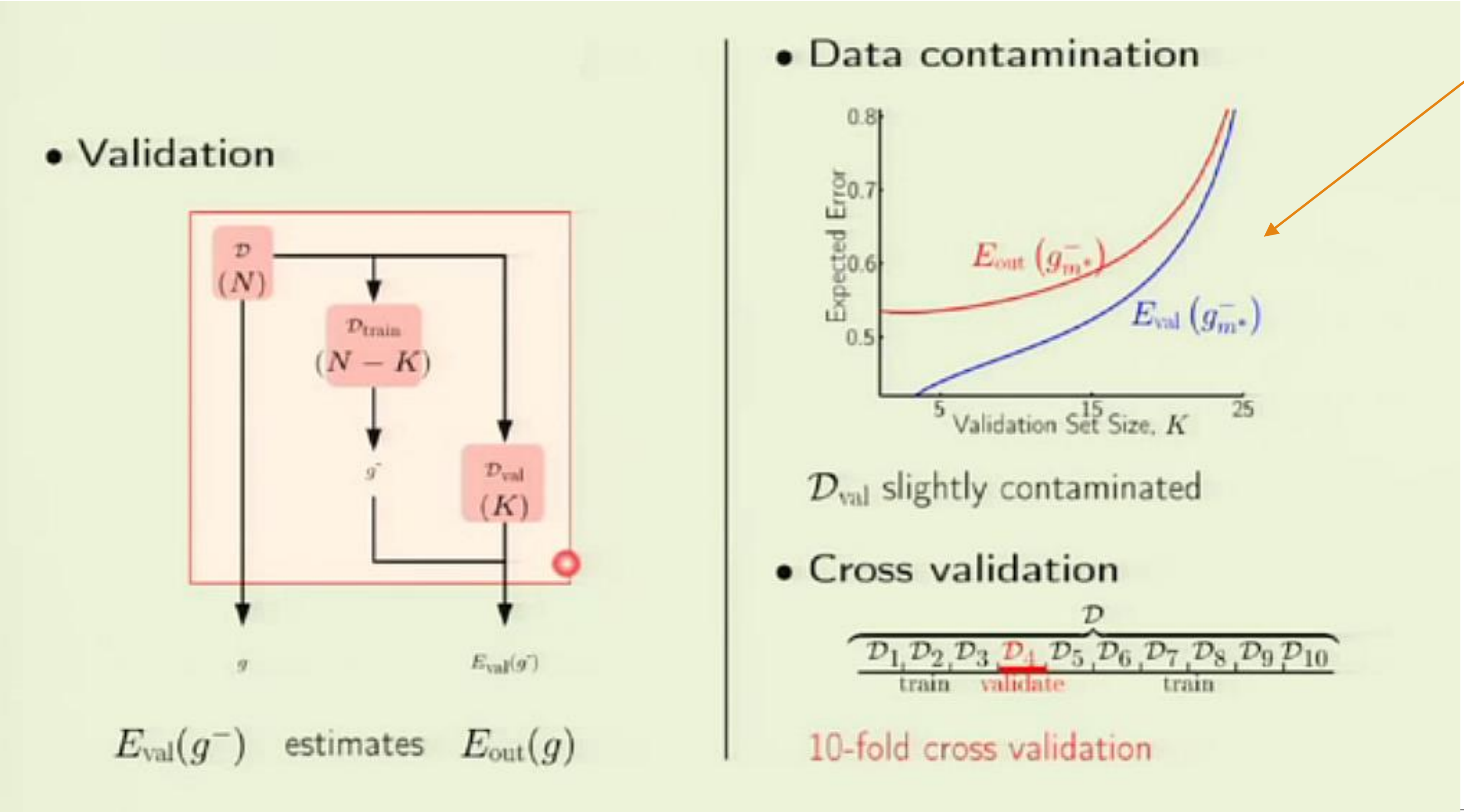
Support Vector Machines

Last Lectures



Optimistic bias

We use 25 examples to exaggerate the effect (bias) and see that **as we increase K, bias (diff. between curves) decreases** ->

Thus, with a **reasonable size** validation set we can estimate a couple of parameters **without contaminating the data** -> Thus, we can assume that the measurement you are getting from the validation set is reliable

e.g. use all data, 10 runs, validation is the way to go

- g- is the reduced hypothesis (we train with a subset)
- g is the original – best possible - hypothesis (to work with the most trained examples)
- E_{val}(g-) = **validation error** on the reduced hypothesis, is used to estimate the
- E_{out}(g), i.e. the **out of sample** error on the hypothesis we are actually delivering > the question is "how accurate is the E_{val} estimate for E_{out}?"
- K → should not be to small or too big for E_{val} estimate to be reliable
- K = rule of thump 20% to give us a reasonable estimate

