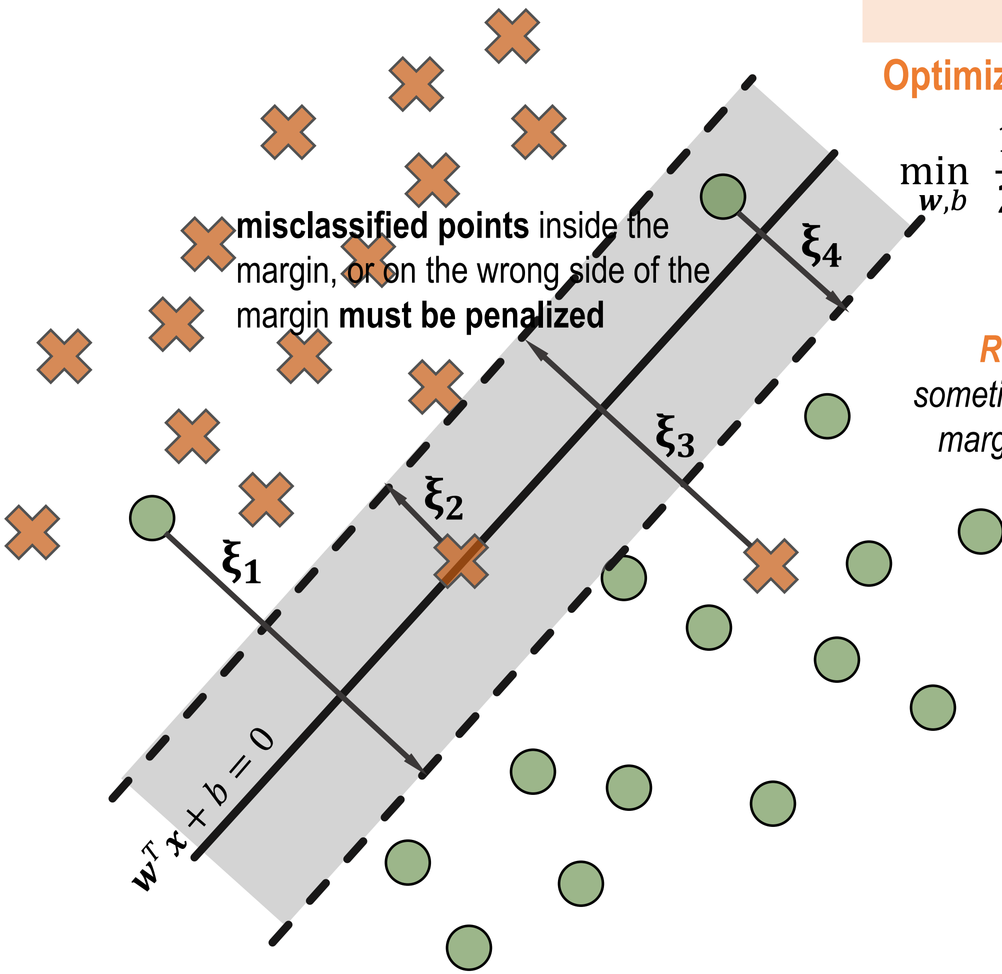
and **misclassifications are minimized**.

Maximize the margin (contrast with the regularization function of Ridge Regression)

Optimization problem

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n} \max\left\{0, 1 - y_i\cdot\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right)\right\}$$

Regularization parameter ($C > 0$), sometimes also denoted λ, trades-off between margin maximization and loss minimization



Penalize each misclassification by the size of the violation, using the hinge loss (contrast with the loss function of the Perceptron)

$$\xi_i = L(f(x_i), y_i) = \max\left\{0, 1 - y_i\cdot\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right)\right\}$$

