# Linear Classifiers and Logistic Regression

4A

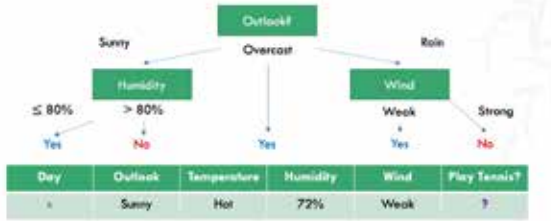
CS 3244
Machine Learning

NUS | Computing

# Recap from Week 03

NUS National University of Singapore

## Shall we play tennis?




| Day | Outlook | Temperature | Humidity | Wind | Play Tennis? |
|---|---|---|---|---|---|
| ... | Sunny | Hot | 72% | Weak | ? |

## Stock Market – Ensemble

We have $T$ reports $h_1, h_2, \ldots, h_T$ predicting whether a stock will go up as $h(x)$.

We can:

1. Select the most trustworthy of them based on their usual performance
   Training performance: $h(x) = h_{t^*}(x)$ with $t^* = \text{argmin}_{t \in \{1,2,\ldots,T\}} \epsilon_{\text{train}}(h_t)$
2. Let each report have a vote
   Uniform Vote: $h(x) = \text{sign}\left(\sum_{t=1}^{T} 1 \cdot h_t(x)\right)$
3. Weight the reports non-uniformly
   Weighted Vote: $h(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot h_t(x)\right)$ where $\alpha_t \geq 0$.
4. Combine the predictions conditionally
   This is decision trees, let's finish it up now!

Key: component hypotheses are diverse

## Q1. How do we pick a feature to split on?

To implement `Choose_Attribute` in DTL for enumerated feature sets with $C$ possible values, we first need a concept of purity. We'll use entropy:

$$H(X) = -\sum_{i=0}^{C} P_i \log_2(P_i)$$

▲ $P_i$ = probability, $p$ = # of positive examples

Example: for a training set containing $p$ positive examples and $n$ negative examples:

$$H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n} log_2\left(\frac{n}{p+n}\right)$$

NUS CS3244: Machine Learning 25

## During COVID-19 Pandemic

Are you sick?

*False negative* highly costly!
People and the economy dies.

*False positive* requires the inconvenience of quarantine.

$f \rightarrow \begin{cases} +1 & \text{sick} \\ \square & \\ -1 & \text{well} \end{cases}$

$h_\theta$

| $f$ \ $h_\theta$ | +1 | −1 |
|---|---|---|
| +1 | 0 | 1 |
| −1 | 10,000 | 0 |

# Project

Groupings need to be redone; sorry!

- Some incomplete submissions were supposed to be completed; please check your status.

Check the #projects channel bookmarks for details on proposal, final presentation and their evaluation metrics

New project groupings hopefully by Wednesday.

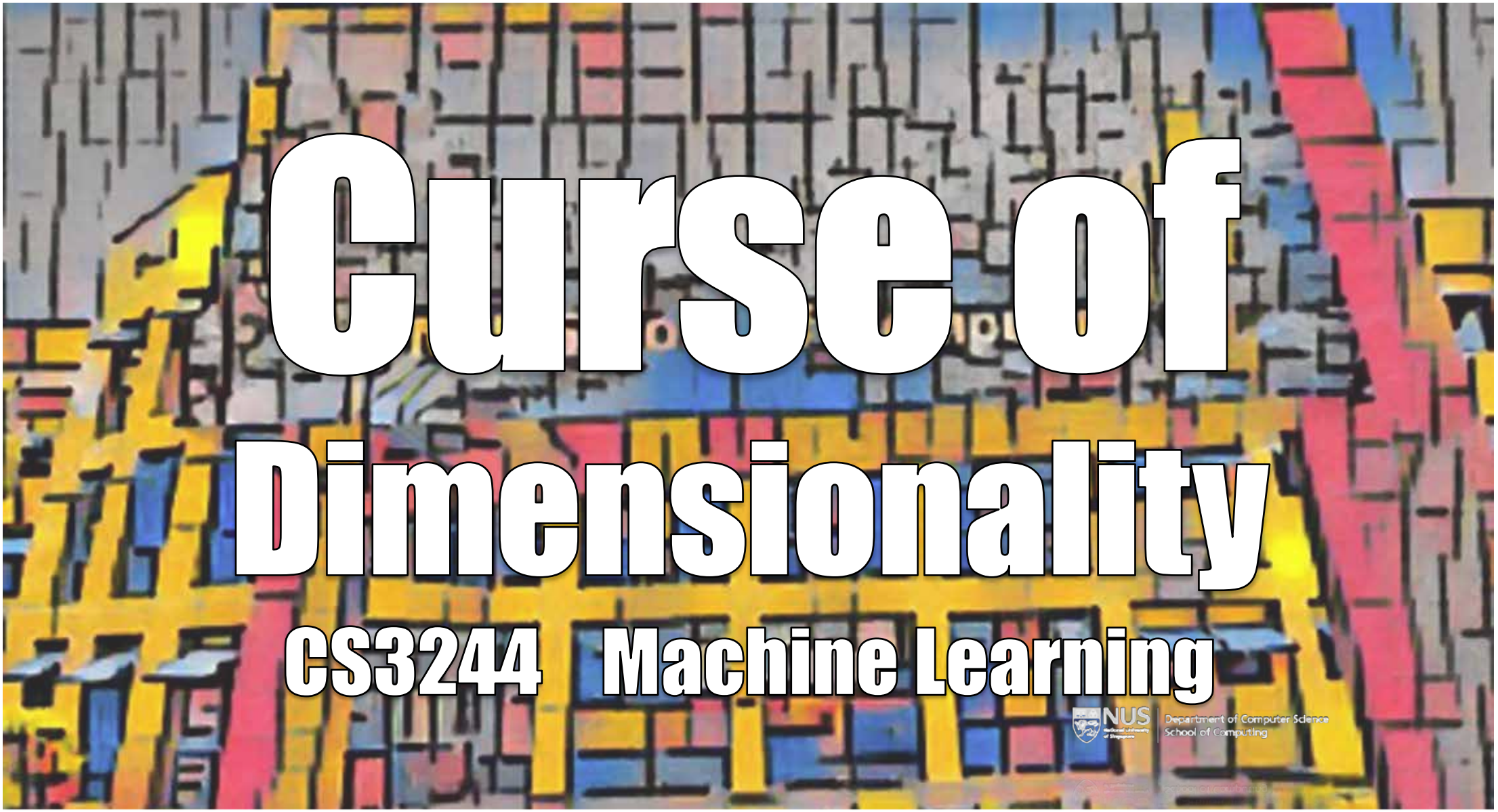
# Forecast for Week 04A

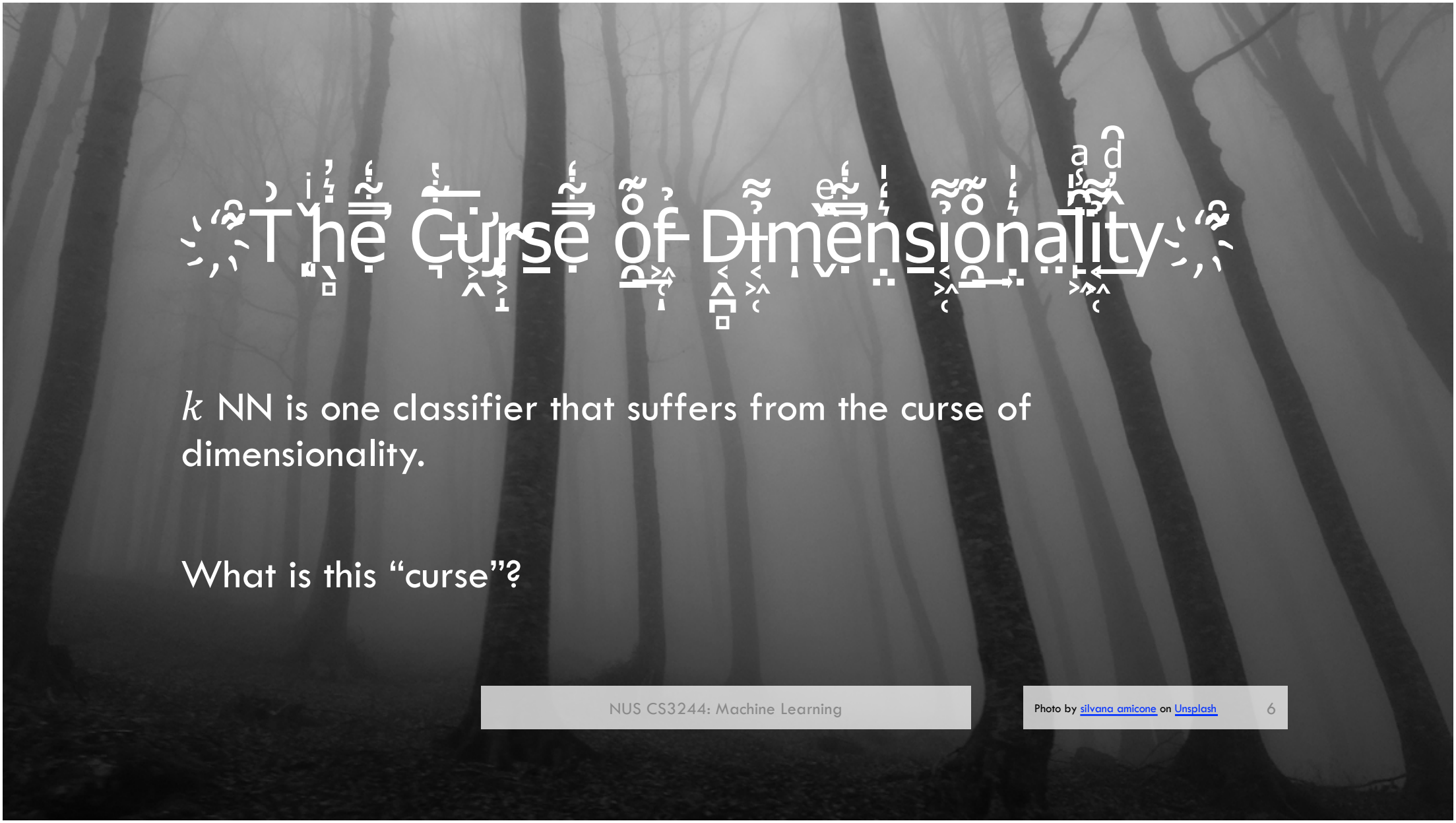

Learning Outcomes for this week

- Describe the basic idea of linear classification;
- Understand how both linear and logistic regression works;

Other important concepts:

- Curse of Dimensionality
- Gradient Descent


Curse of
Dimensionality
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing



# The Curse of Dimensionality

$k$ NN is one classifier that suffers from the curse of dimensionality.

