UVA CS 6316: Machine Learning

Lecture 11c: Support Vector Machine (SVM Optimization and Dual basic)

Dr. Yanjun Qi

University of Virginia

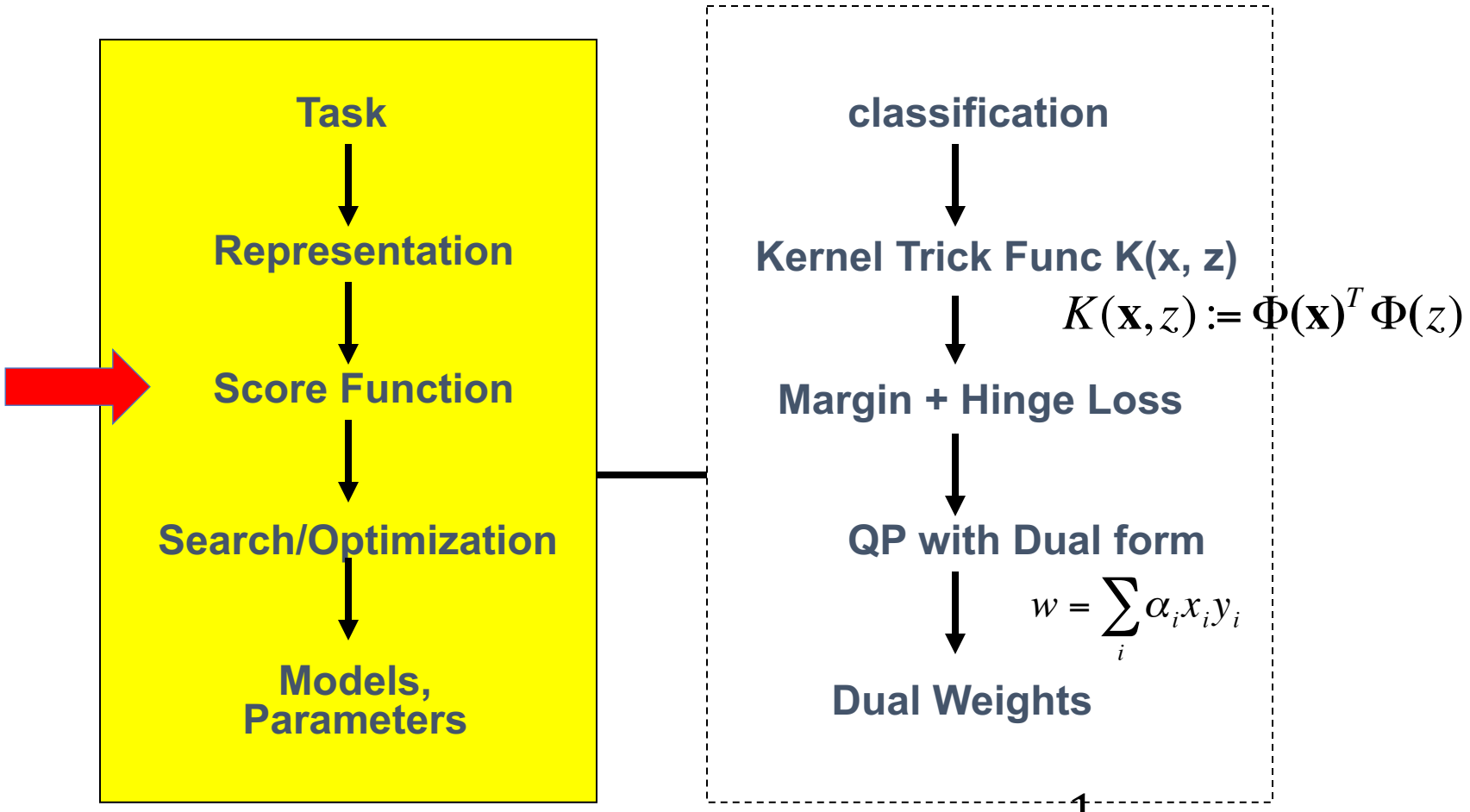
Department of Computer Science

What Left in SVM?

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Linearly Non-separable case (soft SVM)
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide

$K(x, z)$

Support Vector Machine



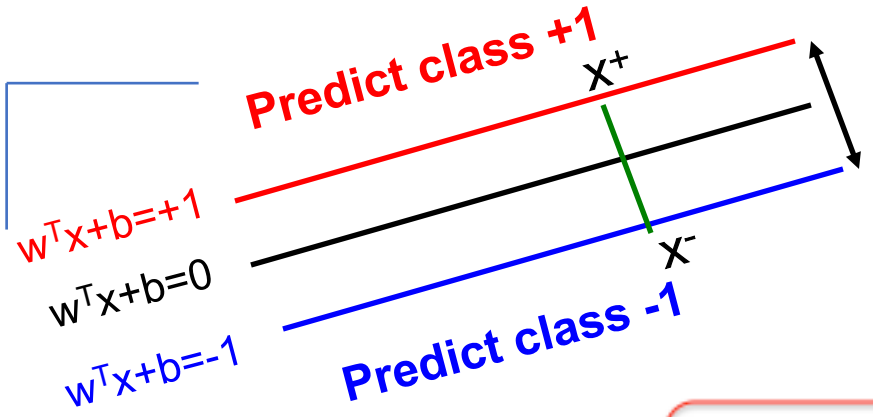
$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + \mathrm{C} \sum_{\mathrm{i}=1}^{\mathrm{n}} \varepsilon_{\mathrm{i}}$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1 - \varepsilon_{\mathrm{i}}$$

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_i \alpha_i \mathrm{y_i} = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

Optimization Step
i.e. learning optimal parameter for SVM



1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Optimization Reformulation

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Min $(w^T w)/2$

subject to the following constraints:

For all x in class + 1

$w^T x+b >= 1$ $y_i = 1$

For all x in class - 1

$w^T x+b <= -1$ $y_i = -1$

A total of n constraints if we have n input samples

$\rightarrow$ pos $y_i = 1$, $w^T x_i + b \geq 1$

$y_i (w^T x_i + b) \geq 1$

$\rightarrow$ neg $y_i = -1$, $w^T x_i + b \leq -1$

$y_i (w^T x_i + b) \geq 1$

Optimization Reformulation

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Min $(w^T w)/2$

subject to the following constraints:

For all x in class + 1

$w^T x+b >= 1$

For all x in class - 1

$w^T x+b <= -1$

A total of n constraints if we have n input samples

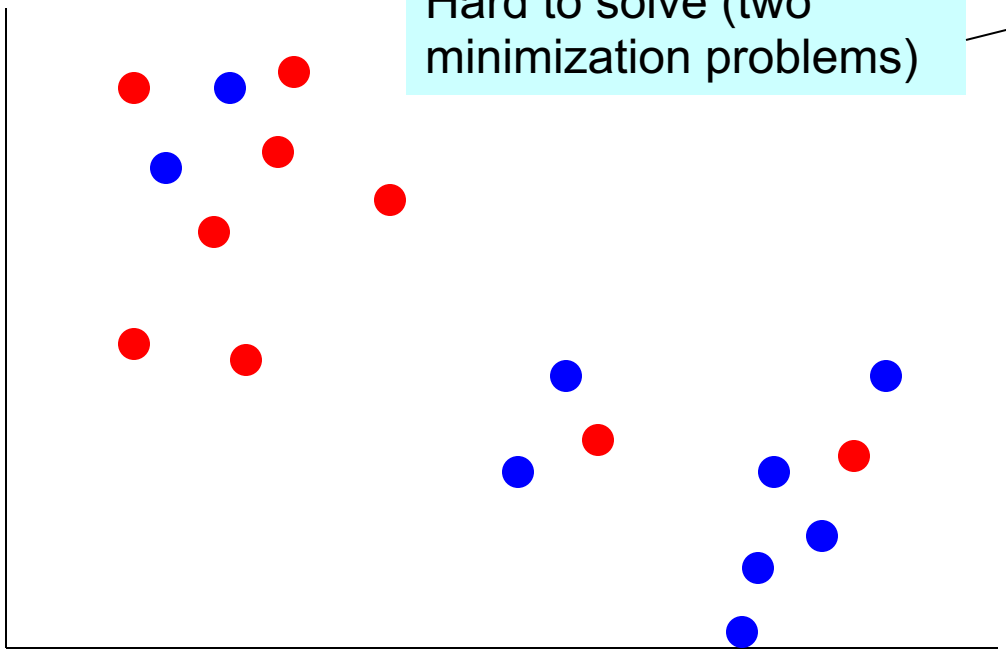
$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 / 2$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1$$

Linearly Non separable case

- So far we assumed that a linear hyperplane can perfectly separate the points
- But this is not usually the case

- noise, outliers



Hard to solve (two minimization problems)

How can we convert this to a QP problem?

- Minimize training errors?

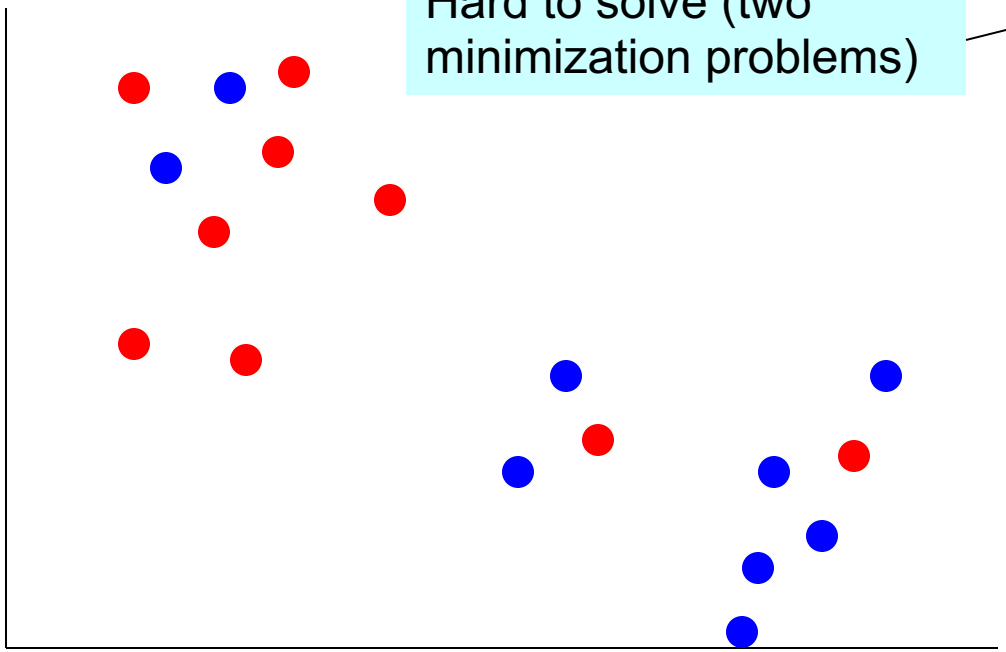
min $w^T w/2$

min #errors

Linearly Non separable case

- So far we assumed that a linear plane can perfectly separate the points
- But this is not usally the case

 - noise, outliers



Hard to solve (two minimization problems)

How can we convert this to a QP problem?

- Minimize training errors?

  min $w^T w/2$

  min #errors

- Penalize training errors:

 min $w^T w/2+C*(\#errors)$

Hard to encode in a QP problem

C ↑ penalize errors more

SVM: $\min_{\vec{w}, b} \; w^T w + C(\#\text{errors})$

Ridge Regression:

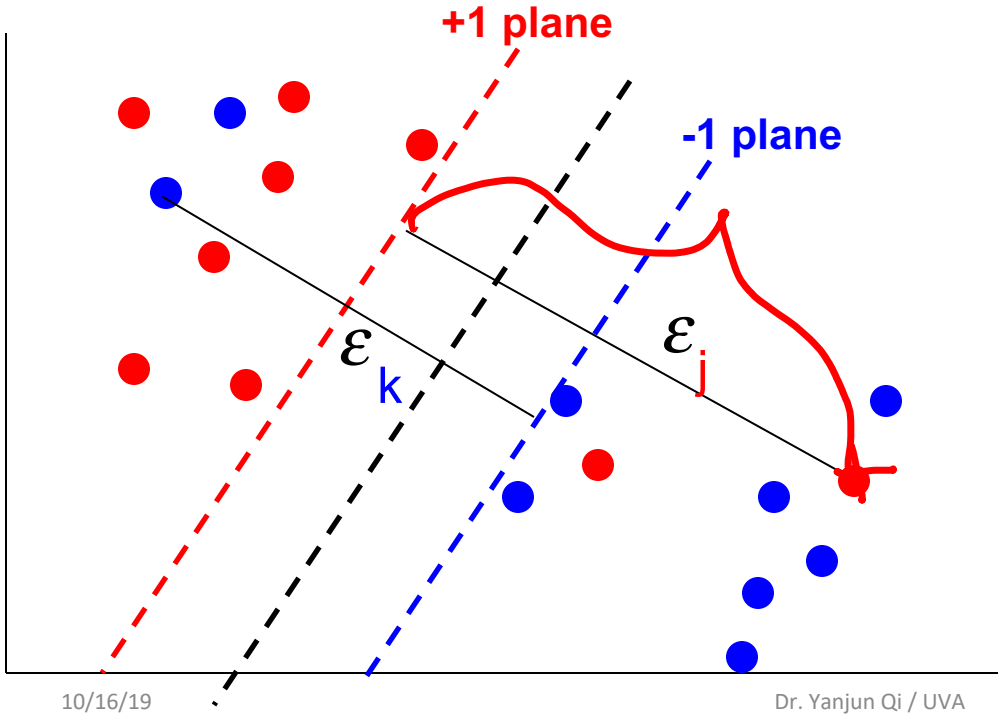
$$\min_{\theta} \; \lambda \theta^T \theta + J(\theta)$$