SVMs

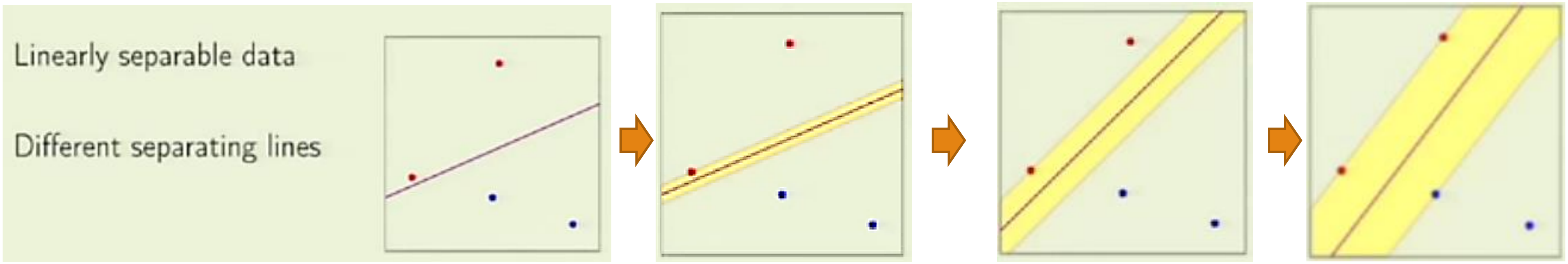
One of the most successful classification method in Machine Learning

Neat Piece of Work:

- There is a principle derivation for the method
- A very nice optimization package that you can use in order to get the solution
- Solution has a very intuitive interpretation

Outline

- Maximizing the margin (margin the main notion in SVMs -> need to maximize margin)
- Formulate the solution (analytical; constrained optimization problem)
- Nonlinear Transformations (expand from linear case to non-linear)



Small Margin | Wider Margin | Best fit Margin

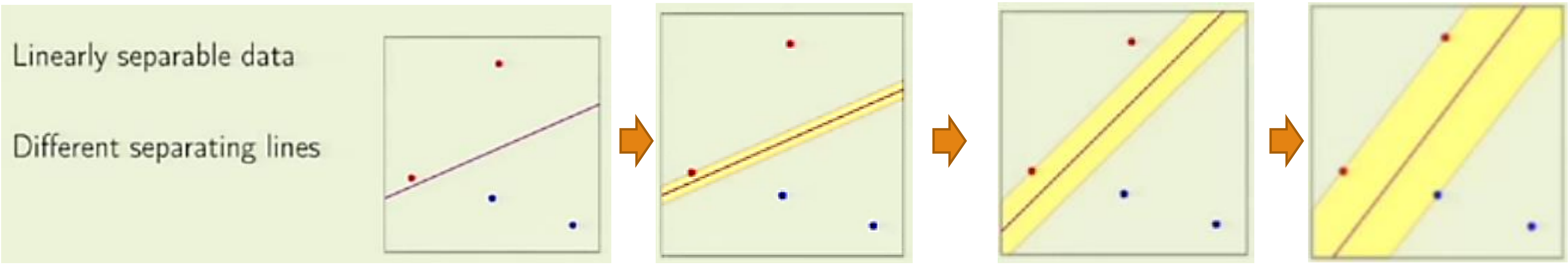
Question

Which one is the best? Knowing that:

- **Classification**: All give zero classification error
- **Generalization**: All deal with a liner problem with 4 points
- **Intuitive Decision**: we would choose the last one

- **Q: Is there any advantage of choosing 1 line over any other?**

- Linearly separable data: there are lines that can separate red from blue
- Different separating lines: We can apply different algorithms -> find different boundaries -> zero error



Small Margin Wider Margin Best fit Margin

New Questions

- Why is the bigger margin better?
- If we decide that ‘bigger margin better’, can we solve for a “w” that maximizes the margin?

e.g. noisy data -> intuition: last case good

- ➢ 1st case (Small Margin) a noisy point can be misclassified
- ➢ Last case, it’s high change that a noisy point can be classified correctly

A discriminant function that is a linear combination of the components of x can be written as

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0, \tag{1}$$

THRESHOLD WEIGHT

where $\mathbf{w}$ is the *weight vector* and w_0 the *bias* or *threshold weight*. A two-category linear classifier implements the following decision rule: Decide ω_1 if $g(\mathbf{x}) > 0$ and ω_2 if $g(\mathbf{x}) < 0$. Thus, $\mathbf{x}$ is assigned to ω_1 if the inner product $\mathbf{w}^t\mathbf{x}$ exceeds the threshold $-w_0$ and ω_2 otherwise. If $g(\mathbf{x}) = 0$, $\mathbf{x}$ can ordinarily be assigned to either class, but in this chapter we shall leave the assignment undefined. Figure 5.1 shows a typical implementation, a clear example of the general structure of a pattern recognition system