What is this "curse"?

# Sparsity with high dimensions

$m = 5$

$n = 1$

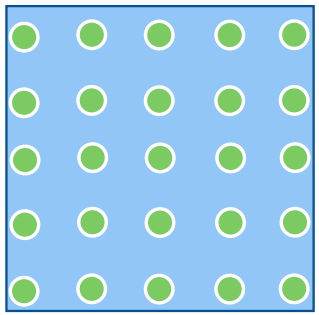


Chart illustration idea: Stanford CS229N

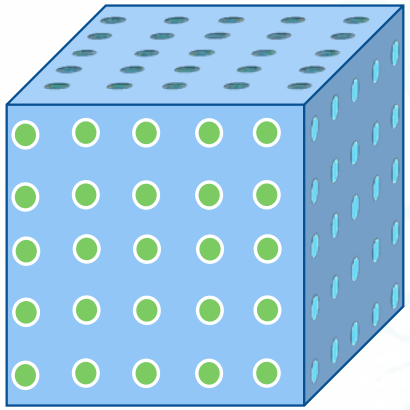
# Sparsity with high dimensions

$m = 5$
$n = 1$

$m = 25$
$n = 2$



$m = 125$
$n = 3$



Sparsity problem: maintaining density of samples depends on exponential growth of the data

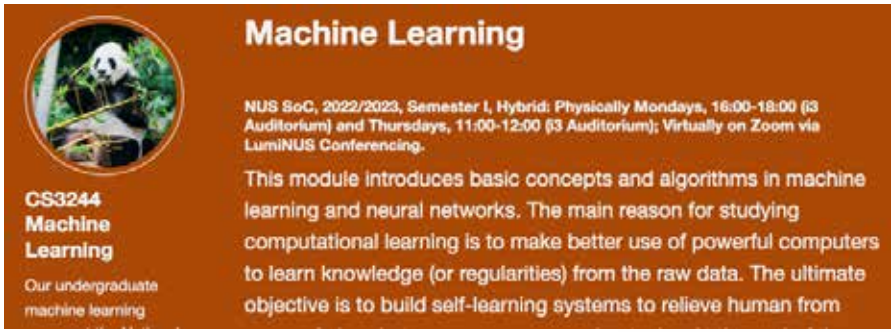


Chart illustration idea: Stanford CS229N

# Unit Hypercube with Colab

Let's Go!

http://www.comp.nus.edu.sg/~cs3244/AY22S1/04.colab.html



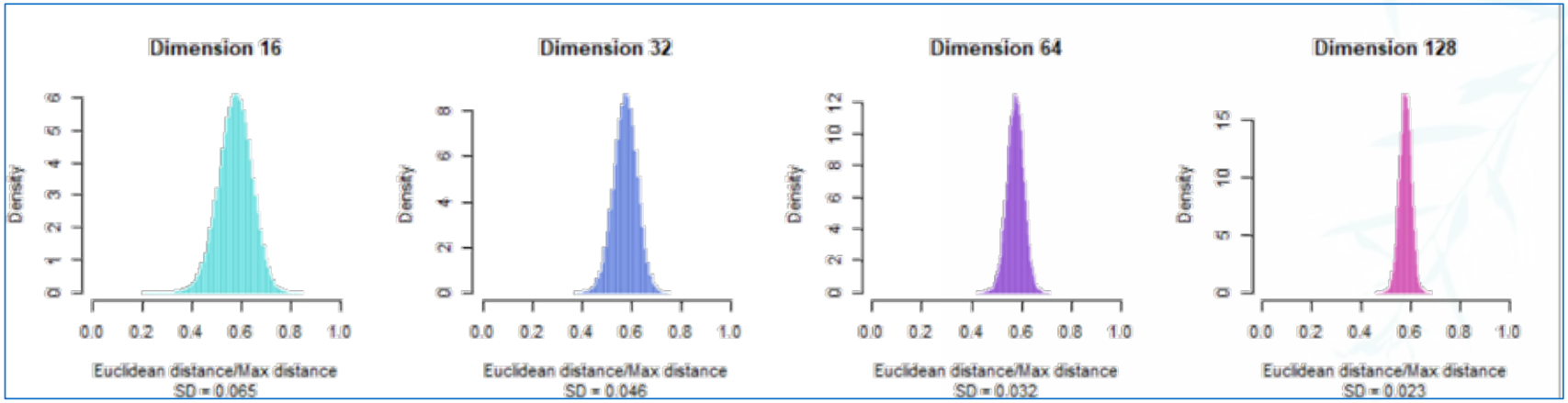
# Standard Deviation $\sigma$ grows slowly


```
1 dimensions:
#   Generated 200 points
Average distance (STD) of test point to 200 samples: 0.908554 (0.627194)
2 dimensions:
#   Generated 200 points
Average distance (STD) of test point to 200 samples: 1.357414 (0.672876)
3 dimensions:
#   Generated 200 points
Average distance (STD) of test point to 200 samples: 2.104206 (0.862900)
4 dimensions:
#   Generated 200 points
Average distance (STD) of test point to 200 samples: 2.857518 (0.851753)
8 dimensions:
#   Generated 200 points
Average distance (STD) of test point to 200 samples: 4.298752 (0.915833)
10 dimensions:
#   Generated 200 points
Average distance (STD) of test point to 200 samples: 4.240121 (0.873999)
100 dimensions:
#   Generated 200 points
Average distance (STD) of test point to 200 samples: 14.180229 (0.844941)
200 dimensions:
#   Generated 200 points
Average distance (STD) of test point to 200 samples: 19.588471 (0.845534)
```

# Curse of Dimensionality

In high dimensional space, most points are nearly the same distance away.

The result: learners that depend on distance break down in high dimensions.




https://stats.stackexchange.com/questions/451027/mathematical-demonstration-of-the-distance-concentration-in-high-dimensions

# Summary – in high dimensions

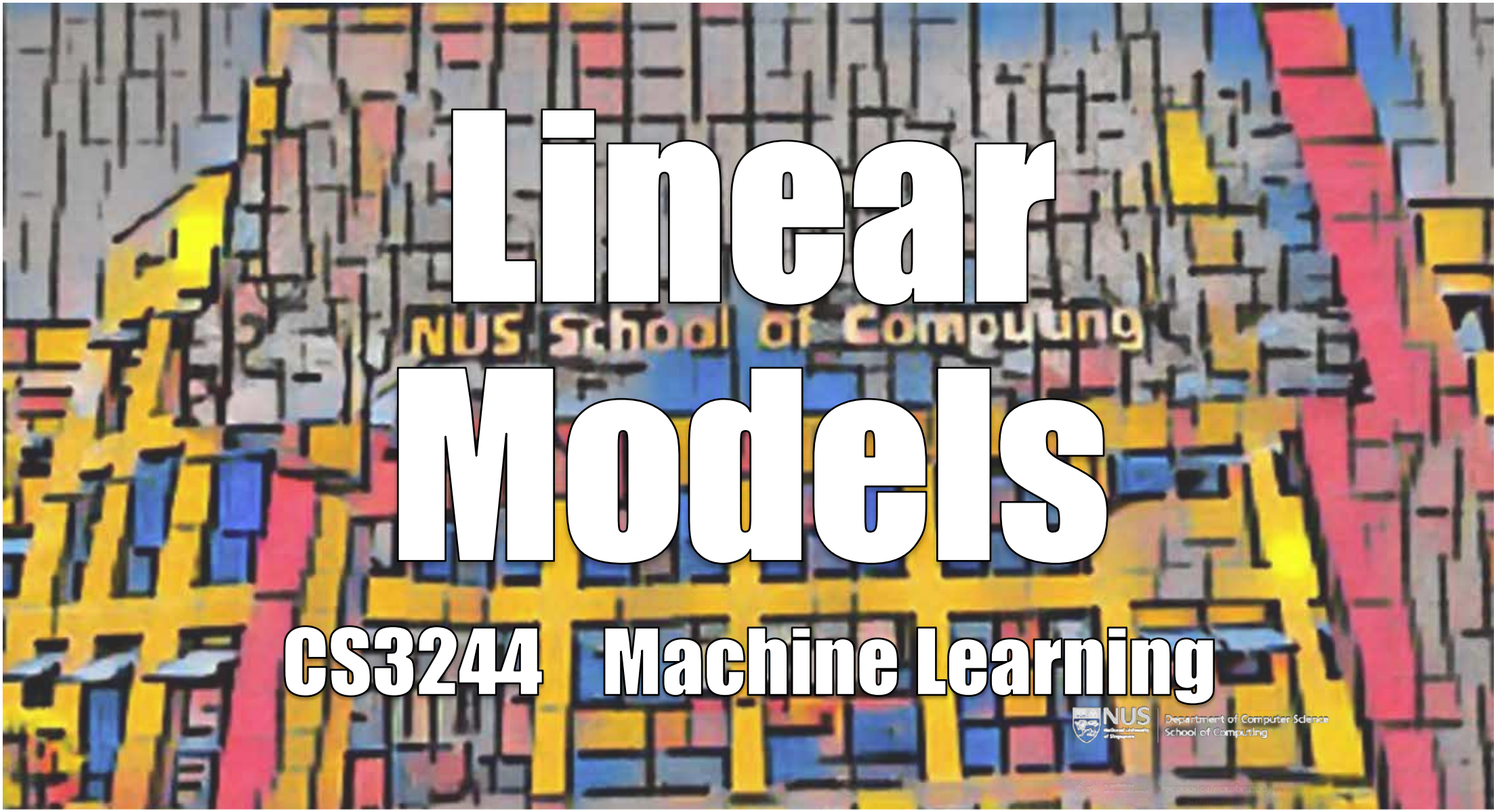
Hard to visualize, but 3B1B does a great job: *Thinking outside the 10-dimensional box* https://www.youtube.com/watch?v=zwAD6dRSVyI

