

# Project Presentations

## Tuesday, 3/1

1. **Group 1** (Hussain, Tanvir; Lewis, Cameron; Villamar, Sandra)
2. **Group 2** (Dong, Meng; Long, Jianzhi; Wen, Bo; Zhang, Haochen)
3. **Group 3** (Chen, Yuzhao; Li, Zonghuan; Song, Yuze; Yan, Ge)
4. **Group 4** (Li, Jiayuan; Xiao, Nan; Yu, Nancy; Zhou, Pei)
5. **Group 5** (Li, Zheng; Tao, Jianyu; Yang, Fengqi)
6. **Group 6** (Bian, Xintong; Jiang, Yufan; Wu, Qiyao)
7. **Group 7** (Chen, Yongxing; Yao, Yanzhi; Zhang, Canwei)
8. **Group 8** (Nukala, Kishore; Pulleti, Sai; Vaidyula, Srikar)

## Thursday, 3/3

1. **Group 9** (Baluja, Michael; Cao, Fangning; Huff, Mikael; Shen, Xuyang)
2. **Group 10** (Arun, Aditya; Long, Heyang; Peng, Haonan)
3. **Group 11** (Cowin, Samuel; Liao, Albert; Mandadi, Sumega)
4. **Group 12** (Jia, Yichen; Jiang, Zhiyun; Li, Zhuofan)
5. **Group 13** (Dandu, Murali; Daru, Srinivas; Pamidi, Sri)
6. **Group 14** (He, Bolin; Huang, Yen-Ting; Wang, Shi; Wang, Tzu-Kao)
7. **Group 15** (Chen, Luobin; Feng, Ruining; Wu, Ximei; Xu, Haoran)

## Tuesday, 3/8

1. **Group 16** (Chen, Rex; Liang, Youwei; Zheng, Xinran)
2. **Group 17** (Aguilar, Matthew; Millhiser, Jacob; O'Boyle, John; Sharpless, Will)
3. **Group 18** (Wang, Haoyu; Wang, Jiawei; Zhang, Yuwei)
4. **Group 19** (Chen, Yinbo; Di, Zonglin; Mu, Jiteng)
5. **Group 20** (Chowdhury, Debalina; He, Scott; Ye, Yiheng)
6. **Group 21** (Lin, Wei-Ru; Ru, Liyang; Zhang, Shaohua)
7. **Group 22** (Bhavsar, Shivad; Blazej, Christopher; Bu, Yinyan; Liu, Haozhe)

## Thursday, 3/10

1. **Group 23** (Chen, Claire; Hsieh, Chia-Wei; Lin, Jui-Yu; Tsai, Ya-Chen)
2. **Group 24** (Cheng, Yu; Yu, Zhaowei; Zaidi, Ali)
3. **Group 25** (Assadi, Parsa; Brugere, Tristan; Pathak, Nikhil; Zou, Yuxin)
4. **Group 28** (Candassamy, Gokulakrishnan; Dixit, Rajeev; Huang, Joyce)
5. **Group 27** (Kok, Hong; Wang, Jacky; Yan, Yijia; Yuan, Zhouyuan)
6. **Group 28** (Luan, Zeting; Yang, Zheng)
7. **Group 29** (Cuawenberghs, Kalyani; Mojtahed, Hamed)

Each presentation will be allocated **9 minutes** (pts will be deducted if you go over 9 minutes)

The **presentation slides** of **ALL GROUPS** are due by **Monday, 2/28 @ 11:59 pm**

Email me the file ([mvasconcelos@eng.ucsd.edu](mailto:mvasconcelos@eng.ucsd.edu)) and name the file **GroupX.pdf**, where **X** is your group number (see previous slide). Use **Group X Presentation** as the subject of your email and cc to all members.

The presentation should discuss the **problem** that you are trying to solve, the **data** that you are using, the **proposed solution(s)**, and the **results** that you have so far (they can later be UPDATED IN THE PROJECT PAPER).

## ECE 271B: Take-Home Quizzes Guidelines

By submitting your quiz solution, you agree to comply with the following.

1. The quiz should be treated as a **take-home test** and be an **INDIVIDUAL** effort. **NO collaboration is allowed.** The submitted work must be yours and must be original.
2. The work that you turn-in to be your own, using the resources that are available to all students in the class.
3. You are not allowed to consult or use resources provided by tutors, previous students in the class, or any websites that provide solutions or help in solving assignments and exams.
4. You will not upload your solutions or any other course materials to any websites or in some other way distribute them outside the class.
5. 0 points will be assigned to any problem that seems to violate these rules and, if recurrent, the incident(s) will be reported to the Academic Integrity Office.

With respect with quiz logistics, you should do the following.

1. Quizzes should be submit in PDF format on Gradescope by **11:59 pm of the due date**. Late submissions will be accepted within 24 hours, but will incur a 20% penalty. After that, there will be no credit.
2. If there are issues that need clarification, feel free to ask on Piazza. However, make sure **not to give the solutions away**. General questions that are not specifically about the problems can, of course, be discussed openly. It follows that if you can frame your question about the problem more generally, you will get a lot more feedback. In general, this also applies to the TAs office hours. If you are stuck in a problem, feel free to go see the TAs. However, TAs will not solve the problem for you. Make sure to ask the question more generally.
3. **Start early** because some problems might need non-trivial amounts of computer time.
4. Unless instructed otherwise, you have to **write all the code** (no packages allowed). If in doubt, ask on Piazza.
5. **All code used to solve the computer problems must be submitted with your quiz.** While we will not be grading code, the TA might need to check it up. If the code is not submitted, 0 points will be assigned to the computer problem.
6. Any request of **quiz regrading** must be submitted on Gradescope **within one week** after the release of the respective graded quiz.
7. Be considerate of the TAs that will be grading your quiz by **submitting a readable PDF document**. Be aware that there is no obligation on the part of the TAs to put effort into deciphering quizzes beyond what is reasonably expected. Typical problems for handwritten documents are: 1) poor handwriting; 2) student writes on both sides of the page and ink bleeds from the back-ground; 3) documents "scanned" by a taking a picture, where there are issues of camera focus or perspective effects that compromise reading; 4) a PDF that is compiled with pages or images upside-down, out of order, or with a skewed perspective. These are issues that severely affect the ability of the TAs to do their job and can be easily avoided with some minimal amount of planning. Now that you are made aware of them, it should be fairly trivial to avoid them. If the TAs are faced with these issues, they can choose not to grade the problem. I give them that discretion.

**Quiz #3 due today @ 11:59pm**

**Quiz #4 posted on Canvas**

**Due date: Tuesday, 3/8**

# **ECE 271B – Winter 2022**

## **The Support Vector Machine**

**Disclaimer:**  
This class will be recorded  
and made available to students asynchronously.

Manuela Vasconcelos  
ECE Department, UCSD

# Today

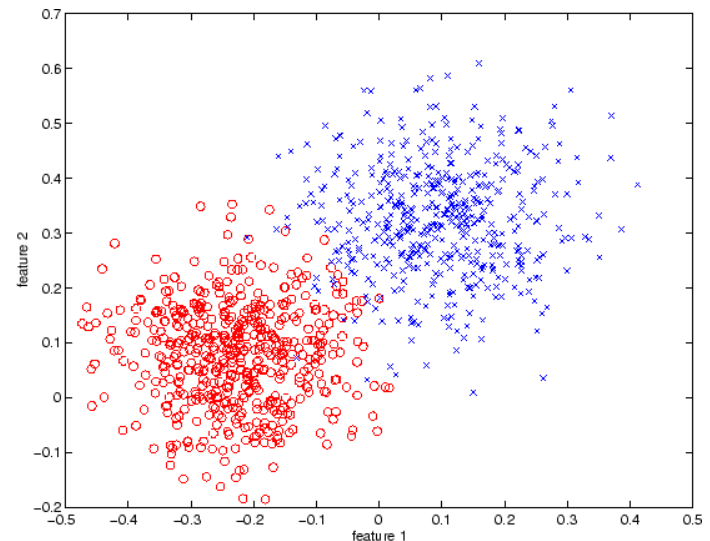
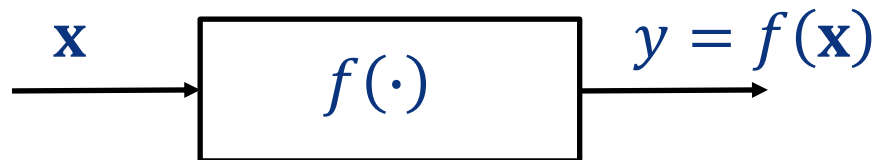
- ▶ we have talked about classification and linear discriminants
- ▶ this led us to the idea that all these learning problems boil down to the solution of an optimization problem
- ▶ we have seen how to solve optimization problems
  - with and without constraints, equalities, inequalities
- ▶ then we did a detour to talk about kernels
  - how do we implement a non-linear boundary on the linear discriminant framework
- ▶ today, we will put everything together by studying the SVM

*Cortes, C. and Vapnik, V.;* Support-Vector Networks. *Machine Learning*, 20 (3): 273–297, 1995.

