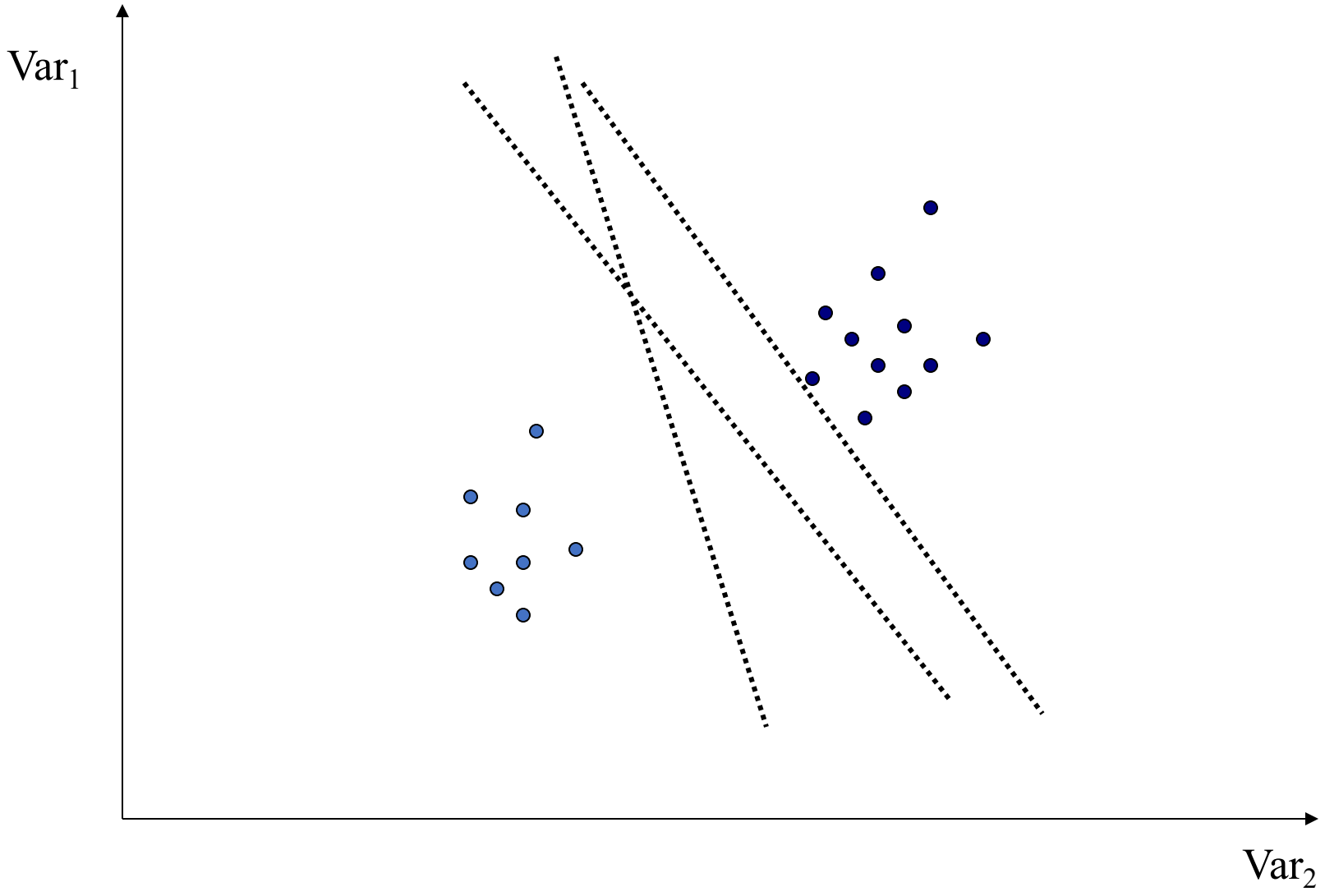
Fall 2022 ECEN 758
Data Mining & Analysis
Support Vector Machines