Linearly Non separable case

- Instead of minimizing the number of misclassified points we can minimize the ***distance*** between these points and their correct plane

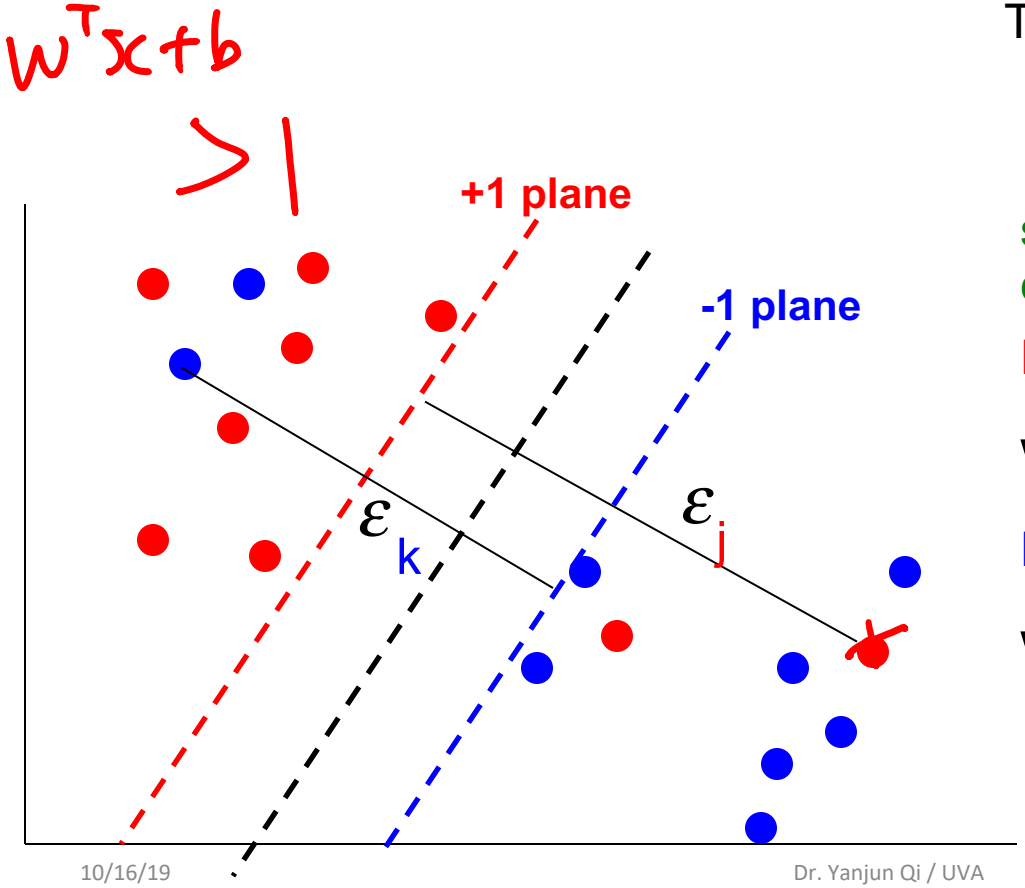
The new optimization problem is:

$$\min_w \frac{w^T w}{2} + C\sum_{i=1}^{n} \varepsilon_i$$



Linearly Non separable case

- Instead of minimizing the number of misclassified points we can minimize the ***distance*** between these points and their correct plane



The new optimization problem is:

$$\min_w \frac{w^T w}{2} + C\sum_{i=1}^{n} \varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

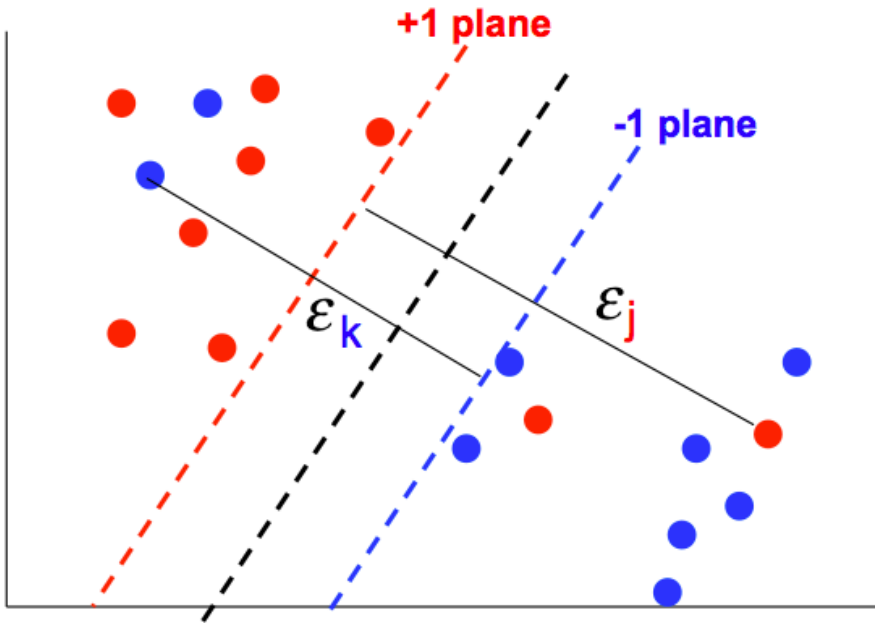
$$w^T x_i + b >= 1 - \varepsilon_i$$

For all x_i in class - 1

$$w^T x_i + b <= -1 + \varepsilon_i$$

Wait. Are we missing something?

Final optimization for linearly non-separable case



The new optimization problem is:

$$\min_w \frac{w^T w}{2} + C\sum_{i=1}^{n} \varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

$w^T x_i + b >= 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x_i + b <= -1 + \varepsilon_i$

} A total of n constraints

For all i

$$\varepsilon_i \geq 0$$

} Another n constraints

Two optimization problems:

For the separable and non separable cases

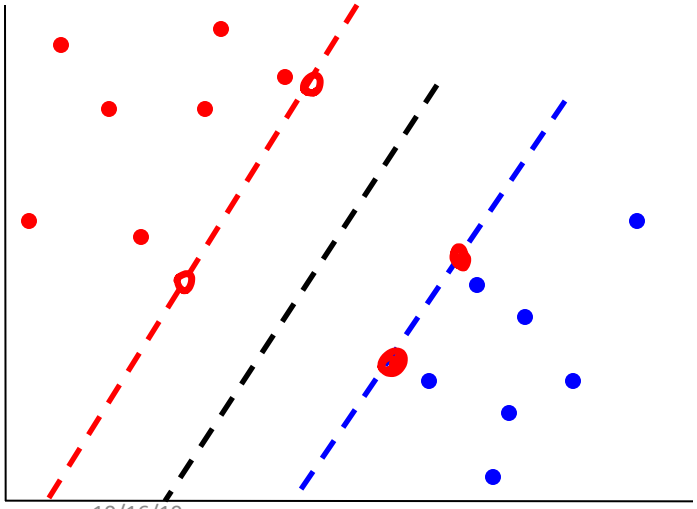
$$\min_w \frac{w^T w}{2}$$

For all x in class + 1

$w^T x+b >= 1$

For all x in class - 1

$w^T x+b <=-1$



$$\min_w \frac{w^T w}{2} + C\sum_{i=1}^{n} \varepsilon_i$$

For all x_i in class + 1

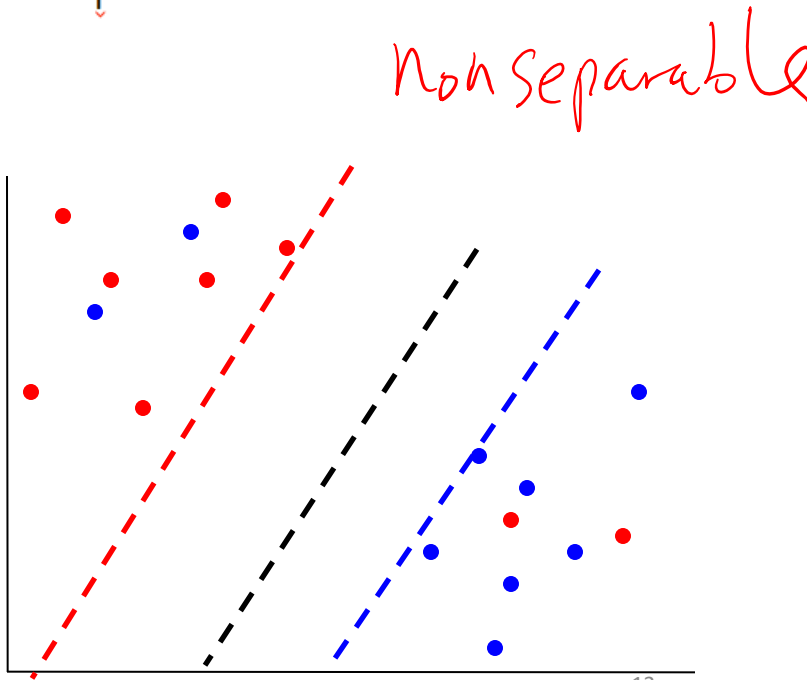
$w^T x_i+b >= 1-\varepsilon_i$

For all x_i in class - 1

$w^T x_i+b <= -1+\varepsilon_i$

For all i

$$\varepsilon_i \geq 0$$

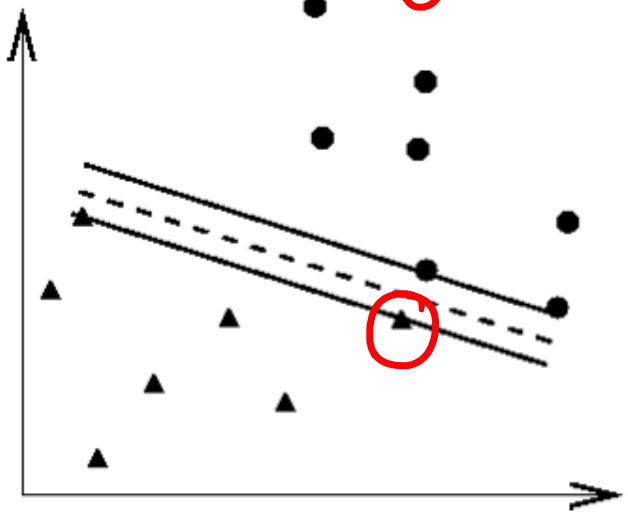


Model Selection, find right C

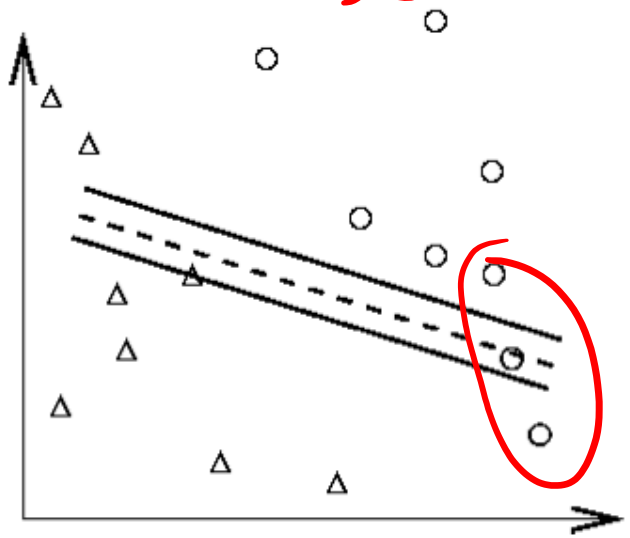
Training

Test

large C



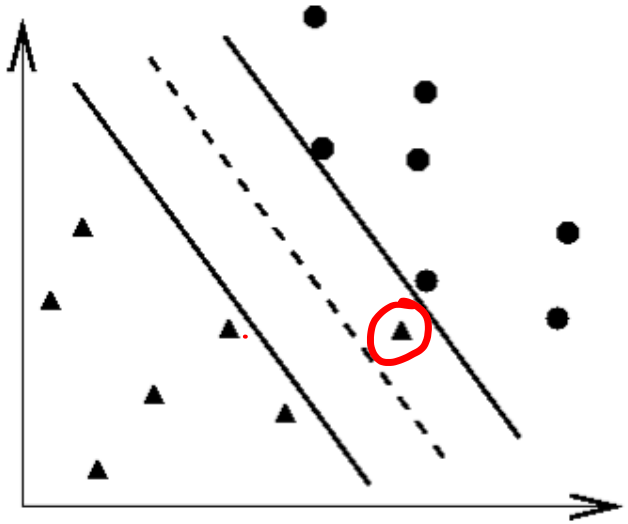
(a) Training data and an overfitting classifier



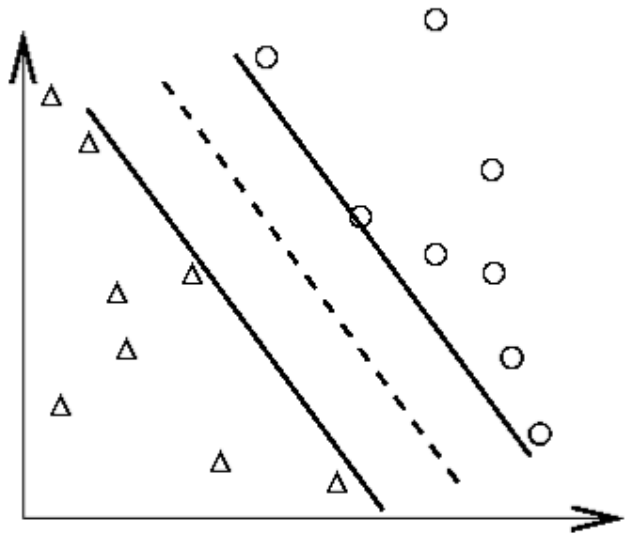
(b) Applying an overfitting classifier on testing data