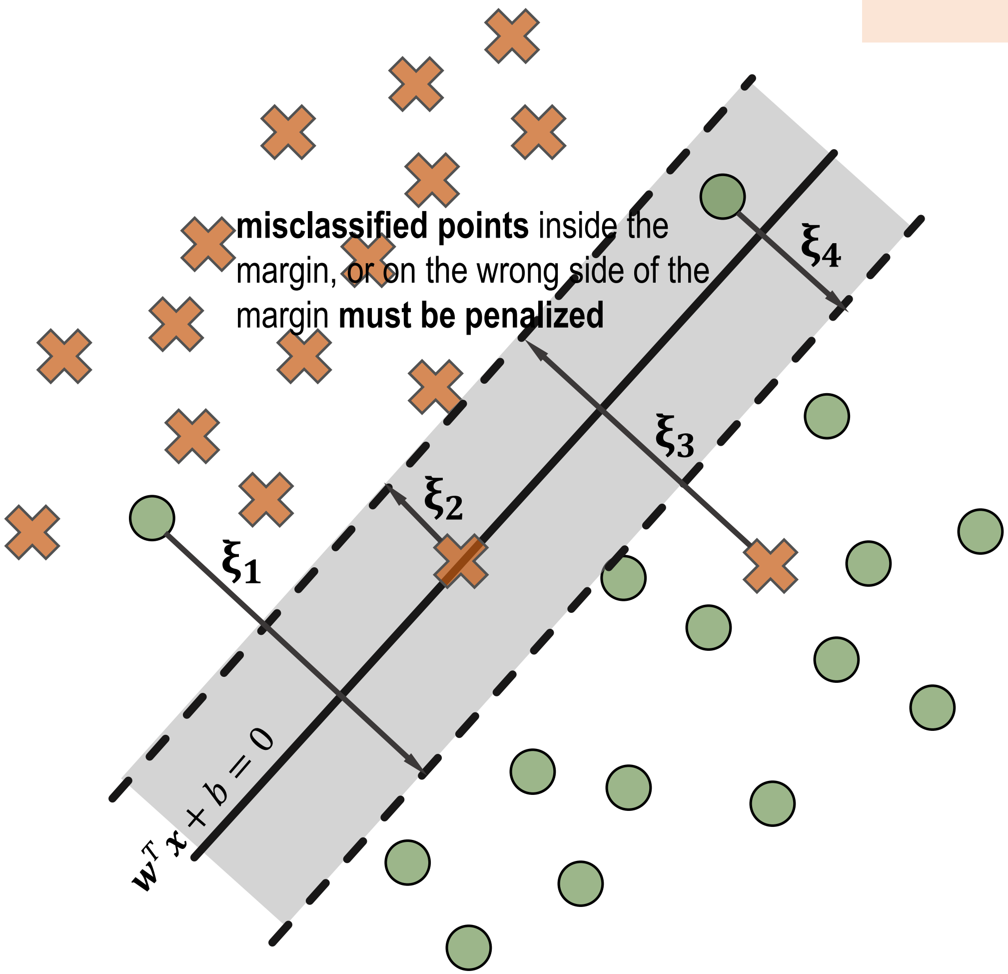
Soft-Margin SVM: Primal Problem

Problem: Find a **linear classifier** $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ with the largest margin such that

$\mathbf{sign}\big(f(\boldsymbol{x})\big) = +1$, when positive example

$\mathbf{sign}\big(f(\boldsymbol{x})\big) = -1$, when negative example

and **misclassifications are minimized**.



soft-margin support vector machine

$$\min \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1\ldots n$$

$$\xi_i \geq 0$$

This model is called the **soft-margin SVM** as it is softens the classification constraints with **slack variables** (ξ_i) and **allows flexibility for misclassifications** by the model

hard-margin support vector machine

$$\min \quad \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad \forall i = 1\ldots n$$

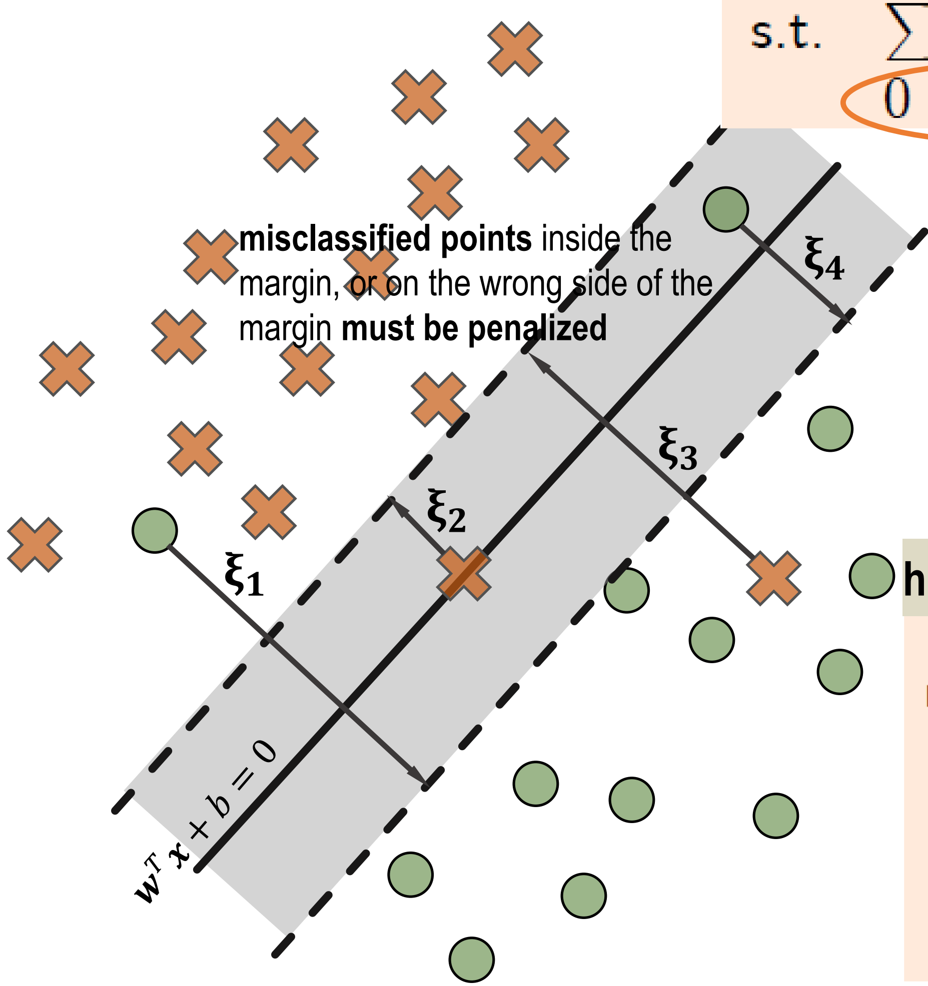
Soft-Margin SVM: Dual Problem

soft-margin svm dual

$$\max \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$0 \le \alpha_i \le C, \quad \forall i = 1 \ldots n$$