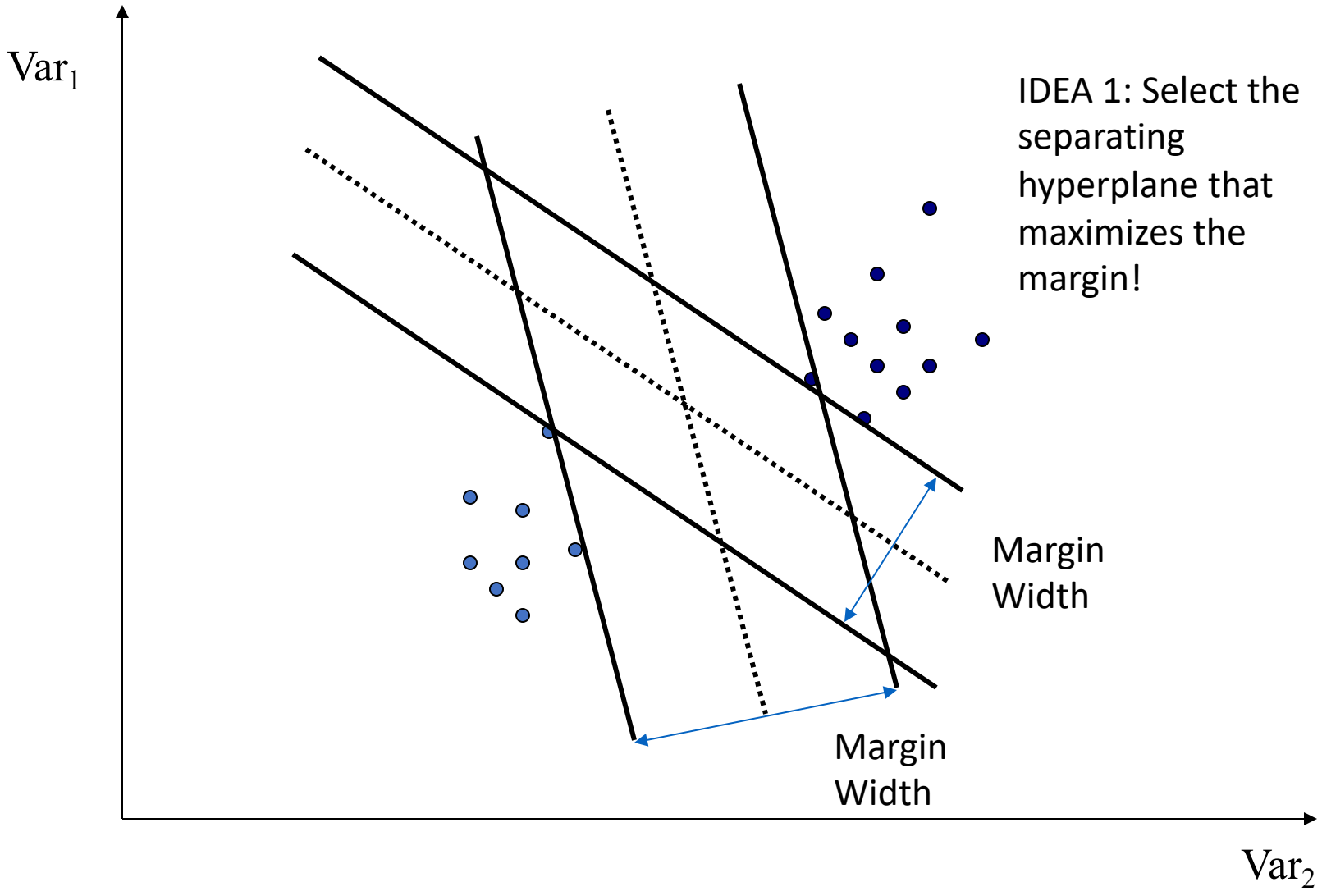
Material adapted from

SVM Linear Separable Case: Quadratic Programs

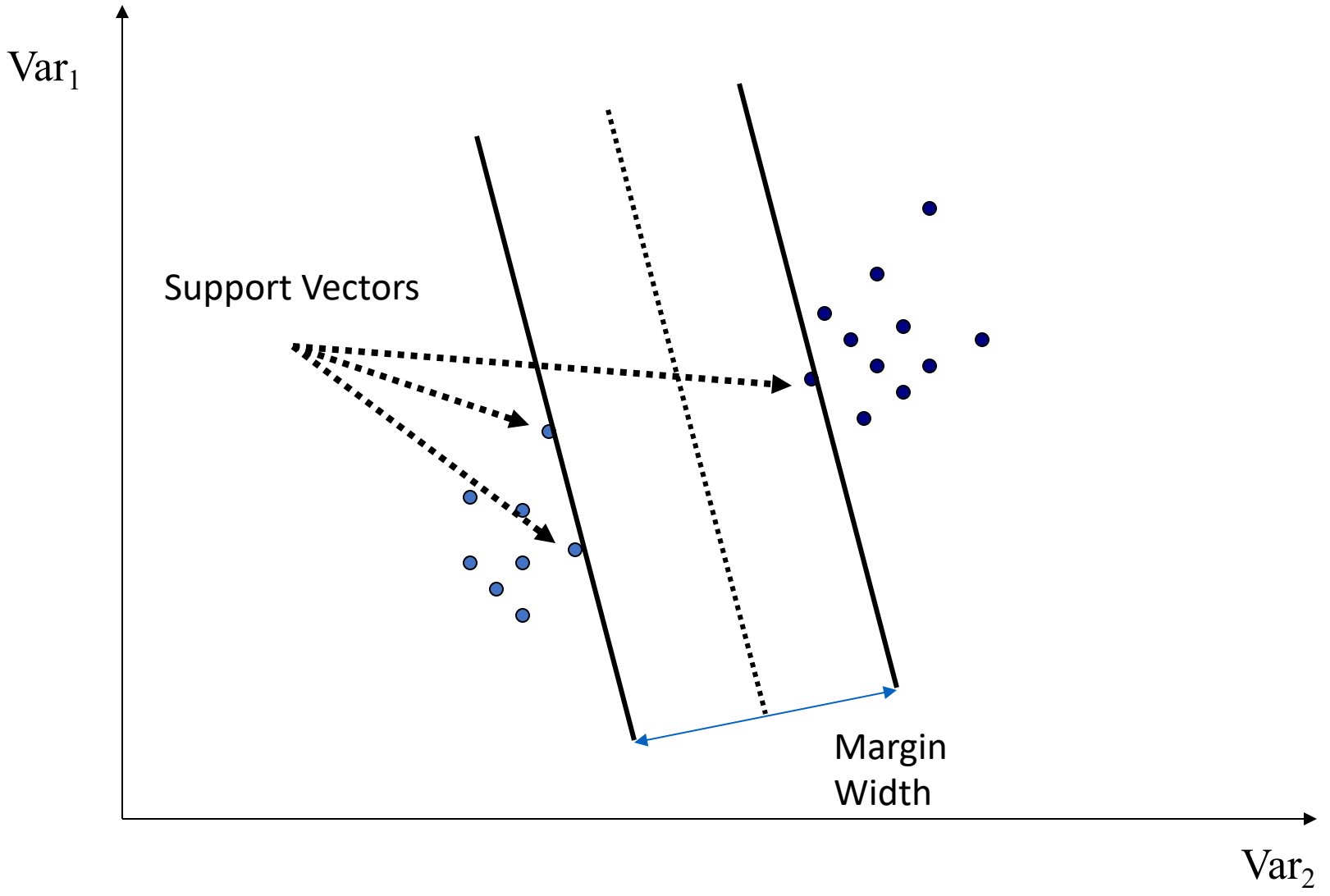
Which Separating Hyperplane to Use?



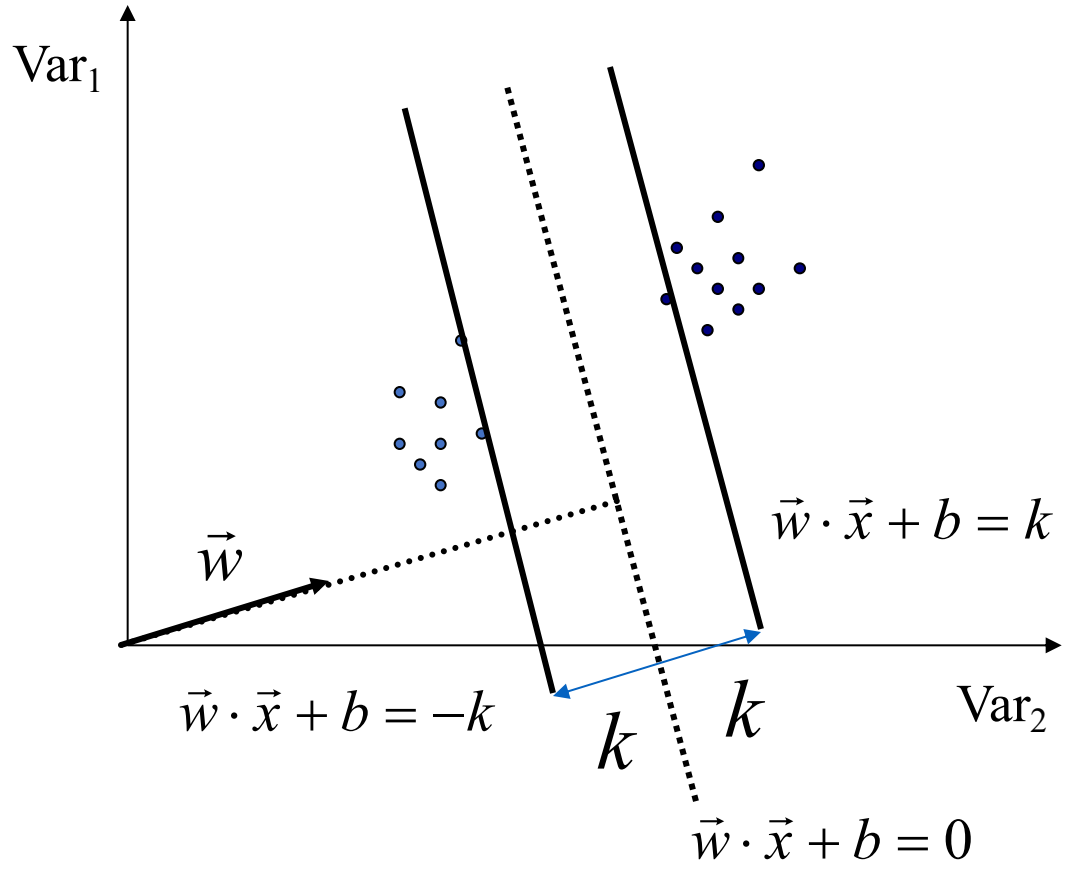
Maximizing the Margin



Support Vectors



Setting Up the Optimization Problem



The width of the margin is:

$$\frac{2|k|}{\|w\|}$$

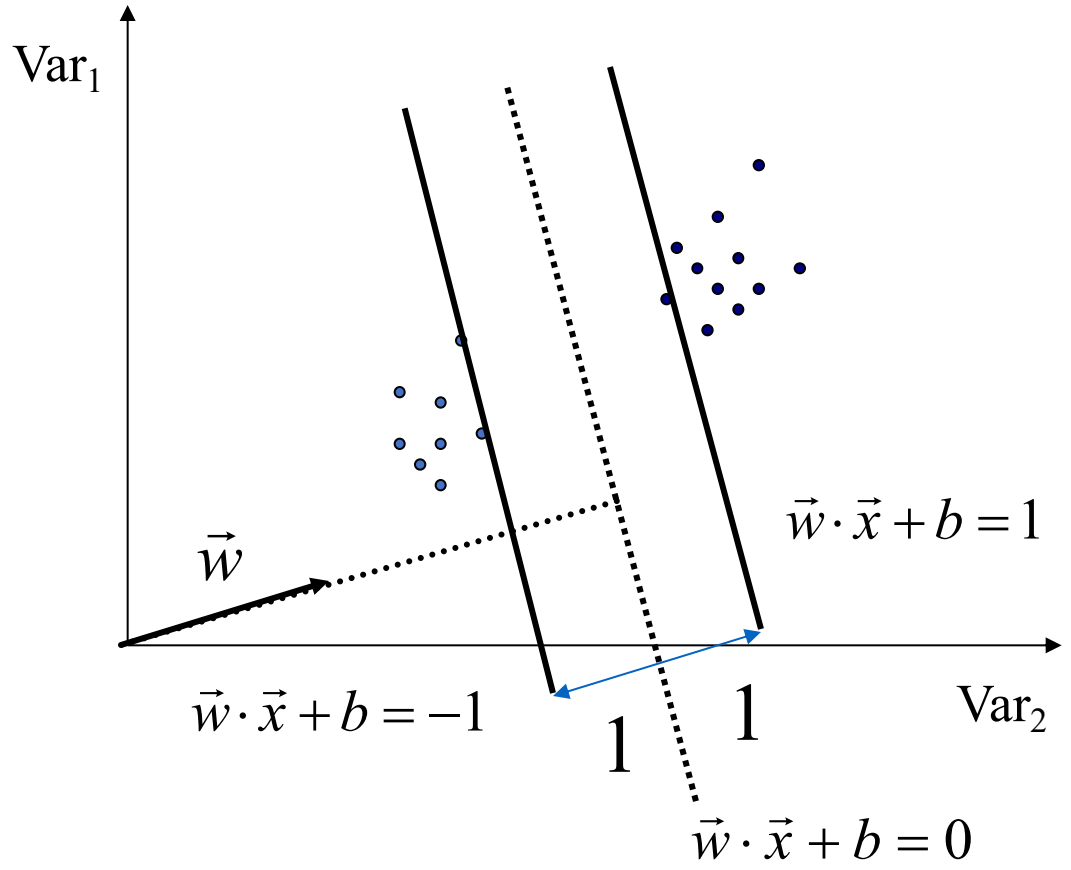
So, the problem is:

$$\max \frac{2|k|}{\|w\|}$$

$$s.t.\ (w \cdot x + b) \geq k,\ \forall x \text{ of class 1}$$

$$(w \cdot x + b) \leq -k,\ \forall x \text{ of class 2}$$

Setting Up the Optimization Problem



There is a scale and unit for data so that *k=1*. Then problem becomes:

$$\max \frac{2}{\|w\|}$$

$$s.t.\ (w\cdot x+b)\geq 1,\ \forall x \text{ of class 1}$$

$$(w\cdot x+b)\leq -1,\ \forall x \text{ of class 2}$$

Setting Up the Optimization Problem

- **If class 1 corresponds to 1 and class 2 corresponds to -1, we can rewrite**

$$(w \cdot x_i + b) \geq 1, \ \forall x_i \ \text{with} \ y_i = 1$$
$$(w \cdot x_i + b) \leq -1, \ \forall x_i \ \text{with} \ y_i = -1$$

- **as**

$$y_i (w \cdot x_i + b) \geq 1, \ \forall x_i$$

- **The problem becomes:**

$$\max \frac{2}{\|w\|} \quad \text{or} \quad \min \frac{1}{2}\|w\|^2$$
$$s.t. \ y_i (w \cdot x_i + b) \geq 1, \ \forall x_i \qquad s.t. \ y_i (w \cdot x_i + b) \geq 1, \ \forall x_i$$

Linear, Hard-Margin SVM Formulation

- **Find *w,b* that solves**

$$\min \frac{1}{2}\|w\|^2$$

$$s.t.\ y_i(w \cdot x_i + b) \geq 1,\ \forall x_i$$

- **Problem is convex so, there is a unique global minimum value (when feasible)**
- **There is also a unique minimizer, i.e. weight and *b* value that provides the minimum**
- **Non-solvable if the data is not linearly separable**
- **Quadratic Programming**
  - Very efficient computationally with modern constraint optimization engines (handles thousands of constraints and training instances).

