The Support Vector Machine

4 B

CS 3244
Machine Learning

NUS National University of Singapore | Computing

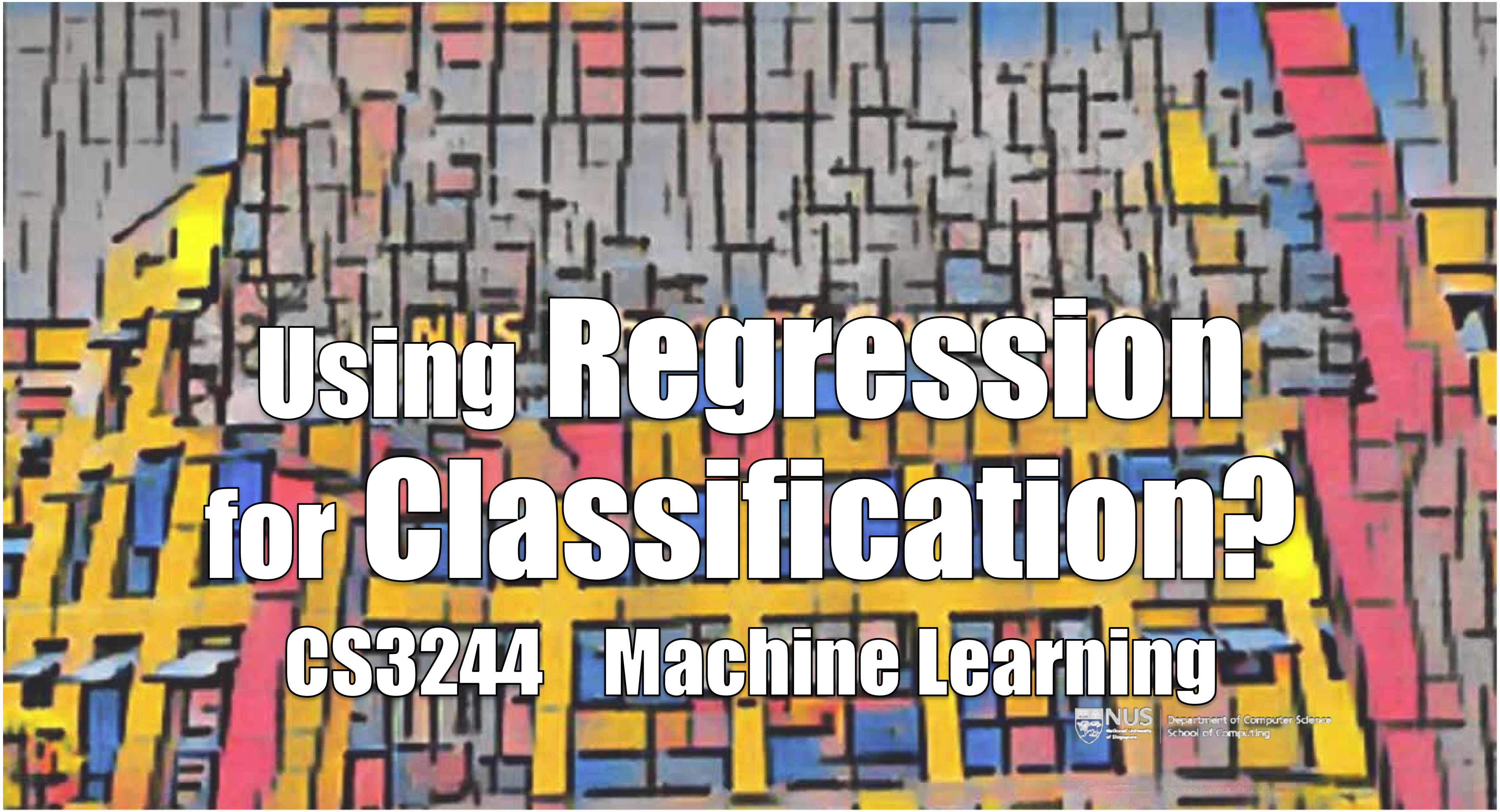
Forecast for Week 04B

Learning Outcomes for this week:

- Understand the Support Vector Machine Classifier as an optimal hyperplane;
- Understand how the optimization function is modified to allow errors (soft SVM).

Other important concepts:

- Noisy Targets
- Non-linear Mappings
- Kernels


Using Regression
for Classification?
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing

Linear regression for classification

Linear regression learns a real-valued function $y = f(\mathbf{x}) \epsilon \mathbb{R}$

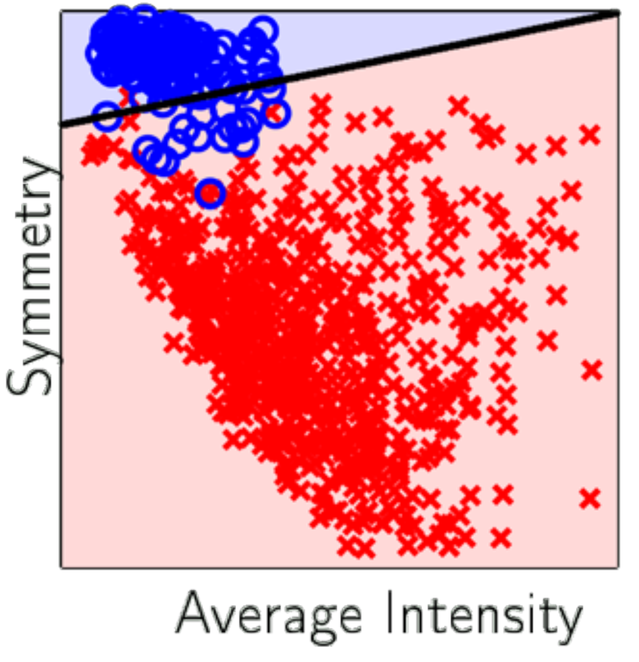
Binary valued functions are also real-valued! $\pm 1 \epsilon \mathbb{R}$

Use linear regression to get θ where $\theta^\top \mathbf{x}^{(j)} \approx y^{(j)} = \pm 1$

In this case, $\text{sign}(\theta^\top \mathbf{x}^{(j)})$ is likely to agree with $y^{(j)} = \pm 1$

Good initial weights for classification

Why linear regression doesn't set good weights for classification



What's wrong with this picture?

Hint: think squared error

Noisy Targets

CS3244 Machine Learning

Noisy targets

The target function isn't always a function $f: \mathcal{X} \rightarrow \mathcal{Y}$

Criterion	Value
Age	32 years
Gender	Male
Salary	40 K
Debt	26 K
…	…
Years in Job	1 year
Years at Current Residence	3 years

Consider two identical customers for loan approval …

could have two different outcomes!

Why? How do we characterize these sources of noise?

Your Turn: what do you think?

The target function isn't always a function $f: \mathcal{X} \rightarrow \mathcal{Y}$

Criterion	Value
Age	32 years
Gender	Male
Salary	40 K
Debt	26 K
…	…
Years in Job	1 year
Years at Current Residence	3 years

Q1: Is misreporting salary a noisy target?

Q2: What other lecture featured noisy targets?

Target distribution

Instead of saying the target is a function, think of it as a distribution: $P(y|\mathbf{x})$

Our data $(\mathbf{x}, y)$ is now generated by the joint distribution: $P(\mathbf{x})P(y|\mathbf{x})$