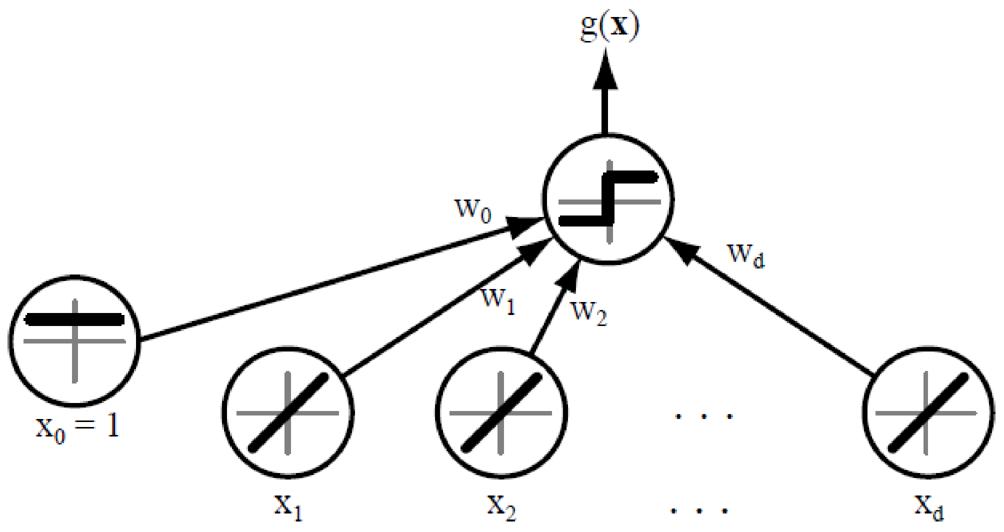
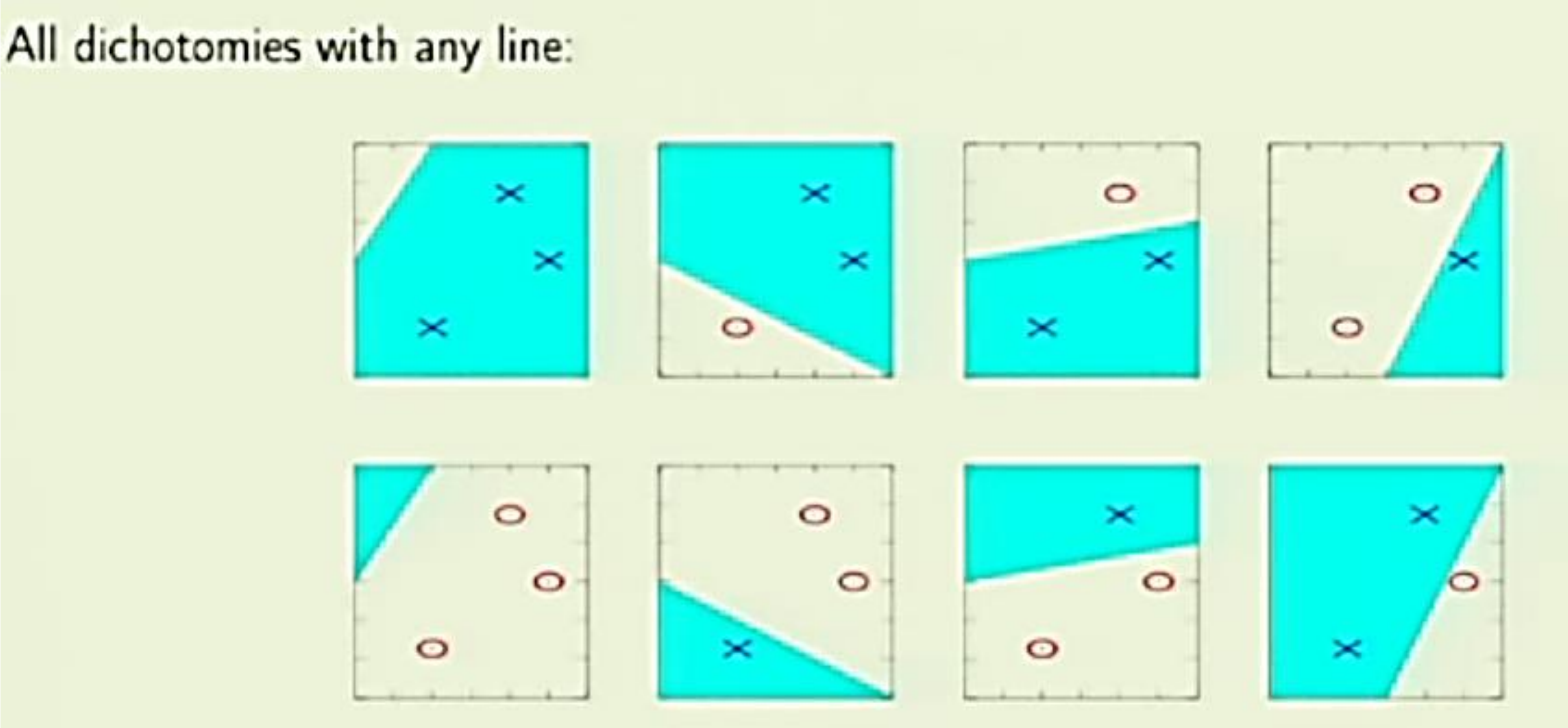
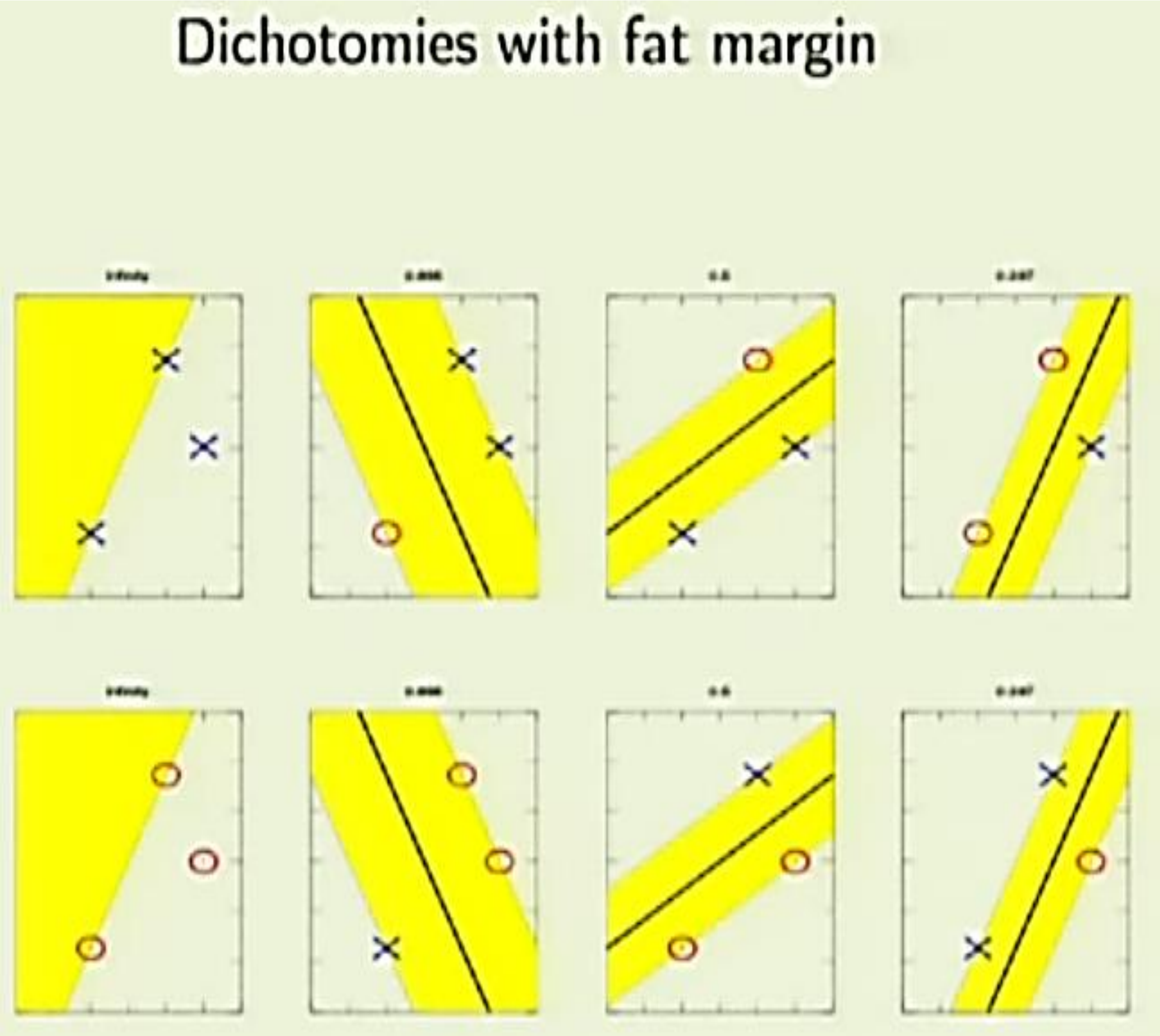