*Vapnik, V.* The Nature of Statistical Learning Theory. Springer, New York, 1996.

# Classification

- ▶ a **classification problem** has two types of variables
  - $\mathbf{x}$  – vector of **observations** (**features**) in the world
  - $y$  – **state** (**class**) of the world
- ▶ e.g.
  - $\mathbf{x} \in \mathcal{X} \in \mathbb{R}^2 = (\text{fever}, \text{blood pressure})$
  - $y \in \mathcal{Y} = \{\text{disease}, \text{no disease}\}$
- ▶  $\mathbf{x}$ ,  $y$  related by (unknown) **function**



- ▶ **goal**: design a **classifier**  $h: \mathcal{X} \rightarrow \mathcal{Y}$  such that  $h(\mathbf{x}) = f(\mathbf{x}), \forall \mathbf{x}$

# Linear Discriminant

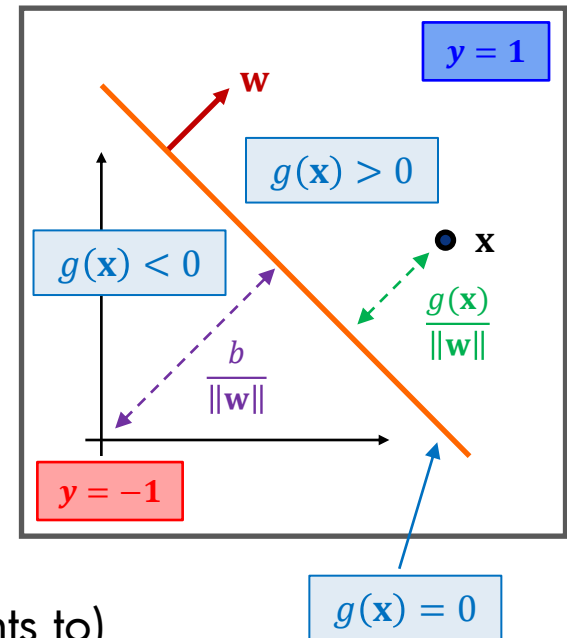
- ▶ the **classifier** implements the **linear decision rule**

$$h(\mathbf{x}) = \begin{cases} 1, & \text{if } g(\mathbf{x}) > 0 \\ -1, & \text{if } g(\mathbf{x}) < 0 \end{cases} = \text{sgn}[g(\mathbf{x})]$$

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

and has the **properties**

- it divides  $\mathcal{X}$  into two “half-planes”
- boundary is the **plane** with
  - **normal**  $\mathbf{w}$
  - distance to the origin  $b/\|\mathbf{w}\|$
- $g(\mathbf{x})/\|\mathbf{w}\|$  is the **distance from point  $\mathbf{x}$  to the boundary**
  - $g(\mathbf{x}) = 0$  for **points on the plane**
  - $g(\mathbf{x}) > 0$  on the “**positive side**” (side  $\mathbf{w}$  points to)
  - $g(\mathbf{x}) < 0$  on the “**negative side**”



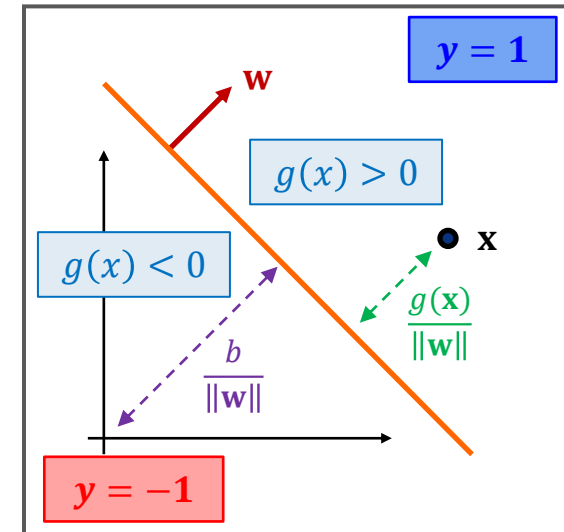
# Linear Discriminant

► given a linearly separable training set  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$

► no errors if and only if,  $\forall i$

- $y_i = 1$  and  $g(\mathbf{x}_i) > 0$  or  
 $y_i = -1$  and  $g(\mathbf{x}_i) < 0$
- i.e.  $y_i g(\mathbf{x}_i) > 0$

► this allows a very concise expression for the situation of “no training error” or “zero empirical risk”



► given a training set  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ , the necessary and sufficient condition to have **zero empirical risk** is

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) > 0, \forall i$$

# The Margin

- ▶ the **margin** is the distance from the boundary to the closest point

$$\gamma = \min_i \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

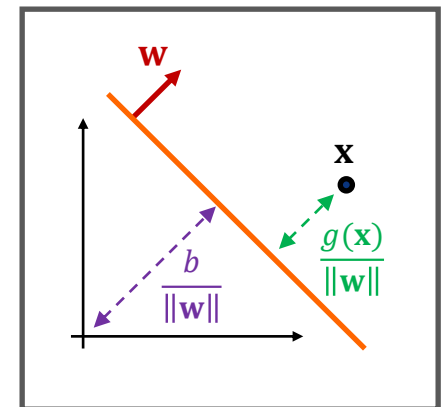
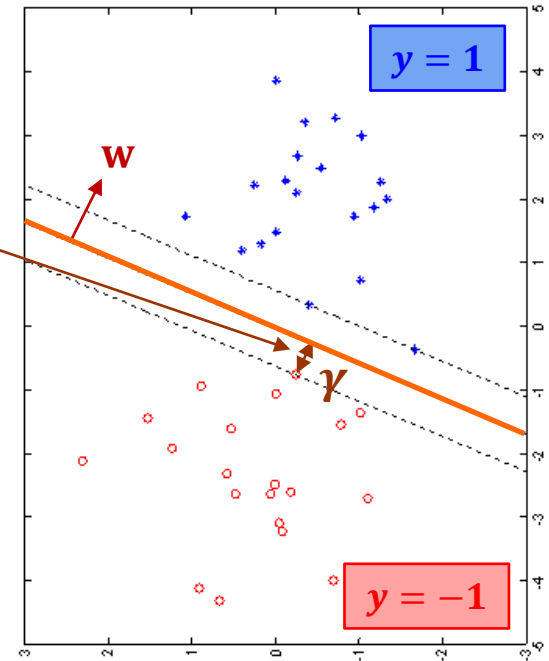
- ▶ among all planes such that

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0, \forall i$$

there will be no error if it is strictly greater than zero

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) > 0, \forall i \iff \gamma > 0$$