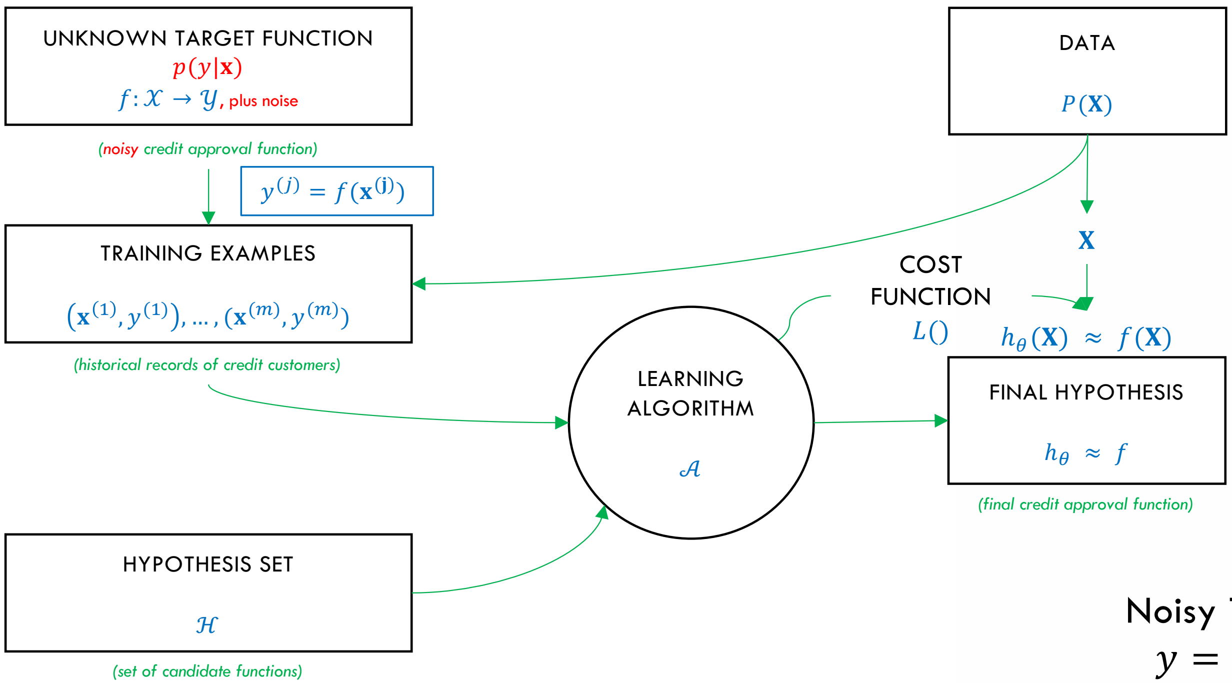
Noisy target = Deterministic target function $f(x) = \mathbb{E}(y|\mathbf{x})$

+ Noise $(y - f(\mathbf{x}))$

A deterministic target is just a special case:

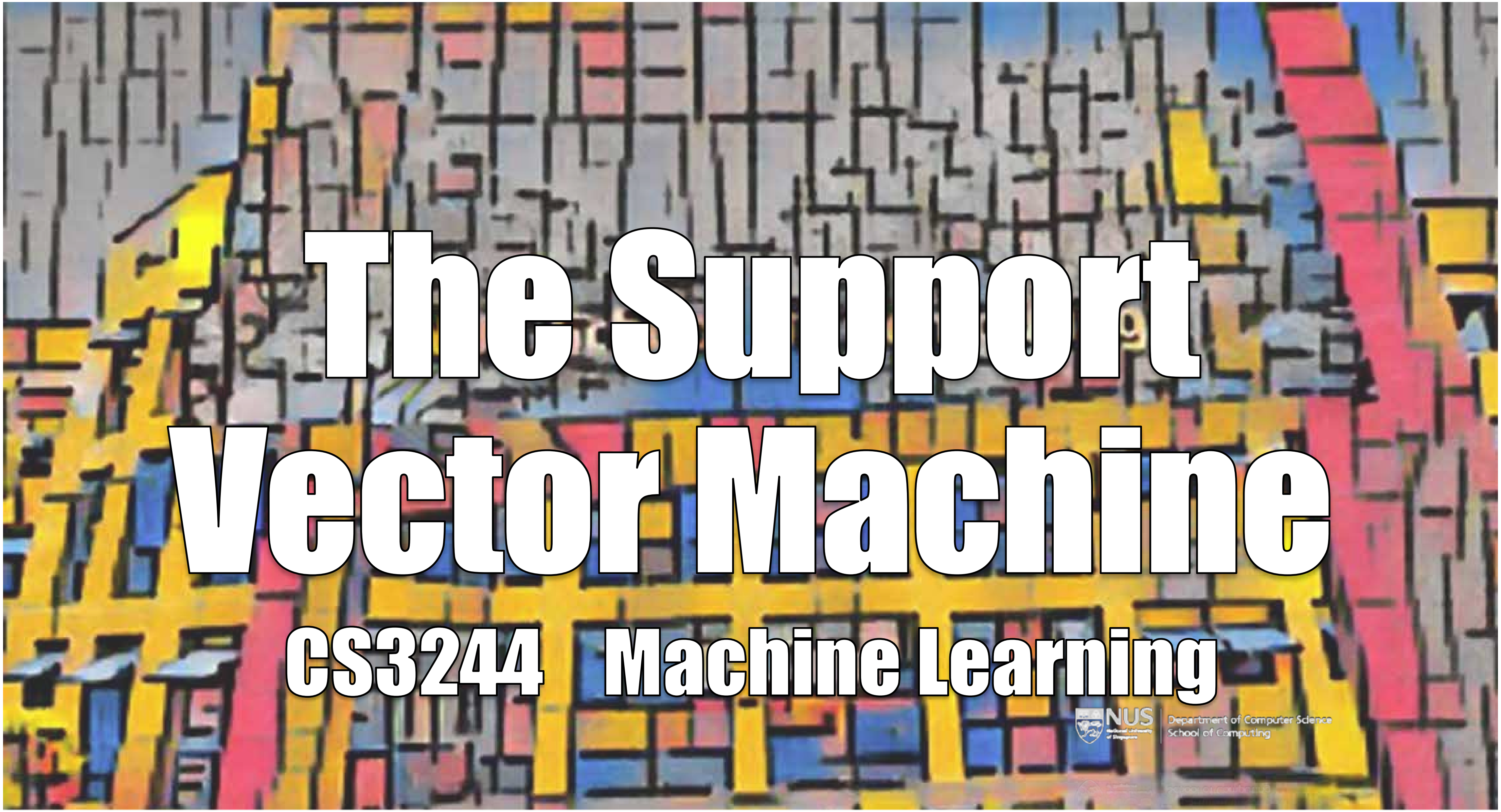
$P(y|\mathbf{x}) = 0$, except for $y = f(\mathbf{x})$

Learning diagram with noisy targets

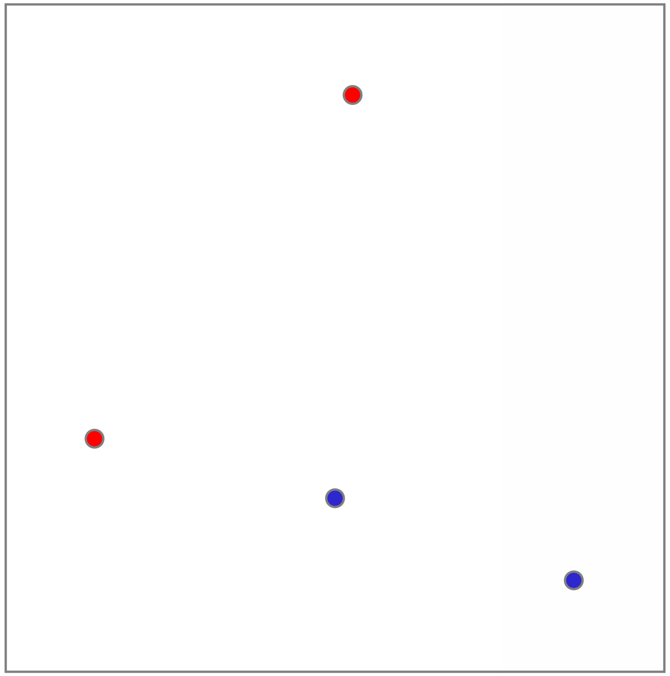


Noisy Target:

$$y = f(\mathbf{x}) \quad \rightarrow \quad y \sim P(y|\mathbf{x})$$


The Support Vector Machine
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing

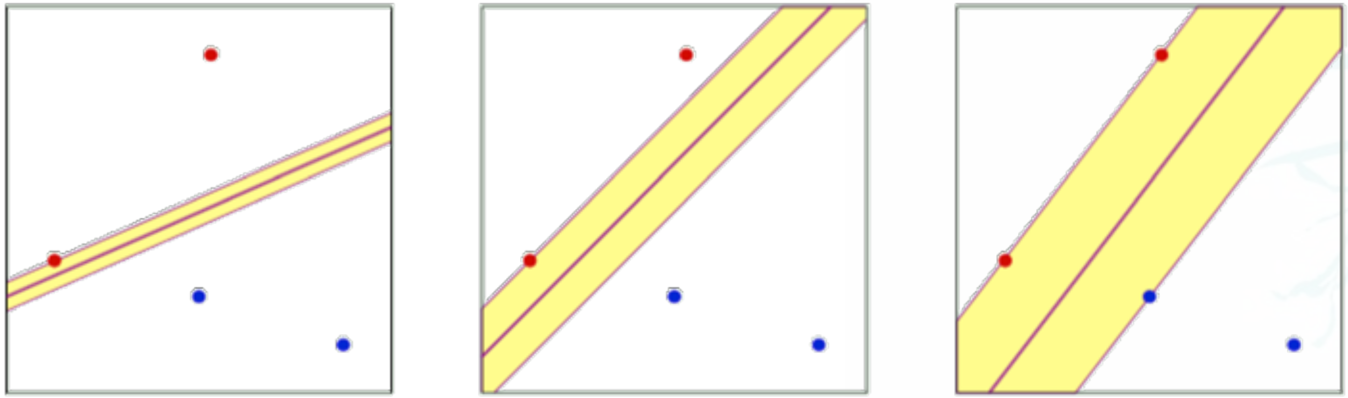
Land Transport Authority



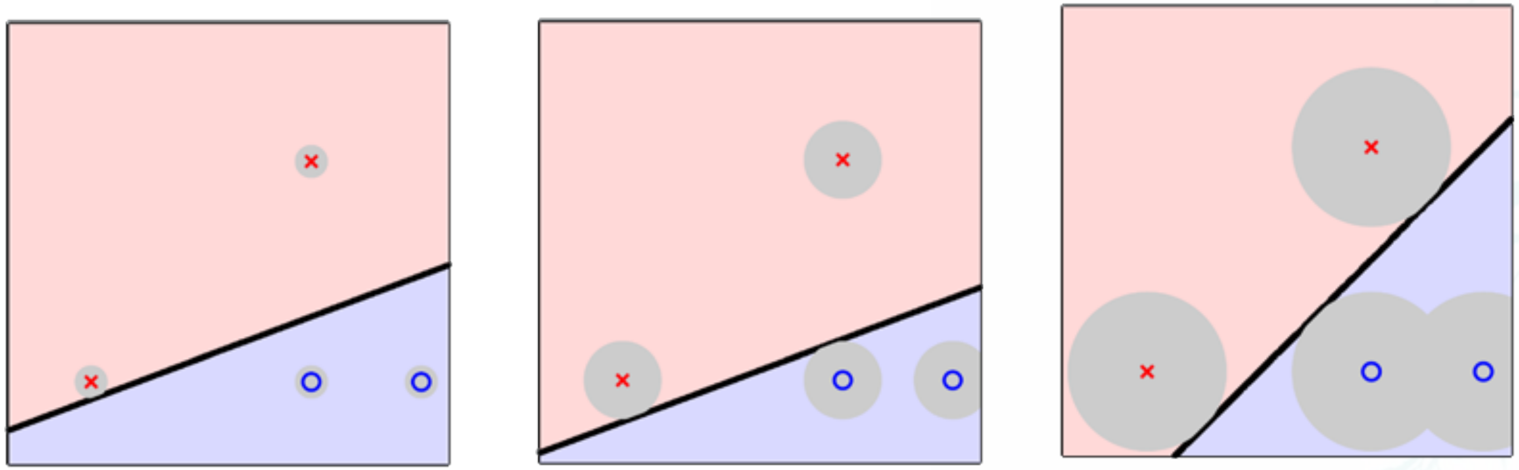
Better linear separation

Two questions:

1. Why is a bigger margin better?
2. Which $\boldsymbol{\theta}$ maximizes the margin?

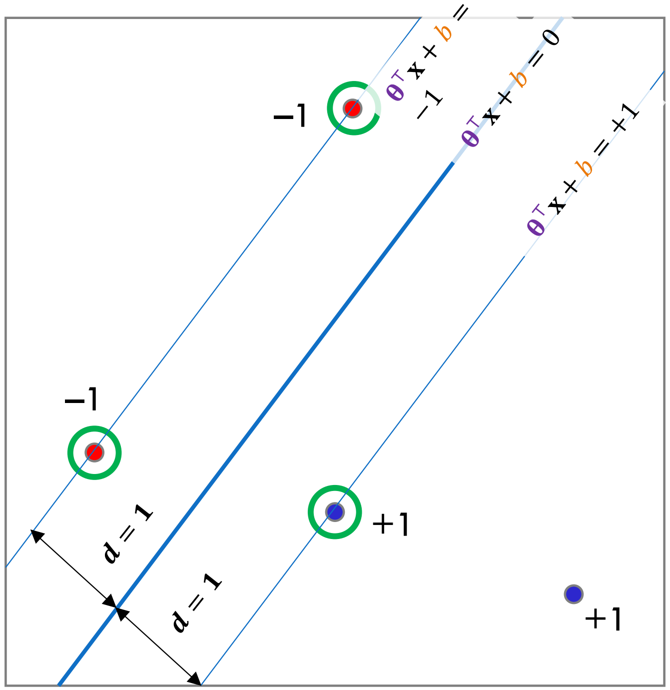


Inherently handles noisy data



d

SVM: An analogizer



Hyperplane equation: $h(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} + b$

+ve points must lie above the hyperplane; −ve points below.

$\boldsymbol{\theta}$ dictates the orientation of the plane;
b dictates the offset (bias).

Define distance d to the optimal plane as “1” (unit distance).

This sets up constrained quadratic optimization problem that identifies the unique $h(\mathbf{x})$.

Note: Only a subset of the dataset **determines** our unique $h(\mathbf{x})$.

These are the support vectors, the most difficult instances to classify.

Inductive Bias of the SVM

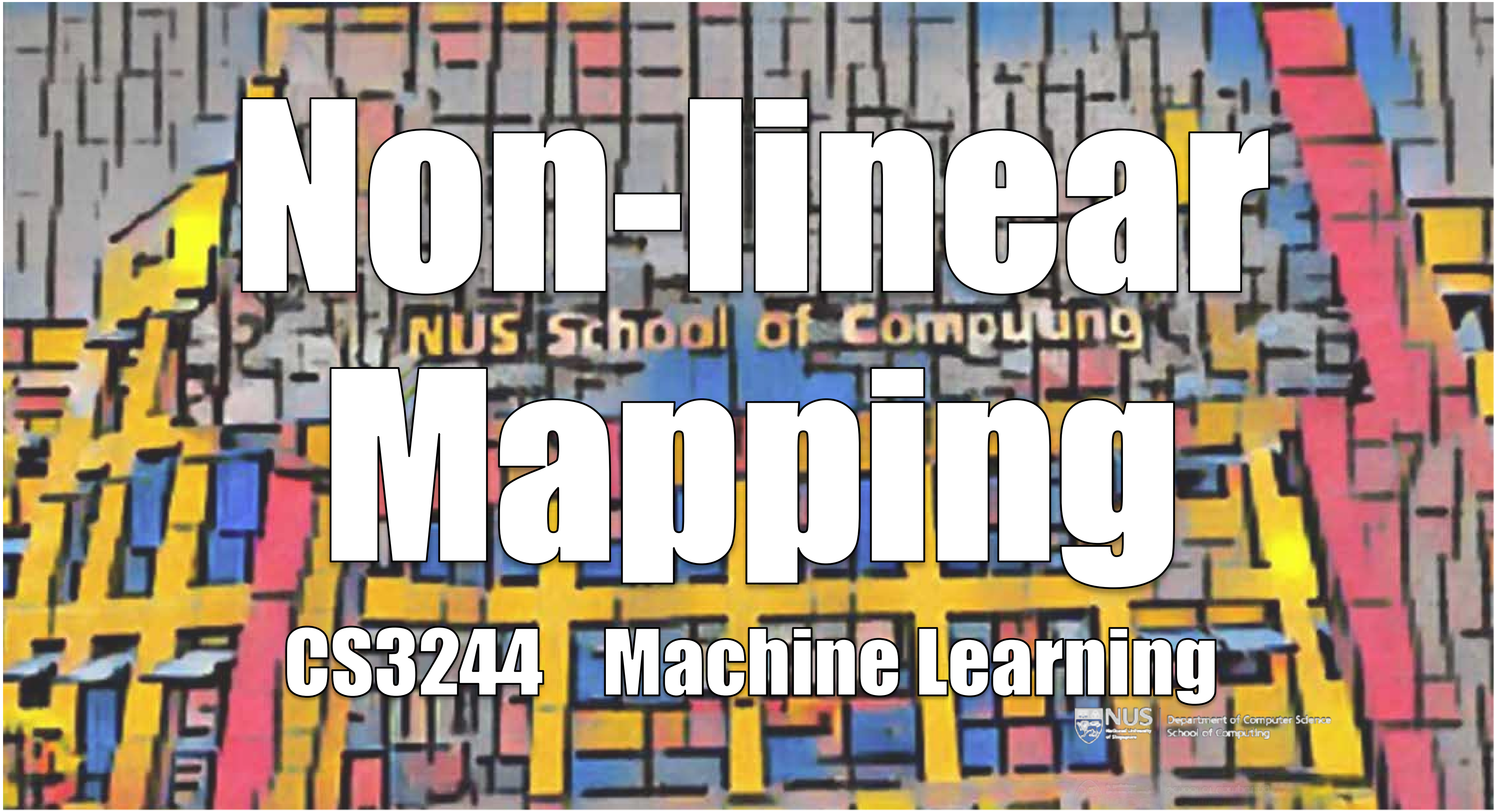
In Zoom breakout or physical subgroups, answer then post:

(5 mins): What do you think the Inductive Bias of the SVM is?