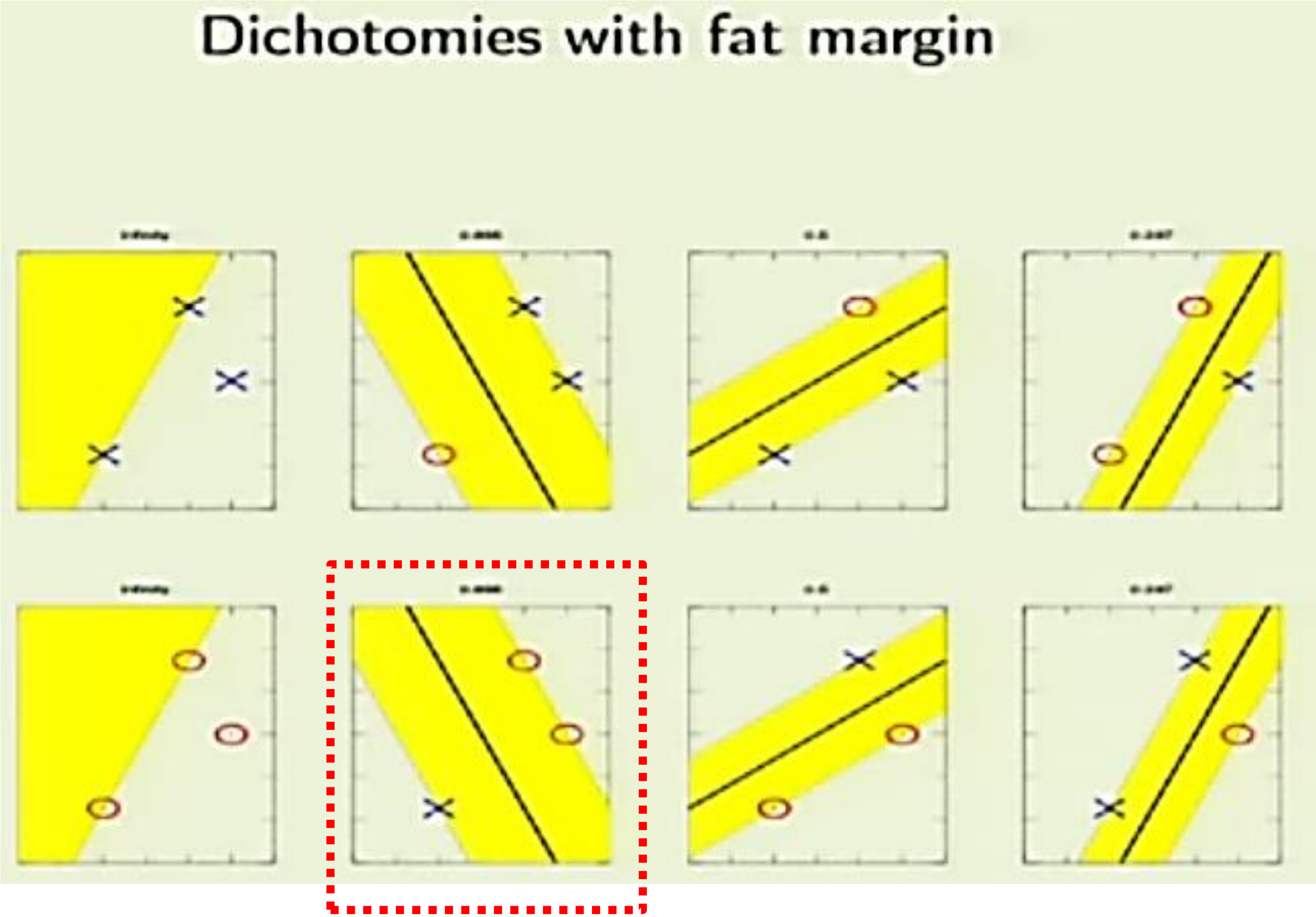
Figure 5.1: A simple linear classifier having d input units, each corresponding to the values of the components of an input vector. Each input feature value x_i is multiplied by its corresponding weight w_i; the output unit sums all these products and emits a $+1$ if $\mathbf{w}^t\mathbf{x} + w_0 > 0$ or a -1 otherwise.