SVM: Linear and Separable Case

Assume that the points are linearly separable, that is, there exists a separating hyperplane that perfectly classifies each point.

The goal of SVMs is to choose the canonical hyperplane, h^*, that yields the maximum margin among all possible separating hyperplanes

$$h^* = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \right\}$$

We can obtain an equivalent minimization formulation:

$$\textbf{Objective Function:} \quad \min_{\mathbf{w},b} \left\{ \frac{\|\mathbf{w}\|^2}{2} \right\}$$

$$\textbf{Linear Constraints:} \quad y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1, \quad \forall \mathbf{x}_i \in \mathbf{D}$$

SVM: Linear and Separable Case

We turn the constrained SVM optimization into an unconstrained one by introducing a Lagrange multiplier α_i for each constraint. The new objective function, called the *Lagrangian*, then becomes

$$\min \; L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

L should be minimized with respect to $\mathbf{w}$ and b, and it should be maximized with respect to α_i.
Taking the derivative of L with respect to $\mathbf{w}$ and b, and setting those to zero, we obtain

$$\frac{\partial}{\partial \mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \mathbf{0} \quad \text{or} \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial}{\partial b} L = \sum_{i=1}^{n} \alpha_i y_i = 0$$

We can see that $\mathbf{w}$ can be expressed as a linear combination of the data points $\mathbf{x}_i$, with the signed Lagrange multipliers, $\alpha_i y_i$, serving as the coefficients.

Further, the sum of the signed Lagrange multipliers, $\alpha_i y_i$, must be zero.

SVM: Linear and Separable Case

Incorporating $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$ into the Lagrangian we obtain the new *dual Lagrangian* objective function, which is specified purely in terms of the Lagrange multipliers:

$$\textbf{Objective Function:} \quad \max_{\boldsymbol{\alpha}} \; L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\textbf{Linear Constraints:} \quad \alpha_i \geq 0, \; \forall i \in \mathbf{D}, \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$ is the vector comprising the Lagrange multipliers.

L_{dual} is a convex quadratic programming problem (note the $\alpha_i \alpha_j$ terms), which admits a unique optimal solution.

SVM: Linear and Separable Case

Once we have obtained the α_i values for $i = 1, \ldots, n$, we can solve for the weight vector $\mathbf{w}$ and the bias b. Each of the Lagrange multipliers α_i satisfies the KKT conditions at the optimal solution:

$$\alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) = 0$$

which gives rise to two cases:

(1) $\alpha_i = 0$, or

(2) $y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0$, which implies $y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$

This is a very important result because if $\alpha_i > 0$, then $y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$, and thus the point $\mathbf{x}_i$ must be a support vector.

On the other hand, if $y_i(\mathbf{w}^T \mathbf{x}_i + b) > 1$, then $\alpha_i = 0$, that is, if a point is not a support vector, then $\alpha_i = 0$.

Linear and Separable Case: Weight Vector and Bias

Once we know α_i for all points, we can compute the weight vector $\mathbf{w}$ by taking the summation only for the support vectors:

$$\mathbf{w} = \sum_{i, \alpha_i > 0} \alpha_i y_i \mathbf{x}_i$$

Only the support vectors determine $\mathbf{w}$, since $\alpha_i = 0$ for other points.
To compute the bias b, we first compute one solution b_i, per support vector, as follows:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1, \text{ which implies } b_i = \frac{1}{y_i} - \mathbf{w}^T \mathbf{x}_i = y_i - \mathbf{w}^T \mathbf{x}_i$$

The bias b is taken as the average value:

$$b = \text{avg}_{\alpha_i > 0}\{b_i\}$$

SVM Classifier

Given the optimal hyperplane function $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, for any new point $\mathbf{z}$, we predict its class as

$$\hat{y} = \text{sign}(h(\mathbf{z})) = \text{sign}(\mathbf{w}^T\mathbf{z} + b)$$

where the sign$(\cdot)$ function returns $+1$ if its argument is positive, and -1 if its argument is negative.

Soft Margin SVM: Linear and Nonseparable Case

The assumption that the dataset be perfectly linearly separable is unrealistic. SVMs can handle non-separable points by introducing *slack variables* ξ_i as follows:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

where $\xi_i \geq 0$ is the slack variable for point $\mathbf{x}_i$, which indicates how much the point violates the separability condition, that is, the point may no longer be at least $1/\|\mathbf{w}\|$ away from the hyperplane.

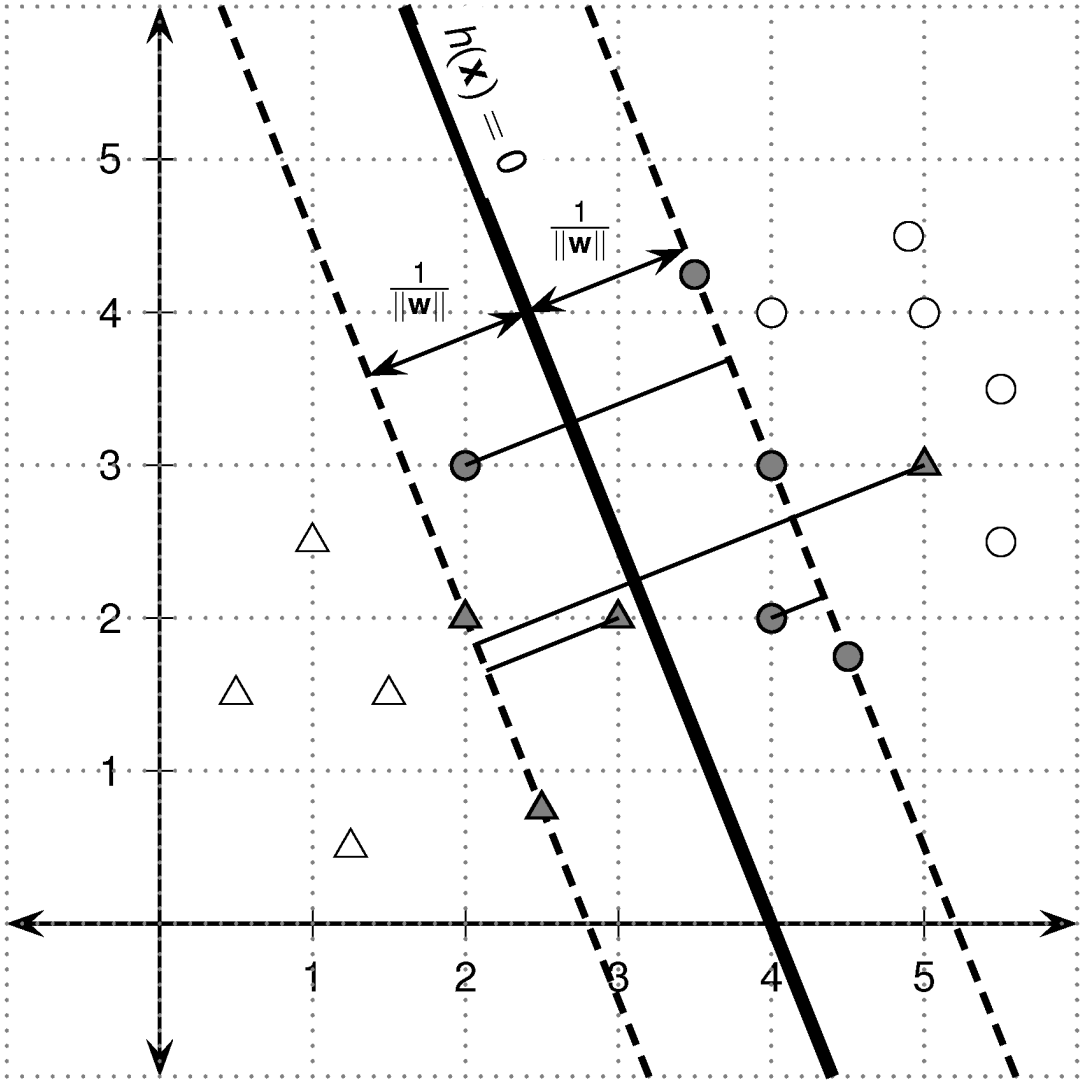
The slack values indicate three types of points. If $\xi_i = 0$, then the corresponding point $\mathbf{x}_i$ is at least $\frac{1}{\|\mathbf{w}\|}$ away from the hyperplane.

If $0 < \xi_i < 1$, then the point is within the margin and still correctly classified, that is, it is on the correct side of the hyperplane.

However, if $\xi_i \geq 1$ then the point is misclassified and appears on the wrong side of the hyperplane.