Ask one member to write it to the #general thread. Upvote others that you like.

Non-linear Mapping

CS3244 Machine Learning



Pre-Lecture Activity from last week

Can, but not accurate.

probably not with a high degree of accuracy, so perhaps the more appropriate term is simply *estimate* rather than *model*. Of course these are dependent on how non-linear the data is : linear classifiers can give good estimates approximately linear data and vice versa

Yes, but the result will not be accurate and thus may not be quite useful

We can attempt to model non-linearity with linear models and probably achieve some level of accuracy, depending on the problem. This may even be preferable due to the relative simplicity of linear models in terms of design and computational requirements. However, non-linear problems would still be better modelled with models that more accurately captures the actual

I think some non-linearity can be modeled by a linear classifier (if has a high correlation); for other non-linearity, a linear classifier can somehow model non-linearity but cannot perfectly match it. For instance, knn is a nonlinear classifier. For each small part, the boundaries are linear segments (approximately). However, the general shape is complex, which cannot be simply represented by linear models.

If the data is far from being linear, it is difficult (perhaps impractical) to accurately model it with linear classifiers. For example, if we consider the quadratic curve (U-shape) and say that points within the U have one label and points outside have another label, the accuracy of having a few best-fit linear classifiers will be quite low. As the number of classifiers increase, the accuracy will increase, but it'll take a very large number of linear classifiers to model the curve accurately. Why not have a quadratic curve classifier instead?

No, cannot.

I think we cannot model non-linearity with a linear classifier accurately. We can at most estimate. Linear classifiers, according to Robby's answer, do not model interactions between features. Mathematically, using a linear regression model would assume that the coefficients of any (x_i)(x_j) term (where i is not equal to j) is always zero.

I think no, since for a linear function, the output is related to input x_i by the weight w_i. Ie. y = Σ(w_i)(x_i). If the output for example increases exponentially wrt to x_i, I dont think this can be expresed using (w_i)(x_i).

Perhaps an exception would be when the non-linear relationship so happens to be almost linear.

Yes, with some transformation

Yes, by scaling the non-linear features of the function to become linear. However, model may not be very accurate after scaling.

it may be possible if some transformations are made to the dataset

Yes i think it is possible to do that with a linear classifier. Though not the most accurate, an example would be the SVM where samples are mapped to a higher dimensional feature space

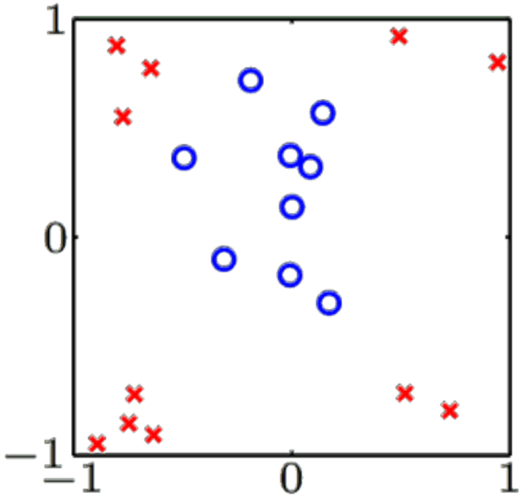
Yes, it it possible if we perform linear transformation, however it might makes the feature dimension larger which leads to high complexity in time.

Yes, but not directly. One indirect way is to add new features to the non-linear model, it will increase the number of dimension when we plot the data and there can be a linear classifier that separate examples in the new model

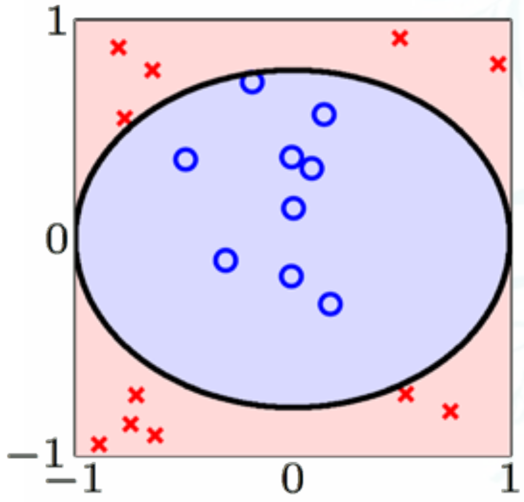
It is possible to model non linear data with a linear classifier but we need to tweak the model. One possible approach is to do a feature transformation where the original attributes are mapped to a new feature space. For example, given non linear data that follows the equation of a circle x^2 + y^2 = r^2 where different labels have different range of radius. In the linear model we can plot (x^2, y^2) in the x and y axis and use linear separator to groups the data.

Linear models are limited

Data:



Hypothesis:



Another example

Credit line is affected by years in current residence x_i, but not in a linear way.

The value range $\left[\left[1 < x_i < 5\right]\right]$ is more significant.

Can we do that with linear models?

But linear *in what?*

Linear regression implements

$$\sum_{i=0}^{n} \theta_i x_i = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

Linear classification implements

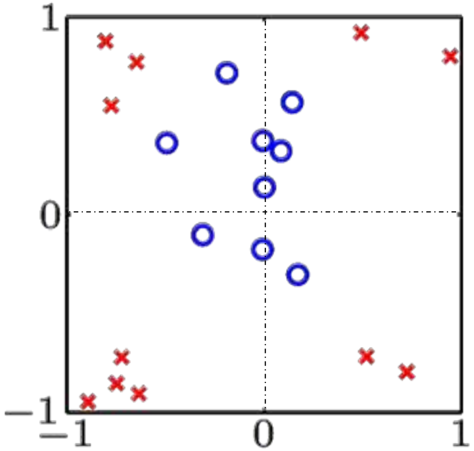
$$\mathrm{sign}\left(\sum_{i=0}^{n} \theta_i x_i\right)$$

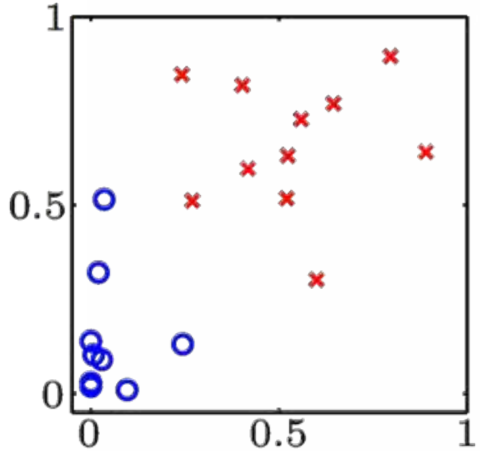
Algorithms work because of the linearity of weights,

but it doesn't say anything about the observed data $\mathbf{x}$.

Transform the data nonlinearly

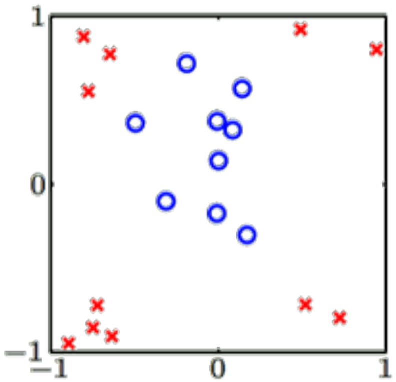
$$(x_1, x_2, \ldots, x_n) \xrightarrow{\Phi} (x_1^2, x_2^2, \ldots, x_n^2)$$



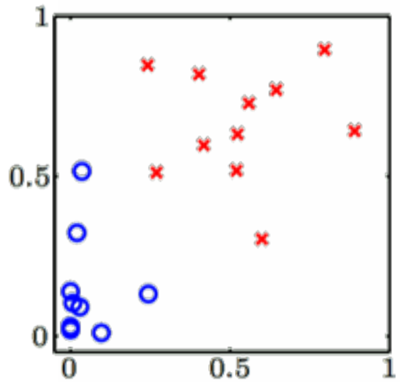


Any $\mathbf{x} \xrightarrow{\Phi} \mathbf{z}$ preserves this linearity!