- If few have 3 points > how many boundaries or dichotomies/lines?
  - 2^3 possible lines
- Having many possible boundaries is bad for generalization
- **Question**: is this affected by the margin? [lines + margin!]



- 3 points again
- We have the max (fat) margin for all cases
- Every time the margin touches all points
- If I want to have classifier with a specific margin > I can rule out cases



- **Assume**: we need a classifier with a margin at least as fat as in the red box to accept it
- **Fat margins (we restrict them) →** implies fewer dichotomies and VC dimension (is a measure of the capacity of a statistical classification algorithm), when compared to the case where <u>we did not restrict</u> them at all

Informally:

- By requiring the margin to be restricted > I have fewer dichotomies
- So, we can **estimate** the **out of sample error** based on the margin
- We will see that **if we have indeed a BIGGER margin** → <u>a BETTER out of sample performance</u>

Thus – Fat margins are good:

At the end of the lecture we will find that the out that the estimated out of sample error/performance is better with a fat margin

Goal:

Find the w, that not only classifies the points correctly, but also achieves so by the biggest possible margin

Finding $\mathbf{w}$ with large margin

Margin = a distance from a plane to a point

Let $\mathbf{x}_n$ be the nearest data point to the line $\mathbf{w}^\mathsf{T}\mathbf{x} = 0$.

Linear equation

We are going to refer to the line as a plane (not going to n-DIM space and hyperplanes)

So, if I have w and x, can we find the distance between the **plane (described by w)** and the point x_n?

That distance will be the margin that we are looking for.