Soft Margin Hyperplane

Shaded points are the support vectors



SVM: Soft Margin or Linearly Non-separable Case

In the nonseparable case, also called the *soft margin* the SVM objective function is

$$\textbf{Objective Function:} \quad \min_{\mathbf{w},b,\xi_i} \left\{ \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{n} (\xi_i)^k \right\}$$

$$\textbf{Linear Constraints:} \quad y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad \forall \mathbf{x}_i \in \mathbf{D}$$

$$\xi_i \geq 0 \quad \forall \mathbf{x}_i \in \mathbf{D}$$

where C and k are constants that incorporate the cost of misclassification.

The term $\sum_{i=1}^{n} (\xi_i)^k$ gives the *loss*, that is, an estimate of the deviation from the separable case.

The scalar C is a *regularization constant* that controls the trade-off between maximizing the margin or minimizing the loss. For example, if $C \to 0$, then the loss component essentially disappears, and the objective defaults to maximizing the margin. On the other hand, if $C \to \infty$, then the margin ceases to have much effect, and the objective function tries to minimize the loss.

SVM: Soft Margin Loss Function

The constant k governs the form of the loss. When $k = 1$, called *hinge loss*, the goal is to minimize the sum of the slack variables, whereas when $k = 2$, called *quadratic loss*, the goal is to minimize the sum of the squared slack variables.

Hinge Loss: Assuming $k = 1$, the SVM dual Lagrangian is given as

$$\max_{\boldsymbol{\alpha}} \quad L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

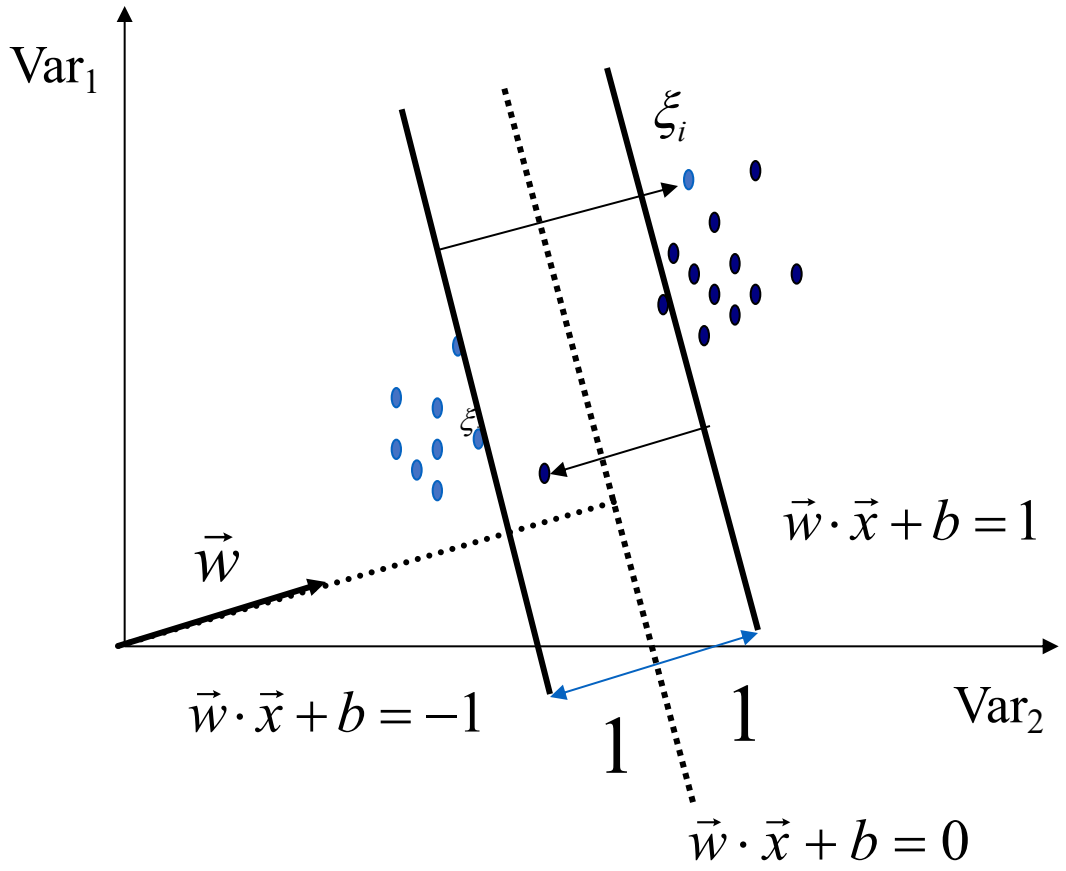
The only difference from the separable case is that $0 \leq \alpha_i \leq C$.

Quadratic Loss: Assuming $k = 2$, the dual objective is:

$$\max_{\boldsymbol{\alpha}} \quad L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i^T \mathbf{x}_j + \frac{1}{2C} \delta_{ij} \right)$$

where δ is the *Kronecker delta* function, defined as $\delta_{ij} = 1$ if and only if $i = j$.