Difficult to get sufficient samples

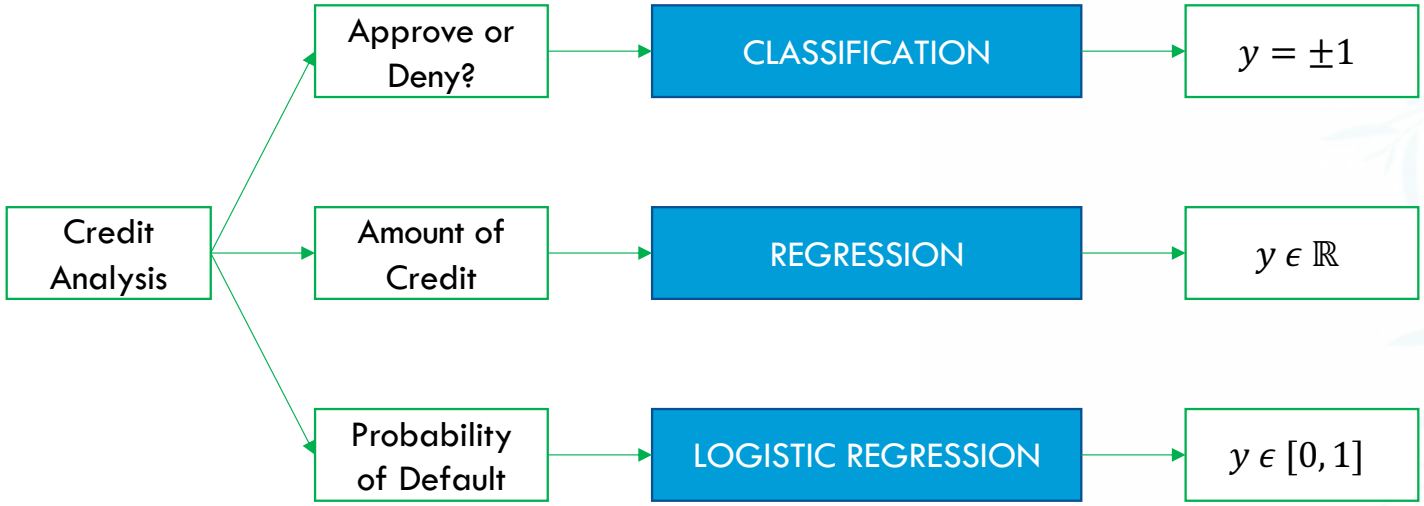
Standard distance metrics become less meaningful

- High dimensional hypercube is mostly "edge" space, far from the center of the hypercube

Solution? Choose or design an appropriate distance metric


Linear
Models
CS3244 Machine Learning
NUS School of Computing
NUS
Department of Computer Science
School of Computing

# Three learning problems

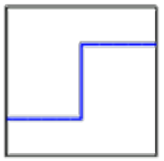



Linear models are the fundamental models.

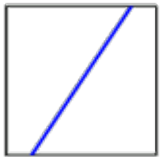
The linear model is the first model to try.

# The linear signal

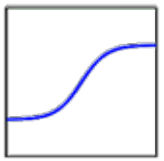
$s = \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}$



$\text{sign}\ (\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x})$ $\quad y = \pm 1$



$\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}$ $\quad y \in \mathbb{R}$



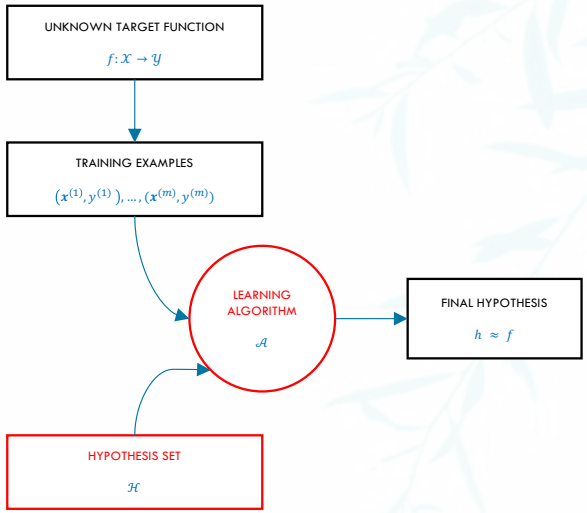
$g(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x})$ $\quad y \in [0, 1]$

# Quick Recap: Choosing a model

We choose a Learning Algorithm $\mathcal{A}$ (equivalent to choosing $\mathcal{H}$), and any associated hyperparameters.

We employ $\mathcal{A}$ to then choose an $h$ parameterized by its $\theta$.

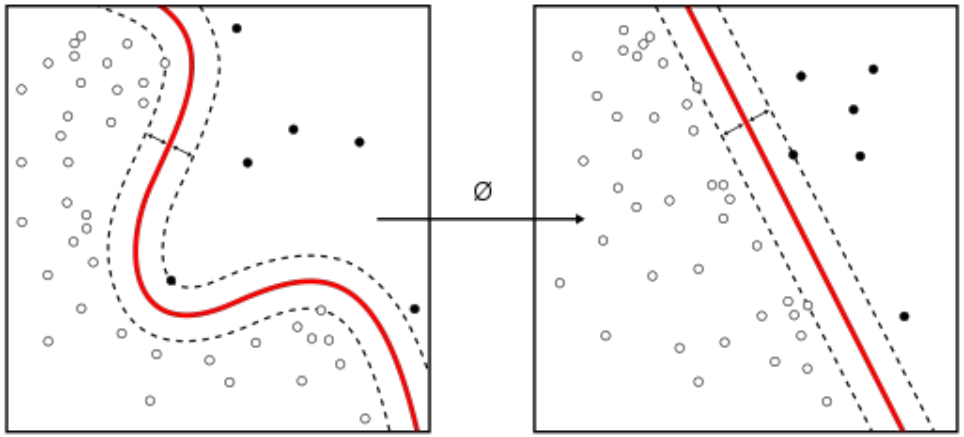
*i*.e., the learning algorithm $\mathcal{A}$ selects $h_\theta \in \mathcal{H}$.



# $\mathcal{H}$ for linear models

$\mathcal{H}$ represents the universe of possible hypotheses that a certain learning algorithm $A$ can generate.

That is, given the choices of $\theta$, what types of decision boundaries can it generate?



# 🏃‍♀️🏃 What do you think? 👍 or 👎


Do we extend credit to this person, if this is all the information we know?

| Criterion | Value |
|---|---|
| Age | 32 years |



Artwork credits: Project DebtRelief

# Credit Approval


Do we extend credit to this person?

| Criterion | Value |
|---|---|
| Age | 32 years |

$$h(x): [x > 25]$$

Logical test, Iverson brackets

Normalize $x$:

$$h(x): [\left(\frac{x}{100}\right) > 0.25]$$

The 'bias weight' $\theta_0$ corresponds to the threshold. $\theta_1$ and $\theta_0$ are parameters!.

Offset (Bias)

$$h(\mathbf{x}): \left(\frac{1}{100}\right) x_1 - (0.25)x_0 > 0$$

Introduce artificial $x_0$ as always equal to 1.



Artwork credits: Project DebtRelief

# Credit Approval


Do we extend credit to this person?

| Criterion | Value |
|---|---|
| Age | 32 years |
| ~~Gender~~ | ~~Male~~ |
| Salary | 40 K |



Artwork credits: Project DebtRelief

# Credit Approval


Do we extend credit to this person?