Technicalities

1. Normalize w => $|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n| > 0$, for all points in the dataset, near and far, $w^T x_n$ will result with a number **plus or minus**, so we take the absolute value.

Q: We like to relate w with the margin >> however, there is a technicality: if we multiply w by 1M does the plane changes? -> No! see equation above (I can multiply with any number and have the same plane)

Thus, any formula that takes w and produces the margin will need to have scale invariance -> so we do this now to simplify the analysis later!

So we can have (no loss of generality): $|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n| = 1$ -> we consider all representations (planes) and pick one that requires, for the minimum point, that the absolute value is 1

- Basically we can scale w up and down until we get the point where the abs. value =1

So, we need the Euclidean distance - we do not compare the performance of each plane for different points but comparing the performance of different planes for the same point

Technicalities

2. Pull out w_0:

Solve the problem (different) w_1-w_d, than w_0-w_d

$\mathbf{w} = (w_1, \cdots, w_d)$ apart from w_0 ← We will call it *b* for bias (not to get confused if we use w_0)

The plane is now $\mathbf{w}^\intercal\mathbf{x} + b = 0$ > **There is no x_0** (was multiplied by b and now its gone)

$|\mathbf{w}^\intercal\mathbf{x}_n| = 1$ Becomes: $|\, w^T x_n + b \,| = 1$

$\mathbf{w}^\intercal\mathbf{x} + b = 0$ Becomes the NEW plane

These are the technicalities we need to get out of our way to simplify our math!

Geometry of the Problem

Computing the distance

The distance between $\mathbf{x}_n$ and the plane $\mathbf{w}^{\mathsf{T}}\mathbf{x} + b = 0$ (1), where $|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| = 1$

The vector $\mathbf{w}$ is $\perp$ to the plane in the $\mathcal{X}$ space: (Input space)

Question: why is it perpendicular to the place?

Answer: Take $\mathbf{x}'$ and $\mathbf{x}''$ on the plane → they need to satisfy the plane equation (1)

=> $\mathbf{w}^{\mathsf{T}}\mathbf{x}' + b = 0$ and $\mathbf{w}^{\mathsf{T}}\mathbf{x}'' + b = 0$

Note: **remember the concept is** >> x_n is a point; we have a plane; thus, we would like to estimate the distance

Geometry of the Problem (see we drop b > not needed)

The vector $\mathbf{w}$ is $\perp$ to the plane in the $\mathcal{X}$ space:

Take $\mathbf{x}'$ and $\mathbf{x}''$ on the plane

$\mathbf{w}^{\mathsf{T}}\mathbf{x}' + b = 0$ and $\mathbf{w}^{\mathsf{T}}\mathbf{x}'' + b = 0$ **We take the difference**

$\Longrightarrow \quad \mathbf{w}^{\mathsf{T}}(\mathbf{x}' - \mathbf{x}'') = 0$

Conclusion: **vector w** is orthogonal to the **vector (x' - x'')**

Interesting: we did not make any restrictions about the x' and x" points, so they can be any points on the plane

Conclusion: **vector w** that defines the plane is orthogonal to every vector to the plane =>
W is orthogonal to the plane !

and the distance is ...

Distance between $\mathbf{x}_n$ and the plane:

Take any point $\mathbf{x}$ on the plane

Projection of $\boxed{\mathbf{x}_n - \mathbf{x}}$ on $\mathbf{w}$ <u>is the distance we are looking for</u>

In order to get the projection, we get the unit vector of w:

$$\hat{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \implies \text{distance} = \left|\hat{\mathbf{w}}^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x})\right|$$

- **Inner Product**: unit vector (transposed) with x_n-x
- **Note**: **w** hat can be + or – (vector facing either x or the other direction)→ we use the ABS of this value

, which is a unit vector, i.e. w divided by its norm

$$\text{distance} = \frac{1}{\|\mathbf{w}\|}\left|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n - \mathbf{w}^{\mathsf{T}}\mathbf{x}\right| = \frac{1}{\|\mathbf{w}\|}\left|\overset{1}{\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b} - \overset{0}{\mathbf{w}^{\mathsf{T}}\mathbf{x} - b}\right| = \frac{1}{\|\mathbf{w}\|}$$

Point 1: by having the plane and insist on a canonical representation of w, by $|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| = 1$ for the nearest point x_n → then, the your **margin (distance)** will be 1/norm of the **w** you use

Point 2: we use this distance → will be able to find out <u>which combinations of w will give me the best possible margin</u>

The optimization problem

Equivalent Problem that is more friendly

Maximize $\frac{1}{\|\mathbf{w}\|}$

subject to $\min_{n=1,2,\ldots,N} |\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| = 1$

This is not a friendly optimization problem; we have minimum and that does not help

→ thus, **we need to "*get rid of the min and abs*" and find an equivalent optimization problem that is <u>more friendly</u>**

So, what do we notice? → Notice: $|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| = y_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b)$

Minimize $\frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}$

subject to $y_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1 \quad$ for $\quad n = 1, 2, \ldots, N$

Notice: $|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| = y_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b)$ Why?