1. Original Data
$\mathbf{x}^{(j)} \in \mathcal{X}$

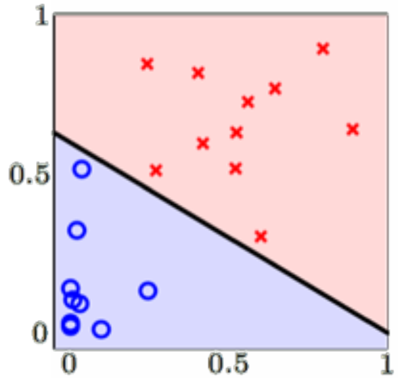


$\xrightarrow{\Phi}$



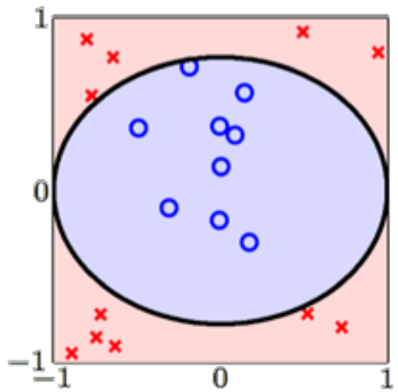
2. Transform the data
$\mathbf{z}^{(j)} = \Phi\left(\mathbf{x}^{(j)}\right) \in \mathcal{Z}$

$\downarrow$



3. Separate the data in
Z space
$\tilde{h}(\mathbf{z}) = \text{sign}\left(\tilde{\boldsymbol{\theta}}^{\top}\mathbf{z}\right)$

$\xleftarrow{\text{'}\Phi^{-1}\text{'}}$



4. Classify in $\boldsymbol{X}$ space
$h_{\tilde{\theta}}(\mathbf{x}) = \tilde{h}\left(\Phi(\mathbf{x})\right)$
$= \text{sign}\left(\tilde{\boldsymbol{\theta}}^{\top}\Phi(\mathbf{x})\right)$

What transforms to what

$$\mathbf{x} = (x_1, x_2, \dots, x_n) \xrightarrow{\Phi} \mathbf{z} = (z_0, z_1, \dots, z_{\tilde{n}})$$

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)} \xrightarrow{\Phi} \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(m)}$$

$$y^{(1)}, y^{(2)}, \dots, y^{(m)} \xrightarrow{\Phi} y^{(1)}, y^{(2)}, \dots, y^{(m)}$$

$\boldsymbol{\theta}$? No weights in $\mathcal{X}$ $\tilde{\boldsymbol{\theta}} = (\theta_1, \theta_2, \dots, \theta_{\tilde{n}})$

$$\begin{aligned} h_{\tilde{\theta}}(\mathbf{x}) &= \operatorname{sign}\left(\tilde{\boldsymbol{\theta}}^\top \mathbf{z}\right) \\ &= \operatorname{sign}(\tilde{\boldsymbol{\theta}}^\top \Phi(\mathbf{x})) \end{aligned}$$

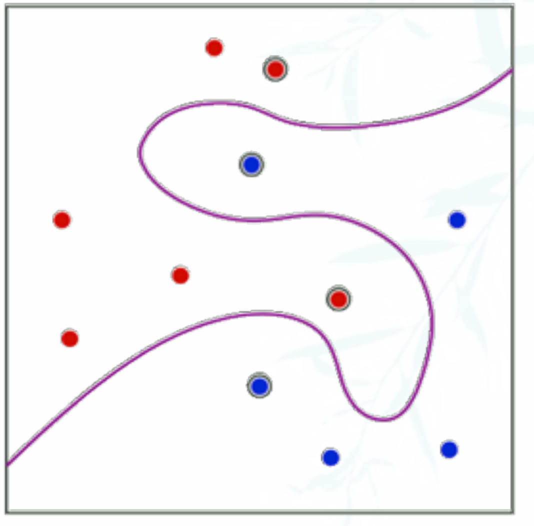
"Support Vectors" in $\mathcal{X}$ space

What if the support vectors live in $\mathcal{Z}$ space?

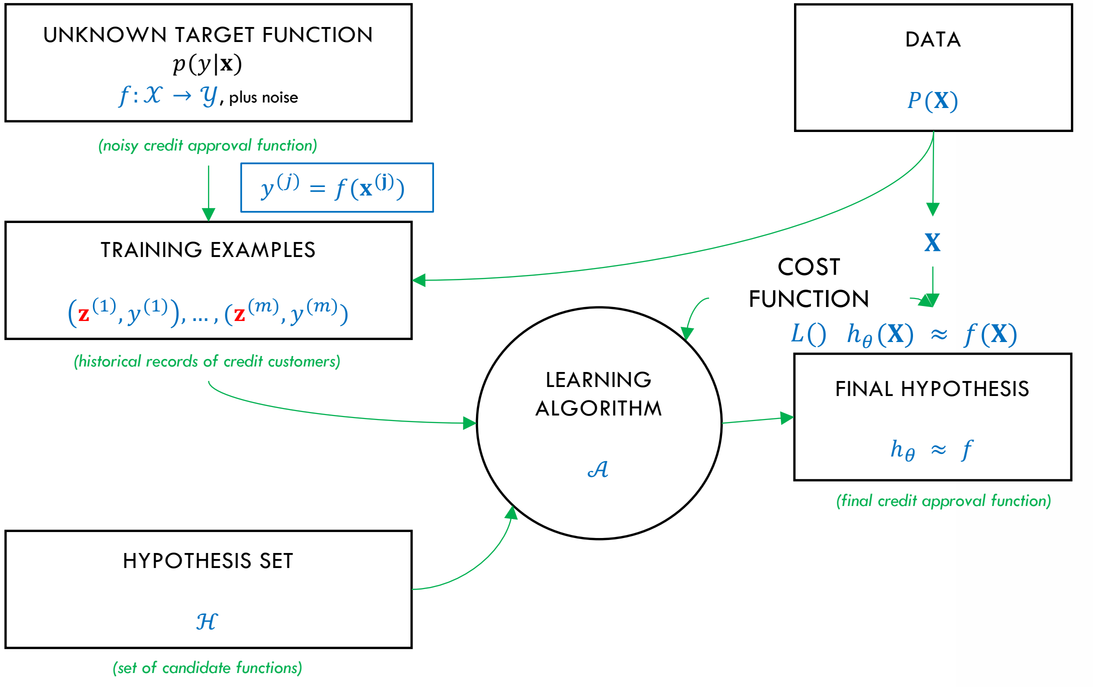
In $\mathcal{X}$ space, we say that we have "pre-images" of support vectors.

The margin is maintained in the $\mathcal{Z}$ space.

Great generalization, since in the model, # of parameters $\propto$ # of support vectors.