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

$$h(\mathbf{x}): \theta_2 x_2 + \left(\frac{1}{100}\right) x_1 + (-0.25) x_0 > 0$$

$$h(\mathbf{x}): \theta_3 x_3 + \theta_2 x_2 + \theta_1 x_1 + \theta_0 x_0 > 0$$

$$h(\mathbf{x}): \sum_{i=1}^{n} \theta_i x_i > 0$$



Artwork credits: Project DebtRelief

# Simplifying with vector notation

$$h(\boldsymbol{x}): \theta_n x_n + \cdots + \theta_0 x_0 > 0$$

Two vectors: $\boldsymbol{\theta} = \theta_n, \ldots, \theta_0$ and $\mathbf{x} = x_n, \ldots, x_0$.

 by default, vectors are _____ vectors

How to multiply?

# Simplifying with vector notation

How to multiply? _____

Which to transpose? $\boldsymbol{\theta}$ or $\mathbf{X}$ ?

$$\begin{bmatrix}\theta_0\\ \dots\\ \theta_n\end{bmatrix}\times[x_0 \quad \dots \quad x_n\,]=$$

$$[\theta_0 \quad \dots \quad \theta_n]\times\begin{bmatrix}x_0\\ \dots\\ x_n\end{bmatrix}=$$

# Simplifying with vector notation

How about with an entire matrix $\mathbf{X}$ of instances?

(Recall $\mathbf{X}$ is the data matrix, $\mathbf{x}$s stacked on top of each other)

Which to transpose? $\boldsymbol{\theta}^\top \mathbf{X}$ or $\mathbf{X}\boldsymbol{\theta}$?

$\mathbf{X}\boldsymbol{\theta} =$

e.g., $\mathbf{X}$ consisting stacked instances of $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$

$$\mathbf{X}\boldsymbol{\theta} =$$

$$= \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \theta_1 x_1^{(1)} + \theta_2 x_2^{(1)} + \theta_3 x_3^{(1)} \\ \theta_1 x_1^{(1)} + \theta_2 x_2^{(2)} + \theta_3 x_3^{(3)} \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{\theta}^\top \mathbf{x}^{(1)} \\ \boldsymbol{\theta}^\top \mathbf{x}^{(2)} \end{bmatrix}$$

$$= \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \end{bmatrix}$$

# A simple $\mathcal{H}$: hyperplane with a threshold (bias)

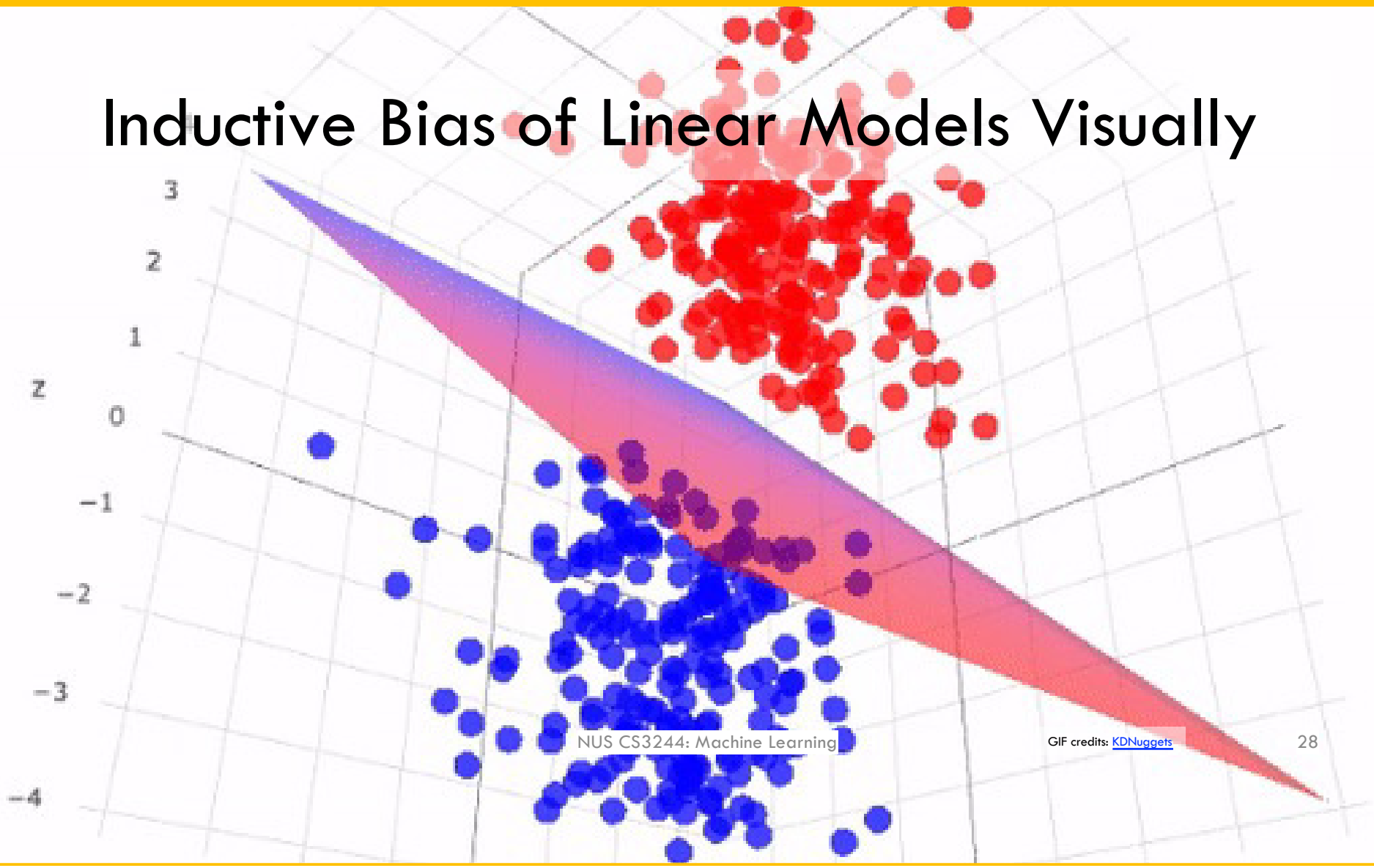
For an input $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ *'attributes of a customer'*

- Approve credit if: $\sum_{i=1}^{n} \theta_i x_i > 0$
- Deny credit otherwise: $\sum_{i=1}^{n} \theta_i x_i < 0$

This linear formula $h \in \mathcal{H}$ can be written as:

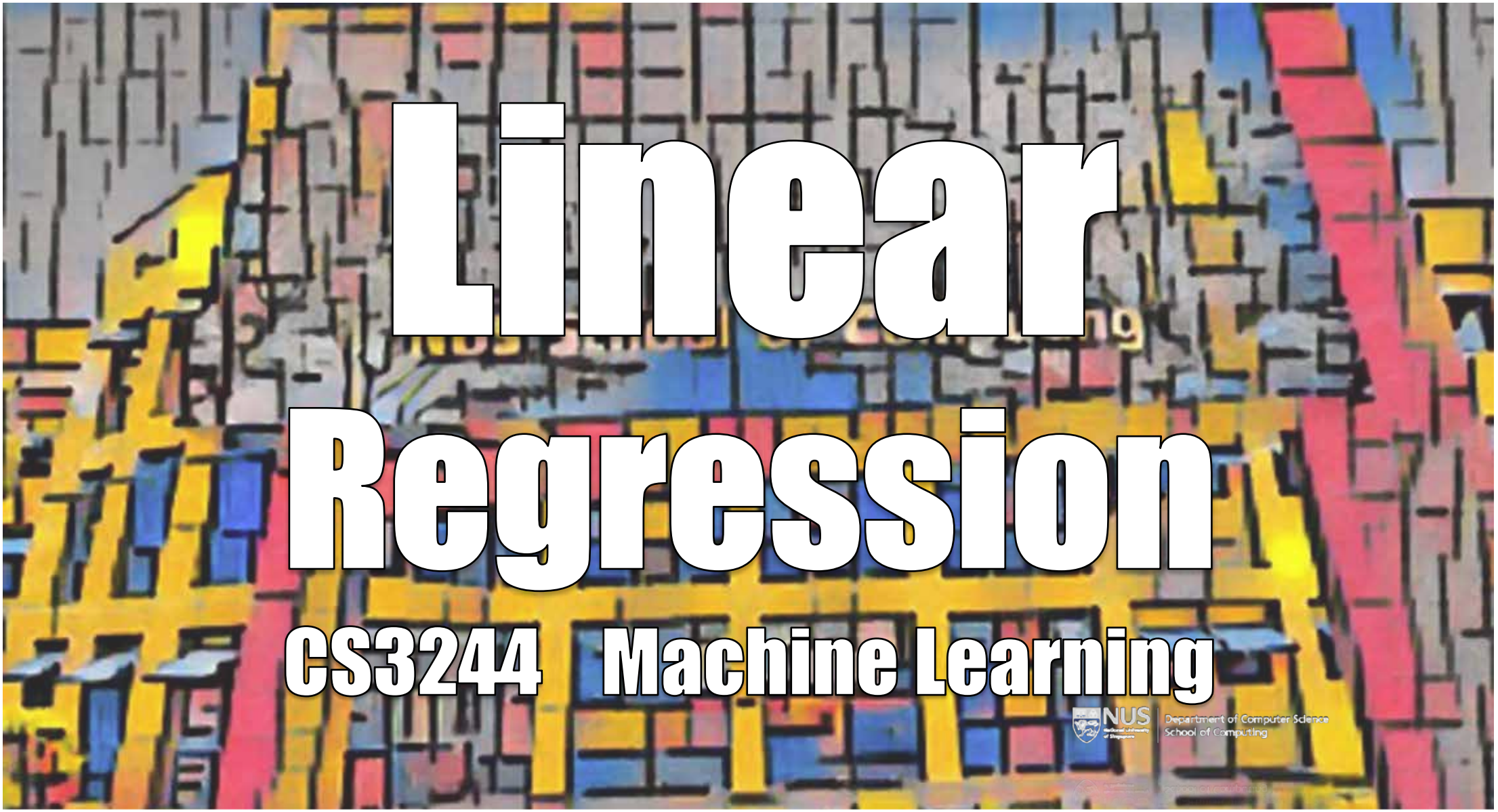
$$h_\theta(x) = \text{sign}\left(\sum_{i=1}^{n} \theta_i x_i\right)$$

# Inductive Bias of Linear Models Visually






GIF credits: KDNuggets


Linear
Regression
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing

# Credit Approval


How much credit do we extend this person?