the only difference between the soft-margin and hard-margin SVM dual problems is that the Lagrange multipliers (α_i training example weights) are upper-bounded by the **regularization parameter** ($0 \le \alpha_i \le C$)

hard-margin svm dual

$$\max \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0,$$

$$\alpha_i \ge 0, \quad \forall i = 1 \ldots n$$

$$0 \le \alpha_i \le \infty$$

Soft-Margin SVM: Dual Problem

soft-margin svm dual

$$\begin{aligned} \max \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & 0 \le \alpha_i \le C, \quad \forall i = 1\ldots n \end{aligned}$$

the regularization constant **is set by the user**;
this parameter trades off between the regularization term (bias) and the loss term (variance)

the dual solution depends **only on the inner products** of the training data;
this is an important property that allows us to extend linear SVMs to learn nonlinear classification functions without explicit transformation

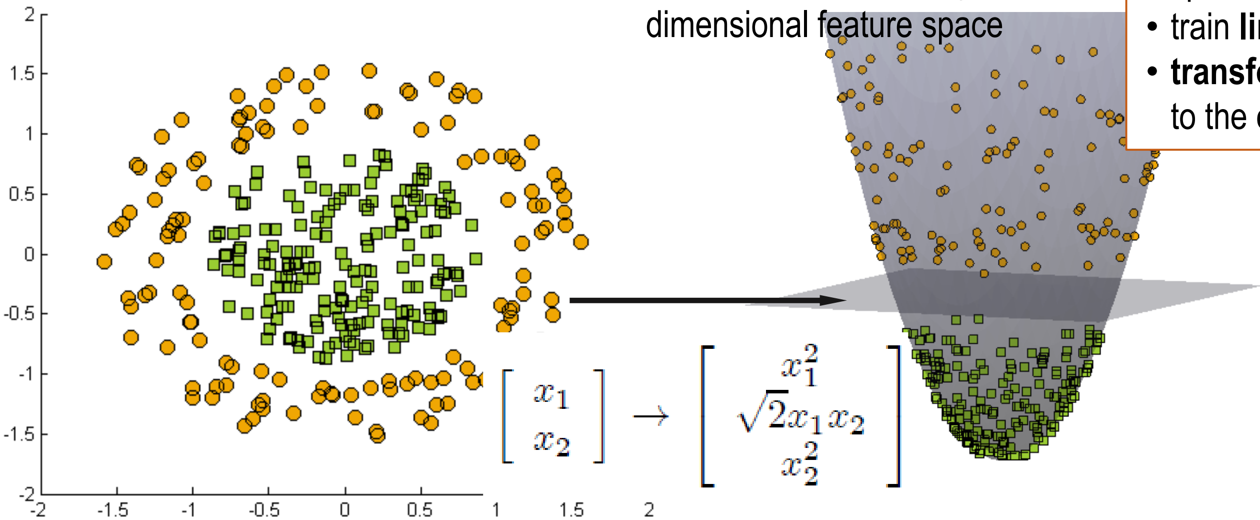
soft-margin support vector machine

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}'\mathbf{x}_i - b) \ge 1 - \xi_i \quad \forall i = 1\ldots n \\ & \xi_i \ge 0 \end{aligned}$$

Nonlinear SVM Classifiers

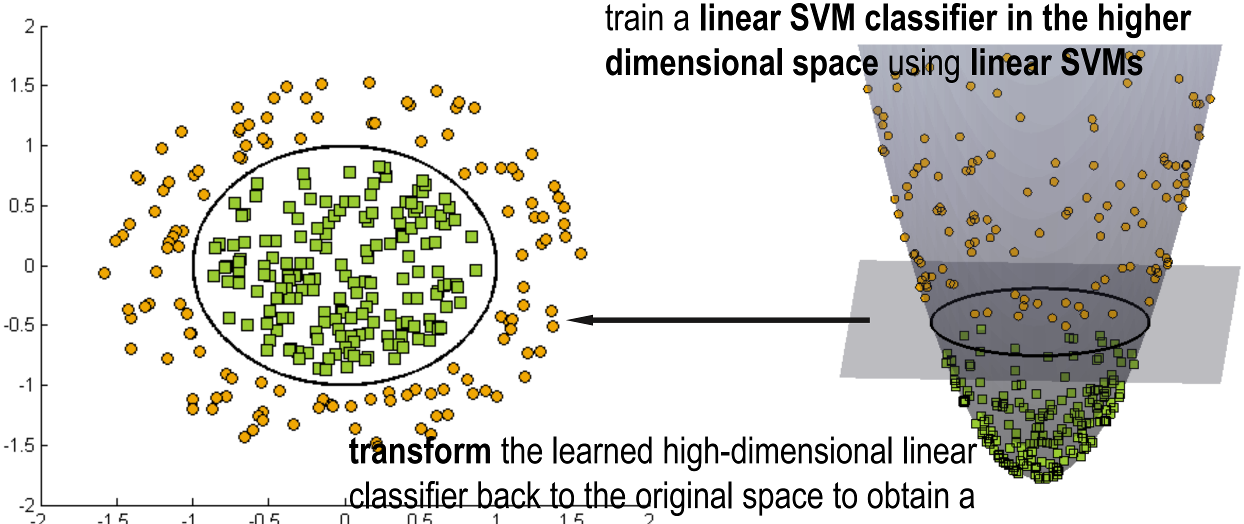
Solution Approach 1 (Explicit Transformation)
- **transform data** to a higher dimensional feature space
- train **linear SVM classifier in the high-dim. space**
- **transform high-dimensional linear classifier** back to the original space to obtain a nonlinear classifier

transform data to a higher dimensional feature space



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

train a **linear SVM classifier in the higher dimensional space** using **linear SVMs**