Final Learning Diagram

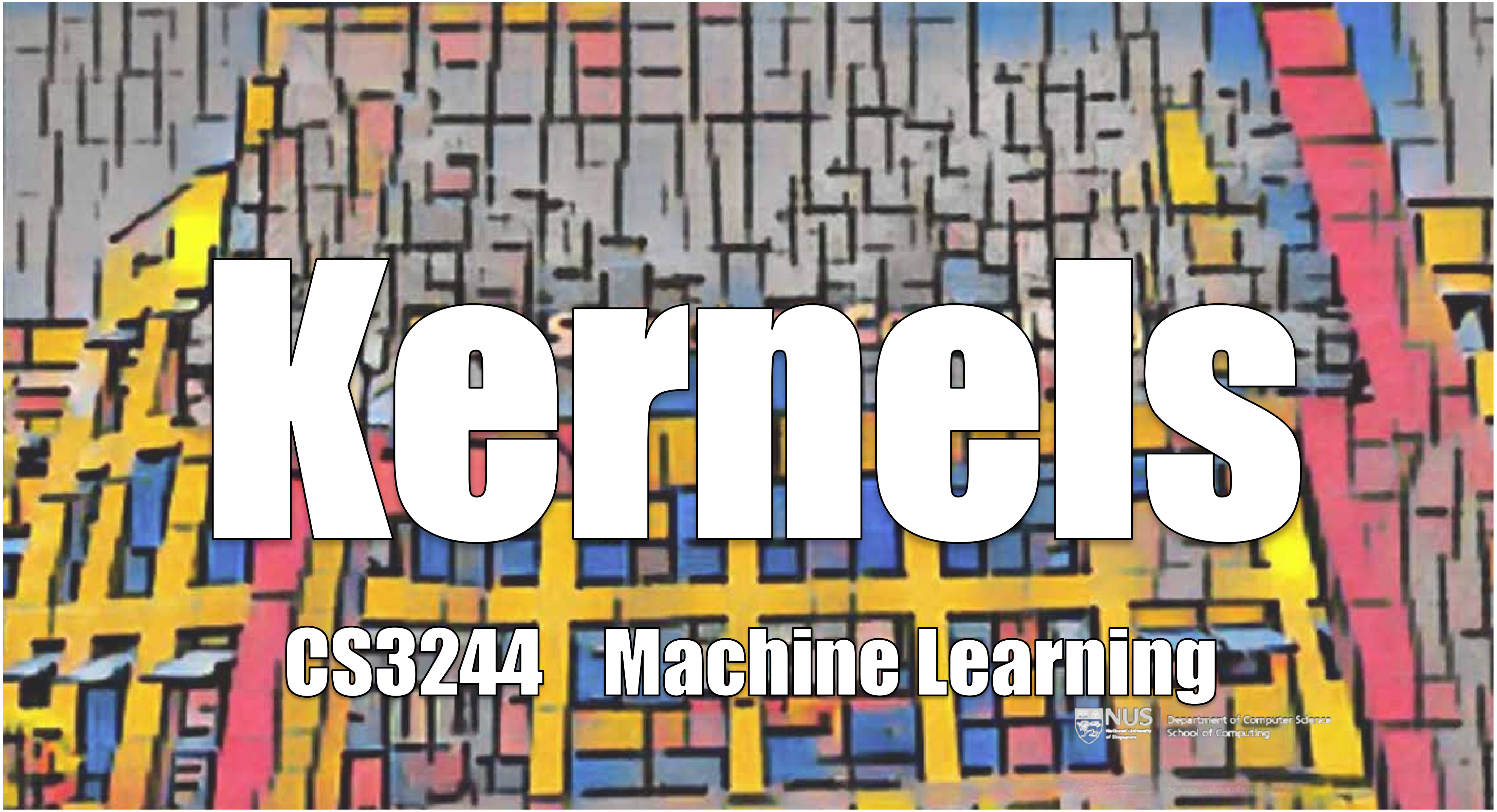


Nonlinear transformation

$\boldsymbol{\theta}^\top \mathbf{x}$ is linear in $\boldsymbol{\theta}$

Any $\mathbf{x} \xrightarrow{\Phi} \mathbf{z}$ preserves this linearity.

E.g., $(x_1, x_2) \xrightarrow{\Phi} (x_1^2, x_2^2)$


Kernels
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing

Pre-Lecture Activity from last week

Kernels

NUS National University of Singapore

Yes. Sometimes, in real-world problems, linear separation is not possible, and there might be curved separating hyperplane for data classification of linearly inseparable data (overlapping data). In such non-linear classification, the input data can be mapped into high-dimensional *feature space* using non-linear functions (feature or kernel functions), and linear classifier is then used for data classification.
Source: https://www.reneshbedre.com/blog/support-vector-machine.html (edited)

> RB **Renesh Bedre**
> **Support Vector Machine (SVM) basics and implementation in Python**
> Implementation of Support vector machine (SVM) in Python for prediction of heart disease. Learn SVM basics, model fitting, model accuracy, and interpretation

I think we can model non-linearity with linear classifiers by composing linear functions together (such as neural networks) or using non-linear kernels inside linear functions (such as logistic regression, it's form is linear but the kernel is non-linear so it can model non-linearity).

👀 1

Yes, it is possible to evaluate **model non-linearity with a linear classifier.** In the hyperplane. **Kernels** enable the linear SVM model to separate nonlinearly separable data points, also can use **Feature Transformation** to Introduce non-linearity into our Linear Model. then should evaluate the non-linearity.

Kernels and the Kernel Trick

A kernel is function that returns a distance (similarity) measure (often an inner product) of two instances:

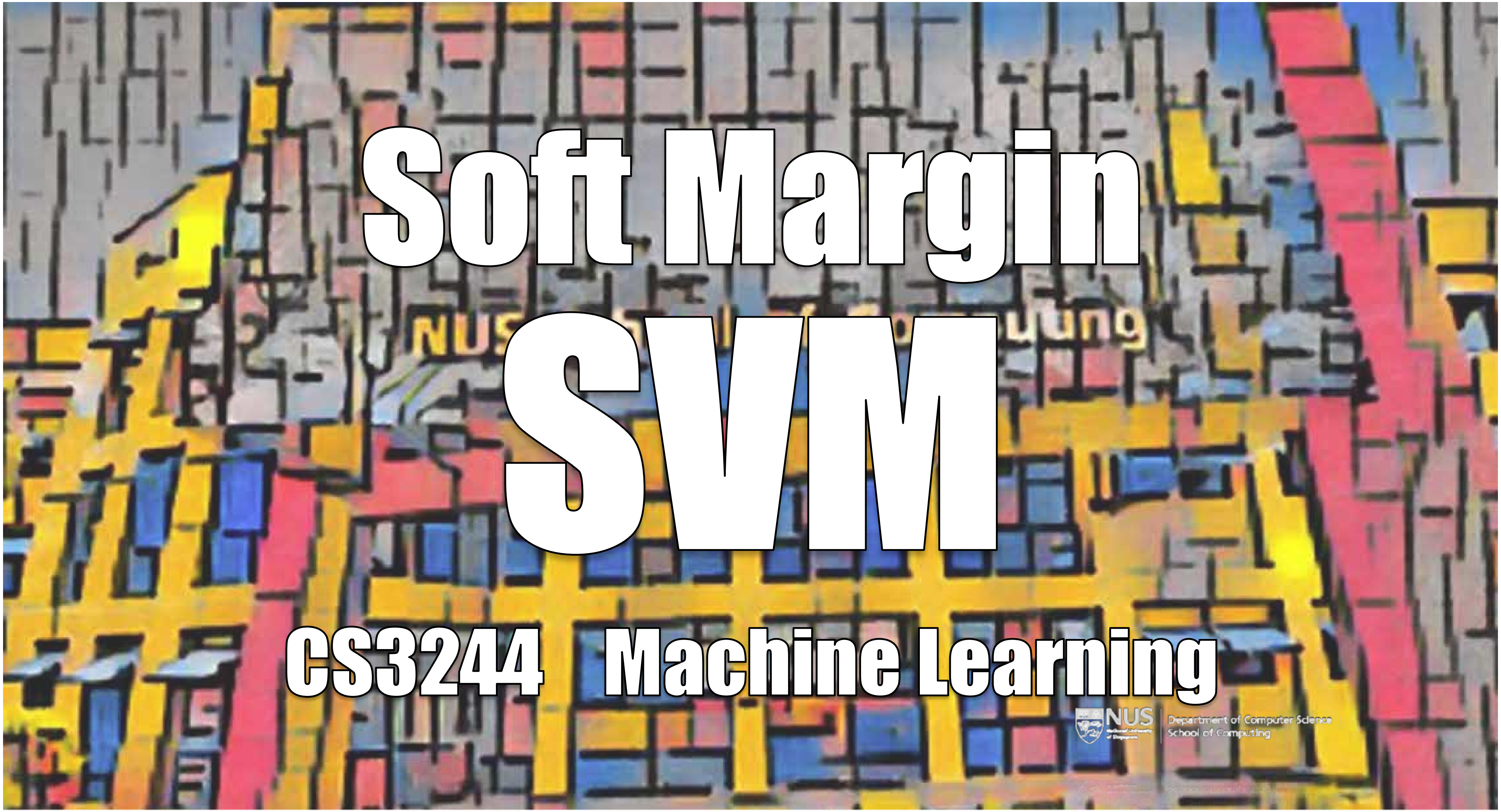
$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{z}^\top \boldsymbol{z}'$$

for some $\mathcal{Z}$ space.