Select the right penalty parameter C

small C



(c) Training data and a better classifier



(d) Applying a better classifier on testing data

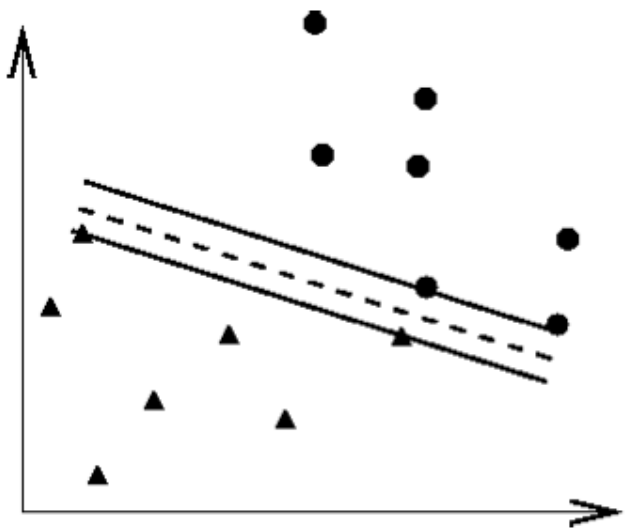
Model Selection, find right C

large C

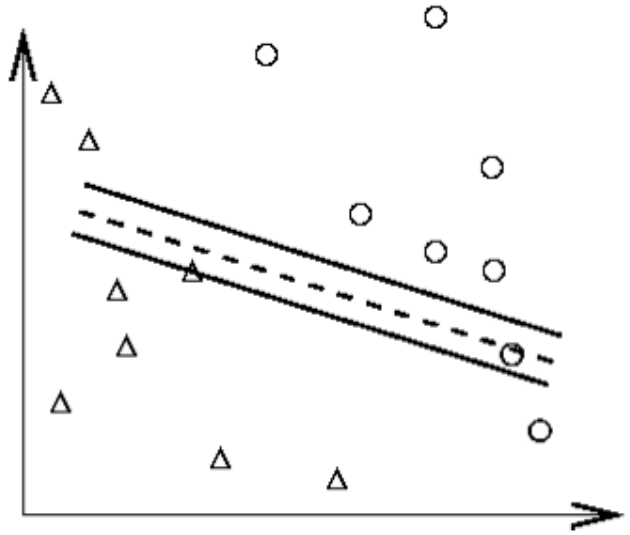
A large value of C means that misclassifications are bad - resulting in smaller margins and less training error (but more expected true error).

A small C results in more training error, hopefully better true error.

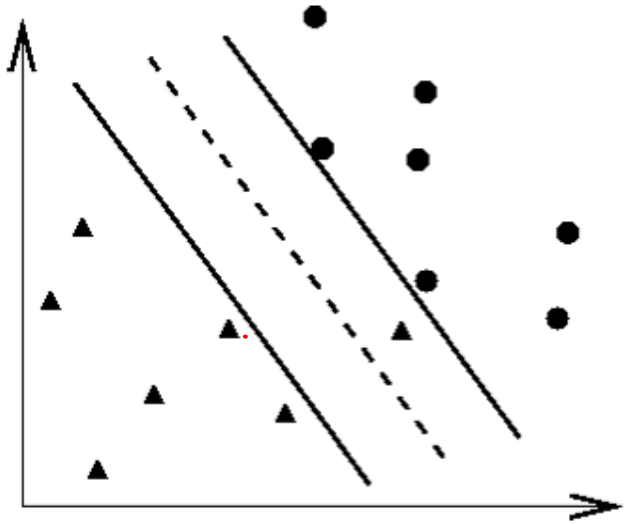
small C



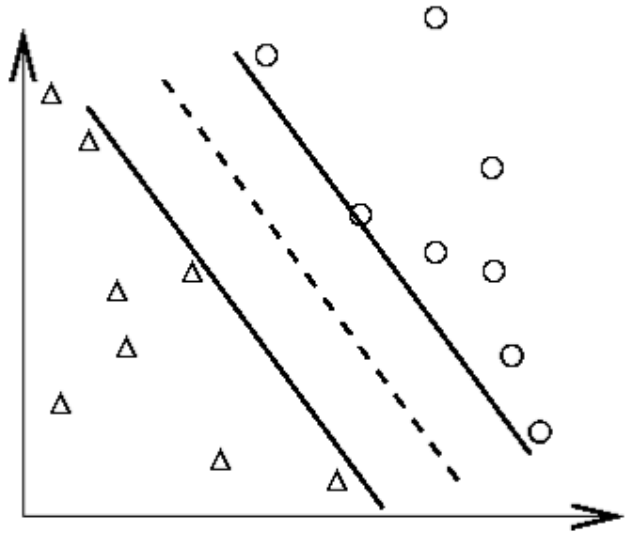
(a) Training data and an overfitting classifier



(b) Applying an overfitting classifier on testing data

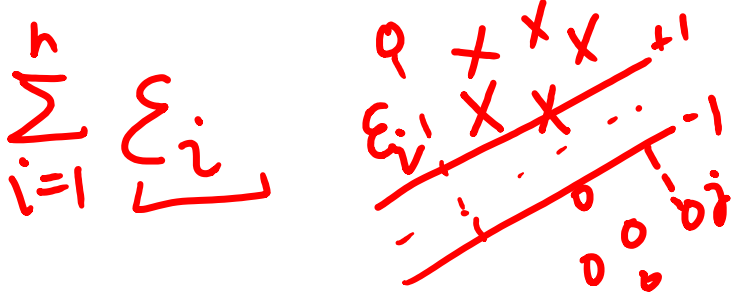


(c) Training data and a better classifier



(d) Applying a better classifier on testing data

Hinge Loss for Soft SVM



$$\min_w \frac{w^T w}{2} + C\sum_{i=1}^{n} \varepsilon_i$$

For all x_i in class + 1

$w^T x_i + b >= 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x_i + b <= -1 + \varepsilon_i$

For all i

$$\varepsilon_i \geq 0$$

soft

$$\sum_{i=1}^{n} \max(0, 1 - y_i f(\mathbf{x}_i))$$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \frac{\mathbf{w}^T \mathbf{w}}{2} + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$

subject to:

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

$$\varepsilon_i \geq 0$$

vs. Hard SVM

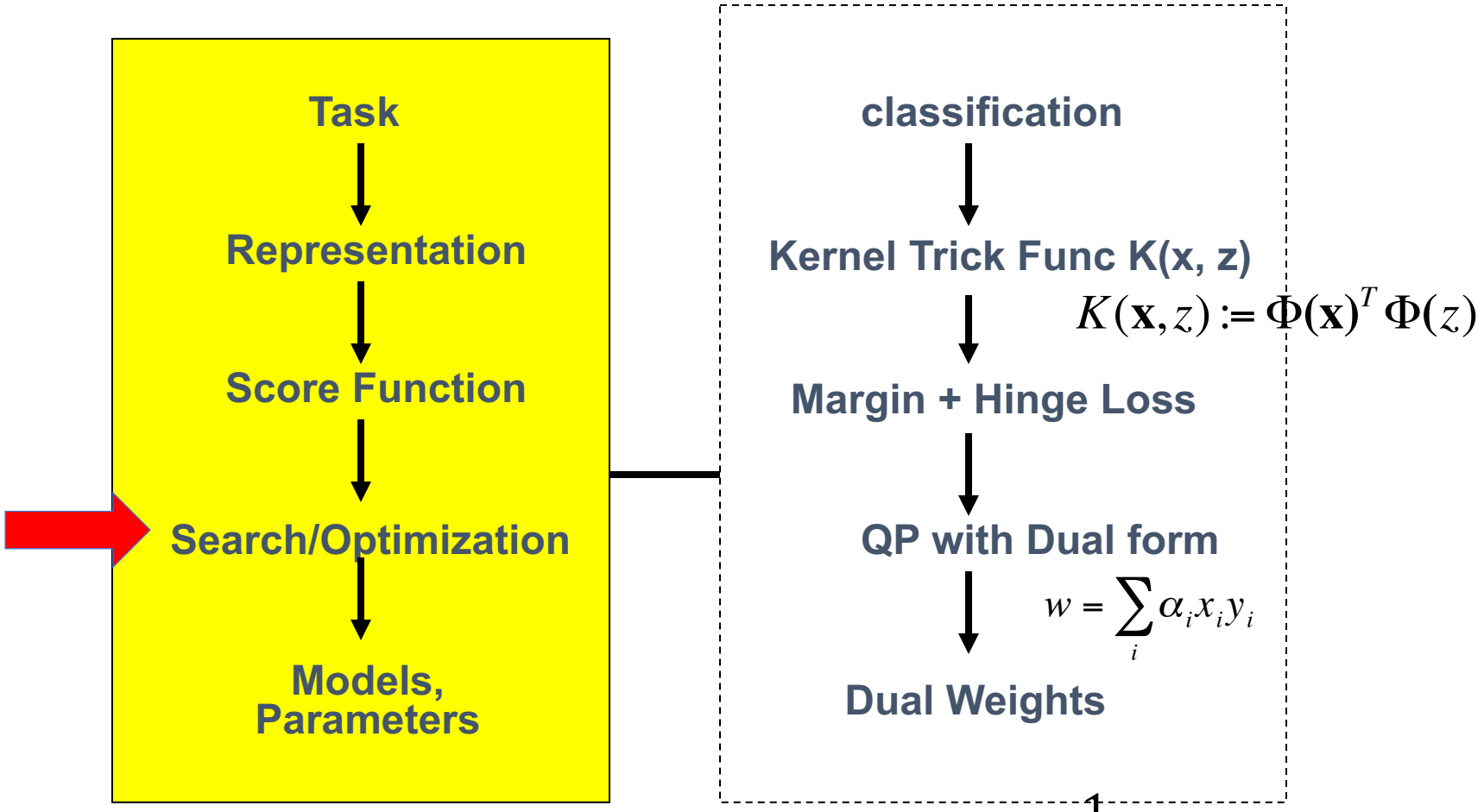
$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 / 2$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

What Left in SVM?

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Linearly Non-separable case (soft SVM)
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide

Support Vector Machine



$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + \mathrm{C} \sum_{\mathrm{i}=1}^{\mathrm{n}} \varepsilon_{\mathrm{i}}$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1 - \varepsilon_{\mathrm{i}}$

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

Two optimization problems: For the separable and non separable cases

Min $(w^T w)/2$

For all x in class + 1

$w^T x+b >= 1$

For all x in class - 1

$w^T x+b <=-1$

$$\min_w \frac{w^T w}{2}+C\sum_{i=1}^{n}\varepsilon_i$$

For all x_i in class + 1

$w^T x_i+b >= 1-\varepsilon_i$

For all x_i in class - 1

$w^T x_i+b <= -1+\varepsilon_i$

For all i

$$\varepsilon_i \geq 0$$

- Instead of solving these QPs directly we will solve a dual formulation of the SVM optimization problem
- The main reason for switching to this type of representation is that it would allow us to use a neat trick that will make our lives easier (and the run time faster)

Optimization Review:
Ingredients

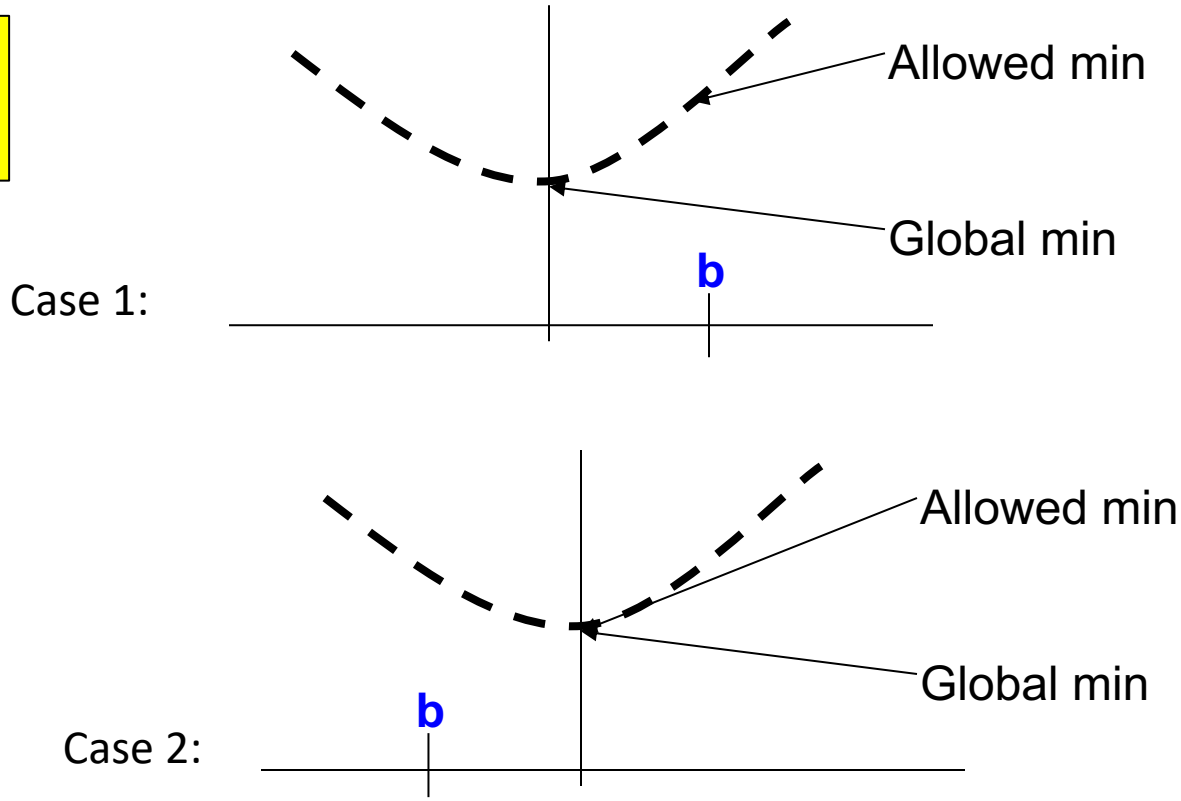
- Objective function
- Variables
- Constraints