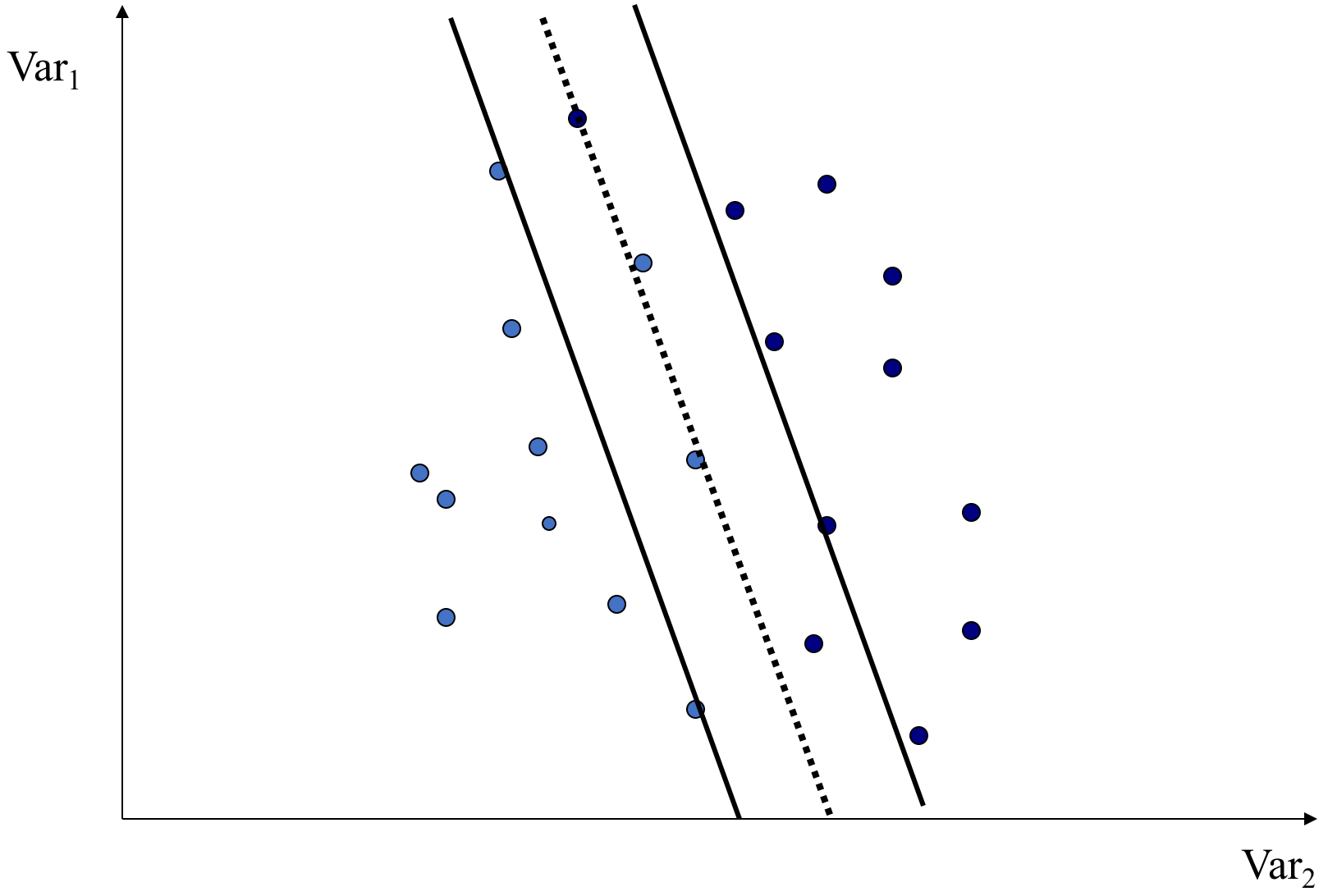
Non-Linearly Separable Data



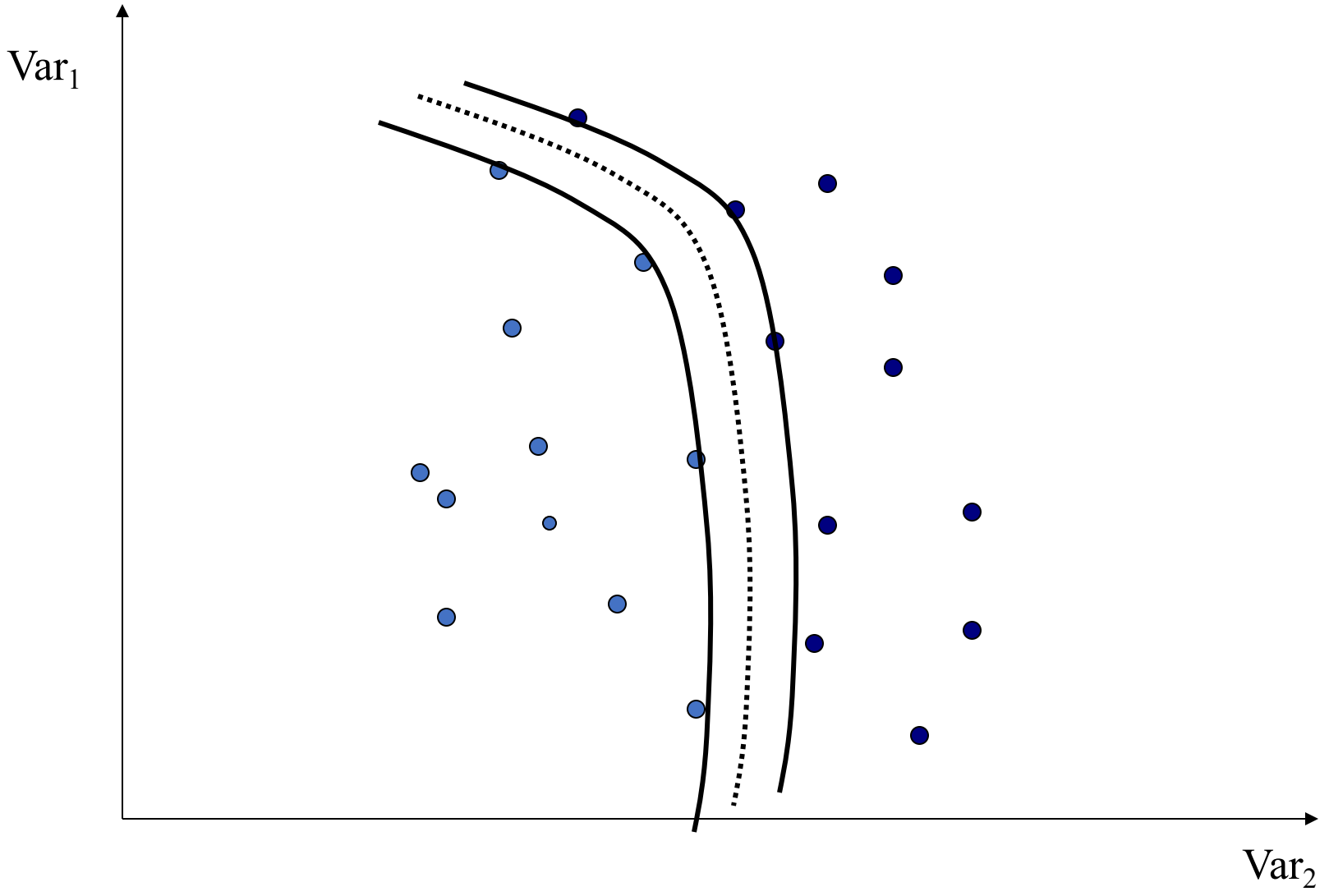
Introduce slack variables ξ_i

Allow some instances to fall within the margin, but penalize them

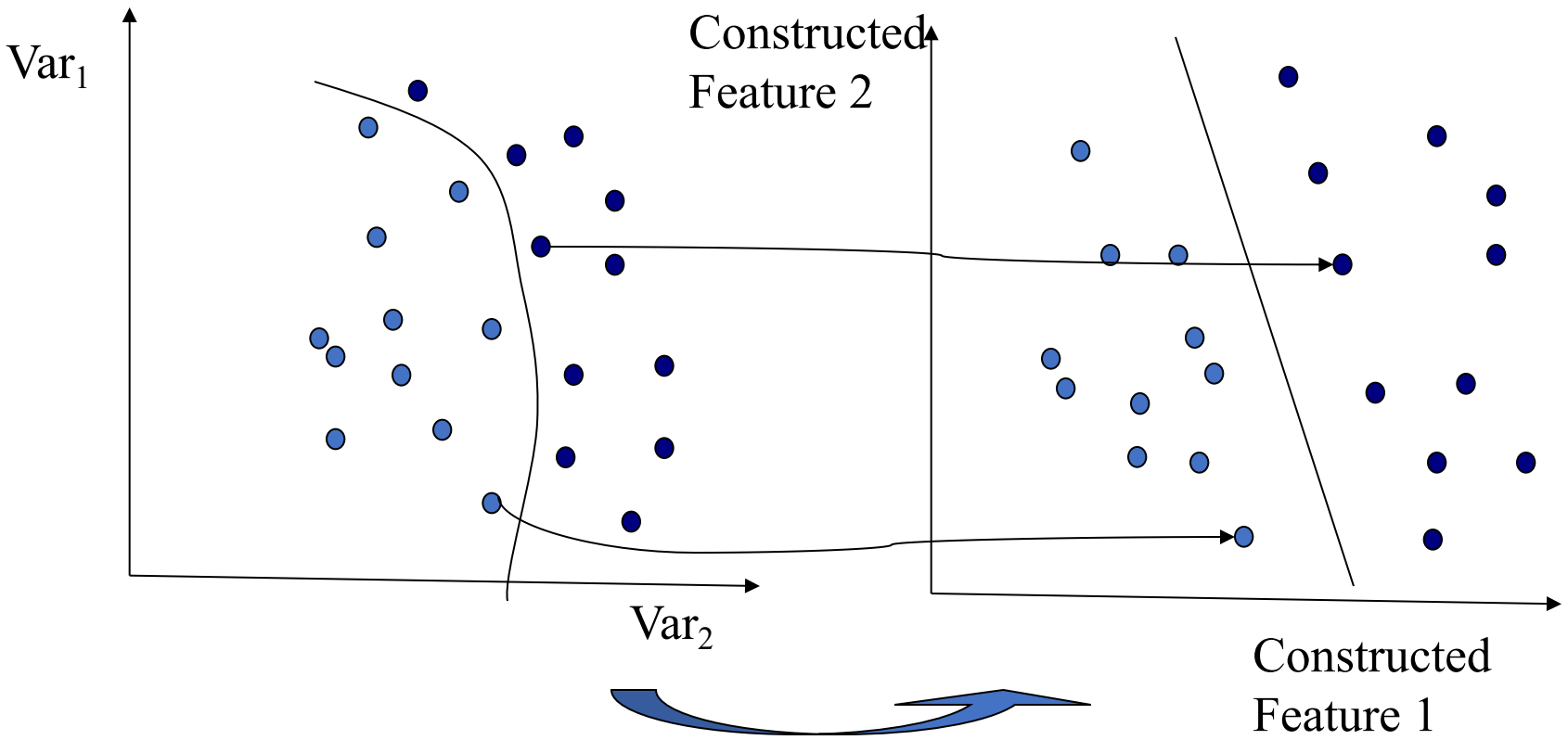
Disadvantages of Linear Decision Surfaces



Advantages of Non-Linear Surfaces



Linear Classifiers in High-Dimensional Spaces



Find function Φ(x) to map to a different space

Mapping Data to a High-Dimensional Space

- **Find function Φ(x) to map to a different space, then SVM formulation becomes:**

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \qquad s.t.\ \ y_i(w\cdot\Phi(x)+b) \geq 1-\xi_i, \forall x_i$$
$$\xi_i \geq 0$$

- **Data appear as Φ(x), weights *w* are now weights in the new space**
- **Explicit mapping expensive if Φ(x) is very high dimensional**
- **Solving the problem without explicitly mapping the data is desirable**

The Dual of the SVM Formulation

- **Original SVM formulation**
  - *n* inequality constraints
  - *n positivity constraints*
  - *n* number of ξ variables

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$s.t.\ \ y_i(w \cdot \Phi(x) + b) \geq 1 - \xi_i, \forall x_i$$
$$\xi_i \geq 0$$

- **The (Wolfe) dual of this problem**
  - one equality constraint
  - *n* positivity constraints
  - *n* number of α variables (Lagrange multipliers)
  - Objective function more complicated

$$\min_{a_i} \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (\Phi(x_i) \cdot \Phi(x_j)) - \sum_i \alpha_i$$

$$s.t.\ \ \mathrm{C} \geq \alpha_\mathrm{i} \geq 0, \forall x_i$$
$$\sum_i \alpha_i y_i = 0$$

- **NOTICE: Data only appear as $\Phi(x_i) \cdot \Phi(x_j)$**

The Kernel Trick

- **$\Phi(x_i) \cdot \Phi(x_j)$: means, map data into new space, then take the inner product of the new vectors**
- **We can find a function such that: $K(x_i \cdot x_j) = \Phi(x_i) \cdot \Phi(x_j)$, i.e., the image of the inner product of the data is the inner product of the images of the data**
- **Then, we do not need to explicitly map the data into the high-dimensional space to solve the optimization problem (for training)**
- **How do we classify without explicitly mapping the new instances?**

$$\text{sgn}(wx + b) = \text{sgn}(\sum_i \alpha_i y_i K(x_i, x) + b)$$

$$\text{where } b \text{ solves } \alpha_j (y_j \sum_i \alpha_i y_i K(x_i, x_j) + b - 1) = 0,$$

$$\text{for any } j \text{ with } \alpha_j \neq 0$$

Examples of Kernels

- **Assume we measure two quantities, e.g. expression level of genes *TrkC* and *SonicHedghog (SH)* and we use the mapping:**

$$\Phi :< x_{TrkC}, x_{SH} >\rightarrow \{x_{TrkC}^2, x_{SH}^2, \sqrt{2}x_{TrkC}x_{SH}, x_{TrkC}, x_{SH}, 1\}$$