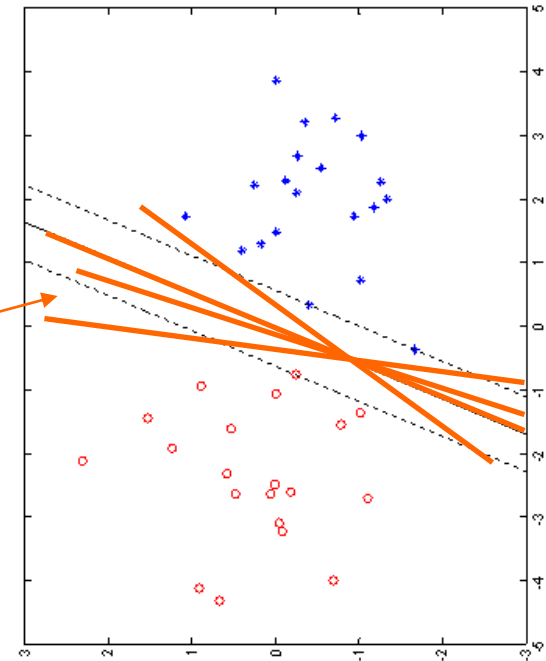
- ▶ note that this is ill-defined in the sense that  $\gamma$  does not change if both  $\mathbf{w}$  and  $b$  are scaled by  $\lambda \rightarrow$  we need a **normalization**





# Maximizing the Margin

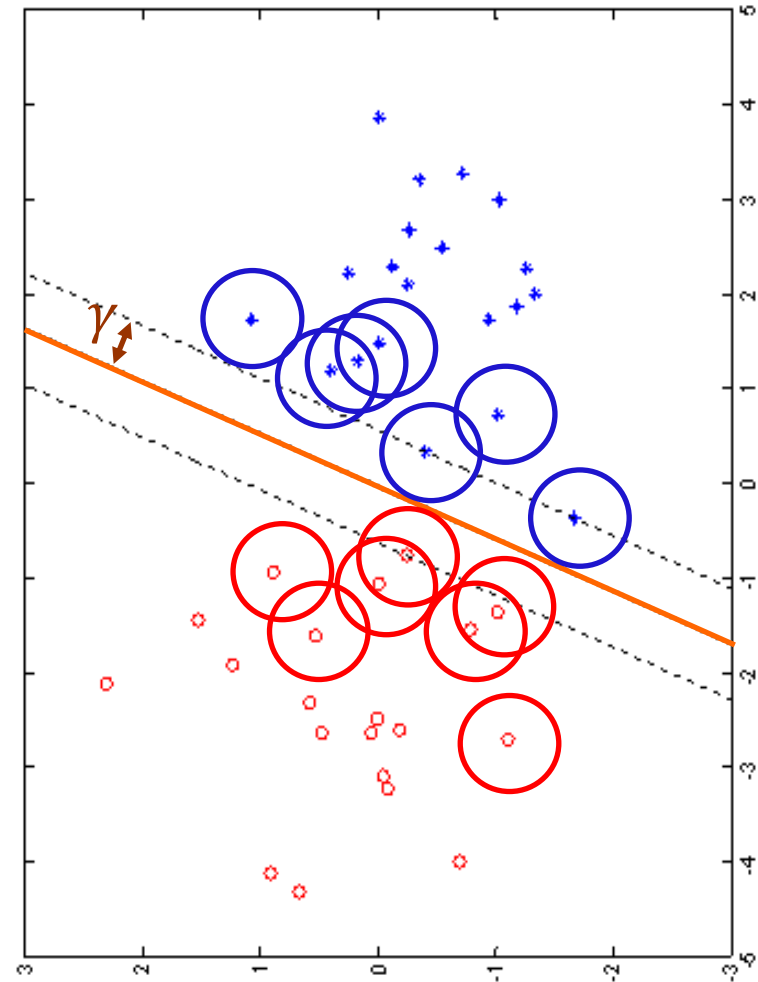
- ▶ let's **assume** we have selected some **normalization**, e.g.  $\|\mathbf{w}\| = 1$
- ▶ the next question is: what is the cost that we are going to optimize?
- ▶ there are **several** planes that separate the classes, which one is **best plane**?
- ▶ recall that in the case of the Perceptron, we have seen that **the margin** determines the **complexity** of the learning problem
  - the Perceptron converges in less than  $(k/\gamma)^2$  iterations
- ▶ it sounds like **maximizing the margin** is a good idea



# Maximizing the Margin

## ► intuition1

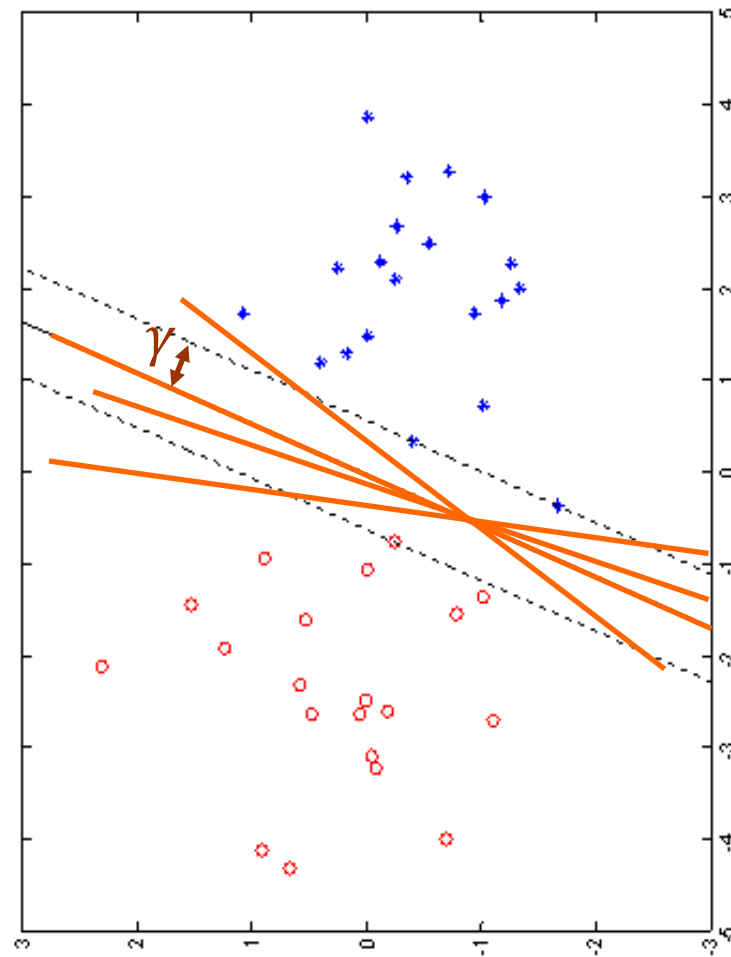
- think of **each point** in the training set as a sample from a **probability density** centered on it
- if we draw another sample, we will not get the same points
- each point is really a pdf with a certain variance
- if we leave a **margin of  $\gamma$**  on the training set, we are safe against this **uncertainty** (as long as the support of the pdf is smaller than  $\gamma$ )
- the larger the  $\gamma$ , the **more robust** the classifier!



# Maximizing the Margin

## ► intuition2

- think of the **plane** as an **uncertain estimate** because it is learned from a sample drawn at random
- since the sample changes from draw to draw, the **plane** parameters are **random variables of non-zero variance**
- instead of a **single plane** we have a **probability distribution over planes**
- the larger the **margin  $\gamma$** , the larger the number of **planes** that will **not** originate errors
- the **larger** the  $\gamma$ , the **larger** the **variance** allowed on the plane parameter estimates!