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

~~Classification: Approve/Deny~~

- Regression: Credit line (real-valued dollar amount)
- Input: <*table on the left*>
- Linear regression output: $h_\theta(\mathbf{x}) = \sum_{i=0}^{n} \theta_i x_i = \boldsymbol{\theta}^\top \mathbf{x}$
  $= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
- Officers decide on credit lines based on historical data $\mathbf{X}$:
  $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

$y^{(j)} \in \mathbb{R}$ is the credit line for customer $x^{(j)}$;
regression tries to replicate this.



Artwork credits: Project DebtRelief

# Linear Regression

$$X\theta$$

# $\mathbb{R}$-valued cost function

How well does $h_\theta(\mathbf{x}) = \theta^\top \mathbf{x}$ approximate $f(\mathbf{x})$?

In linear regression, we use *squared error*: $(h_\theta(\mathbf{x}) - f(\mathbf{x}))^2$

Training error:

# $\mathbb{R}$-valued cost function

How well does $h_\theta(\mathbf{x}) = \theta^\top \mathbf{x}$ approximate $f(\mathbf{x})$?

In linear regression, we use *squared error*: $(h_\theta(\mathbf{x}) - f(\mathbf{x}))^2$

Training error:

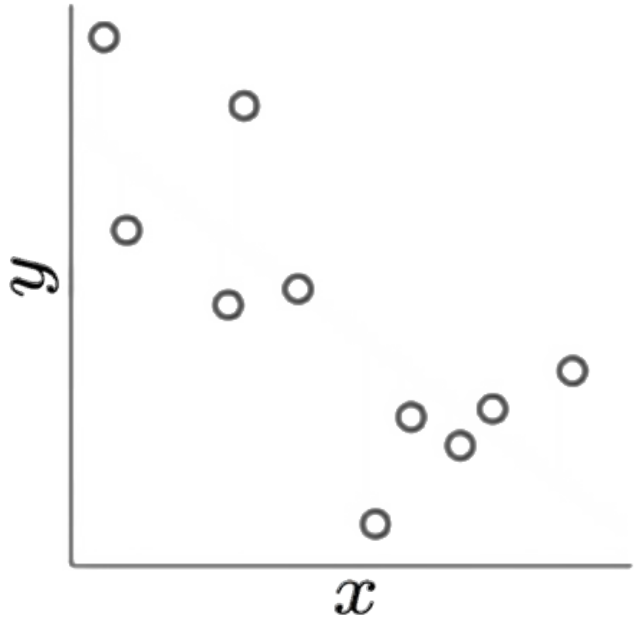
$$L_{train}(h) = \frac{1}{m}\sum_{j=1}^{m} (h_\theta(x^{(j)}) - y^{(j)})^2$$

*The sweet spot: Plausible AND Convenient*

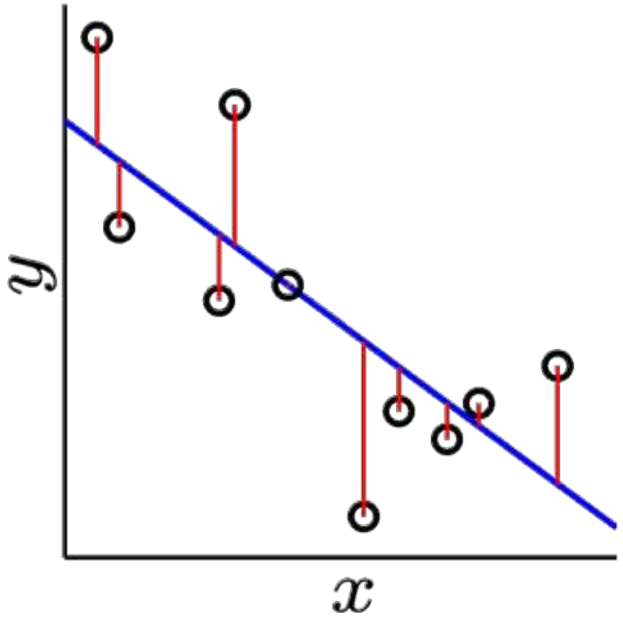
Pointwise Average

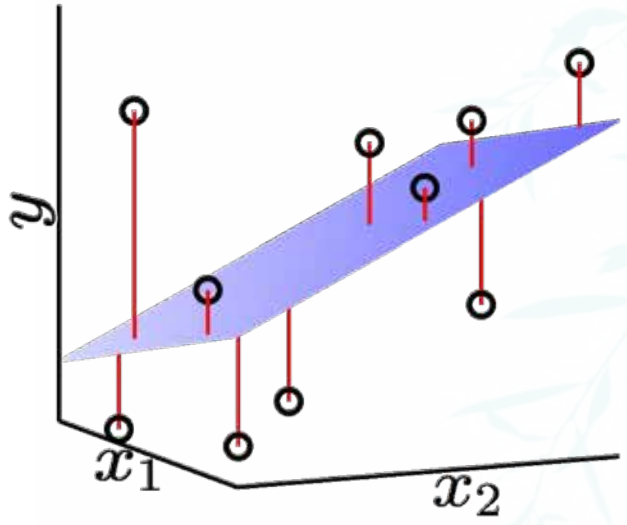
How bad is our prediction?