- **Consider the function:**

$$K(x \cdot z) = (x \cdot z + 1)^2$$

- **We can verify that:**

$$\Phi(x) \cdot \Phi(z) =$$
$$x_{TrkC}^2 z_{TrkC}^2 + x_{SH}^2 z_{SH}^2 + 2x_{TrkC}x_{SH}z_{TrkC}z_{SH} + x_{TrkC}z_{TrkC} + x_{SH}z_{SH} + 1 =$$
$$= (x_{TrkC}z_{TrkC} + x_{SH}z_{SH} + 1)^2 = (x \cdot z + 1)^2 = K(x \cdot z)$$

Polynomial and Gaussian Kernels

$$K(x \cdot z) = (x \cdot z + 1)^p$$

- **is called the polynomial kernel of degree *p*.**
- **For *p=2*, if we measure 7,000 genes using the kernel once means calculating a summation product with 7,000 terms then taking the square of this number**
- **Mapping explicitly to the high-dimensional space means calculating approximately 49,000,000 = 7,000**2 new features for both training instances, then taking the inner product of that (another 49,000,000 terms to sum)**
- **In general, using the Kernel trick provides huge computational savings over explicit mapping!**
- **Another commonly used Kernel is the Gaussian (maps to a dimensional space with number of dimensions equal to the number of training cases):**

$$K(x \cdot z) = \exp(-\|x - z\| / 2\sigma^2)$$

The Mercer Condition

- **Is there a mapping $\Phi(x)$ for any symmetric function $K(x,z)$? No**
- **The SVM dual formulation requires calculation $K(x_i, x_j)$ for each pair of training instances. The array $G_{ij} = K(x_i, x_j)$ is called the Gram matrix**
- **There is a feature space $\Phi(x)$ when the Kernel is such that G is always semi-positive definite (Mercer condition)**

Variable Selection with SVMs

- **Recursive Feature Elimination**
  - Train a linear SVM
  - Remove the variables with the lowest weights (those variables affect classification the least), e.g., remove the lowest 50% of variables
  - Retrain the SVM with remaining variables and repeat until classification is reduced
- **Very successful**
- **Other formulations exist where minimizing the number of variables is folded into the optimization problem**
- **Similar algorithm exist for non-linear SVMs**
- **Some of the best and most efficient variable selection methods**

Multiclass Classification & SVMs

Multiclass classification

- **Introduction**

- **Combining binary classifiers**
  - One-vs-all
  - All-vs-all

- **Training a single classifier**
  - Multiclass SVM

What is multiclass classification?

- **An input can belong to one of K classes**

- **Training data: Input associated with class label (a number from 1 to K)**
- **Prediction: Given a new input, predict the class label**

Each input belongs to exactly one class. Not more, not less.

- **Otherwise, the problem is not multiclass classification**
- **If an input can be assigned multiple labels (think tags for emails rather than folders), it is called *multi-label classification***

Example applications: Images

- *Input*: hand-written character; *Output*: which character?

all map to the letter A

- *Input*: a photograph of an object; *Output*: which of a set of categories of objects is it?
  - Eg: the Caltech 256 dataset

Car tire

Car tire

Duck

laptop

Example applications: Language

- ***Input*: a news article; *Output*: which section of the newspaper should it belong to?**

- ***Input*: an email; *Output*: which folder should an email be placed into?**

- ***Input*: an audio command given to a car; *Output*: which of a set of actions should be executed?**

Binary to multiclass

- **Can we use a binary classifier to construct a multiclass classifier?**
  - Decompose the prediction into multiple binary decisions

- **How to decompose?**
  - One-vs-all
  - All-vs-all
  - Error correcting codes

General setting

- **Input $x \in R^n$**
  - The inputs are represented by their feature vectors
- **Output $y \in \{1, 2, \cdots, K\}$**
  - These classes represent domain-specific labels

- **Learning: Given a dataset $D = \{<x_i, y_i>\}$**
  - Need to specify a learning algorithm that takes uses D to construct a function that can predict **y** given **x**
  - Goal: find a predictor that does well on the training data and has low generalization error

- **Prediction/Inference: Given an example x and the learned function (often also called the model)**
  - Using the learned function, compute the class label for **x**

1. One-vs-all classification

- **Assumption: Each class individually separable from *all* the others**

- **Learning: Given a dataset D = {<x_i, y_i>},**

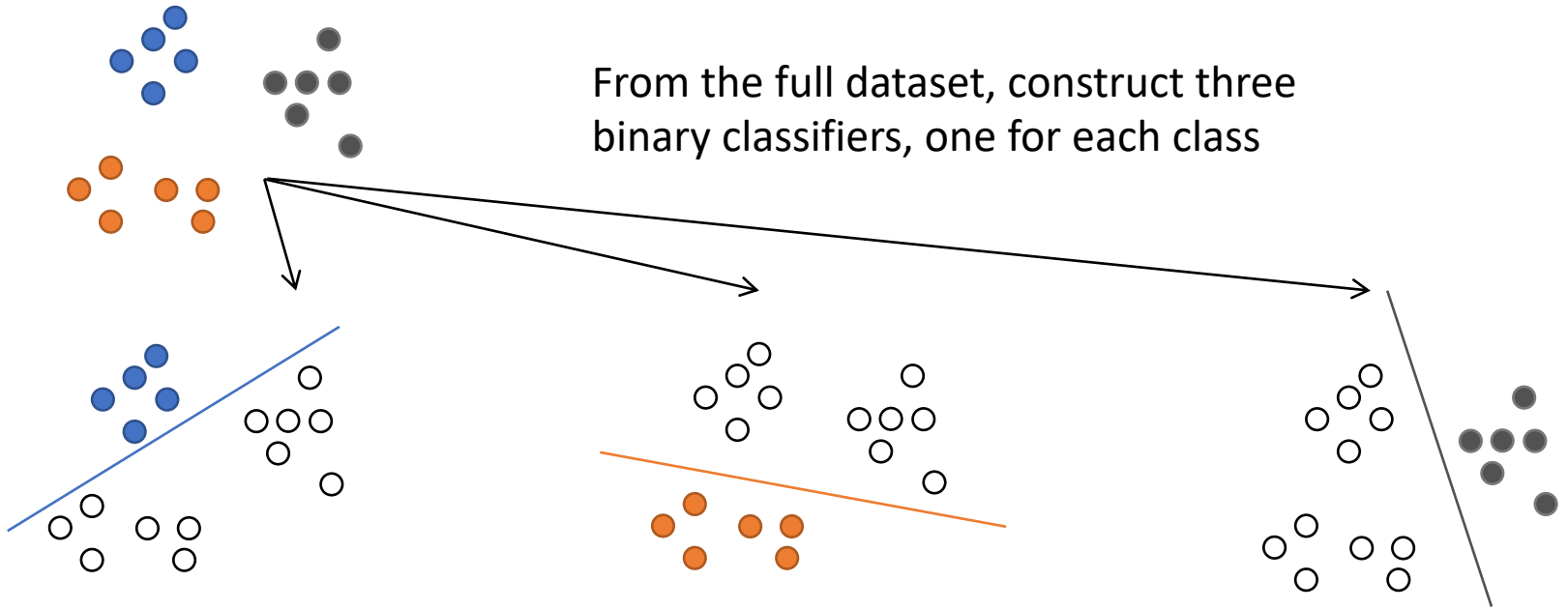
  Note: $\mathbf{x}_i \in R^n$, $\mathbf{y}_i \in \{1, 2, \cdots, K\}$

  - Decompose into K binary classification tasks
  - For class k, construct a binary classification task as:
    - Positive examples: Elements of D with label k
    - Negative examples: All other elements of D
  - Train K binary classifiers $\mathbf{w}_1, \mathbf{w}_2, \cdots \mathbf{w}_K$ using any learning algorithm we have seen

- **Prediction: "*Winner Takes All*"**

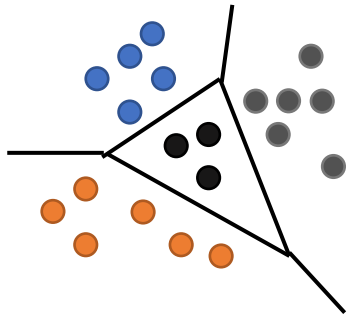
$$\mathbf{argmax}_i \ \mathbf{w}_i^T\mathbf{x}$$

Visualizing One-vs-all



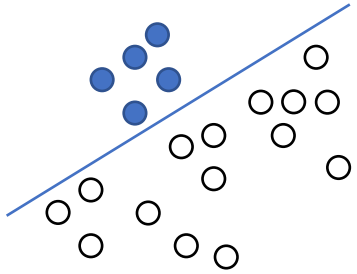
Winner Take All will predict the right answer. Only the correct label will have a positive score

One-vs-all may not always work

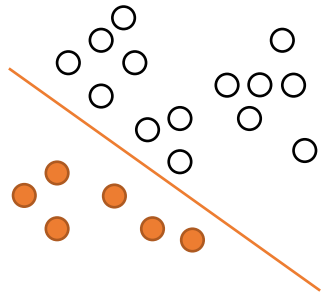


Black points are not separable with a single binary classifier

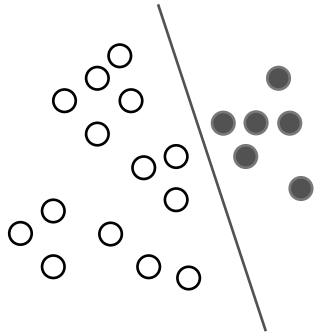
The decomposition will not work for these cases!