Find values of the variables
that minimize or maximize the objective function
while satisfying the constraints

Optimization Review:
Constrained Optimization

$\min_u u^2$

s.t. $u >= b$

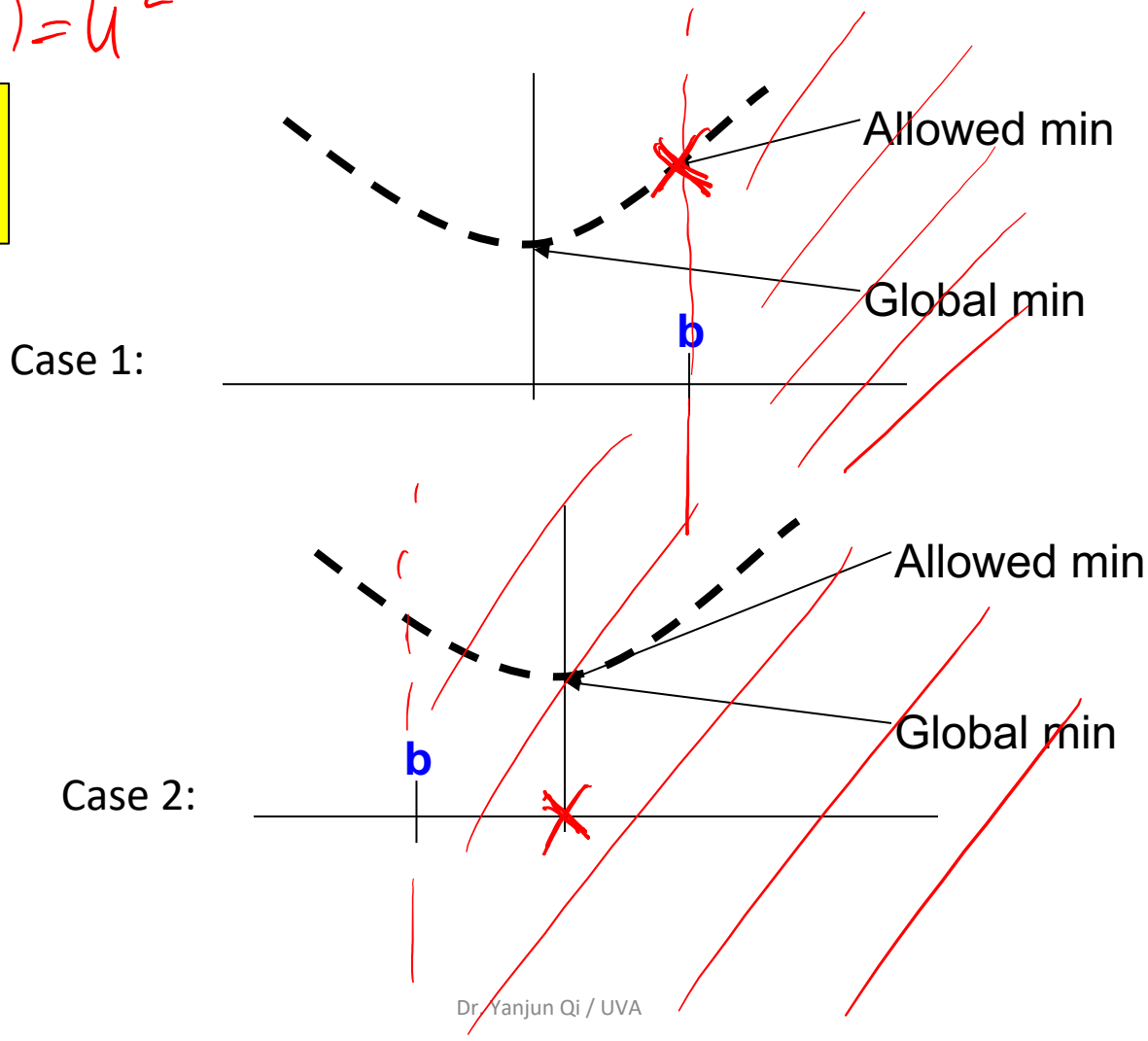


Optimization Review:
Constrained Optimization

$f(u) = u^2$

$\min_u u^2$

s.t. $u >= b$



Optimization Review:
Constrained Optimization

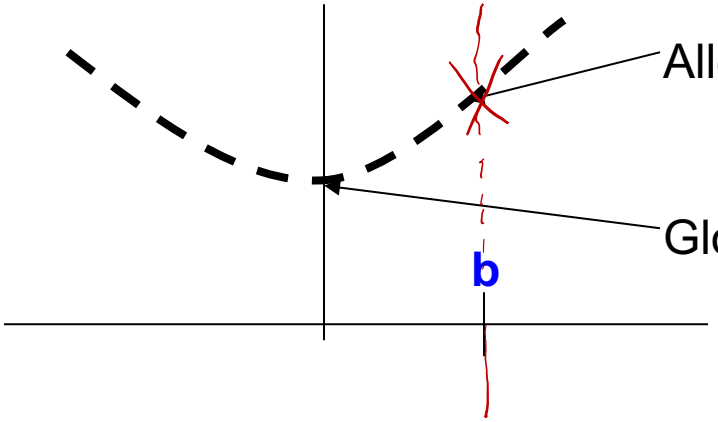
$f(u)$

$\min_u u^2$

s.t. $u >= b$

[Subject to]

Case 1:

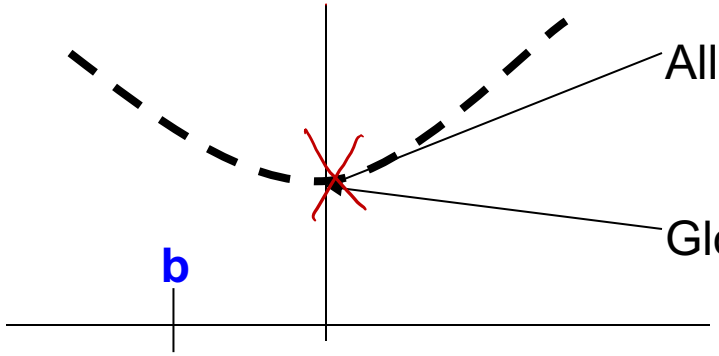


$b > 0$

$f(u^*) = b^2$

$u^* = b$

Case 2:



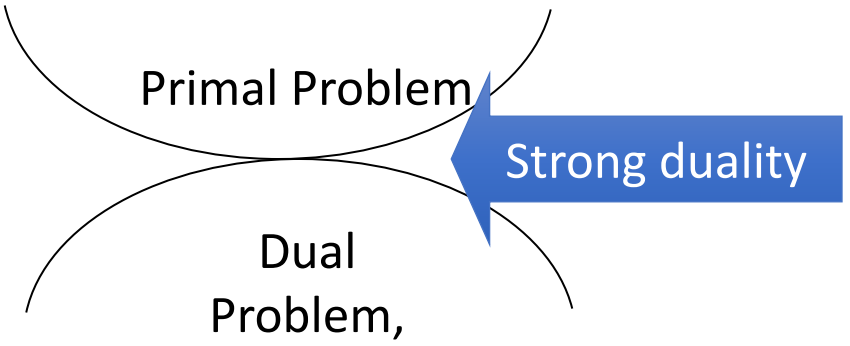
$b < 0$

$f(u^*) = 0$

$u^* = 0$

Optimization Review: Dual Problem (Extra)

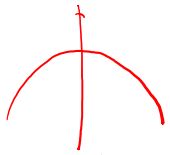
- Solving dual problem if the dual form is easier than primal form
- Need to change primal minimization to dual maximization (OR ➔ Need to change primal maximization to dual minimization)
- Only valid when the original optimization problem is convex/concave (strong duality)



Concrete derivation in L11Extra-SVMoptimDual

$$f(u): \begin{cases} \min u^2 \\ s.t.\ u \geq b \end{cases}$$

$$g(\alpha): \begin{cases} \max -\frac{\alpha^2}{4} + b\alpha = \max\left\{-\left(\frac{\alpha}{2}-b\right)^2 + b^2\right\} \\ s.t.\ \alpha \geq 0 \end{cases}$$



$$\begin{cases} \text{if } b \geq 0,\ b = \alpha/2, & u^* = b,\ g^* = b^2 \\ \text{if } b < 0,\ b \neq \alpha/2, & \alpha^* = 0,\ g^* = 0 \end{cases}$$

$$\Rightarrow \alpha(b-u) = 0 \qquad \text{KKT condition}$$

Optimization Review:
Lagrangian Duality (Extra)

- The Primal Problem

Primal:

$$\min_w \quad f_0(w)$$
$$\text{s.t.} \quad f_i(w) \le 0, \quad i = 1,\ldots,k$$

The generalized Lagrangian:

"Method of Lagrange multipliers" convert to a higher-dimensional problem

$$L(w,\alpha) = f_0(w) + \sum_{i=1}^{k} \alpha_i f_i(w)$$

the α's ($\alpha_i \ge 0$) are called the Lagarangian multipliers

Lemma:

$$\max_{\alpha,\alpha_i \ge 0} L(w,\alpha) = \begin{cases} f_0(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{o/w} \end{cases}$$

A re-written Primal:

$$\min_w \max_{\alpha,\alpha_i \ge 0} L(w,\alpha)$$

Optimization Review:
Lagrangian Duality, cont. (Extra)

- Recall the Primal Problem:

$$\min_w \max_{\alpha,\alpha_i \geq 0} \mathcal{L}(w,\alpha)$$

- The Dual Problem:

$$\max_{\alpha,\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha)$$

- **Theorem (weak duality):**

$$d^* = \max_{\alpha,\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha) \quad \leq \quad \min_w \max_{\alpha,\alpha_i \geq 0} \mathcal{L}(w,\alpha) = p^*$$

- **Theorem (strong duality):**

  Iff there exist a saddle point of $\mathcal{L}(w,\alpha)$

  we have $d^* = p^*$

Dual representation of the hard SVM QP

- We will start with the linearly separable case
- Instead of encoding the correct classification rule and constraint we will use Lagrange multiplies to encode it as part of the our minimization problem

Min $(w^T w)/2$

s.t.

$(w^T x_i + b) y_i >= 1$

Recall that Lagrange multipliers can be applied to turn the following problem:

$$L_{primal}(w, b, \alpha) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \sum_{i=1}^{N}\alpha_i\left(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1\right)$$

The Dual Problem (Extra)

$$\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$$

Dual formulation

- We minimize $\mathcal{L}$ with respect to w and b first:

$$\nabla_w \mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{train} \alpha_i y_i x_i = 0, \quad (*)$$

$$\nabla_b \mathcal{L}(w,b,\alpha) = \sum_{i=1}^{train} \alpha_i y_i = 0, \quad (**)$$

Note that **(*)** implies:

$$w = \sum_{i=1}^{train} \alpha_i y_i x_i \quad (***)$$

- Plus (***) back to $\mathcal{L}$, and using (**), we have:

$$\max_{\alpha_i \geq 0} \mathcal{L}(w,b,\alpha) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j=1} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

Summary: Dual for hard SVM (Extra)

Solving for **w** that gives maximum margin:

1. Combine objective function and constraints into new objective function, using Lagrange multipliers \alpha_i

$$L_{primal} = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \sum_{i=1}^{N}\alpha_i\left(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1\right)$$

2. To minimize this Lagrangian, we take derivatives of **w** and *b* and set them to 0:

Summary: Dual for hard SVM (Extra)

3. Substituting and rearranging gives the dual of the Lagrangian:

$$L_{dual} = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

which we try to maximize (not minimize).