# Normalization

$$\gamma = \min_i \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

► now, let's look at the **normalization**

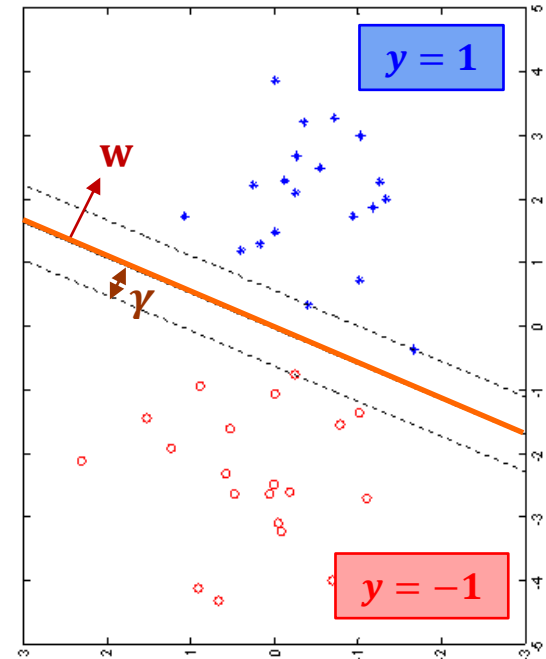
► natural choice is  $\|\mathbf{w}\| = 1$

► the **margin is maximized** by solving

$$\begin{aligned} \max_{\mathbf{w}, b} \min_i |\mathbf{w}^T \mathbf{x}_i + b| \\ \text{subject to } \|\mathbf{w}\|^2 = 1 \end{aligned}$$

► this is **somewhat complex**

- need to find the **points that achieve the minimum without** knowing what  $\mathbf{w}$  and  $b$  are
- optimization problem with **quadratic constraints**



# Normalization

$$\gamma = \min_i \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

- ▶ a more convenient **normalization** is to make  $|g(\mathbf{x})| = 1$  for the **closest point** i.e.

$$\min_i |\mathbf{w}^T \mathbf{x}_i + b| = 1$$

for which

$$\gamma = \frac{1}{\|\mathbf{w}\|}$$

- ▶ we have seen that the **empirical error is zero** iff  
(Lecture 6)

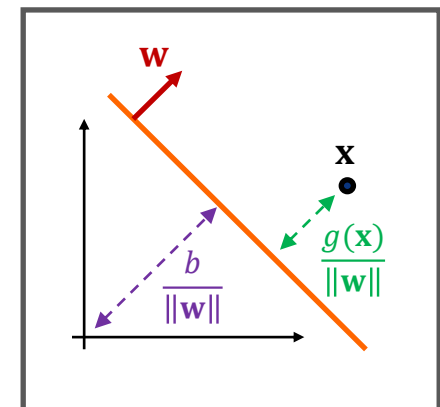
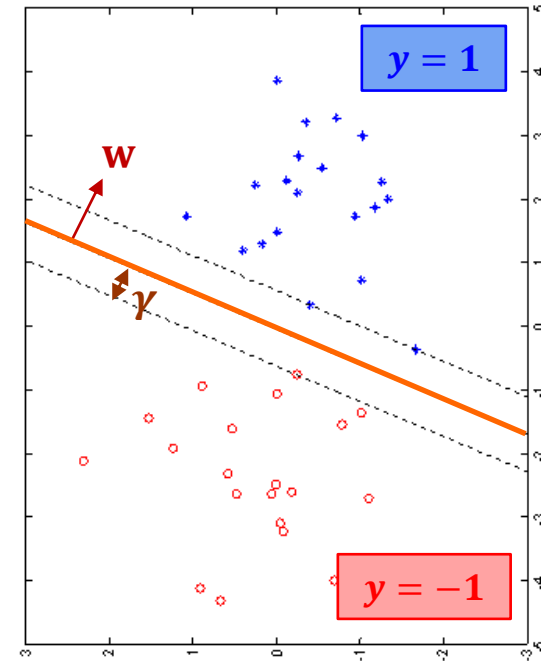
$$y_i(\mathbf{w}^T \mathbf{x}_i + b) > 0, \forall i$$

and to also satisfy  $|\mathbf{w}^T \mathbf{x}_i + b| \geq 1, \forall i$ , we need

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i$$

- ▶ the **SVM** is the classifier that **maximizes the margin** under these constraints

$$\min_{\mathbf{w}, b} \|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i$$



# The Support Vector Machine

$$\min_{\mathbf{w}, b} \|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i$$

► last class, we saw that

► **Theorem: (strong duality)** Consider the problem

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{e}_j^T \mathbf{x} - d_j \leq 0$$

where  $\mathcal{X}$  and  $f$  are convex, and the optimal value  $f^*$  is finite. Then, there is at least **one Lagrange multiplier** vector and there is **no duality gap**.

► this means the **SVM problem can be solved by solving its dual**

# SVM: The Dual Problem

- ▶ for the primal

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i$$

- ▶ the dual problem is

$$\max_{\alpha \geq 0} q(\alpha)$$

$$q(\alpha) = \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$$

where the Lagrangian is

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i \alpha_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

- ▶ setting derivatives to zero w.r.t. to the primal variables  $\mathbf{w}$  and  $b$

$$\nabla_{\mathbf{w}} L = 0 \Leftrightarrow \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0 \Leftrightarrow \mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\nabla_b L = 0 \Leftrightarrow \sum_i \alpha_i y_i = 0$$

# SVM: The Dual Problem

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i \alpha_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

► plugging back

$$\mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\sum_i \alpha_i y_i = 0$$

we get the Lagrangian

$$\begin{aligned} L(\mathbf{w}^*, b, \alpha) &= \frac{1}{2} \|\mathbf{w}^*\|^2 - \sum_i \alpha_i [y_i(\mathbf{w}^{*T} \mathbf{x}_i + b) - 1] \\ &= \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \underbrace{\sum_i \alpha_i y_i b}_0 + \sum_i \alpha_i \\ &= -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i \end{aligned}$$



# SVM: The Dual Problem

$$\max_{\alpha \geq 0} q(\alpha)$$

$$q(\alpha) = \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$$

$$L(\mathbf{w}^*, b, \alpha) = -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i$$

► and the **dual problem** is

$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i \right\} \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0$$

► once this is solved, the **vector**

$$\mathbf{w}^* = \sum_i \alpha_i^* y_i \mathbf{x}_i$$

is the **normal** to the **maximum margin plane**

► **note:** the dual solution does not determine the **optimal**  $b^*$  since  $b$  drops off when we take derivatives

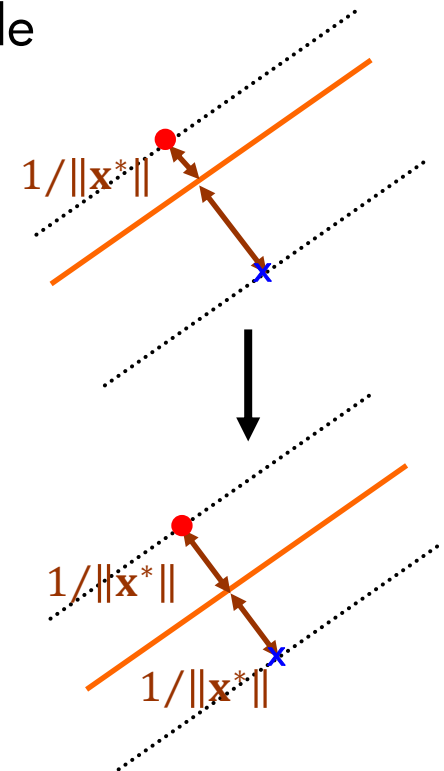
# SVM: The Dual Problem

- ▶ there are various possibilities for **determining**  $b^*$
- ▶ a simple one is to do the following
  - pick one point  $\mathbf{x}^+$  on the margin on the  $y = 1$  side and one point  $\mathbf{x}^-$  on the  $y = -1$  side
  - use the **margin constraint** ( $|\mathbf{w}^T \mathbf{x}_i + b| = 1$ )

$$\left. \begin{array}{l} \mathbf{w}^T \mathbf{x}^+ + b = 1 \\ \mathbf{w}^T \mathbf{x}^- + b = -1 \end{array} \right\} \Leftrightarrow \boxed{b^* = -\frac{\mathbf{w}^{*T} (\mathbf{x}^+ + \mathbf{x}^-)}{2}}$$