---

[1] Getting rid of ABS:

- **We are only considering the points that are classified correctly** (that separate the data correctly)
- Then, we're **choosing** among them **those that maximize the margin >** since they are classifying the data correctly, the **signal (wx_n+b)** <u>agrees with the **label**</u> **y_n (+1 or -1)**

[2] Getting rid of the min:

- Instead of *maximizing the 1/norm of w* > we *minimize ½ $w^T \cdot w$*

$$\text{subject to} \quad y_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1 \quad \text{for} \quad n = 1, 2, \ldots, N$$

This is an inequality constraint that is linear in nature

$$\text{Minimize } \frac{1}{2} \mathbf{w}^{\mathsf{T}}\mathbf{w}$$

Friendly Optimization Problem

$$\text{subject to } y_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1 \quad \text{for} \quad n = 1, 2, \ldots, N$$

This quantity is the same as the signal (wx_n+b)

- If the min is 1, then yn(wx_n+b) is fine

Important Points:

- Maybe the optimization will result having all of these points make this quantity strictly > 1
- So, if for a certain point → this quantity >1 → and then, we get the minimum of ½ $w^T \cdot w$ then this is the point I am going to have

We cannot get the min **w**, when all values are strictly greater than 1 =>

Conclusion: "When we solve the above optimization problem, the solution necessarily satisfies the inequality constraint, with at least one of the points resulting = 1", so this new friendly opt. problem to find the best margin, **is equal to** the unfriendly opt. problem we had in the beginning

Constraint Optimization Problem - Overview

$$\text{Minimize} \quad \frac{1}{2} \mathbf{w}^{\mathsf{T}}\mathbf{w}$$

$$\text{subject to} \quad y_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1 \quad \text{for} \quad n = 1, 2, \ldots, N$$

DOMAIN:

$$\mathbf{w} \in \mathbb{R}^d, \; b \in \mathbb{R}$$

- d is the dimension
- B is a real number

Question: Constrained Optimization Problem: how to solve it?

- We need an analytic way to solve it: **form a Lagrange** and the constrained problem becomes unconstrained etc.

- The small problem is that we have to convert the inequality problem to equality → can we square and then, solve he equality problem?

Lagrange Formulation

We minimize $\frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}$ subject to $y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) \geq 1$

Objective Function

$$\alpha_n\left(y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) - 1\right)$$

1. We removed inequality
2. We have a Lagrange multiplier a_n

   a_n or you may see it as λ_i in the notes

$$\frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} - \sum_{n=1}^{N} \alpha_n\left(y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) - 1\right)$$

- **We have a new that formula makes sense**
- **We minimize w.r.t** w, b **and maximizing** w.r.t $a_n \geq 0$
- We have new variable Lagrange Multipliers (vector **a**)
- There are n multiplies, one for every point in the set

$$\text{Minimize} \quad \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} - \sum_{n=1}^{N} \alpha_n\left(y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) - 1\right)$$

Lagrange Formulation

$$\text{Minimize} \quad \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} - \sum_{n=1}^{N} \alpha_n \left(y_n \left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) - 1\right) \quad (1)$$

<u>Working with the unconstrained part</u>:
we just need to <u>optimize</u> L w.r.t w and **b**
and the following conditions result:

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n = \mathbf{0} \quad (2)$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{n=1}^{N} \alpha_n y_n = 0 \quad (3)$$

- We take the gradient of L with respect to w
- We take the derivative of L with respect to b

Next Step: Substitute 2,3 to eq. 1, such that the maximization of **a** (Lagrangian) – tricky since **a** has a range – becomes free of **w** and **b**

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n \quad \text{and} \quad \sum_{n=1}^{N} \alpha_n y_n = 0$$

The goal is to come up with an equation that is a function of the Lagrangian a only!

Lagrange Formulation

Minimize $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\intercal}\mathbf{w} - \sum_{n=1}^{N} \alpha_n (y_n (\mathbf{w}^{\intercal}\mathbf{x}_n + b) - 1)$ **(1)**

in the Lagrangian $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\intercal}\mathbf{w} - \sum_{n=1}^{N} \alpha_n (y_n (\mathbf{w}^{\intercal}\mathbf{x}_n + b) - 1)$

we get $\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^{\intercal}\mathbf{x}_m$

Constraints: Maximize w.r.t. to $\boldsymbol{\alpha}$ subject to $\alpha_n \geq 0$ for $n = 1, \cdots, N$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$

We need this!