Why is it a "*trick*"? We can directly compute this closed-form K function, but the actual transformation may be difficult or intractable to compute:

Polynomial kernel: $(1 + \mathbf{x}^\top \mathbf{x}')^{\mathrm{Q}}$

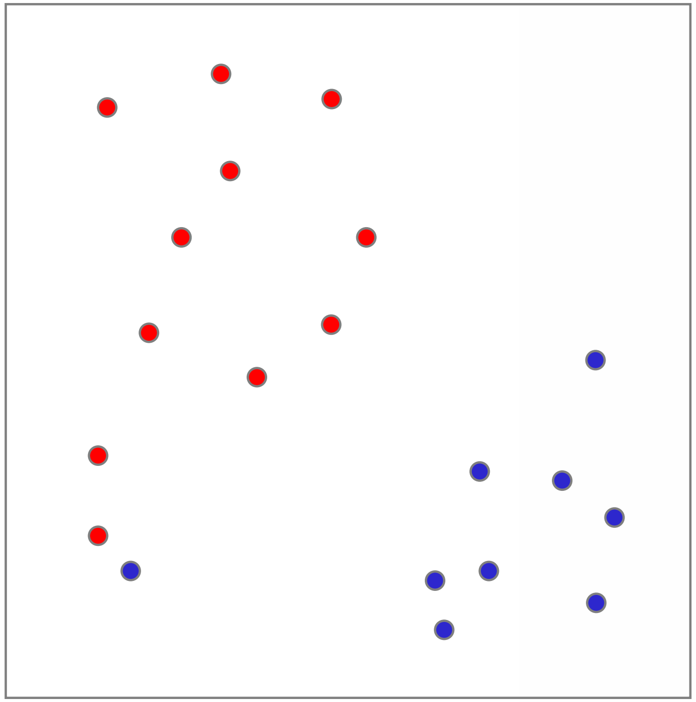
Some $\mathcal{A}$ compare distances between points (e.g., SVM, kNN). We can plug-in a suitable kernel K to compute similarity.



Soft Margin SVM

CS3244 Machine Learning

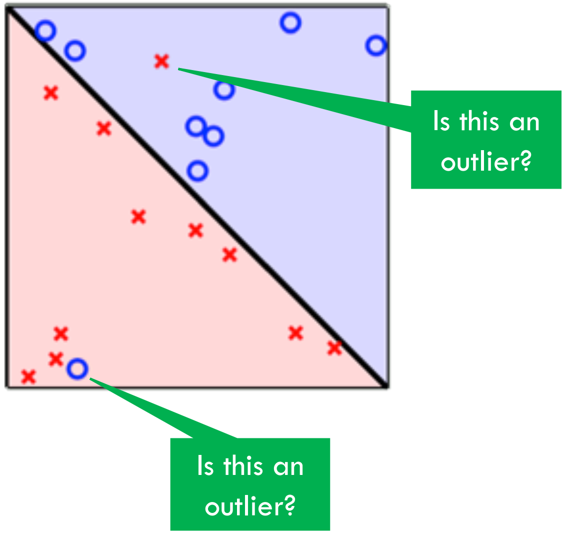
Land Transport Authority, Round 2



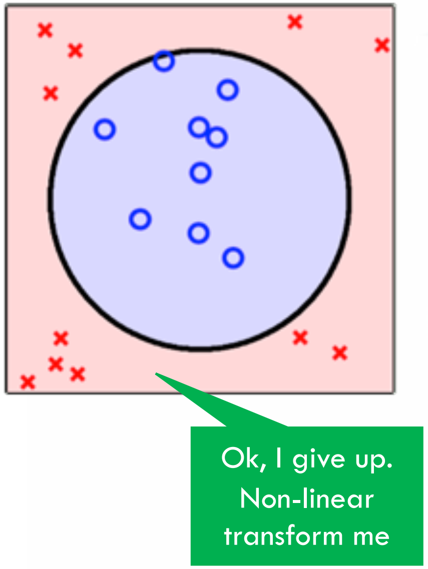
Tree Image credits: MuadLib @ Reddit

Two types of non-separable

Slightly



Seriously



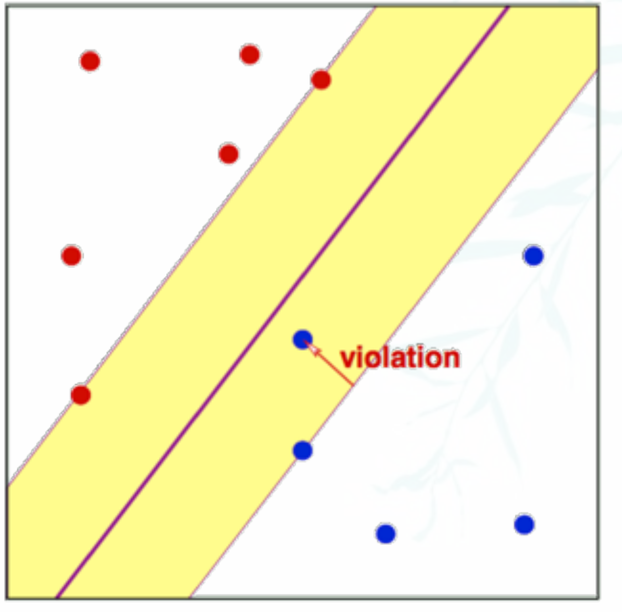
Cost Function w Slack Variables

Margin violation: $y^{(*)}\left(\boldsymbol{\theta}^{\top}\mathbf{x}^{(*)} + b\right) \geq 1$ fails

Quantify this:

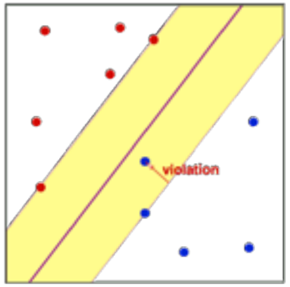
$y^{(*)}\left(\boldsymbol{\theta}^{\top}\mathbf{x}^{(*)} + b\right) \geq 1 - \xi^{(*)}$
where $\xi^{(*)} \geq 0$

Slack variable: Soft error on $(x^{(*)}, y^{(*)})$

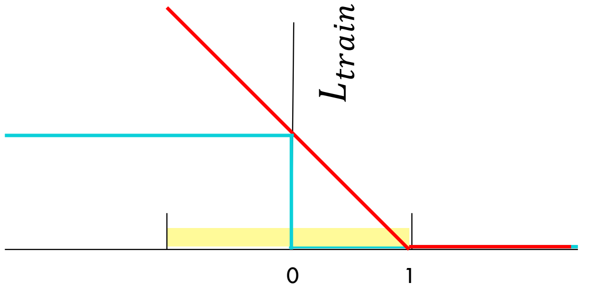


Total violation: $\sum_{j=1}^{m} \xi^{(j)}$

SVM's loss function: Hinge Loss



$$L_{train}(\boldsymbol{\theta}) = \sum_{j=1}^{m} max\left(0, 1 - y^{(j)}\left(\boldsymbol{\theta}^\top \mathbf{x}^{(j)} + b\right)\right)$$

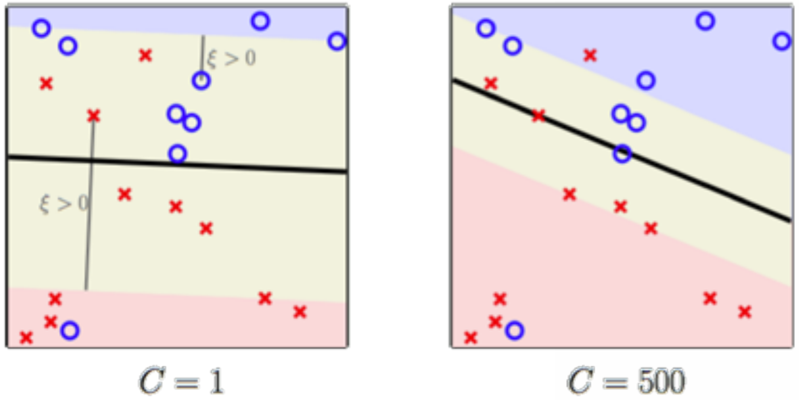


green = 0-1 loss
red = hinge loss

Soft margin SVM penalizes *misclassifications and correct classifications that fall inside the margin.*