4. Once we have the *alpha*$_i$*, we can substitute into previous equations to get **w** and *b*.
5. This defines **w** and *b* as linear combinations of the training data.

$$w = \sum_{i=1}^{train} \alpha_i y_i x_i$$

Summary: Dual SVM for linearly separable case

O(n) #para

w, b #para

O(p)

Dual formulation

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

n α_i

Min $(w^T w)/2$

subject to the following inequality constraints:

For all x in class + 1

$w^T x+b >= 1$

For all x in class - 1

$w^T x+b <= -1$

} A total of n constraints if we have n input samples

Easier than original QP, more efficient algorithms exist to find α_i, e.g. SMO (see extra slides)

Dual formulation for linearly non-separable case

prima linear soft margin + Hinge $\xi_i \forall i$ $\{w, b$

$O(n+p+1)$

Dual target function:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0, \forall i$$

Hyperparameter C should be tuned through k-folds CV

The only difference is that the \alpha are now bounded

upper

$O(n)$ #para

This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on α_i now

Once again, efficient algorithm exist to find α_i

Prediction via Dual Weights for linear case

$f(x_{ts}) = \text{Sign}(w^T x_{ts} + b)$

$w = \sum_{i=1}^{n} \alpha_i y_i x_i$

Dual target function:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0, \forall i$$

Hyperparameter C should be tuned through k-folds CV

The only difference is that the \alpha are now bounded

To evaluate a new sample x_{ts} we need to compute:

$$w^T x_{ts} + b = \sum_{i \in \text{supportV}} \alpha_i y_i \mathbf{x_i}^T \mathbf{x}_{ts} + b$$

This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on α_i now

Once again, efficient algorithm exist to find α_i

Dual SVM – Training using Kernel Matrix

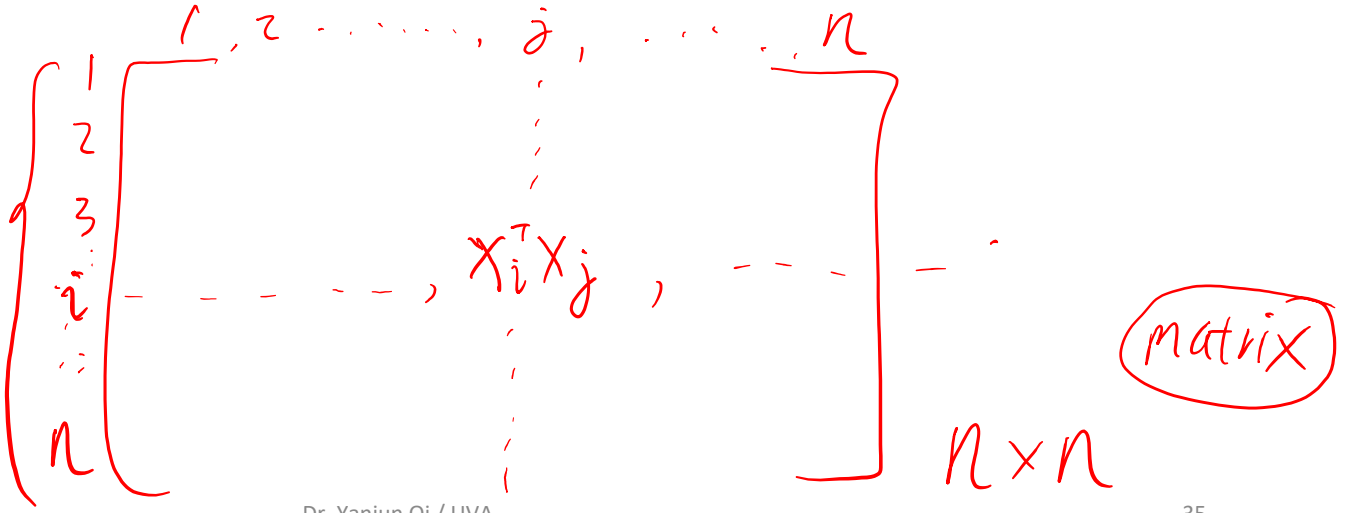
Our dual target function:

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}$$

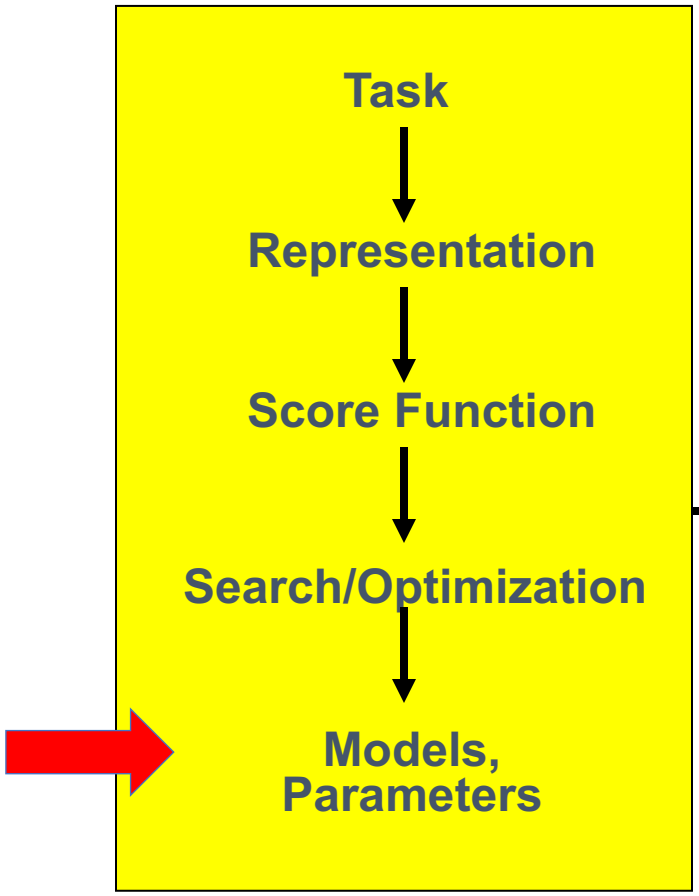
$$\sum_i \alpha_i y_i = 0$$

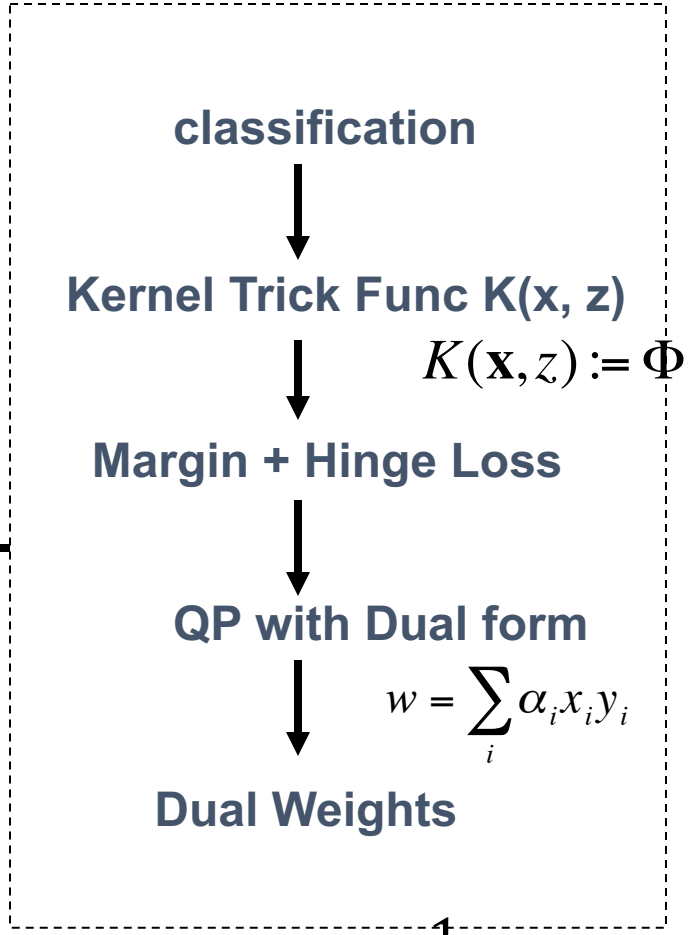
$$C \geq \alpha_i \geq 0, \forall i$$

Dot product among all training samples



Support Vector Machine





$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + \mathrm{C} \sum_{\mathrm{i}=1}^{\mathrm{n}} \varepsilon_{\mathrm{i}}$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1 - \varepsilon_{\mathrm{i}}$

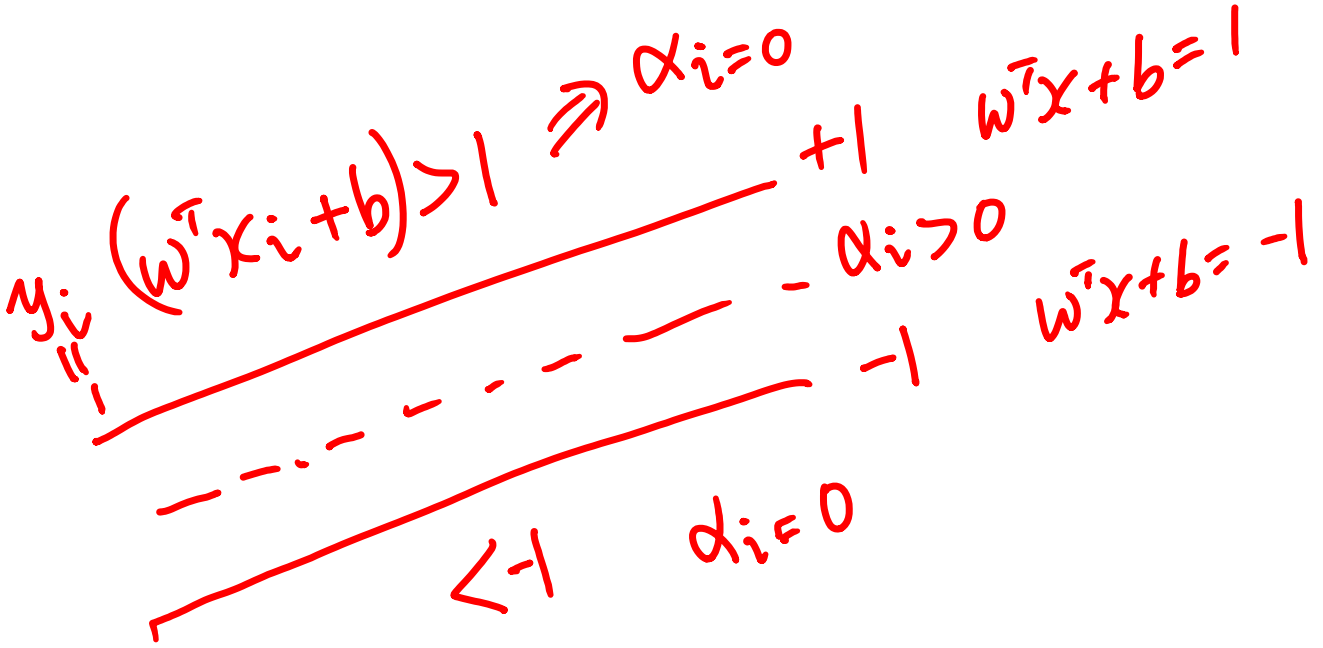
$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

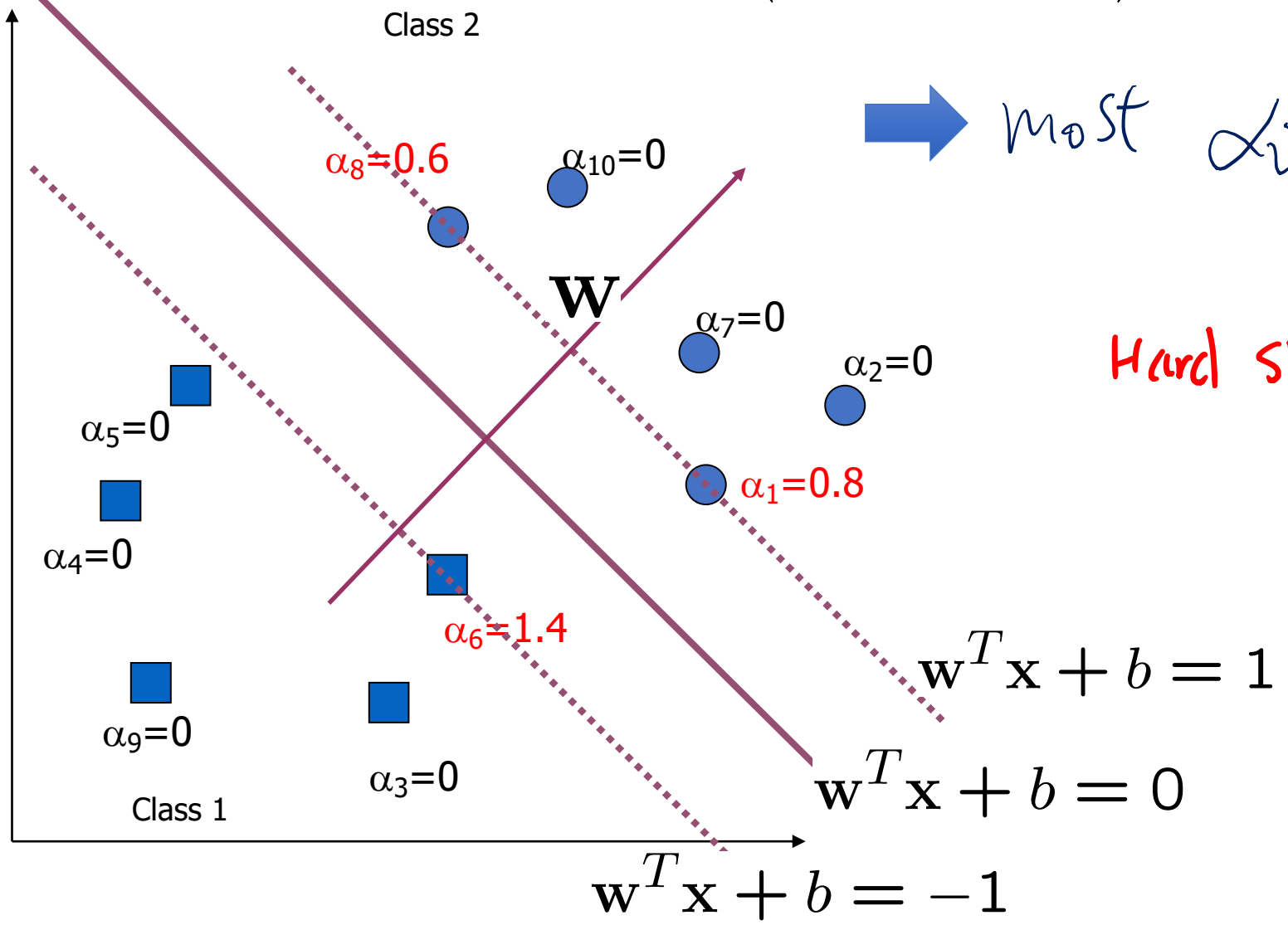
Support vectors: non-zero α_i

- only a few α_i can be nonzero!!