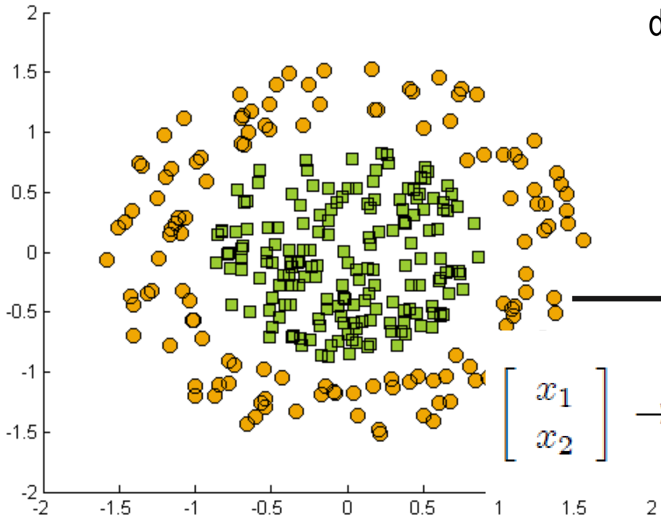
transform the learned high-dimensional linear classifier back to the original space to obtain a nonlinear classifier

Nonlinear SVM Classifiers

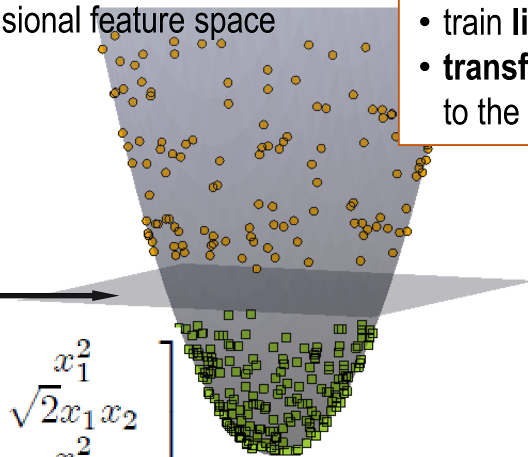
Solution Approach 1 (Explicit Transformation)

- **transform data** to a higher dimensional feature space
- train **linear SVM classifier in the high-dim. space**
- **transform high-dimensional linear classifier** back to the original space to obtain a nonlinear classifier

transform data to a higher dimensional feature space



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$



$$\Phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}$$

Constant Term

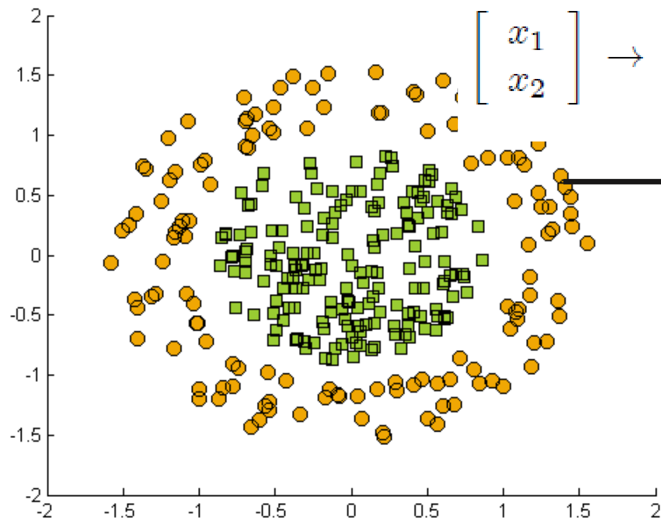
Linear Terms

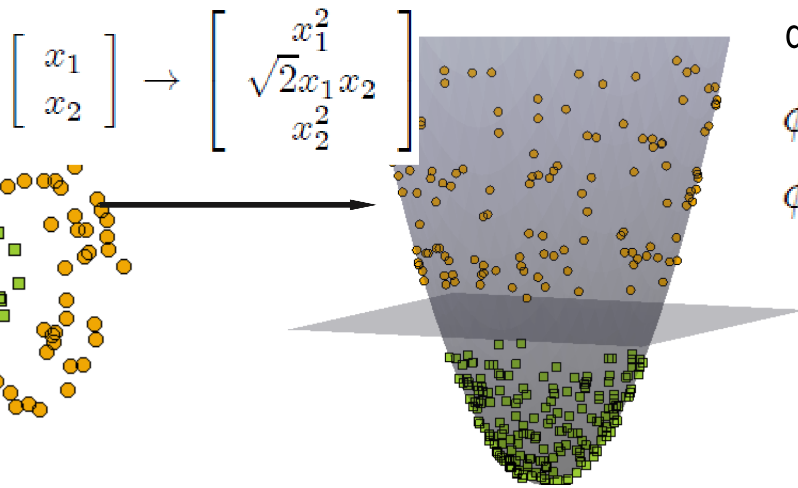
Pure Quadratic Terms

Quadratic Cross-Terms

if we have m training features, the size of the transformation grows very fast; **explicit transformations** can become **very expensive**