▶ **note:**

- the **maximum margin** solution guarantees that there is always at least one point “on the margin” on each side
- if **not**, we could move the plane and get a larger margin



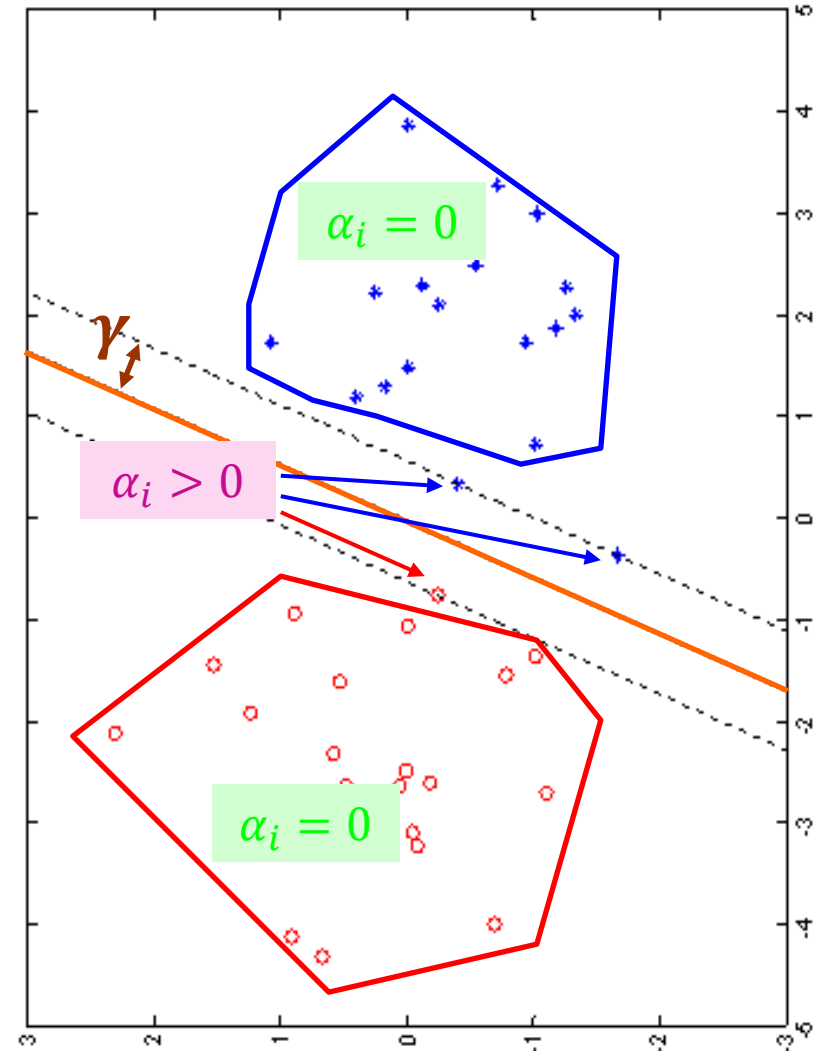
$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i$$

# Support Vectors

- ▶ from the KKT conditions, a **active** (**inactive**) constraint has **non-zero** (**zero**) Lagrange multiplier  $\alpha_i$
- ▶ so,
  - $\alpha_i > 0$  and  $y_i(\mathbf{w}^{*T} \mathbf{x}_i + b^*) = 1$
  - or
  - $\alpha_i = 0$  and  $y_i(\mathbf{w}^{*T} \mathbf{x}_i + b^*) > 1$
- ▶ hence  $\alpha_i > 0$  only for points

$$|\mathbf{w}^{*T} \mathbf{x}_i + b^*| = 1$$

which are those that **lie at a distance equal to the margin**



# Support Vectors

- points with  $\alpha_i > 0$  “support” the optimal hyperplane ( $\mathbf{w}^*, b^*$ ) and, for this, they are called

support vectors

- note that the decision rule is

$$f(\mathbf{x}) = \text{sgn}[\mathbf{w}^{*T} \mathbf{x} + b^*]$$

$$\mathbf{w}^* = \sum_i \alpha_i^* y_i \mathbf{x}_i = \text{sgn} \left[ \sum_i y_i \alpha_i^* \mathbf{x}_i^T \left( \mathbf{x} - \frac{\mathbf{x}^+ + \mathbf{x}^-}{2} \right) \right]$$

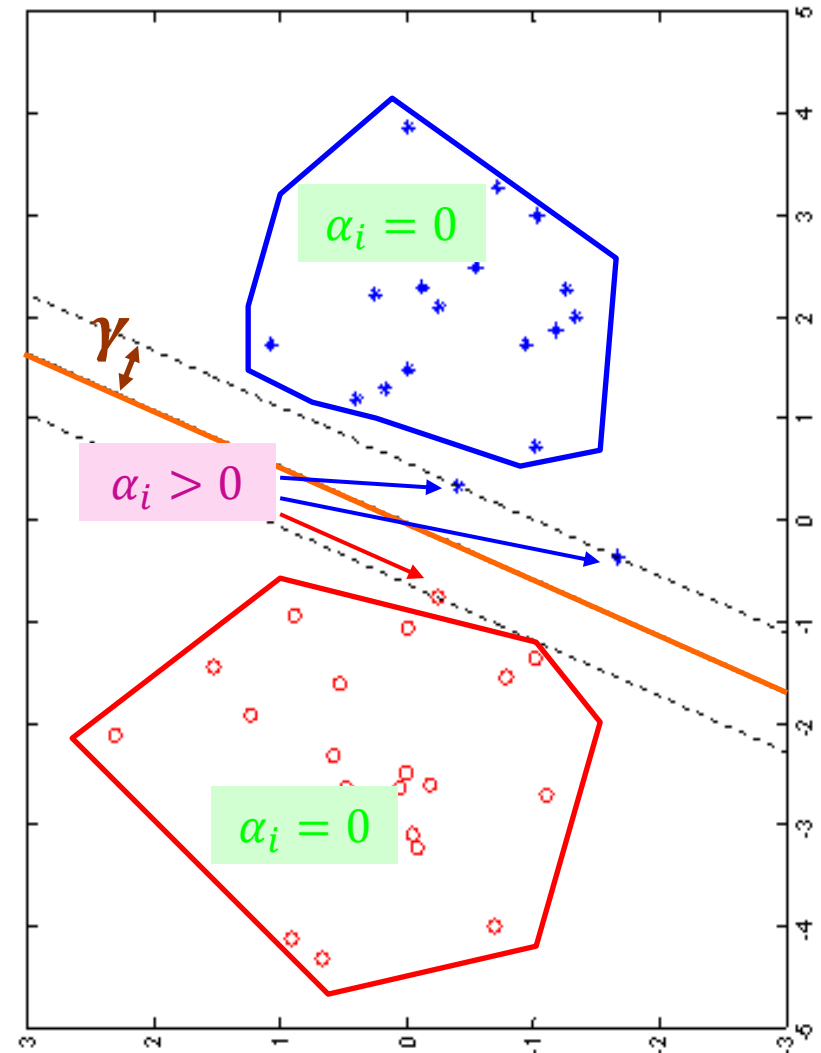
$$b^* = -\frac{\mathbf{w}^{*T}(\mathbf{x}^+ + \mathbf{x}^-)}{2}$$

$$= \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* \mathbf{x}_i^T \left( \mathbf{x} - \frac{\mathbf{x}^+ + \mathbf{x}^-}{2} \right) \right]$$