Initial Constraints: $\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$

No need any more – does not depend on w

Thus, we need to work on solving this **constrained optimization problem** using **quadratic programming**

… The Solution – Quadratic Programming

We need to translate the objective and the constraints we have into the coefficients that we will pass onto the package called quadratic programming

$$\max_{\alpha} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^{\intercal} \mathbf{x}_m \rightarrow \min_{\alpha} \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^{\intercal} \mathbf{x}_m - \sum_{n=1}^{N} \alpha_n \qquad (4)$$

NEXT STEP: Isolate the coefficients from alphas,

- Where, **alphas** are the parameters (these are not passes to QP)
- What we pass to QP are the coefficients y_n and y_m decided by y_s and x_s → see matrix next slide
- Thus, QP will work and provide us with the alphas that minimize equation (4)

… The Solution – Quadratic Programming

$$\min_{\alpha} \quad \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{x}_n^{\intercal}\mathbf{x}_m - \sum_{n=1}^{N} \alpha_n \qquad (4)$$

$$\min_{\alpha} \quad \frac{1}{2}\alpha^{\intercal} \underbrace{\begin{bmatrix} y_1 y_1 \, \mathbf{x}_1^{\intercal}\mathbf{x}_1 & y_1 y_2 \, \mathbf{x}_1^{\intercal}\mathbf{x}_2 & \dots & y_1 y_N \, \mathbf{x}_1^{\intercal}\mathbf{x}_N \\ y_2 y_1 \, \mathbf{x}_2^{\intercal}\mathbf{x}_1 & y_2 y_2 \, \mathbf{x}_2^{\intercal}\mathbf{x}_2 & \dots & y_2 y_N \, \mathbf{x}_2^{\intercal}\mathbf{x}_N \\ \dots & \dots & \dots & \dots \\ y_N y_1 \, \mathbf{x}_N^{\intercal}\mathbf{x}_1 & y_N y_2 \, \mathbf{x}_N^{\intercal}\mathbf{x}_2 & \dots & y_N y_N \, \mathbf{x}_N^{\intercal}\mathbf{x}_N \end{bmatrix}}_{\text{quadratic coefficients}} \alpha + \underbrace{(-\mathbf{1}^{\intercal})}_{\text{linear}} \alpha$$

- **We have**: Quadratic term a^T and a
- **In the bracket**: the coefficients in the double summation:
  - These are red from the training data
  - We have have yi and xi and we generate the multiplication factors
- <u>What is passed to QP</u>: The **matrix** ; the **sum of alphas** (a set of linear coefficients) ; the **constraints** (i) the $\mathbf{y}^T$ as a vector and (ii) **a** range

$$\text{subject to} \quad \underbrace{\mathbf{y}^{\intercal}\alpha = 0}_{\text{linear constraint}}$$

$$\underbrace{\mathbf{0}}_{\text{lower bounds}} \leq \alpha \leq \underbrace{\infty}_{\text{upper bounds}}$$

… The Solution – Quadratic Programming

$$\min_{\alpha} \quad \frac{1}{2}\alpha^{\intercal} \underbrace{\begin{bmatrix} y_1y_1\ \mathbf{x}_1^{\intercal}\mathbf{x}_1 & y_1y_2\ \mathbf{x}_1^{\intercal}\mathbf{x}_2 & \dots & y_1y_N\ \mathbf{x}_1^{\intercal}\mathbf{x}_N \\ y_2y_1\ \mathbf{x}_2^{\intercal}\mathbf{x}_1 & y_2y_2\ \mathbf{x}_2^{\intercal}\mathbf{x}_2 & \dots & y_2y_N\ \mathbf{x}_2^{\intercal}\mathbf{x}_N \\ \dots & \dots & \dots & \dots \\ y_Ny_1\ \mathbf{x}_N^{\intercal}\mathbf{x}_1 & y_Ny_2\ \mathbf{x}_N^{\intercal}\mathbf{x}_2 & \dots & y_Ny_N\ \mathbf{x}_N^{\intercal}\mathbf{x}_N \end{bmatrix}}_{\text{quadratic coefficients}} \alpha + \underbrace{(-\mathbf{1}^{\intercal})}_{\text{linear}} \alpha$$

$$\text{subject to} \quad \underbrace{\mathbf{y}^{\intercal}\alpha = 0}_{\text{linear constraint}}$$

$$\underbrace{\mathbf{0}}_{\text{lower bounds}} \leq \alpha \leq \underbrace{\infty}_{\text{upper bounds}}$$

Or even simpler:

$$\min_{\alpha} \quad \frac{1}{2}\alpha^{\intercal}Q\alpha - \mathbf{1}^{\intercal}\alpha \qquad \text{subject to} \qquad \mathbf{y}^{\intercal}\alpha = 0; \quad \alpha \geq \mathbf{0}$$

- Linear Equality Constraint
- Other range constraints

- <u>What is passed to QP</u>:
  - The **matrix**
  - The **sum of alphas** (a set of linear coefficients)
  - The **constraints**
    - $\mathbf{y}^{T}$ as a vector
    - Range for **a**

QP will give us back the alphas

The rest of this lecture can be found at:

[HERE](https://www.youtube.com/watch?v=eHsErlPJWUU)
or
https://www.youtube.com/watch?v=eHsErlPJWUU