The Kernel Trick



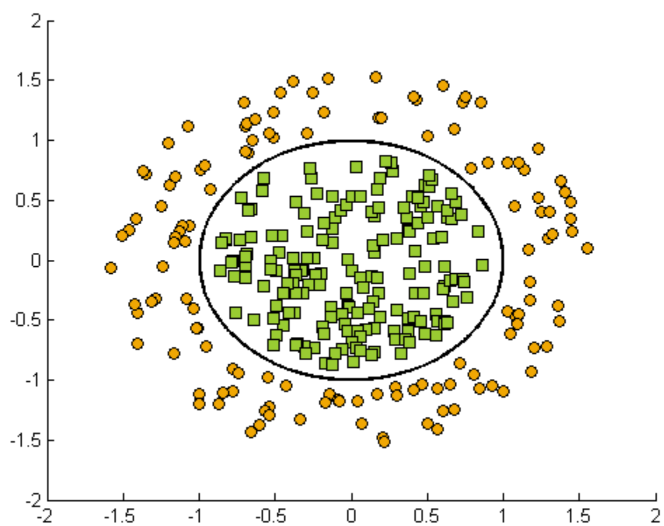


data in higher-dimensional space

$$\phi(\mathbf{x}) = (x_1^2,\ \sqrt{2}x_1x_2,\ x_2^2)$$
$$\phi(\mathbf{z}) = (z_1^2,\ \sqrt{2}z_1z_2,\ z_2^2)$$

when learning a linear SVM, recall that the dual solution depends **only on the inner products** of the training data, so we only need to compute inner products in the higher-dimensional space:

$$\begin{aligned} \phi(\mathbf{x})^T\phi(\mathbf{z}) &= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 \\ &= (x_1z_1 + x_2z_2)^2 = (\mathbf{x}^T\mathbf{z})^2 \\ &= (\mathbf{x}^T\mathbf{z})^2 \end{aligned}$$

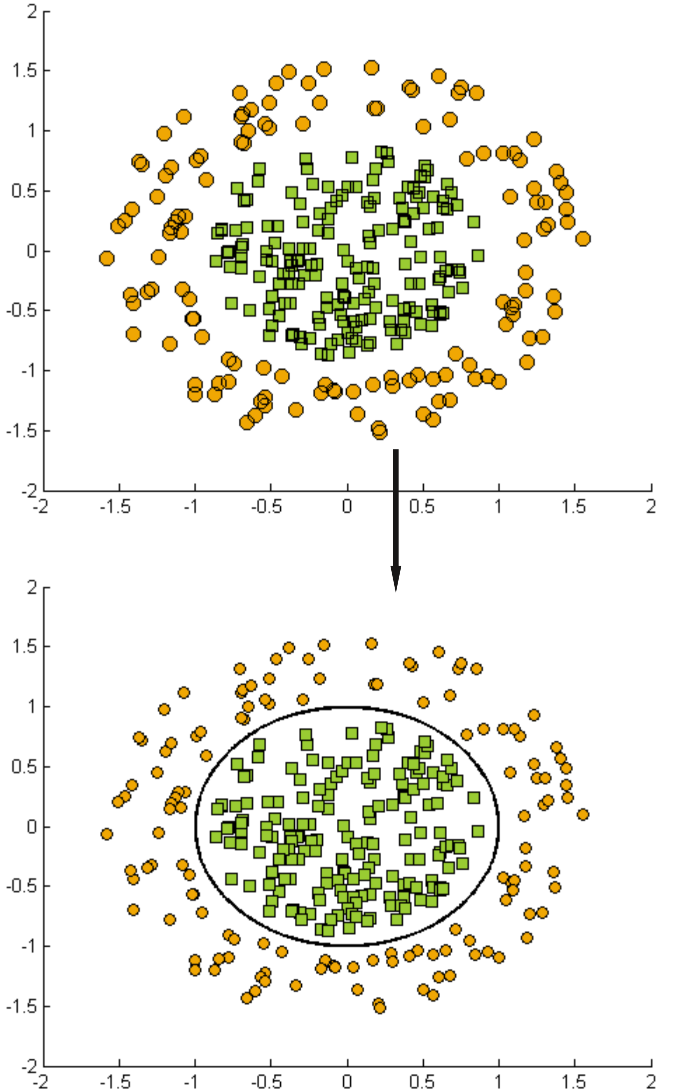


the function $\kappa(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T\boldsymbol{z})^2$ is an example of a **kernel function**

the kernel function relates the inner-products in the original and transformed spaces; with a kernel, ***we can avoid explicit transformation***

Kernel SVMs

n



linear support vector machine

$$\max \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \quad \forall i = 1 \dots n$$

Solution Approach 2 (Kernel SVMs)

- instead of inner-products $\boldsymbol{x}_i^T\boldsymbol{x}_j$, compute the **kernel function** $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T\boldsymbol{x}_j)^2$ between all pairs of training examples
- use the formulation and algorithm of the linear SVM directly, simply replacing the **inner-product matrix** with a **kernel matrix**

kernel support vector machine

$$\max \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \kappa(\mathbf{x}_i,\ \mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \quad \forall i = 1 \dots n$$