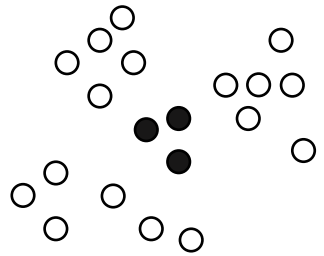
$\mathbf{w}_{blue}^T\mathbf{x} > 0$ for **blue inputs**



$\mathbf{w}_{red}^T\mathbf{x} > 0$ for **red inputs**



$\mathbf{w}_{green}^T\mathbf{x} > 0$ for **green inputs**



???

One-vs-all classification: Summary

- **Easy to learn**
  - Use any binary classifier learning algorithm
- **Problems**
  - No theoretical justification
  - Calibration issues
    - We are comparing scores produced by K classifiers trained independently. No reason for the scores to be in the same numerical range!
  - Might not always work
    - Yet, works fairly well in many cases, especially if the underlying binary classifiers are tuned, regularized

2. All-vs-all classification

Sometimes called one-vs-one

- **Assumption: *Every* pair of classes is separable**
- **Learning: Given a dataset D = {<x_i, y_i>},**

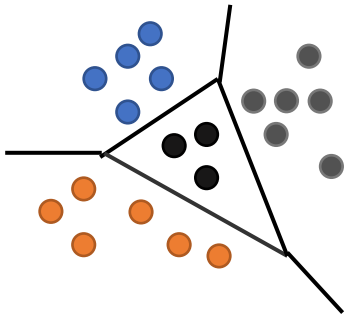
  Note: $\mathbf{x}_i \in R^n$, $\mathbf{y}_i \in \{1, 2, \cdots, K\}$

  - For every pair of labels (j, k), create a binary classifier with:
    - Positive examples: All examples with label j
    - Negative examples: All examples with label k

  - Train $\binom{K}{2} = \frac{K(K-1)}{2}$ classifiers in all

- **Prediction: More complex, each label get K-1 votes**
  - How to combine the votes? Many methods
    - Majority: Pick the label with maximum votes
    - Organize a tournament between the labels

All-vs-all classification



- **Every pair of labels is linearly separable here**
  - When a pair of labels is considered, all others are ignored
- **Problems**
  1. $O(K^2)$ weight vectors to train and store
  2. Size of training set for a pair of labels could be very small, leading to overfitting
  3. Prediction is often ad-hoc and might be unstable

     Eg: What if two classes get the same number of votes? For a tournament, what is the sequence in which the labels compete?

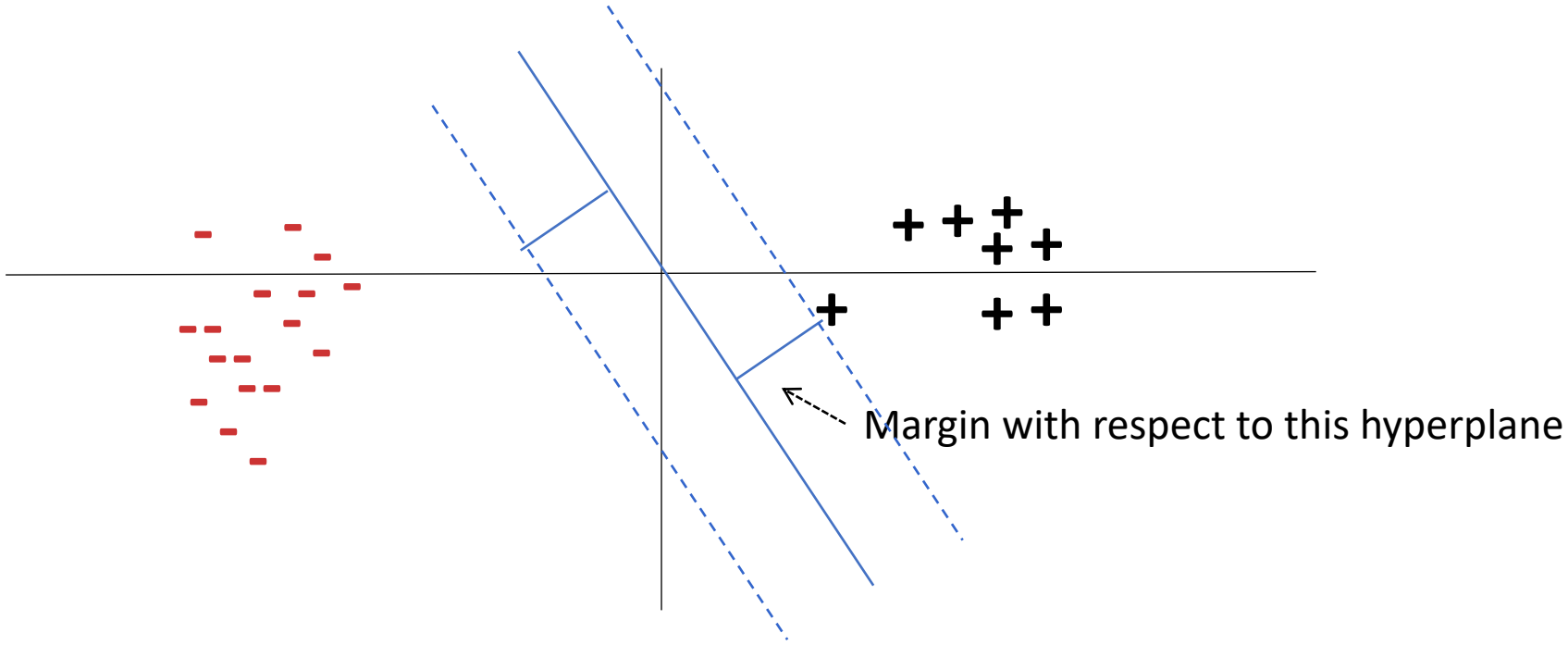
Summary: One-vs-All vs. All vs. All

- **Assume m examples, k class labels.**
  - For simplicity, say, m/k in each.
- **One vs. All:**
  - classifier f_i: m/k (+) and (k-1)m/k (-)
  - Decision:
  - Evaluate k linear classifiers and do Winner Takes All (WTA):
  - $f(x) = \text{argmax}_i \, f_i(x) = \text{argmax}_i \, w_i^T x$
- **All vs. All:**
  - Classifier f_{ij}: m/k (+) and m/k (-)
  - More expressivity, but less examples to learn from.
  - Decision:
  - Evaluate k^2 linear classifiers; decision sometimes unstable.
- **What type of learning methods would prefer All vs. All (efficiency-wise)?**

(Think about Dual/Primal)

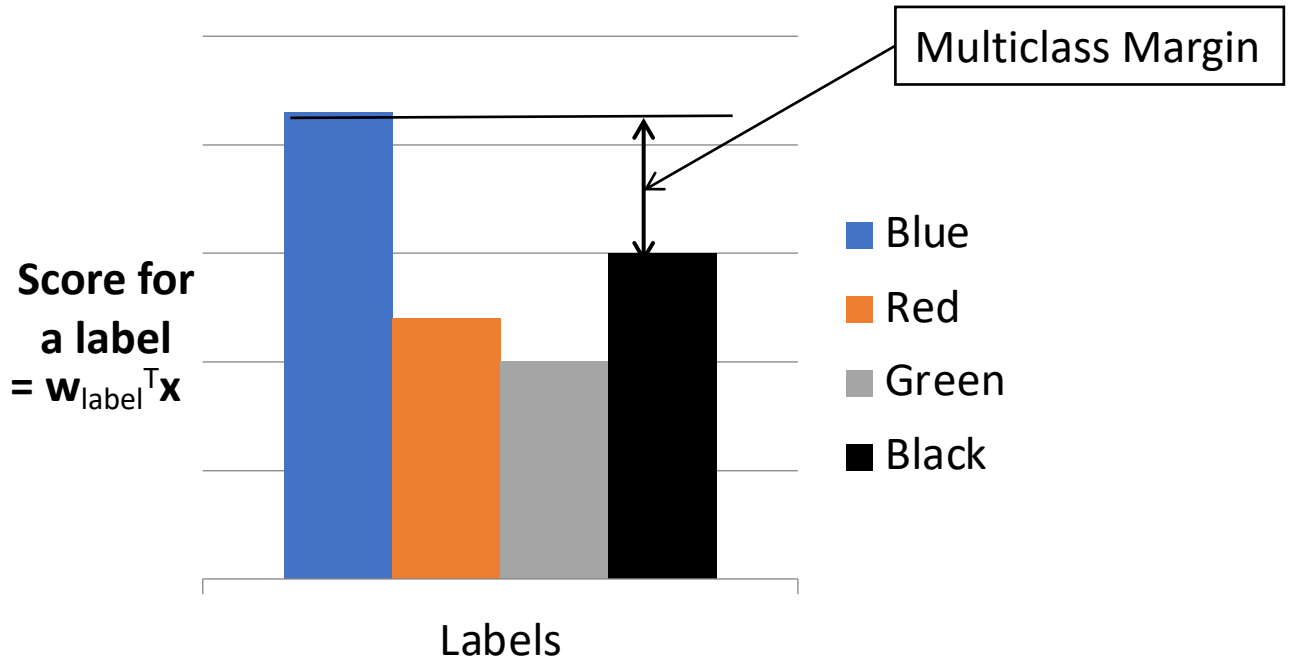
Recall: Margin for binary classifiers

- **The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.**



Multiclass margin

Defined as the score difference between the highest scoring label and the second one