where

$$SV = \{i \mid \alpha_i^* > 0\}$$

is the set of support vectors



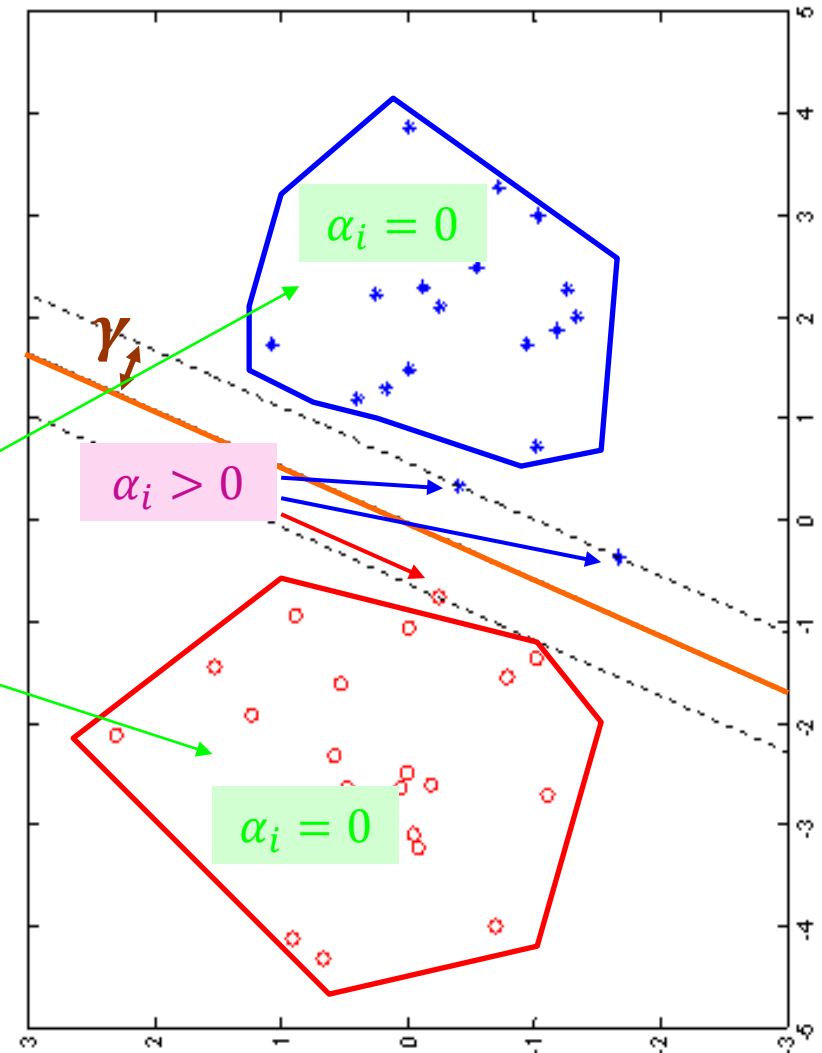
# Support Vectors

- ▶ since the decision rule is

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in \text{SV}} y_i \alpha_i^* \mathbf{x}_i^T \left( \mathbf{x} - \frac{\mathbf{x}^+ + \mathbf{x}^-}{2} \right) \right]$$

we only need the **support vectors** to completely define the classifier

- we can literally throw away all other points!
- ▶ the Lagrange multipliers can also be seen as a measure of importance of each point
  - points with  $\alpha_i = 0$  have no influence: small perturbation does not change the solution



# Perceptron Learning

- note the similarities with the dual Perceptron

```
set  $\alpha_i = 0, b = 0$ 
set  $R = \max_i \|\mathbf{x}_i\|$ 
do {
  for  $i = 1:n$  {
    if  $y_i(\sum_{j=1}^n \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b) < 0$  then {
      •  $\alpha_i = \alpha_i + 1$ 
      •  $b = b + \eta y_i R^2$ 
    }
  }
} until no errors
```

for the dual Perceptron,

- $\alpha_i = 0$  means that the point was never misclassified
- this means that we have an “easy” point, far from the boundary
- very unlikely to happen for a support vector
- in any case, keep in mind that the Perceptron does not maximize the margin!

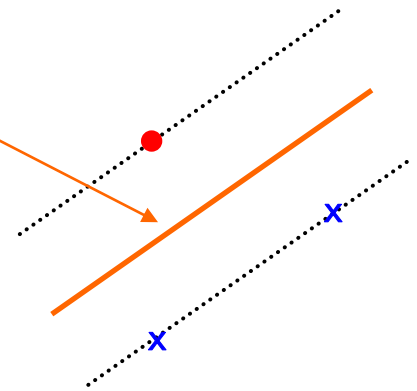
# The Robustness of SVMs

- ▶ in 271A, you talked a lot about the “**curse of dimensionality**”  
number of examples required to achieve certain precision is exponential in the number of dimensions
- ▶ it turns out that **SVMs** are remarkably robust to dimensionality
  - not uncommon to see successful applications on 10,000D+ spaces
- ▶ two main reasons for this:

1) all that the SVM does is to learn a **plane**

although the **number of dimensions** may be **large**, the number of parameters is relatively **small** and there is no much room for overfitting

in fact,  $d + 1$  points are **enough** to specify the decision rule in  $\mathbb{R}^d$ !



# SVMs as Feature Selectors

- ▶ the second reason is that the space is not really that large

## 2) the SVM is a feature selector

to see this, let's look at the decision function

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* \mathbf{x}_i^T \mathbf{x} + b^* \right]$$

this is a **thresholding** of the quantity

$$\sum_{i \in SV} y_i \alpha_i^* \mathbf{x}_i^T \mathbf{x}$$

and note that each of the terms  $\mathbf{x}_i^T \mathbf{x}$  is the projection of the vector to classify ( $\mathbf{x}$ ) into the training (support) vector  $\mathbf{x}_i$

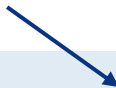


# SVMs as Feature Selectors

- ▶ defining  $\mathbf{z}$  as the vector of the projection onto all support vectors

$$\mathbf{z}(\mathbf{x}) = (\mathbf{x}^T \mathbf{x}_{i_1}, \dots, \mathbf{x}^T \mathbf{x}_{i_k})^T$$

- ▶ the decision function is a plane in the  $\mathbf{z}$  – space

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* \mathbf{x}_i^T \mathbf{x} + b^* \right] = \text{sgn} \left[ \sum_k w_k^* z_k(\mathbf{x}) + b^* \right]$$


with

$$\mathbf{w}^* = (\alpha_{i_1}^* y_{i_1}, \dots, \alpha_{i_k}^* y_{i_k})^T$$

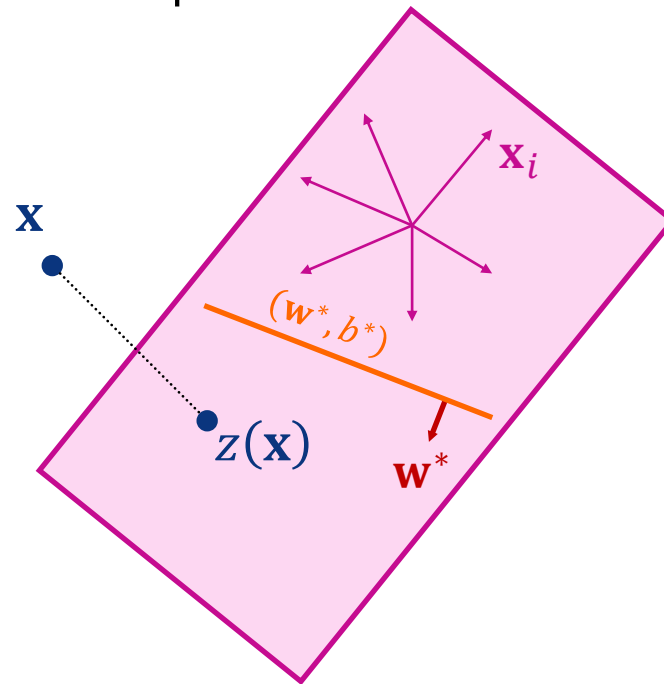
- ▶ this means that

- the classifier operates on the **span of the support vectors**
- the **SVM** performs **feature selection** automatically

# SVMs as Feature Selectors

► geometrically, we have:

- 1) projection on the span of the support vectors
- 2) **classifier** on this space



$$\mathbf{z}(\mathbf{x}) = (\mathbf{x}^T \mathbf{x}_{i_1}, \dots, \mathbf{x}^T \mathbf{x}_{i_k})^T$$

$$\mathbf{w}^* = (\alpha_{i_1}^* y_{i_1}, \dots, \alpha_{i_k}^* y_{i_k})^T$$

$$SV = \{i \mid \alpha_i^* > 0\}$$

- the effective dimension is  $|SV|$  and, typically,  $|SV| \ll n$

# In Summary

## ► SVM training

1) solve the **optimization problem**

$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i \right\} \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0$$

2) then **compute**

$$\mathbf{w}^* = \sum_{i \in SV} \alpha_i^* y_i \mathbf{x}_i$$

$$b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (\mathbf{x}_i^T \mathbf{x}^+ + \mathbf{x}_i^T \mathbf{x}^-)$$

## ► decision function

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* \mathbf{x}_i^T \mathbf{x} + b^* \right]$$

# SVM: Practical Implementations

- ▶ in practice, we need an algorithm for solving the **optimization problem** of the training stage
  - this is still a **complex** problem
  - there has been a **large amount** of research in this area
  - coming up with “your own” algorithm is not going to be competitive
  - luckily, there are **various packages** available, e.g.
    - libSVM: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>
    - SVM light: [http://www.cs.cornell.edu/People/tj/svm\\_light/](http://www.cs.cornell.edu/People/tj/svm_light/)
    - SVM fu: <http://five-percent-nation.mit.edu/SvmFu/>
    - various others (see <http://www.support-vector.net/software.html>)
  - also, **many papers** and **books** on algorithms

# SVM: Kernelization

- note that all equations depend only on  $\mathbf{x}_i^T \mathbf{x}_j$
- the “kernel trick” is **trivial**: replace by  $K(\mathbf{x}_i, \mathbf{x}_j)$

$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i \right\}$$

subject to  $\sum_i \alpha_i y_i = 0$

$$b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (\mathbf{x}_i^T \mathbf{x}^+ + \mathbf{x}_i^T \mathbf{x}^-)$$

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* \mathbf{x}_i^T \mathbf{x} + b^* \right]$$

## 1) training

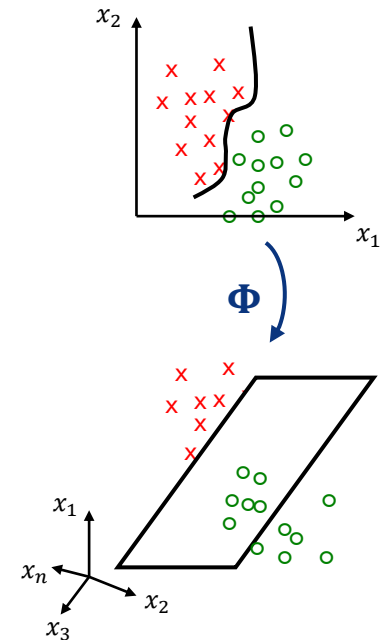
$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_i \alpha_i \right\}$$

subject to  $\sum_i \alpha_i y_i = 0$

$$b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (K(\mathbf{x}_i, \mathbf{x}^+) + K(\mathbf{x}_i, \mathbf{x}^-))$$

## 2) decision function

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) + b^* \right]$$



# SVM: Kernelization

## ► notes:

- as usual this follows from the fact that **nothing** of what we did really **requires** us to be in input-space  $\mathbb{R}^d$  ( $d = \dim[\mathcal{X}]$ )
- we could have **simply** used the notation  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$  for the dot product and all the equations would still **hold**
- the **only difference** is that we can **no longer** recover  $\mathbf{w}^*$  **explicitly** without determining the feature transformation  $\Phi$ , since

$$\mathbf{w}^* = \sum_{i \in SV} \alpha_i^* y_i \Phi(\mathbf{x}_i)$$

- this could have **infinite** dimension  
(e.g., this is a sum of Gaussians when we use the Gaussian kernel)
- but, **luckily**, we **do not really need**  $\mathbf{w}^*$ , **only** the decision function

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) + b^* \right]$$

# Input–Space Interpretation

- ▶ when we introduce a kernel, what is the SVM doing in the input–space?
- ▶ let's look again at the **decision function**

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) + b^* \right]$$

with

$$b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (K(\mathbf{x}_i, \mathbf{x}^+) + K(\mathbf{x}_i, \mathbf{x}^-))$$

- ▶ note that
  - $\mathbf{x}^+$  and  $\mathbf{x}^-$  are **support vectors**
  - assuming that the kernel has reduced support when compared to the distance between support vectors 

next slide

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) + b^* \right]$$

# Input–Space Interpretation

- assuming that the kernel has reduced support when compared to the distance between support vectors

$$\begin{aligned} b^* &= -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (K(\mathbf{x}_i, \mathbf{x}^+) + K(\mathbf{x}_i, \mathbf{x}^-)) \\ &\approx -\frac{1}{2} [\alpha_+^* K(\mathbf{x}^+, \mathbf{x}^+) - \alpha_-^* K(\mathbf{x}^-, \mathbf{x}^-)] \\ &\approx 0 \end{aligned}$$

where we have also assumed that  $\alpha_+ \approx \alpha_-$

note: these assumptions are not crucial, but simplify what follows

- namely the **decision function** is

$$f(\mathbf{x}) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) \right]$$



# Input–Space Interpretation

$$f(\mathbf{x}) = \operatorname{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) \right] \quad \text{or}$$

$$f(\mathbf{x}) = \begin{cases} 1, & \sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) \geq 0 \\ -1, & \sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) < 0 \end{cases}$$

► rewriting

$$\sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) = \sum_{i \in SV | y_i \geq 0} \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) - \sum_{i \in SV | y_i < 0} \alpha_i^* K(\mathbf{x}_i, \mathbf{x})$$

► this is

$$f(\mathbf{x}) = \begin{cases} 1, & \sum_{i \in SV | y_i \geq 0} \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) \geq \sum_{i \in SV | y_i < 0} \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) \\ -1, & \text{otherwise} \end{cases}$$

# Input–Space Interpretation

$$f(\mathbf{x}) = \begin{cases} 1, & \sum_{i \in SV|y_i \geq 0} \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) \geq \sum_{i \in SV|y_i < 0} \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) \\ -1, & \text{otherwise} \end{cases}$$

► or

$$f(\mathbf{x}) = \begin{cases} 1, & \frac{1}{\sum_{i \in SV|y_i < 0} \alpha_i^*} \sum_{i \in SV|y_i \geq 0} \pi_i^* K(\mathbf{x}_i, \mathbf{x}) \geq \frac{1}{\sum_{i \in SV|y_i \geq 0} \alpha_i^*} \sum_{i \in SV|y_i < 0} \beta_i^* K(\mathbf{x}_i, \mathbf{x}) \\ -1, & \text{otherwise} \end{cases}$$

with

$$\pi_i^* = \frac{\alpha_i^*}{\sum_{i \in SV|y_i \geq 0} \alpha_i^*}, i|y_i \geq 0$$

$$\beta_i^* = \frac{\alpha_i^*}{\sum_{i \in SV|y_i < 0} \alpha_i^*}, i|y_i < 0$$

► which is the same as

$$f(\mathbf{x}) = \begin{cases} 1, & \frac{\sum_{i \in SV|y_i \geq 0} \pi_i^* K(\mathbf{x}_i, \mathbf{x})}{\sum_{i \in SV|y_i < 0} \beta_i^* K(\mathbf{x}_i, \mathbf{x})} \geq \frac{\sum_{i \in SV|y_i < 0} \alpha_i^*}{\sum_{i \in SV|y_i \geq 0} \alpha_i^*} \\ -1, & \text{otherwise} \end{cases}$$