Examples of Kernel Functions

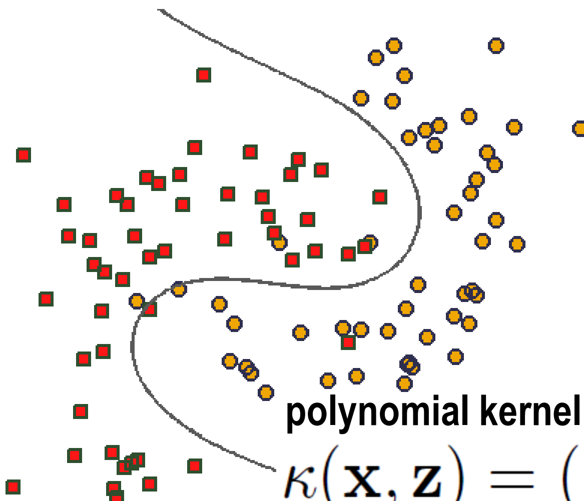
Some popular kernels

- Linear kernel: $\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$
- Polynomial kernel: $\kappa(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + c)^d$, $c, d \geq 0$
- Gaussian kernel: $\kappa(\mathbf{x}, \mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma}}$, $\sigma > 0$
- Sigmoid kernel: $\kappa(\mathbf{x}, \mathbf{z}) = \tanh^{-1} \eta \langle \mathbf{x}, \mathbf{z} \rangle + \theta$

Kernels can also be constructed from other kernels:

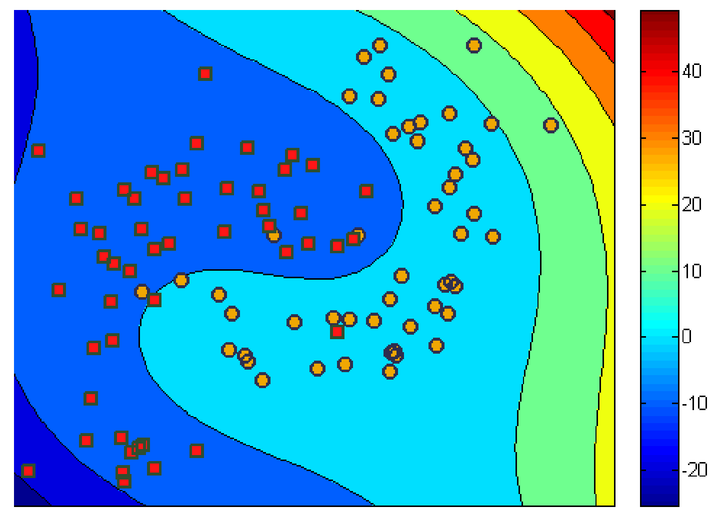
- Conical (not linear) combinations, $\kappa(\mathbf{x}, \mathbf{z}) = a_1 \kappa_1(\mathbf{x}, \mathbf{z}) + a_2 \kappa_2(\mathbf{x}, \mathbf{z})$
- Products of kernels, $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z}) \kappa_2(\mathbf{x}, \mathbf{z})$
- Products of functions, $\kappa(\mathbf{x}, \mathbf{z}) = f_1(\mathbf{x}) f_2(\mathbf{z})$, f_1, f_2 are real valued functions.

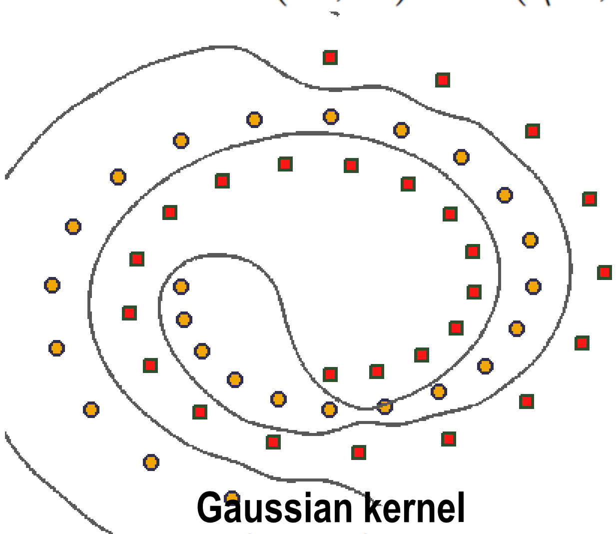
Some Popular Kernels



polynomial kernel

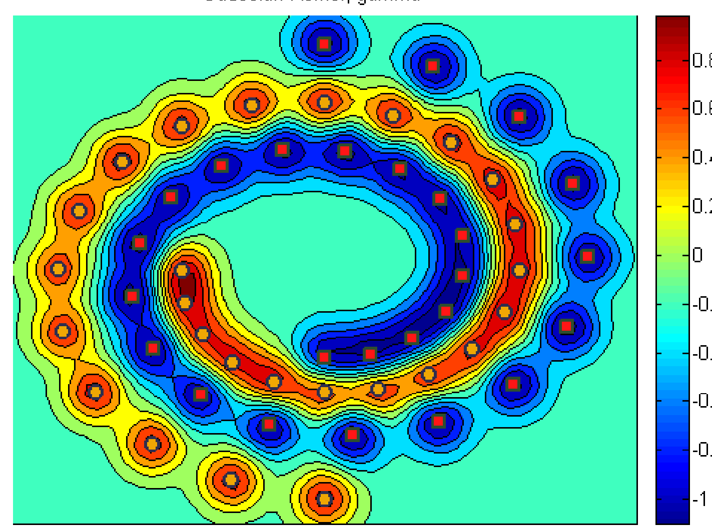
$$\kappa(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + 1)^d$$