$$\forall i \Rightarrow \alpha_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) = 0, \quad i = 1, \ldots, n$$



$$\alpha_i \left(y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right) = 0, \quad i = 1, \ldots, n$$



Most $\alpha_i = 0$

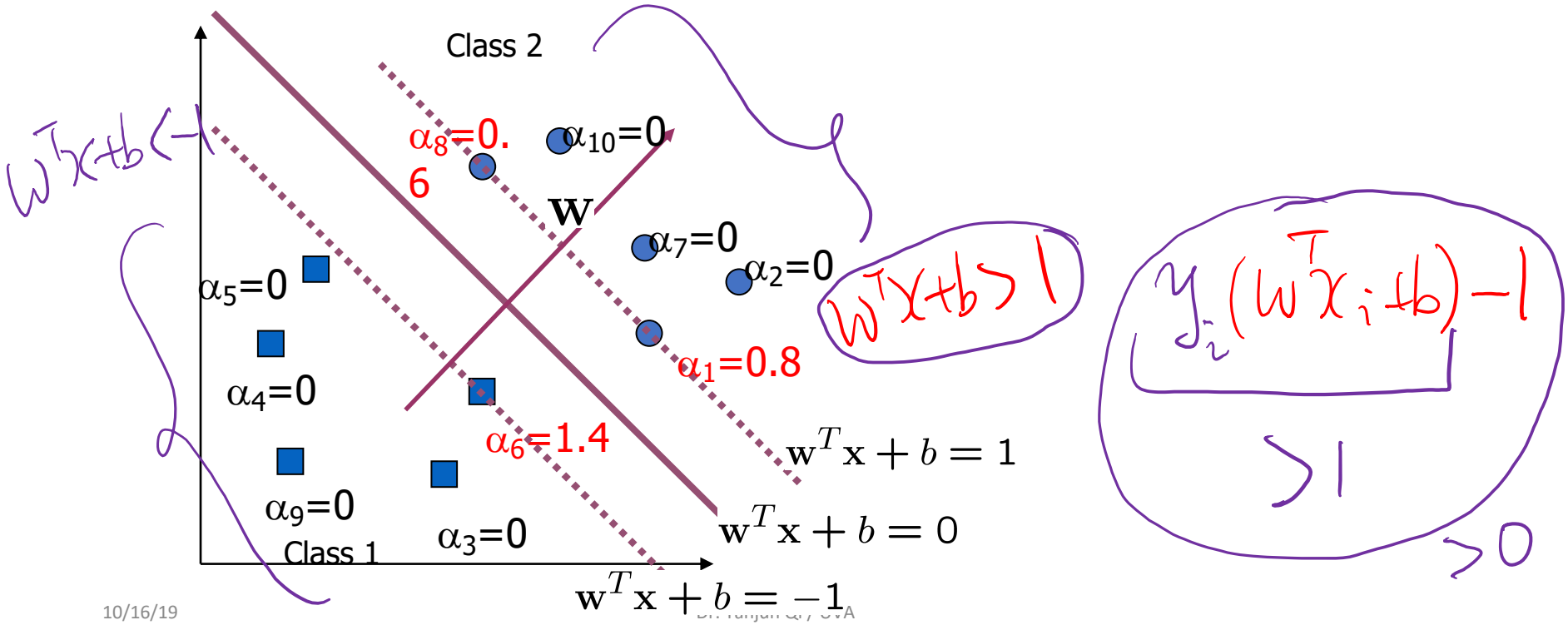
Hard SVM

- only a few α_i can be nonzero! Ie.

$$y_i(\mathbf{w}\cdot\mathbf{x}_i+b)=1$$

$$\alpha_i\left(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1\right)=0,\quad i=1,\ldots,n$$

for most $\alpha_i \Rightarrow \alpha_i = 0$



Support vectors: non-zero α_i

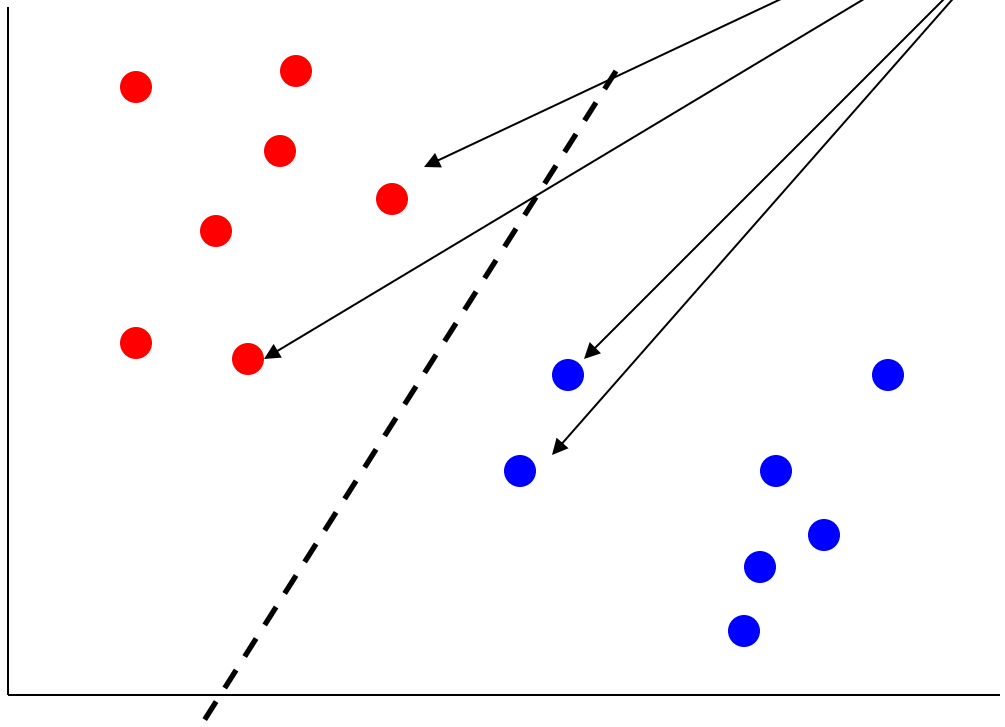
- only a few α_i can be nonzero!!

$$\alpha_i \left(y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right) = 0, \quad i = 1, \ldots, n$$

for most $\alpha_i \Rightarrow \alpha_i = 0$

We call the training data points whose α_i's are nonzero the **support vectors** (SV)

Dual SVM - interpretation



training points

$$w = \sum_i \alpha_i x_i y_i$$

$\alpha_i \geq 0$

For α_i that are 0, no influence

Dual SVM– Testing

Dot product with (“all” ??) training samples

To evaluate a new sample x_{ts} we need to compute:

$$\widehat{y}_{ts} = \text{sign}(\mathbf{w}^T x_{ts} + b) = \text{sign}\left(\sum_{i=1..n} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_{ts} + b\right)$$

$$\Rightarrow \widehat{y}_{ts} = \text{sign}\left(\sum_{i \in SupportVectors} \alpha_i y_i \left(\mathbf{x}_i^T \mathbf{x}_{ts}\right) + b\right)$$

$\alpha_i > 0$

For \alpha_i that are 0, no influence

Support Vectors for the Soft-SVM

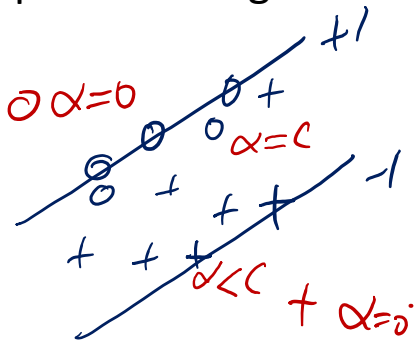
$$\alpha_i \left(y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \varepsilon_i \right) = 0, \quad i = 1, \ldots, n$$

- Support vectors are
  - Samples on the margin:

$$y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) = 1,$$

$$0 < \alpha_i < C$$

  - Samples violating the margin (mostly inside the margin area):



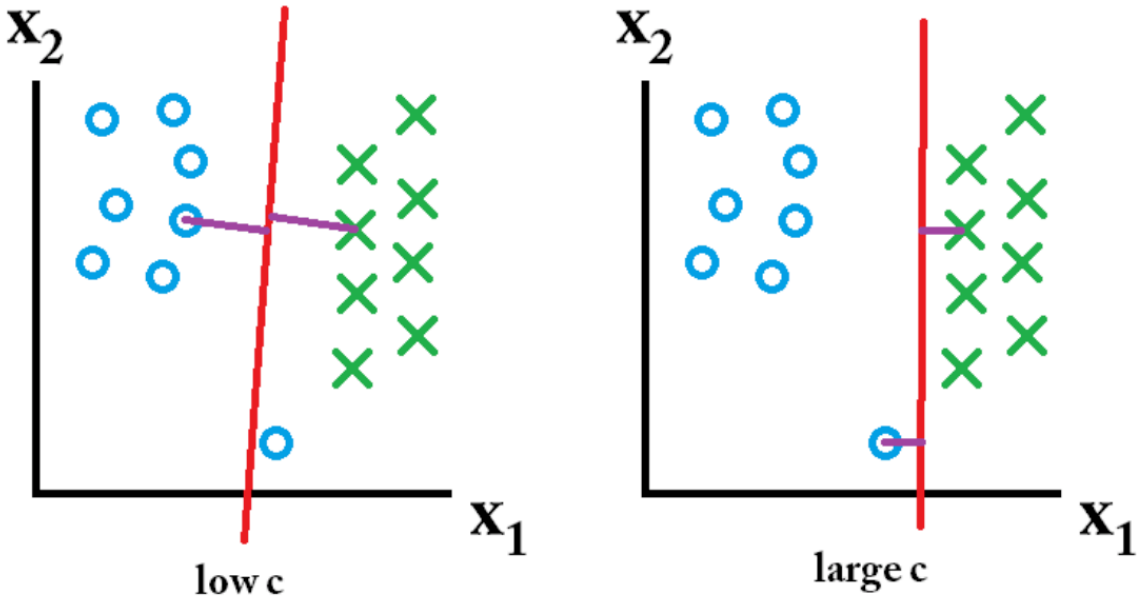
$$y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) < 1,$$

$$\alpha_i = C$$

More in L11Extra-SVMoptimDual

Value C and Number of Support Vectors (no clear relation!!!)

C is how hard we want to punish misclassified examples in the training set.
So for large values of C, it would make sense that if misclassifed examples are severely punished, then it will choose a small margin with not many support vectors.
For small values of C, it would broaden the margin, and as a result, end up getting more points inside of the margin (if not easily separable), so there would be possibly more support vectors.
I thought that this StackExchange post described it pretty well.
As in this example, the small c has more "support vectors" than the large c



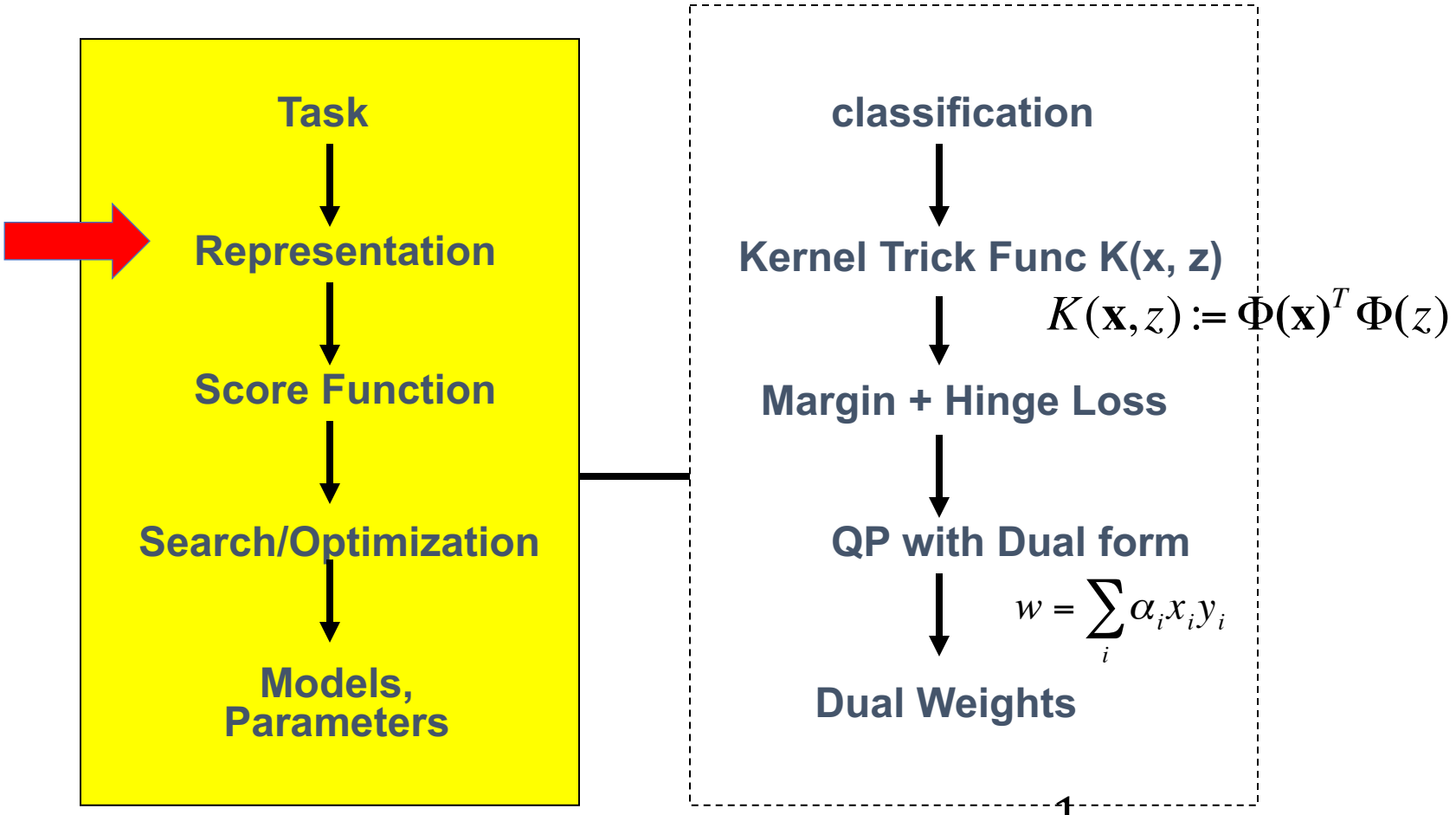
https://github.com/scikit-learn/scikit-learn/issues/7955

https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

Summary of SVM

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide
    - ✓ File format / LIBSVM
    - ✓ Feature preprocsssing
    - ✓ Model selection
    - ✓ Pipeline procedure