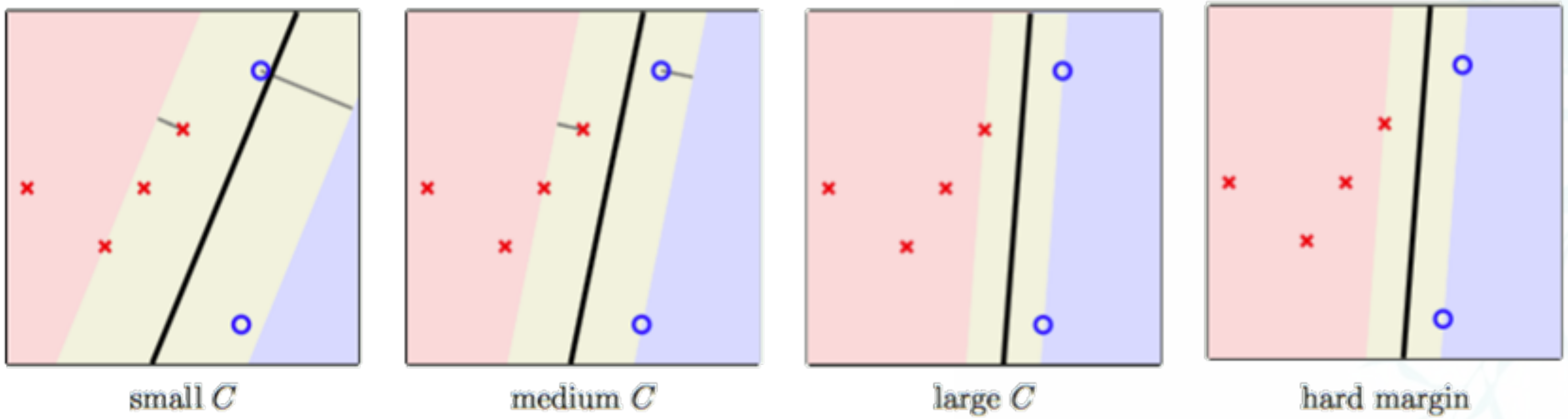
In hard margin SVM, there are, by definition, no misclassifications.

Soft Error C parameter



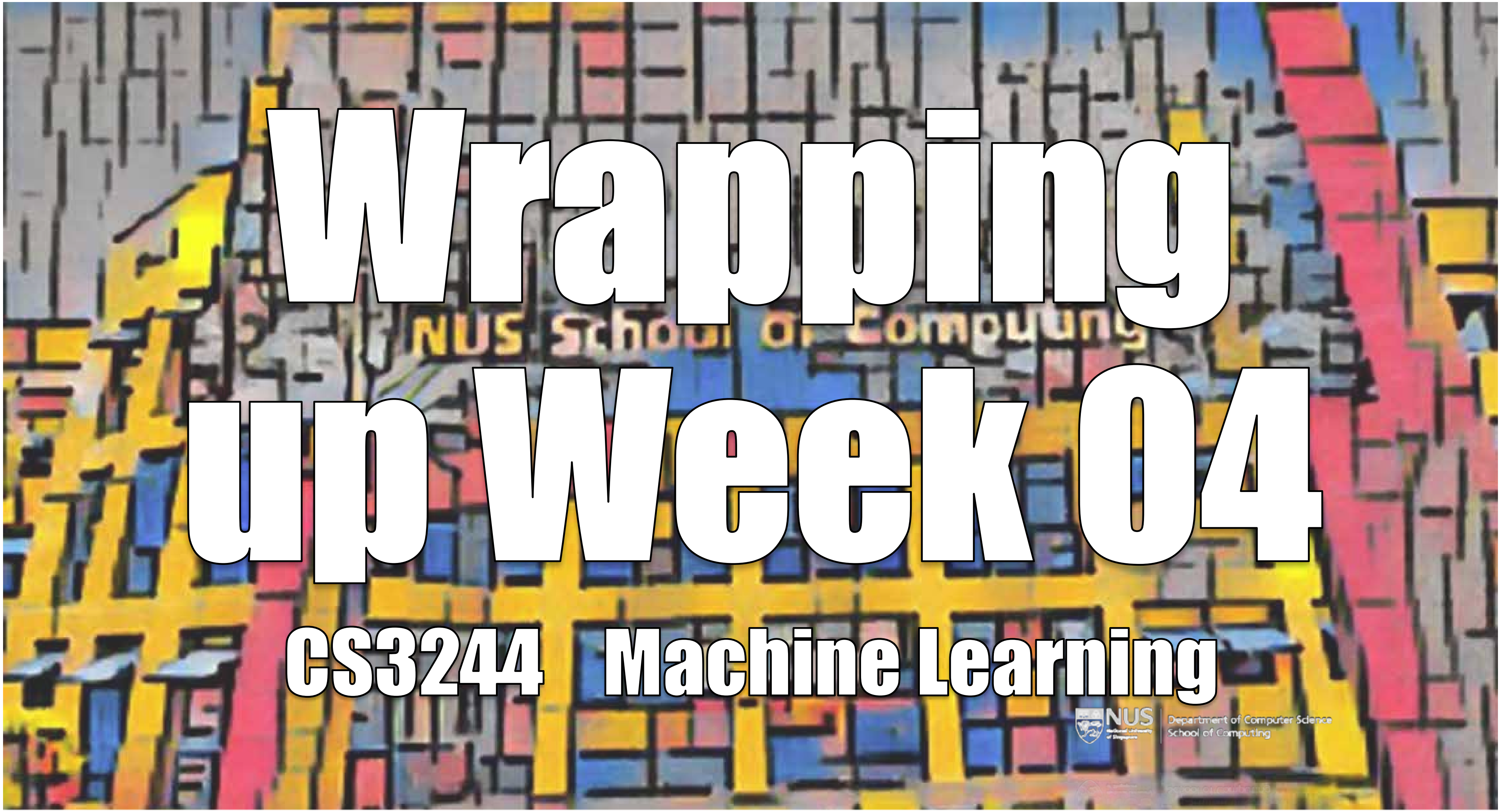
Soft error "badness". Higher values indicate less tolerance.

Effect of Varying C



Wrapping up Week 04

CS3244 Machine Learning



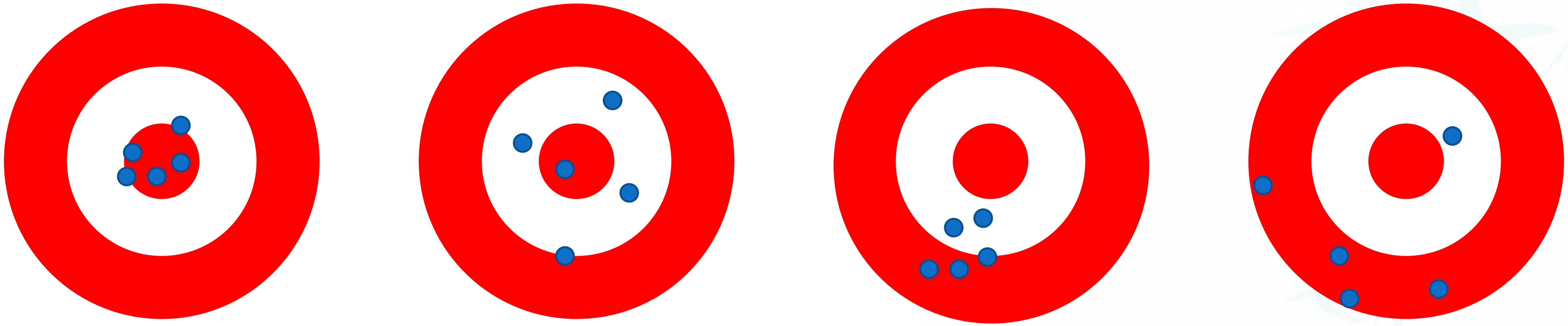
What did we learn this week?

- Describe the basic idea of linear classification;
- Understand how both linear and logistic regression works;
- Understand the Support Vector Machine Classifier as an optimal hyperplane;
- Understand how the optimization function is modified to allow errors (soft SVM).

Other important concepts:

- Curse of Dimensionality
- Gradient Descent
- Noisy Targets
- Non-linear Mappings

Outlook for next week



Assigned Task (due before next Mon)

Read the post https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229 (4 mins)

Post a 1–2 sentence answer to the topic in your tutorial group: #tg-xx

Describe kNN or decision trees with respect to variance.

[Don't worry if you're not sure about the math,
we'll cover this again in Week 05.]