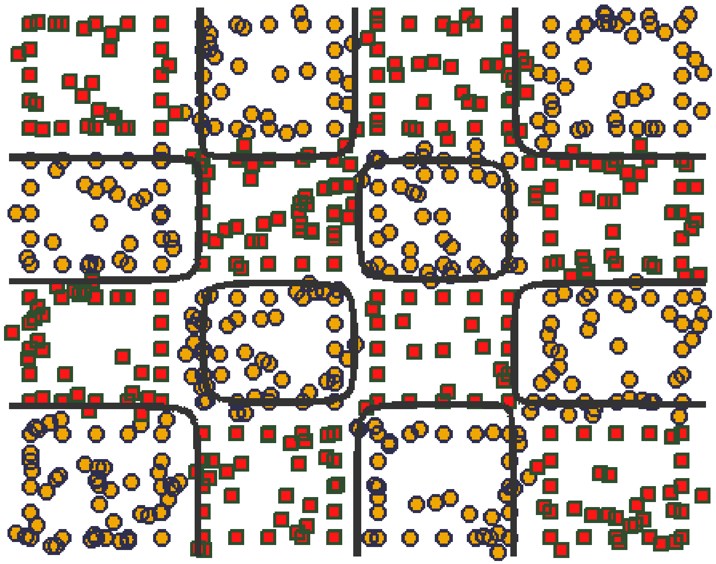


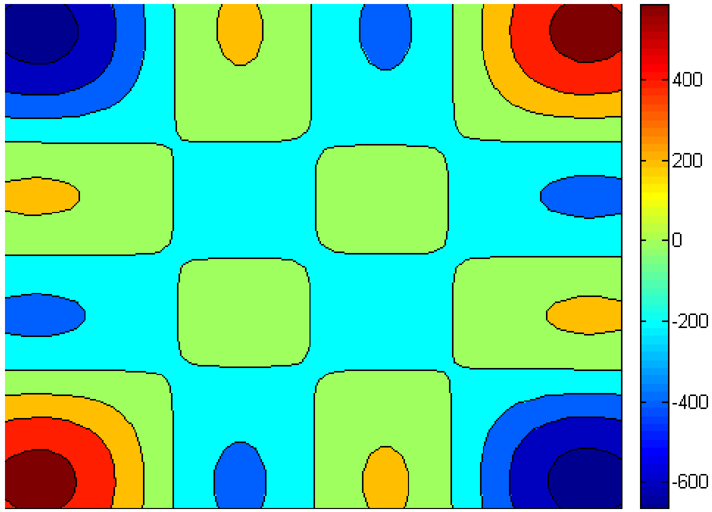
Gaussian kernel

$$\kappa(\mathbf{x}, \mathbf{z}) = \exp - \frac{\|\mathbf{x} - \mathbf{z}\|^2}{\sigma}$$



Overfitting with Kernels





observe that the Gaussian kernel can be written as

$$\exp\left(\frac{-\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2}\right) = \exp\left(\frac{-\|\boldsymbol{x}\|^2 + 2\boldsymbol{x}^T\boldsymbol{z} - \|\boldsymbol{z}\|^2}{2\sigma^2}\right) = \exp(-\|\boldsymbol{x}\|^2)\exp(-\|\boldsymbol{z}\|^2)\exp\left(\frac{\boldsymbol{x}^T\boldsymbol{z}}{\sigma^2}\right)$$

using the Taylor expansion for $\exp(\cdot)$, we have

$$\exp\left(\frac{\boldsymbol{x}^T\boldsymbol{z}}{\sigma^2}\right) = \sum_{k=0}^{\infty} \frac{(\boldsymbol{x}^T\boldsymbol{z})^k}{2\sigma^{2k}\cdot k!}$$

Gaussian kernels can represent **polynomials of every degree**, which means they can overfit, especially when features spaces are larger

margin maximization (regularization) helps learn robust models; **selection of kernel parameter** (σ) is also **critical**

Regularization and Overfitting

soft-margin support vector machine

$$\min \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1\ldots n$$
$$\xi_i \geq 0$$

The **regularization parameter, C**, is chosen **a priori**, and defines the relative **trade-off between norm** (bias/complexity) and **loss** (error/variance)

We want to find classifiers that *minimize* (**regularization** + **C loss**)