Support Vector Machine



$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + \mathrm{C} \sum_{\mathrm{i}=1}^{\mathrm{n}} \varepsilon_{\mathrm{i}}$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1 - \varepsilon_{\mathrm{i}}$

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

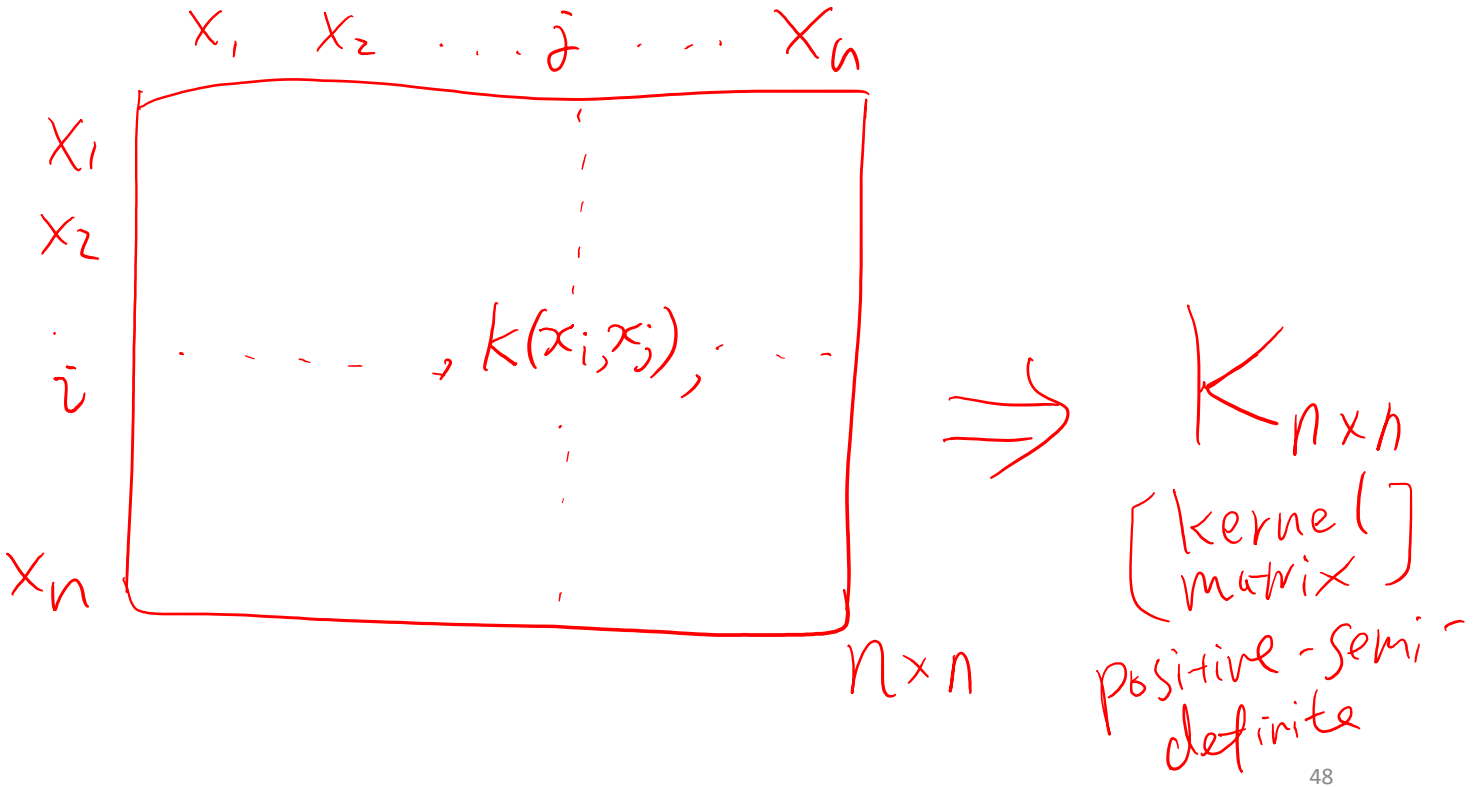
Why do SVMs work?

$X \rightarrow \phi(x)$ e.g. RBF

- If we are using huge features spaces (e.g., with kernels), how come we are not overfitting the data?
  - ✓ Number of parameters remains the same (and most are set to 0) $O(n)$, α_i $i=1, \cdots, n$
  - ✓ While we have a lot of inputs, at the end we only care about the support vectors and these are usually a small group of samples
  - ✓ The maximizing of the margin acts as a sort of regularization term leading to reduced overfitting

Kernel Matrix

- Kernel function creates the kernel matrix, which summarize all the (train) data



Summary: Modification Due to Kernel Trick

- Change all inner products to kernel functions
- For training,

Original Linear

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$C > \alpha_i \geq 0, \forall i \in train$$

With kernel function - nonlinear

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$C > \alpha_i \geq 0, \forall i \in train$$

VC: $\phi(x)$ large

Summary:
Modification Due to Kernel Function

- For testing, the new data **x_ts**

Original Linear

$$\widehat{y_{ts}} = \text{sign}\left(\sum_{\text{i}\in\text{supportVectorn}} \alpha_i \text{y}_\text{i} \mathbf{x_i}^T \mathbf{x}_{ts} + b \right)$$

With kernel function - nonlinear

$$\widehat{y_{ts}} = \text{sign}\left(\sum_{\text{i}\in\text{supportVectors}} \alpha_i \text{y}_\text{i} K(\mathbf{x_i}, \mathbf{x}_{ts}) + b \right)$$

Kernel Trick: Implicit Basis Representation

- For some kernels (e.g. RBF) the implicit transform basis form \phi(**x**) is infinite-dimensional!
  - But calculations with kernel are done in original space, so computational burden and curse of dimensionality aren't a problem.

O(p)

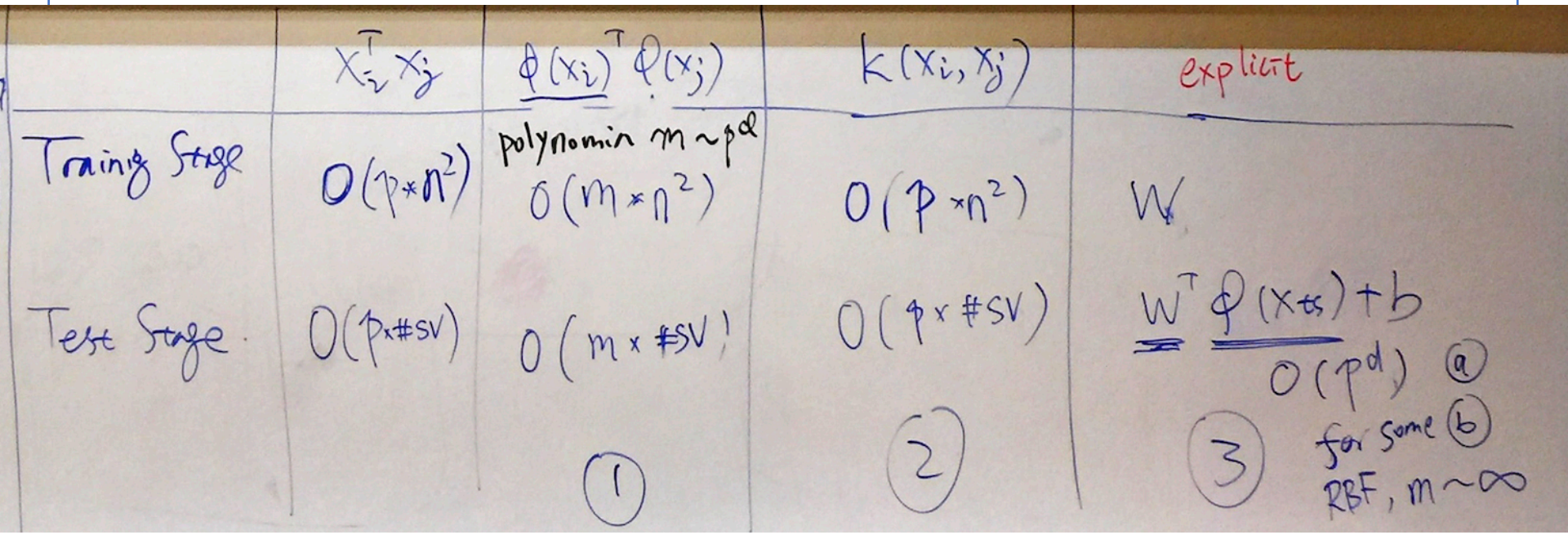
$$K(\mathbf{x},z)=\exp\left(-r\left\|\mathbf{x}-z\right\|^2\right)$$

*O(p*n^2)* operations in building a RBF-kernel matrix for training

➔ Gaussian RBF Kernel corresponds to an infinite-dimensional vector space.

YouTube video of Caltech: Abu-Mostafa explaining this in more detailhttps://www.youtube.com/watch?v=XUj5JbQihlU&t=25m53s

Time Cost Comparisons



Kernel Functions (Extra)

- In practical use of SVM, only the kernel function (and not basis function) is specified

- Kernel function can be thought of as a similarity measure between the input objects

- Not all similarity measure can be used as kernel function, however Mercer's condition states that any positive semi-definite kernel $K(x, y)$, i.e.

$$\sum_{i,j} K(x_i, x_j) c_i c_j \geq 0$$

can be expressed as a dot product in a high dimensional space.

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarize all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, tree kernel, graph kernel, ...)
  - Kernel trick has helped Non-traditional data like strings and trees able to be used as input to SVM, instead of feature vectors
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try for most applications.

Mercer Kernel vs. Smoothing Kernel (Extra)

- The Kernels used in Support Vector Machines are different from the Kernels used in LocalWeighted /Kernel Regression.