# Illustration of linear regression



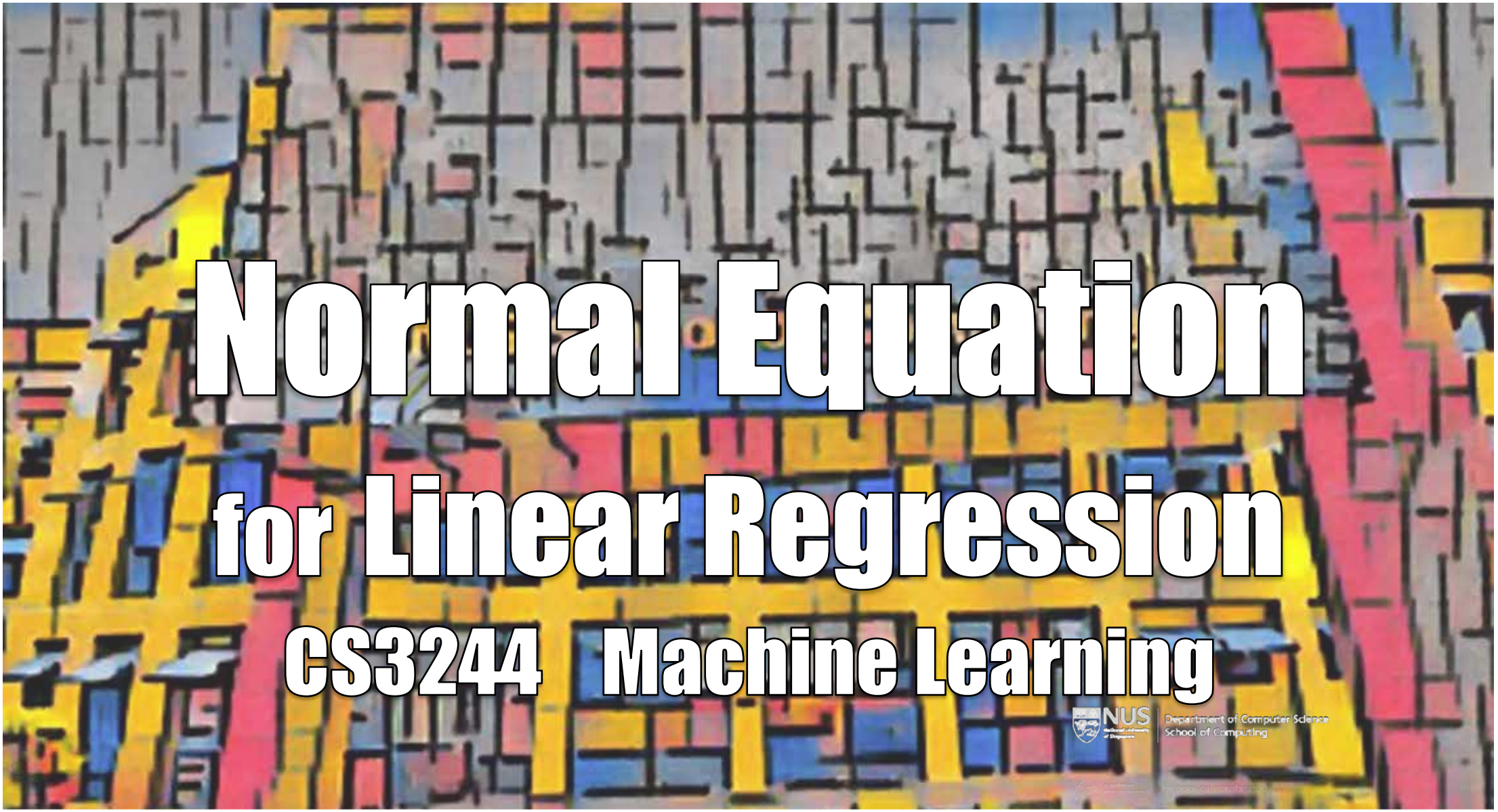
# Illustration of linear regression




Hypothesis: $h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$

$h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

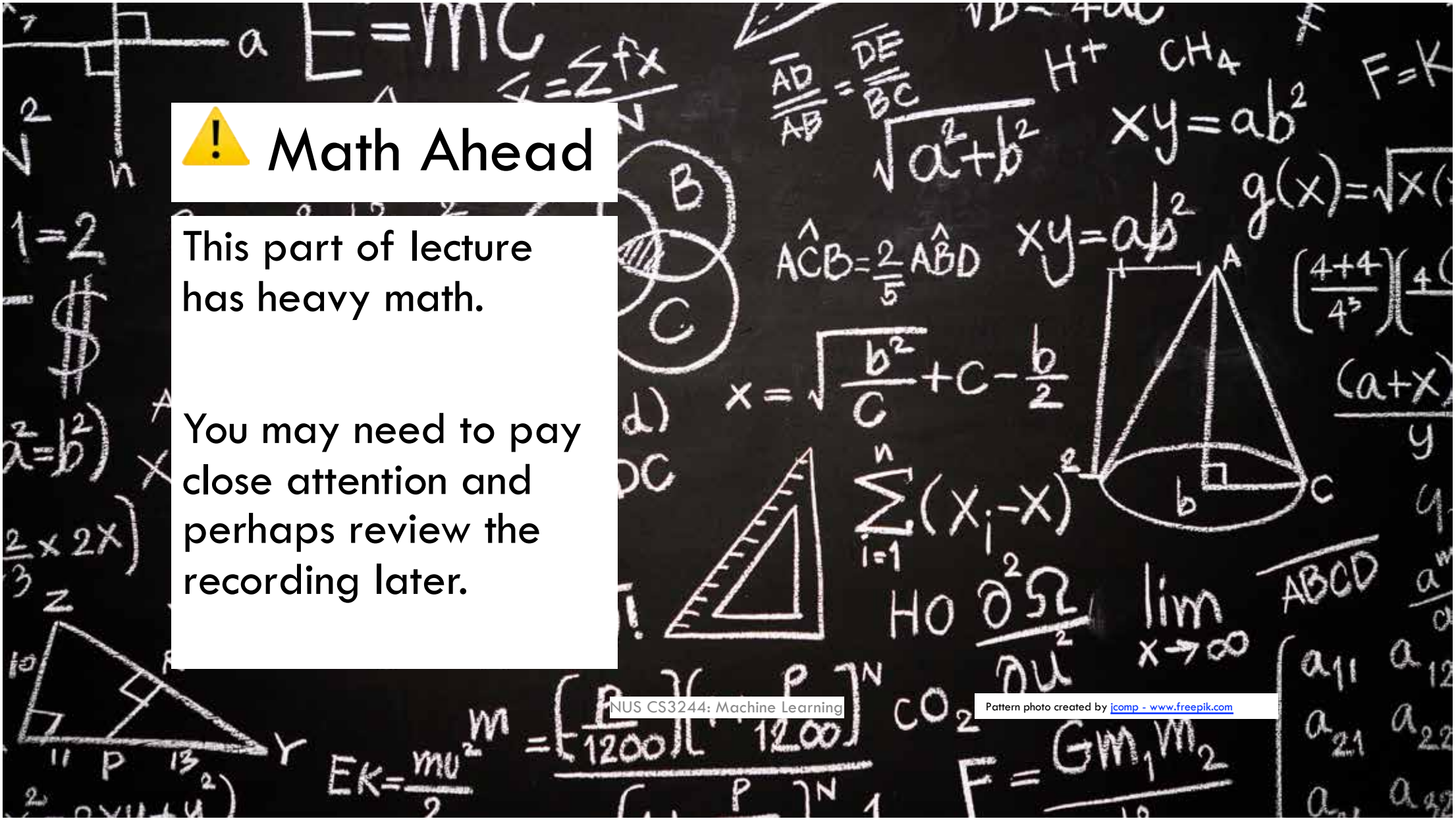

Normal Equation
for Linear Regression
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing

# Linear Regression – Summary

Two methods for solving

1. Gradient Descent

2. Normal Equation



# ⚠ Math Ahead

This part of lecture has heavy math.

You may need to pay close attention and perhaps review the recording later.

# The expression for $L_{train}$

$$L_{train} = \frac{1}{m}\sum_{j=1}^{m}(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}^{(j)} - y^{(j)})^2$$

$$= \frac{1}{m}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2$$

where $\mathbf{X} = \begin{bmatrix} -\mathbf{x}^{(1)\mathsf{T}}- \\ -\mathbf{x}^{(2)\mathsf{T}}- \\ -\mathbf{x}^{(3)\mathsf{T}}- \\ \vdots \\ -\mathbf{x}^{(m)\mathsf{T}}- \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix}$

# Minimizing $L_{train}$

Let's re-write $L_{train}$ first.

$$L_{train} = \frac{1}{m} \|\mathbf{X\theta} - \mathbf{y}\|^2$$

# Minimizing $L_{train}$

Let's re-write $L_{train}$ first.

$$L_{train} = \frac{1}{m}\|\mathbf{X\theta} - \mathbf{y}\|^2$$
$$= \frac{1}{m}(\mathbf{X\theta} - \mathbf{y})^\top(\mathbf{X\theta} - \mathbf{y})$$
$$= \frac{1}{m}\left( (\mathbf{X\theta})^\top\mathbf{X\theta} - (\mathbf{X\theta})^\top\mathbf{y} - \mathbf{y}^\top\mathbf{X\theta} + \mathbf{y}^\top\mathbf{y} \right)$$
$$= \frac{1}{m}(\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X\theta} - 2(\mathbf{X\theta})^\top\mathbf{y} + \mathbf{y}^\top\mathbf{y})$$

$\mathbf{y}$ and $\mathbf{X\theta}$ are both vectors, ordering in multiplication doesn't matter

Then solve for $\boldsymbol{\theta}$:

$$\frac{\partial L_{train}}{\partial \boldsymbol{\theta}} \equiv$$

$\nabla L \equiv$ A vector with all $n$ component partial derivatives of $L$.

$$\frac{\partial L_{train}}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial L}{\partial \theta_0}(\theta_0, \dots, \theta_n) \\ \dots \\ \frac{\partial L}{\partial \theta_n}(\theta_0, \dots, \theta_n) \end{bmatrix}$$

*Jacobian: stack of gradients as partial derivatives.*

# Minimizing $L_{train}$

Let's re-write $L_{train}$ first.

$$L_{train} = \frac{1}{m}\|\mathbf{X\theta} - \mathbf{y}\|^2$$
$$= \frac{1}{m}(\mathbf{X\theta} - \mathbf{y})^\top(\mathbf{X\theta} - \mathbf{y})$$
$$= \frac{1}{m}(\ (\mathbf{X\theta})^\top\mathbf{X\theta} - (\mathbf{X\theta})^\top\mathbf{y}$$
$$-\mathbf{y}^\top\mathbf{X\theta} \quad + \mathbf{y}^\top\mathbf{y}\ )$$
$$= \frac{1}{m}(\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X\theta} - 2(\mathbf{X\theta})^\top\mathbf{y} + \mathbf{y}^\top\mathbf{y})$$

Then solve for $\boldsymbol{\theta}$:

$$\frac{\partial L_{train}}{\partial \boldsymbol{\theta}} \equiv \mathbf{X}^\top\mathbf{X\theta} - \mathbf{X}^\top\mathbf{y} = \vec{\mathbf{0}}$$

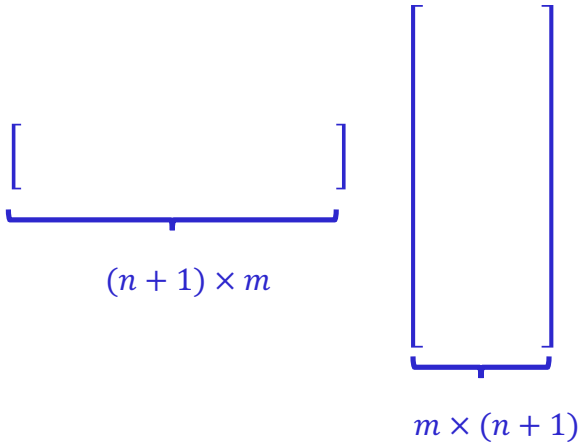
Column of zeroes

$$\mathbf{X}^\top\mathbf{X\theta} = \mathbf{X}^\top\mathbf{y}$$
$$\boldsymbol{\theta} = \left((\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right)\mathbf{y}$$

$(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ is called the **pseudo inverse** of $\mathbf{X}$