Multiclass SVM (Intuition)

- **Recall: Binary SVM**
  - Maximize margin
  - Equivalently,

    Minimize norm of weights such that the closest points to the hyperplane have a score §1

- **Multiclass SVM**
  - Each label has a different weight vector (like one-vs-all)
  - Maximize multiclass margin
  - Equivalently,

    Minimize total norm of the weights such that the true label is scored at least 1 more than the second best one

Multiclass SVM in the separable case

Recall hard binary SVM

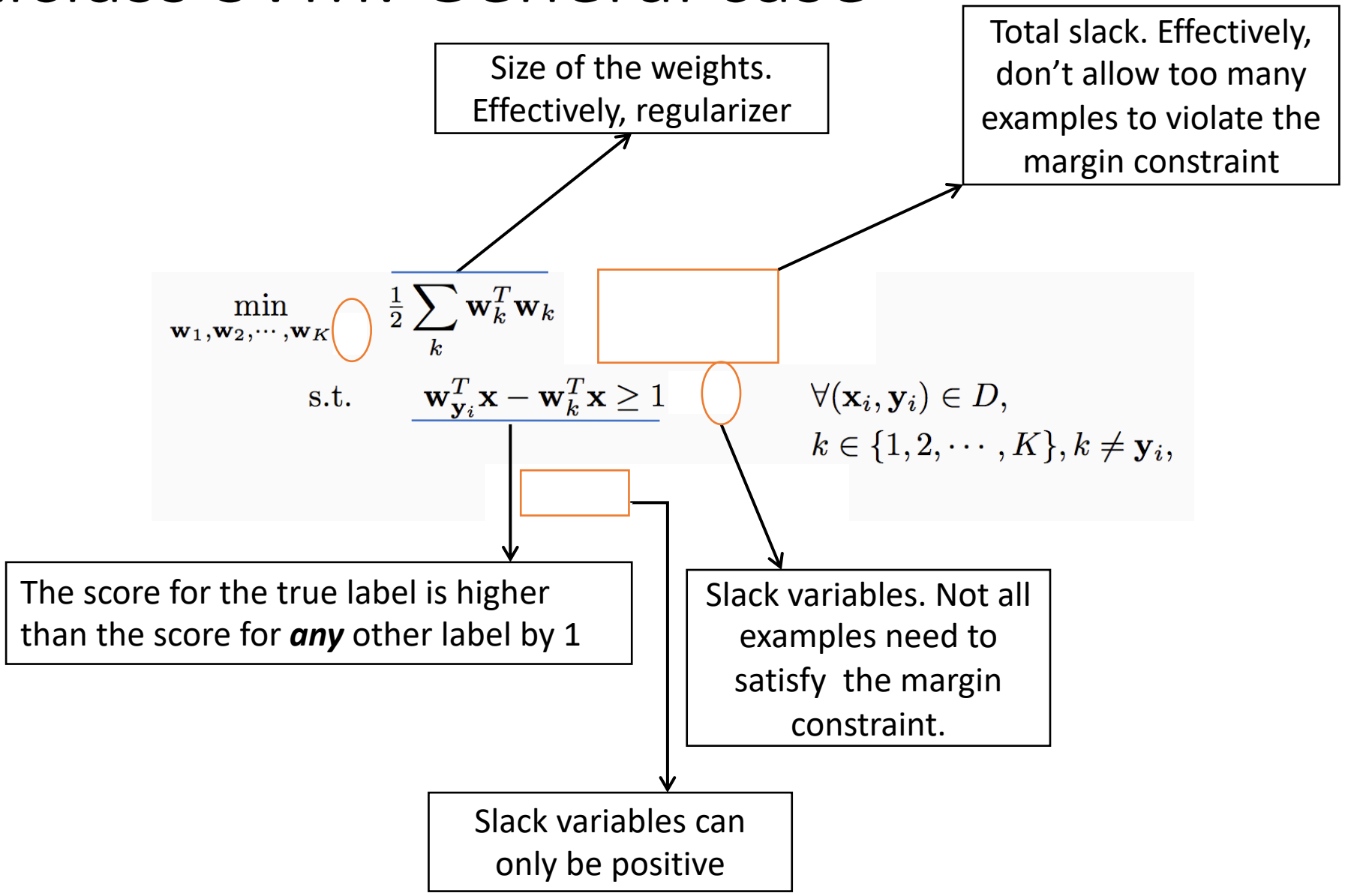
$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.} \forall i, \quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$$

Size of the weights. Effectively, regularizer

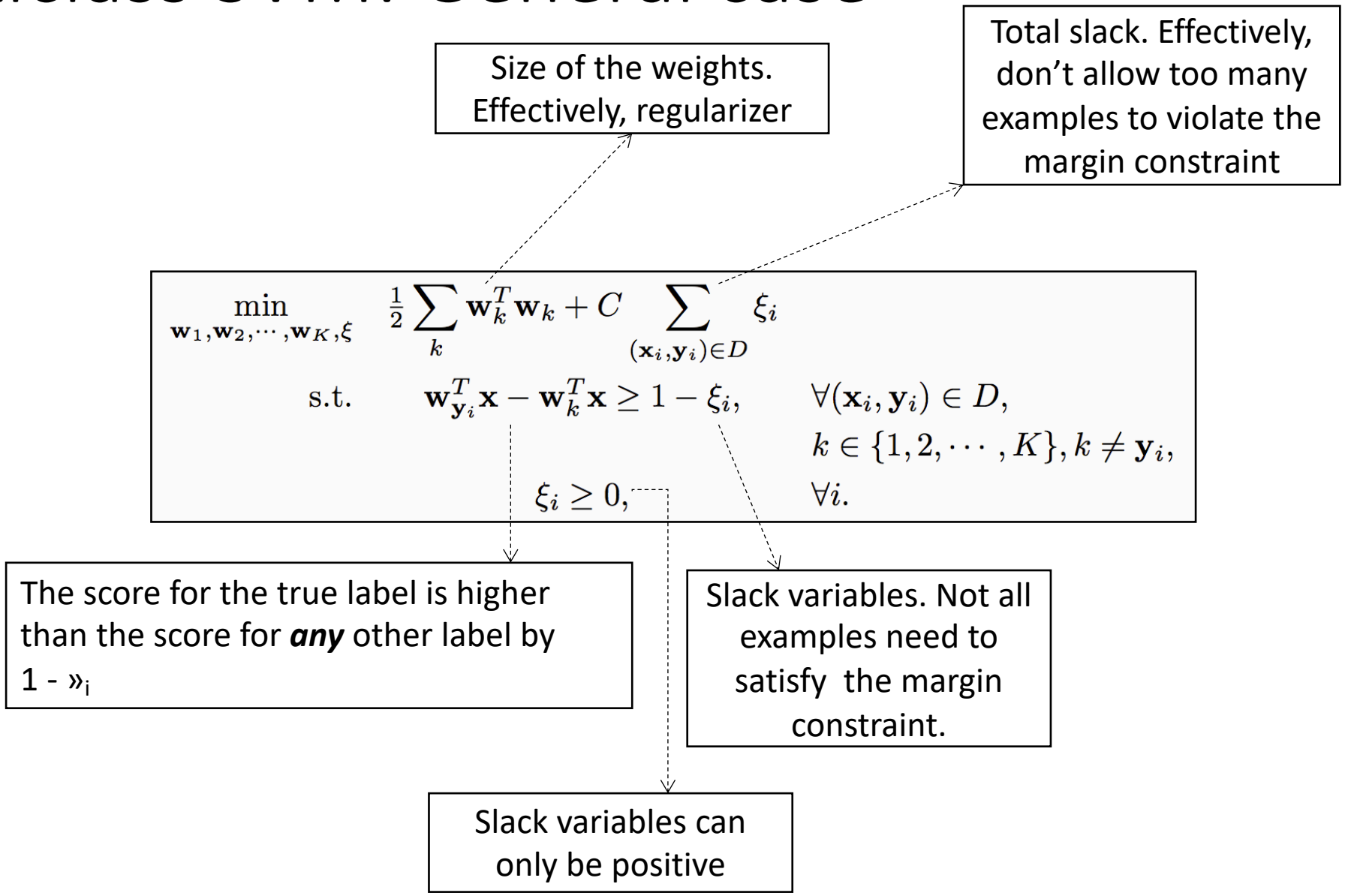
$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T\mathbf{w}_k$$
$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 \qquad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D,$$
$$k \in \{1, 2, \cdots, K\}, k \neq \mathbf{y}_i,$$

The score for the true label is higher than the score for ***any*** other label by 1

Multiclass SVM: General case



Multiclass SVM: General case



Multiclass SVM

- **Generalizes binary SVM algorithm**
  - If we have only two classes, this reduces to the binary (up to scale)
- **Comes with similar generalization guarantees as the binary SVM**
- **Can be trained using different optimization methods**
  - Stochastic sub-gradient descent can be generalized
    - Try as exercise

MultiClass SVMs

- **One-versus-all**
  - Train *n* binary classifiers, one for each class against all other classes.
  - Predicted class is the class of the most confident classifier
- **One-versus-one**
  - Train *n(n-1)/2* classifiers, each discriminating between a pair of classes
  - Several strategies for selecting the final classification based on the output of the binary SVMs
- **Truly MultiClass SVMs**
  - Generalize the SVM formulation to multiple categories