- We can think
  - Support Vector Machines' kernels as **Mercer Kernels**
  - Local Weighted / Kernel Regression's kernels as **Smoothing Kernels**

Summary of SVM

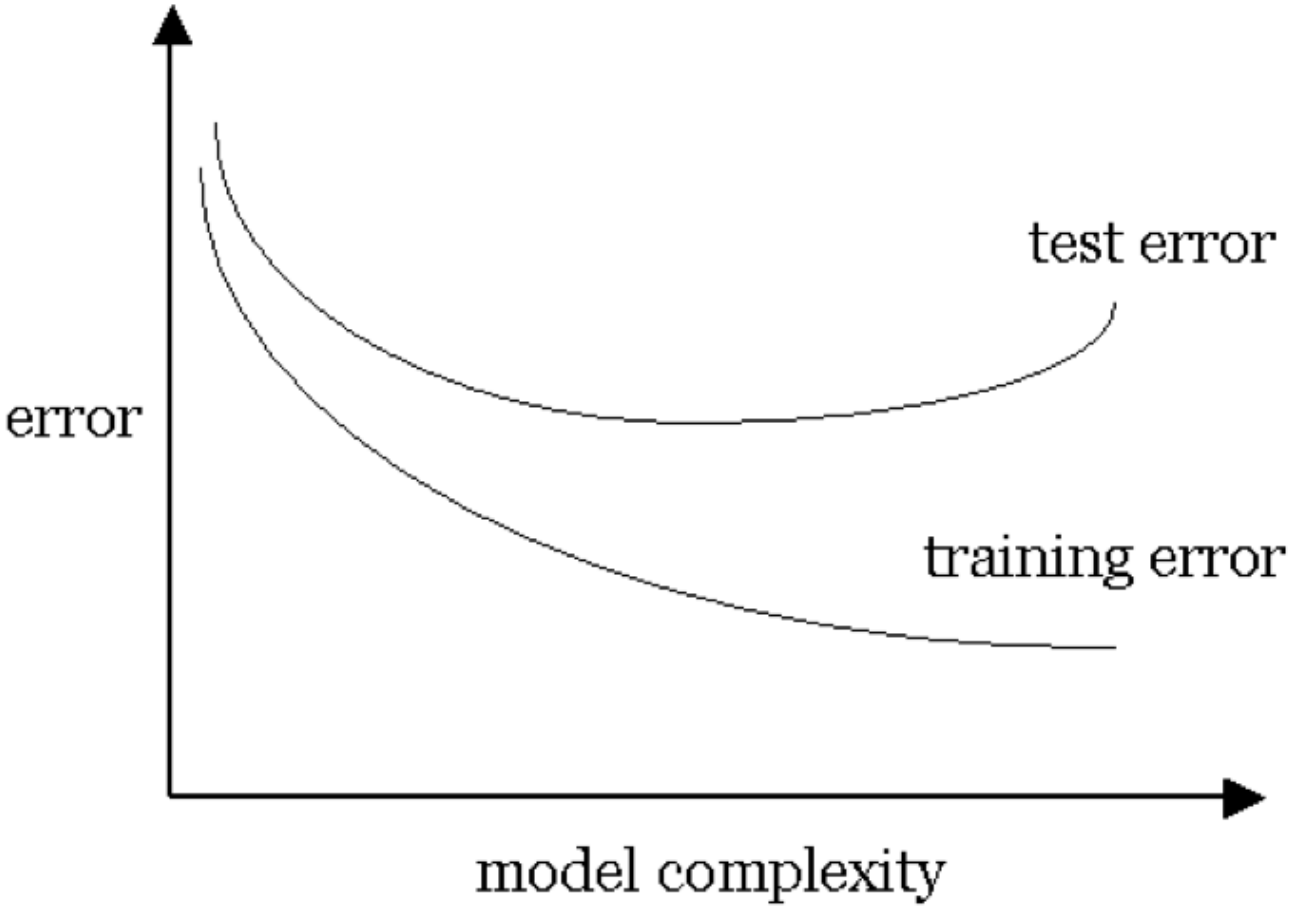
- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide
    - ✓ File format / LIBSVM
    - ✓ Feature preprocsssing
    - ✓ Model selection
    - ✓ Pipeline procedure

Practical Guide to SVM

- From authors of as LIBSVM:
  - A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, 2003-2010
  - http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

(c) Model Selection

Our goal: find the model M which minimizes the test error:

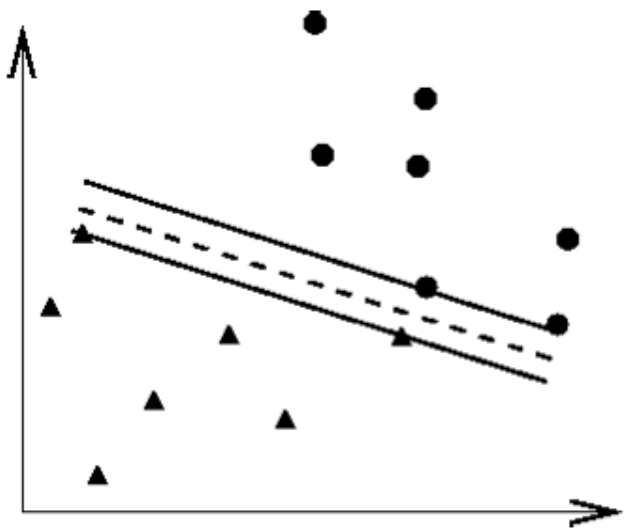


Model Selection, find right C

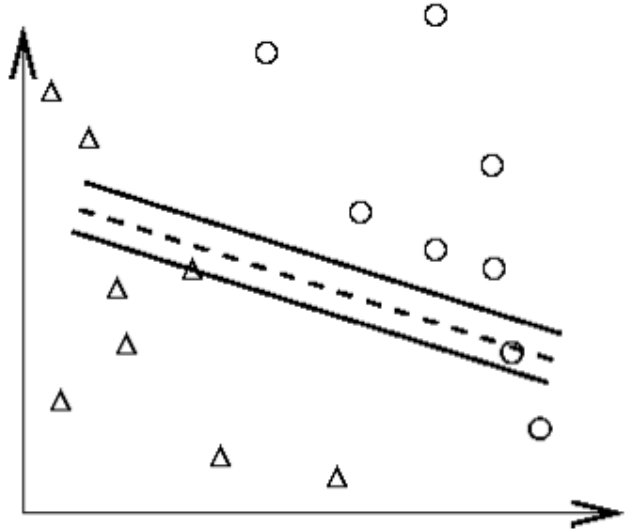
large C

Select the right penalty parameter C

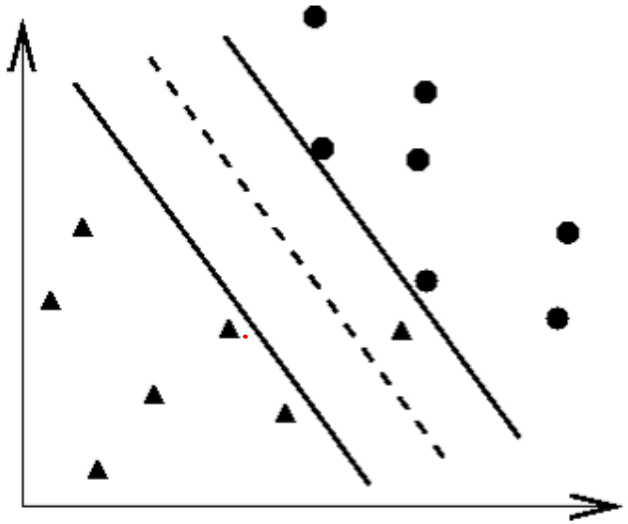
small C



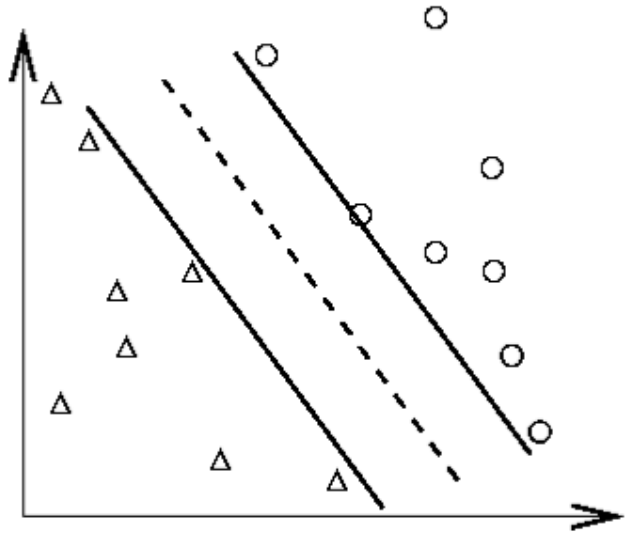
(a) Training data and an overfitting classifier



(b) Applying an overfitting classifier on testing data



(c) Training data and a better classifier



(d) Applying a better classifier on testing data

(c) Model Selection

- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0.$

  two parameters for an RBF kernel: C and γ

- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i{}^T \mathbf{x}_j + r)^d, \gamma > 0.$

  Three parameters for a polynomial kernel

A Practical Guide to Support Vector Classification

(d) Pipeline Procedures

- (I) train / test
- (II) k-folds cross validation
- (III) k-CV on train to choose hyperparameter / then test

Evaluation Choice-III:

Many beginners use the following procedure now:

- Transform data to the format of an SVM package
- Randomly try a few kernels and parameters
- Test

We propose that beginners try the following procedure first:

- Transform data to the format of an SVM package
- Conduct simple scaling on the data
- Consider the RBF kernel $K(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x}-\mathbf{y}\|^2}$
- Use cross-validation to find the best parameter C and γ
- Use the best parameter C and γ to train the whole training set[5]
- Test

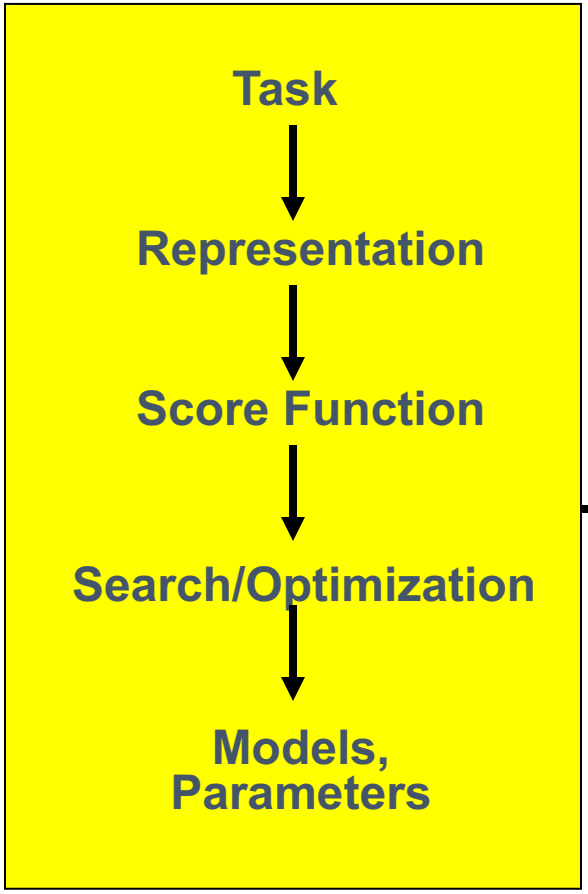
We use option III for HW

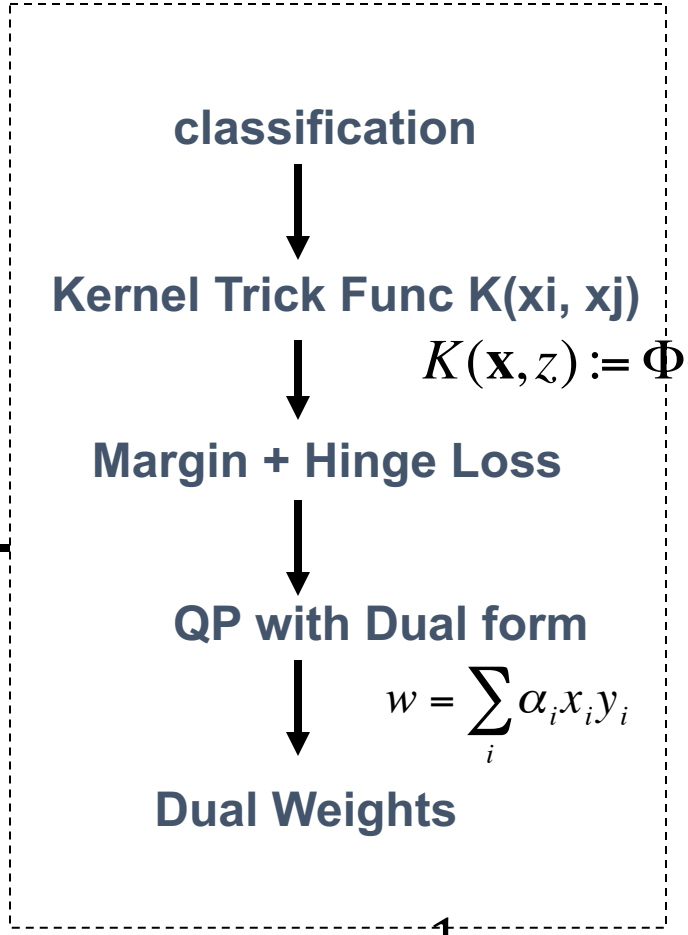
A Practical Guide to Support Vector Classification

Summary of SVM

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide
    - ✓ File format / LIBSVM
    - ✓ Feature preprocsssing
    - ✓ Model selection
    - ✓ Pipeline procedure

Summary: Support Vector Machine





$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + \mathrm{C} \sum_{\mathrm{i}=1}^{\mathrm{n}} \varepsilon_{\mathrm{i}}$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1 - \varepsilon_{\mathrm{i}}$

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

(Later)

kNN: $\hat{y}_{ts} = \frac{1}{k} \sum_{i \in kNeighbors\text{-}of\ x_{ts}} y_i$

find k neighbor of x_{ts} $\sim O(n_x)$

SVM: $\hat{y}_{ts} = \sum_{i \in SV} \alpha_i y_i k(\vec{x}_i, \vec{x}_{ts}) + b$

para $\sim O(n)$

Logistic Regression / Linear Classifier

$\hat{y}_{ts} = \sigma(W^T x_{ts} + b)$

para $\sim O(p)$

Why SVM Works? (Extra)

- Vapnik argues that the fundamental problem is not the number of parameters to be estimated. Rather, the problem is about the flexibility of a classifier

- Vapnik argues that the flexibility of a classifier should not be characterized by the number of parameters, but by the capacity of a classifier
  - This is formalized by the "VC-dimension" of a classifier

- The SVM objective can also be justified by structural risk minimization: the empirical risk (training error), plus a term related to the generalization ability of the classifier, is minimized

- Another view: the SVM loss function is analogous to ridge regression. The term $\frac{1}{2}||w||^2$ "shrinks" the parameters towards zero to avoid overfitting

References

- Big thanks to Prof. Ziv Bar-Joseph and Prof. Eric Xing @ CMU for allowing me to reuse some of his slides
- Elements of Statistical Learning, by Hastie, Tibshirani and Friedman
- Prof. Andrew Moore @ CMU's slides
- Tutorial slides from Dr. Tie-Yan Liu, MSR Asia
- A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, 2003-2010
- Tutorial slides from Stanford "Convex Optimization I — Boyd & Vandenberghe