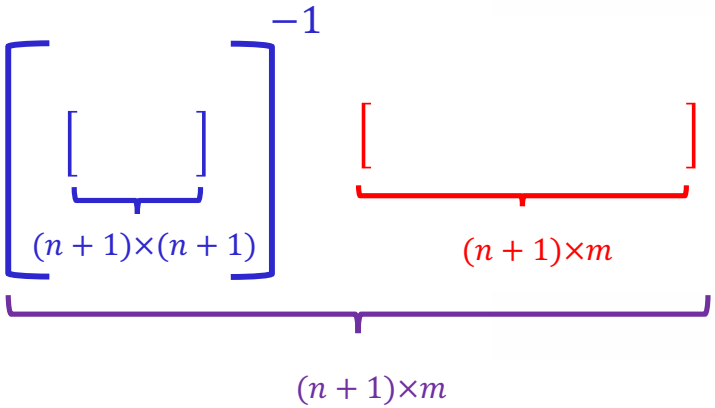
# The pseudo inverse $\mathbf{X}^\dagger$

$$\mathbf{X}^\dagger = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$$



# The pseudo inverse $\mathbf{X}^\dagger$

$$\mathbf{X}^\dagger = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$$




**Pseudo inverse** is expensive to compute if $n$ or $m$ is large.

# Linear regression by Normal Equation

1. Construct the matrix $\mathbf{X}$ and the vector $y$ from the data set as follows:

$$\mathbf{X} = \begin{bmatrix} -\mathbf{x}^{(1)\top} - \\ -\mathbf{x}^{(2)\top} - \\ -\mathbf{x}^{(3)\top} - \\ \vdots \\ -\mathbf{x}^{(m)\top} - \end{bmatrix}, y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

input data matrix target vector

1. Compute the pseudo inverse $\mathbf{X}^{\dagger} = (\mathbf{X}^{\top}\mathbf{X})^{-1}\mathbf{X}^{\top}$
2. Return $\theta = \mathbf{X}^{\dagger}y$

*The normal equation for LR. One-step learning!*

# Linear Regression – Summary

Two methods for solving

1. Gradient Descent
   - Works well, even when $n$ is large
   - Works even if $\mathbf{X}^\top\mathbf{X}$ is non-invertible

🤔 *To think about: What are some reasons that $\mathbf{X}^\top\mathbf{X}$ might not be invertible?*

2. Normal Equation
   - Don't need to choose $\alpha$
   - Don't need to iterate
   - Need to compute $(\mathbf{X}^\top\mathbf{X})^{-1}$, an $O(n^3)$ operation

# Why is it called "regression"?

The term is usually attributed to Sir Francis Galton. He observed that children's heights tended to be less extreme than their parents; the heights of the children ***regressed*** to the average adult height; hence he titled his paper "*Regression towards Mediocrity in Hereditary Stature*."

For Galton, "regression" referred only to the tendency of extreme data values to "revert" to the overall mean value. (Minitab Blog, 1)



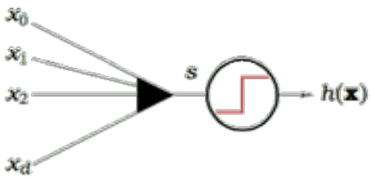
Photo by David Straight on Unsplash


Logistic
Regression
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing

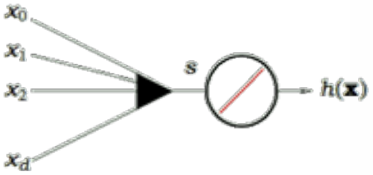
# A third linear model

$$s = \sum_{i=0}^{n} \theta_i x_i$$

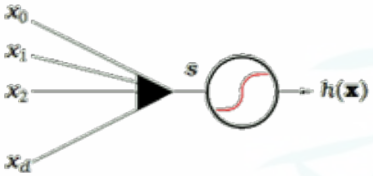
Linear classification




Linear regression




Logistic regression



# The logistic function $g$

We choose a form for $g(\cdot)$ out of convenience.
We use the logistic function, which has the form:

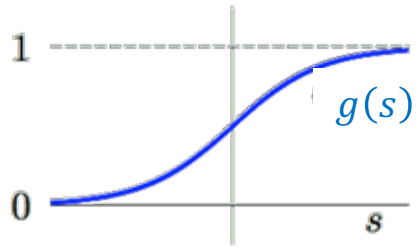
$$g(s) = \frac{exp^s}{1 + exp^s}$$

# The logistic function $g$

We choose a form for $g(\cdot)$ out of convenience.
We use the logistic function, which has the form:

$$g(s) = \frac{exp^s}{1 + exp^s}$$

$g(s)$: needs to map any $x$: $\mathbb{R}$ to $y$: $[0,1]$




Soft threshold: uncertainty

Sigmoid: flattened out 's' shape

# Probability Interpretation

$h_\theta(\mathbf{x}) = g(s)$ is interpreted as a probability

Logistic regression $\equiv y \in [0,1]$

Example: *Prediction of heart attacks*

Input $\mathbf{x}$: cholesterol level, age, weight, diabetic, etc.

output $g(s)$: Likelihood of a heart attack

The signal: $s = \theta^\top \mathbf{x}$

"risk score"



Photo credits: Wikipedia - Chinese Buffet

# Labels are binary, yet our outputs are probabilities

$$\mathbf{X} = \left(\mathbf{x}^{(1)}, y^{(1)} = \pm 1\right), \ldots \left(\mathbf{x}^{(m)}, y^{(m)} = \pm 1\right)$$

$\mathbf{x}^{(j)}$ – a person's health information

$y^{(j)} = \pm 1$ – did they have a heart attack?

**Why?**

We cannot measure a probability.

We can only see the occurrence of an event and try to infer a probability.

# Genuine Probability

Data $(\mathbf{x}, y)$ with binary $y$, generated by a noisy target:

$$P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1; \\ 1 - f(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

The target $f: \mathbb{R}^n \to [0,1]$ is the probability

Learn $h_\theta(\mathbf{x}) = g(\boldsymbol{\theta}^\top \mathbf{x}) \approx f(\mathbf{x})$

# Cost Function for Logistic Regression

For logistic regression,

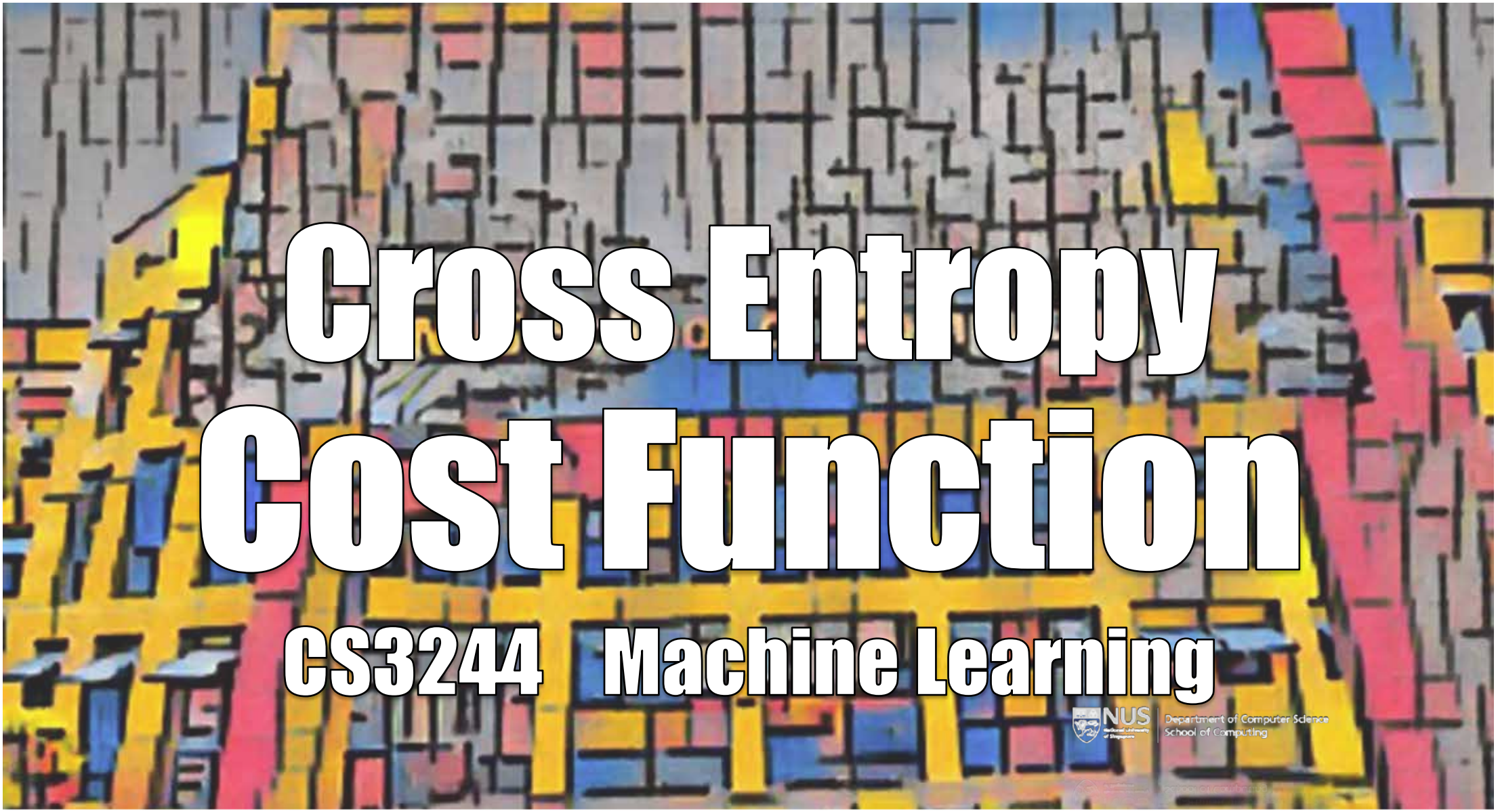
$$L_{train}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{j=1}^{m} ln(1 + exp^{-y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}})$$