# Input–Space Interpretation

$$f(\mathbf{x}) = \begin{cases} 1, & \frac{\sum_{i \in SV|y_i \geq 0} \pi_i^* K(\mathbf{x}_i, \mathbf{x})}{\sum_{i \in SV|y_i < 0} \beta_i^* K(\mathbf{x}_i, \mathbf{x})} \geq \frac{\sum_{i \in SV|y_i < 0} \alpha_i^*}{\sum_{i \in SV|y_i \geq 0} \alpha_i^*} \\ -1, & \text{otherwise} \end{cases}$$

► note that this is the **Bayesian decision rule (BDR)** for

1) **class 1** with **likelihood**

$$\sum_{i \in SV|y_i \geq 0} \pi_i^* K(\mathbf{x}_i, \mathbf{x})$$

and **prior**

$$\sum_{i \in SV|y_i \geq 0} \alpha_i^* / \sum_{i \in SV} \alpha_i^*$$

2) **class 2** with **likelihood**

$$\sum_{i \in SV|y_i < 0} \beta_i^* K(\mathbf{x}_i, \mathbf{x})$$

and **prior**

$$\sum_{i \in SV|y_i < 0} \alpha_i^* / \sum_{i \in SV} \alpha_i^*$$

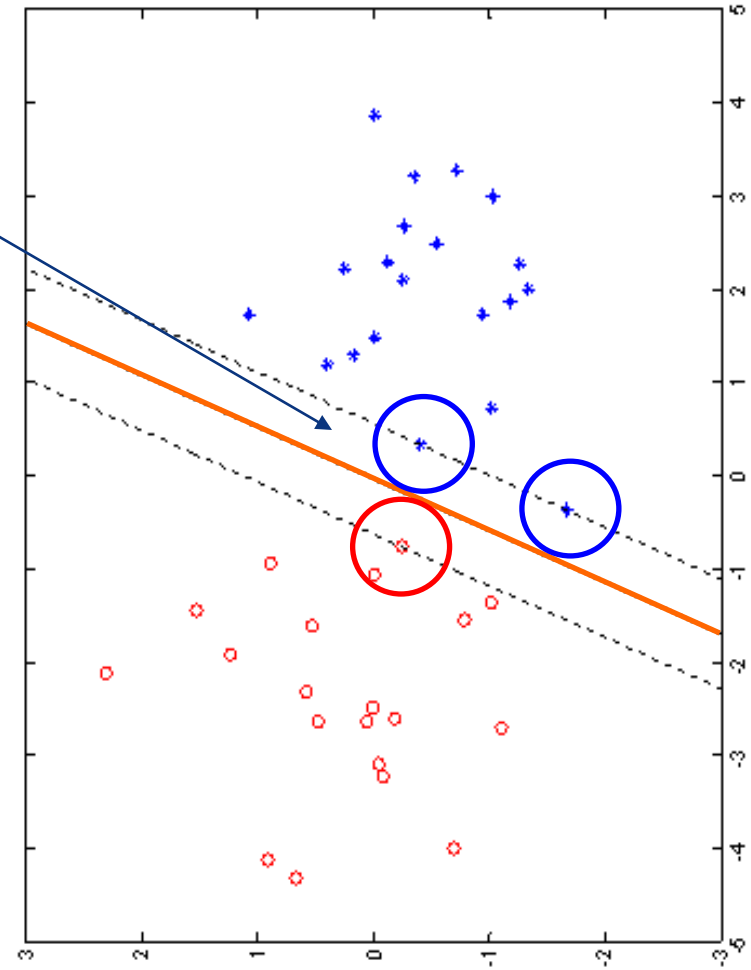
► these **likelihood** functions

- are a **kernel density estimate** if  $K(\cdot, \mathbf{x}_i)$  is a valid pdf
- **peculiar** kernel estimate that
  - **only** places **kernels** around the **support vectors**
  - all **other** points are ignored

# Input–Space Interpretation

► this is a discriminant form of density estimation


- concentrate modeling power where it matters the most, i.e. near **classification boundary**
- smart, since points away from the boundary are always well classified, even if the density estimates in their region are poor
- the SVM can be seen as a **highly efficient** combination of the BDR with kernel density estimates
- recall that one major problem of kernel estimates is the complexity of the decision function,  $O(n)$
- with the SVM, the complexity is only  $O(|SV|)$  but nothing is lost



# Input–Space Interpretation

► note on the approximations made:

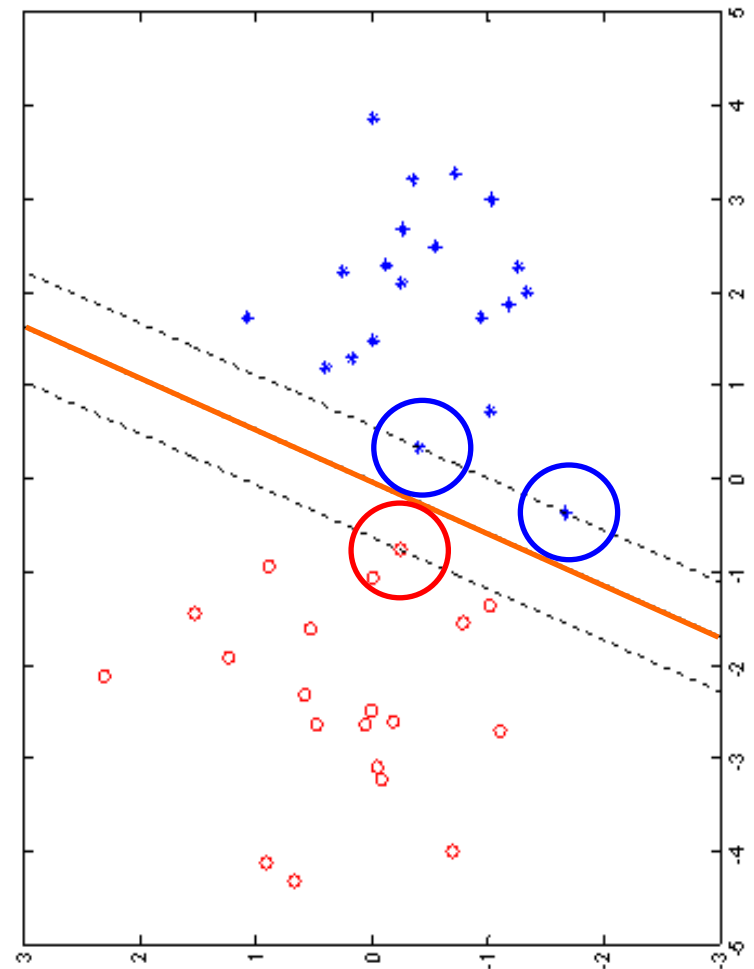
- this result was derived assuming  $b \approx 0$
- in practice,  $b$  is frequently left as a parameter, which is used to **trade–off false positives for misses**
- here, that can be done by **controlling** the **BDR threshold**

$$f(\mathbf{x}) = \begin{cases} 1, & \frac{\sum_{i \in SV|y_i \geq 0} \pi_i^* K(\mathbf{x}_i, \mathbf{x})}{\sum_{i \in SV|y_i < 0} \beta_i^* K(\mathbf{x}_i, \mathbf{x})} \geq T \\ -1, & \text{otherwise} \end{cases}$$


- hence, there is really not much practical difference, even when the assumption of  $b^* = 0$  does not hold!

# Limitations of the SVM

- ▶ appealing, but also points out the limitations of the SVM:
  - major problem of **kernel density estimation** is the **choice of bandwidth**
    - if too **small**, the estimates have too **much variance**
    - if too **large**, the estimates have too **much bias**
  - this problem **appears again** in the **SVM**
    - **no** generic “**optimal**” procedure to find **the kernel** or **its parameters**
    - requires **trial and error**
    - note, however, that this is **less of a headache** since only a **few** kernels have to be evaluated



- usually, we pick an **arbitrary kernel**, e.g. Gaussian
- then, determine **kernel parameters**, e.g. variance, by **trial and error**