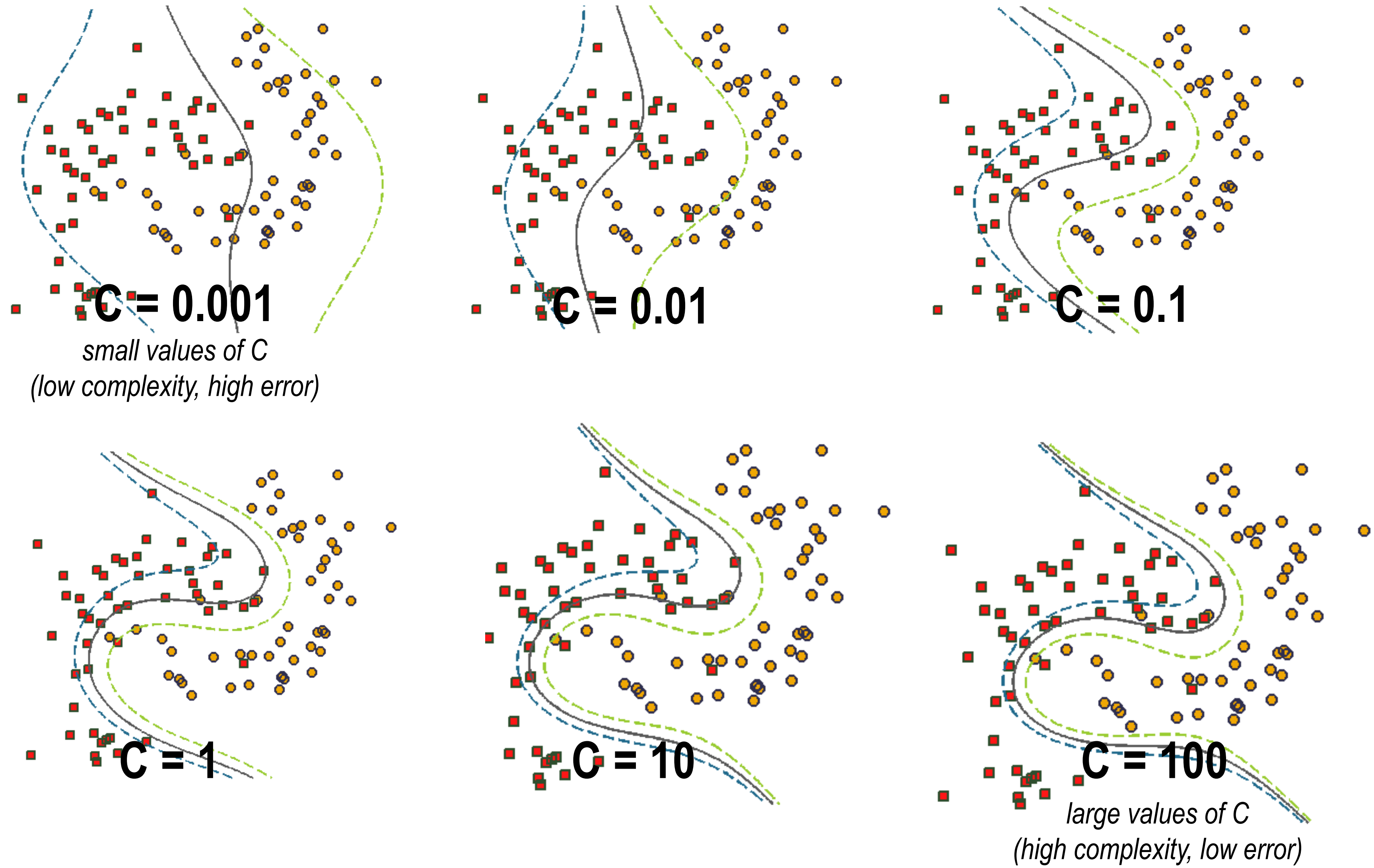
Regularization
- introduces **inductive bias** over solutions
- controls the **complexity** of the solution
- imposes **smoothness restriction** on solutions

As **C increases**, the effect of the **regularization decreases** and the **SVM tends to overfit the data**

soft-margin svm dual

$$\max \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$
$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$
$$0 \leq \alpha_i \leq C, \quad \forall i = 1\ldots n$$

The Effect of C on Classification



SVM Modeling Choices

- Select the kernel function to use (important but often trickiest part of SVM)
  - In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try and usually support by off-the-shelf software
- Select the parameter of the kernel function and the value of ***c***
  - You can use the values suggested by the SVM software

    see www.kernel-machines.org/software.html for a list of available software
  - You can set apart a validation set to determine the values of the parameter

SVM Implementations Over The Years

- **Earliest solution approaches**: Quadratic Programming Solvers (CPLEX, LOQO, Matlab QP, SeDuMi)
- **Decomposition methods**: SVM chunking (Osuna et. al., 1997); SVMlight (Joachims, 1999)
- **Sequential Minimization Optimization** (Platt, 1999); implementation: LIBSVM (Chang et. al., 2000)
- **Interior Point Methods** (Munson and Ferris, 2006),

  Successive Over-relaxation (Mangasarian, 2004)
- **Co-ordinate Descent Algorithms** (Keerthi et. al., 2009), **Bundle Methods** (Teo et. al., 2010)
- **Present:** scikit-learn's Stochastic Gradient Descent can learn linear SVMs; *also has a dedicated SVM package that can handle binary and multi-class classification, regression, one-class classification and kernels*

Support Vector Machines

- Advantages of SVMs
  - polynomial-time exact optimization rather than approximate methods
    - unlike decision trees and neural networks
  - Kernels allow very flexible hypotheses
  - Can be applied to very complex data types, e.g., graphs, sequences
- Disadvantages of SVMs
  - Must choose a good kernel and kernel parameters
  - Very large problems are computationally intractable
    - quadratic in number of examples
    - problems with more than 20k examples are very difficult to solve exactly