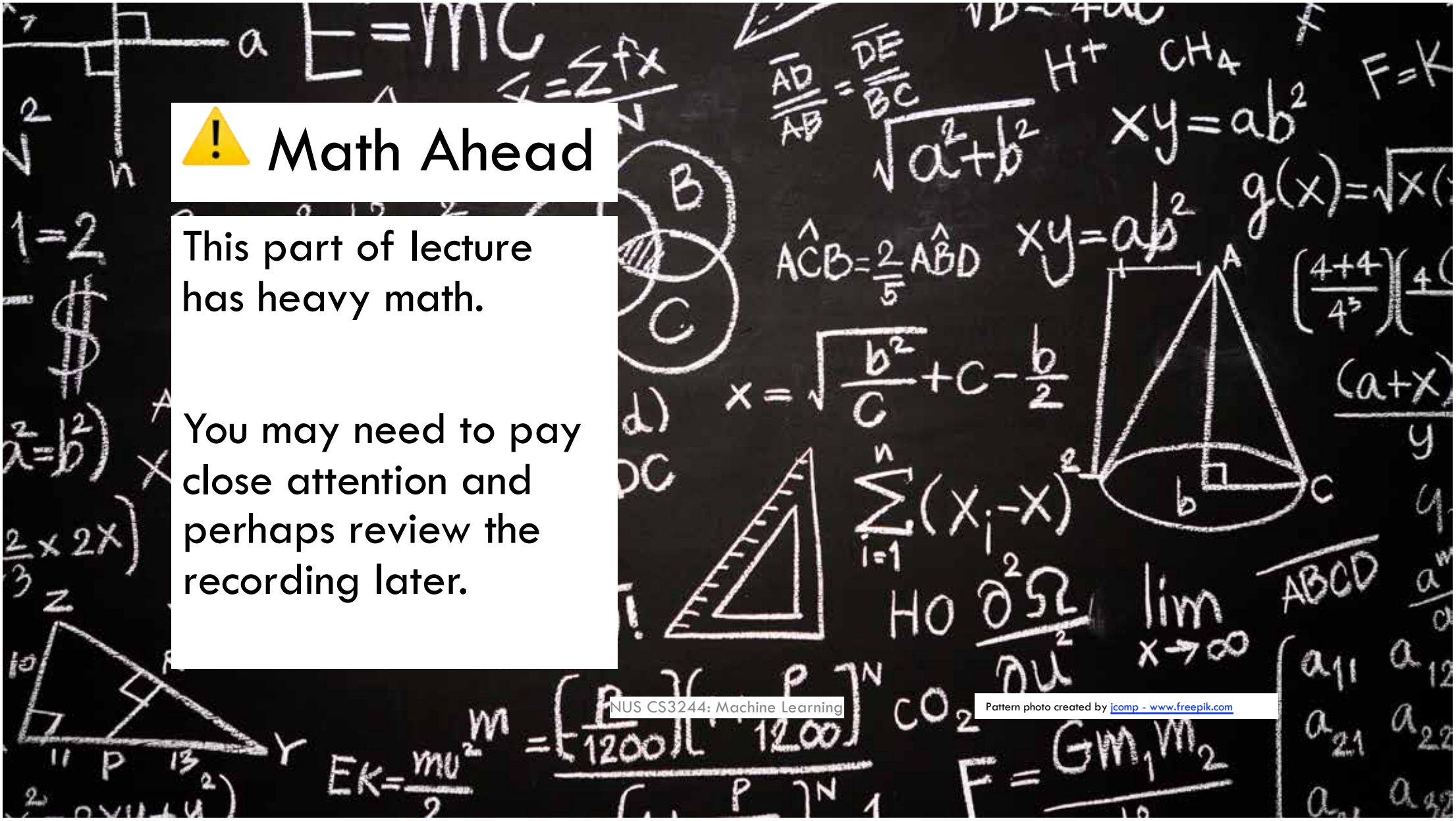
Iterative Solution; yucky (or interesting) math

Compare to linear regression:

$$L_{train}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{j=1}^{m}\left(\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)} - y^{(j)}\right)^2$$

Closed-form solution; but iteration also possible


Cross Entropy
Cost Function
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing



# ⚠ Math Ahead

This part of lecture has heavy math.

You may need to pay close attention and perhaps review the recording later.

# Logistic Regression Cost Function

For each $(\mathbf{x}, y)$, $y$ is generated by probability $f(\mathbf{x})$

Plausible cost function based on likelihood:

If $h_\theta = f$, how "likely" is it to obtain output $y$ from input $\mathbf{x}$?

$$P(y|\mathbf{x}) = \begin{cases} h_\theta(\mathbf{x}) & \text{for } y = +1; \\ 1 - h_\theta(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

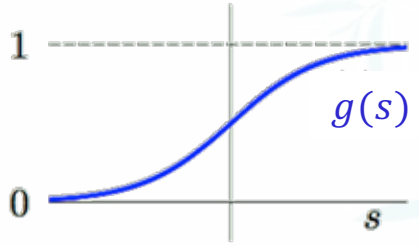
# Formula for likelihood

$$P(y|\mathbf{x}) = \begin{cases} h_\theta(\mathbf{x}) & \text{for } y = +1; \\ 1 - h_\theta(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

Substitute $h_\theta(\mathbf{x}) = g(\boldsymbol{\theta}^\top \mathbf{x})$,

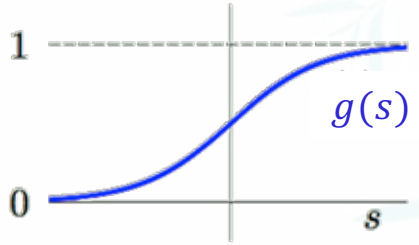
noting $g(-s) = 1 - g(s)$

$$P(y|\mathbf{x}) = g(y \cdot \boldsymbol{\theta}^\top \mathbf{x})$$



# Formula for likelihood

$$P(y|\mathbf{x}) = \begin{cases} h_\theta(\mathbf{x}) & \text{for } y = +1; \\ 1 - h_\theta(\mathbf{x}) & \text{for } y = -1. \end{cases}$$




Substitute $h_\theta(\mathbf{x}) = g(\boldsymbol{\theta}^\top \mathbf{x})$,
noting $g(-s) = 1 - g(s)$

$$P(y|\mathbf{x}) = g(y \cdot \boldsymbol{\theta}^\top \mathbf{x})$$

Likelihood of $\mathbf{X} = (\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})$
where samples are i.i.d. is:

$$P(y^{(1)}, .., y^{(m)} | \mathbf{x}^{(1)}, .., \mathbf{x}^{(m)}) = \prod_{j=1}^{m} P(y^{(j)} | \mathbf{x}^{(j)}) = \prod_{j=1}^{m} g(y^{(j)} \boldsymbol{\theta}^\top \mathbf{x}^{(j)})$$

# Maximizing the likelihood ≡ Minimizing cross entropy

$$= max \prod_{j=1}^{m} g\left(y^{(j)} \boldsymbol{\theta}^{\top} \mathbf{x}^{(j)}\right)$$

# Maximizing the likelihood ≡ Minimizing cross entropy

$$= max \quad \prod_{j=1}^{m} g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)$$

$$\Leftrightarrow max \quad \frac{1}{\mathrm{m}}\ln\left(\prod_{j=1}^{m} g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)\right)$$

$$= max \quad \frac{1}{m}\sum_{j=1}^{m} \ln g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)$$

$$\Leftrightarrow min \quad -\frac{1}{m}\sum_{j=1}^{m} \ln g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)$$

# Maximizing the likelihood ≡ Minimizing cross entropy

$$= max \quad \prod_{j=1}^{m} g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)$$

$$\Leftrightarrow max \quad \frac{1}{m}\ln\left(\prod_{j=1}^{m} g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)\right)$$

$$= max \quad \frac{1}{m}\sum_{j=1}^{m}\ln g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)$$

$$\Leftrightarrow min \quad -\frac{1}{m}\sum_{j=1}^{m}\ln g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)$$

$$= min \quad \frac{1}{m}\sum_{j=1}^{m}\ln\left(\frac{1}{g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)}\right)$$

$$g(s) = \frac{e^{s}}{1+e^{s}} = \left[g(s) = \frac{1}{1+e^{-s}}\right]$$

$$L_{train}(\theta) = \frac{1}{m}\sum_{j=1}^{m}\underbrace{\ln(1+e^{-y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}})}$$

"cross entropy" error

# Maximizing the likelihood ≡ Minimizing cross entropy

$$= max \quad \prod_{j=1}^{m} g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)$$

$$\Leftrightarrow max \quad \frac{1}{m}\ln\left(\prod_{j=1}^{m} g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)\right)$$

$$= max \quad \frac{1}{m}\sum_{j=1}^{m}\ln g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)$$

$$\Leftrightarrow min \quad -\frac{1}{m}\sum_{j=1}^{m}\ln g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)$$

$$= min \quad \frac{1}{m}\sum_{j=1}^{m}\ln\left(\frac{1}{g\left(y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}\right)}\right)$$

$$g(s) = \frac{e^{s}}{1+e^{s}} = \left[g(s) = \frac{1}{1+e^{-s}}\right]$$

$$= min \quad \frac{1}{m}\sum_{j=1}^{m}\ln\left(1+e^{-y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}}\right)$$

Plug in

$$L_{train}(\theta) = \frac{1}{m}\sum_{j=1}^{m}\underbrace{\ln(1+e^{-y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}})}$$

"cross entropy" error

# Summary: Minimizing $L_{train}$

For logistic regression,

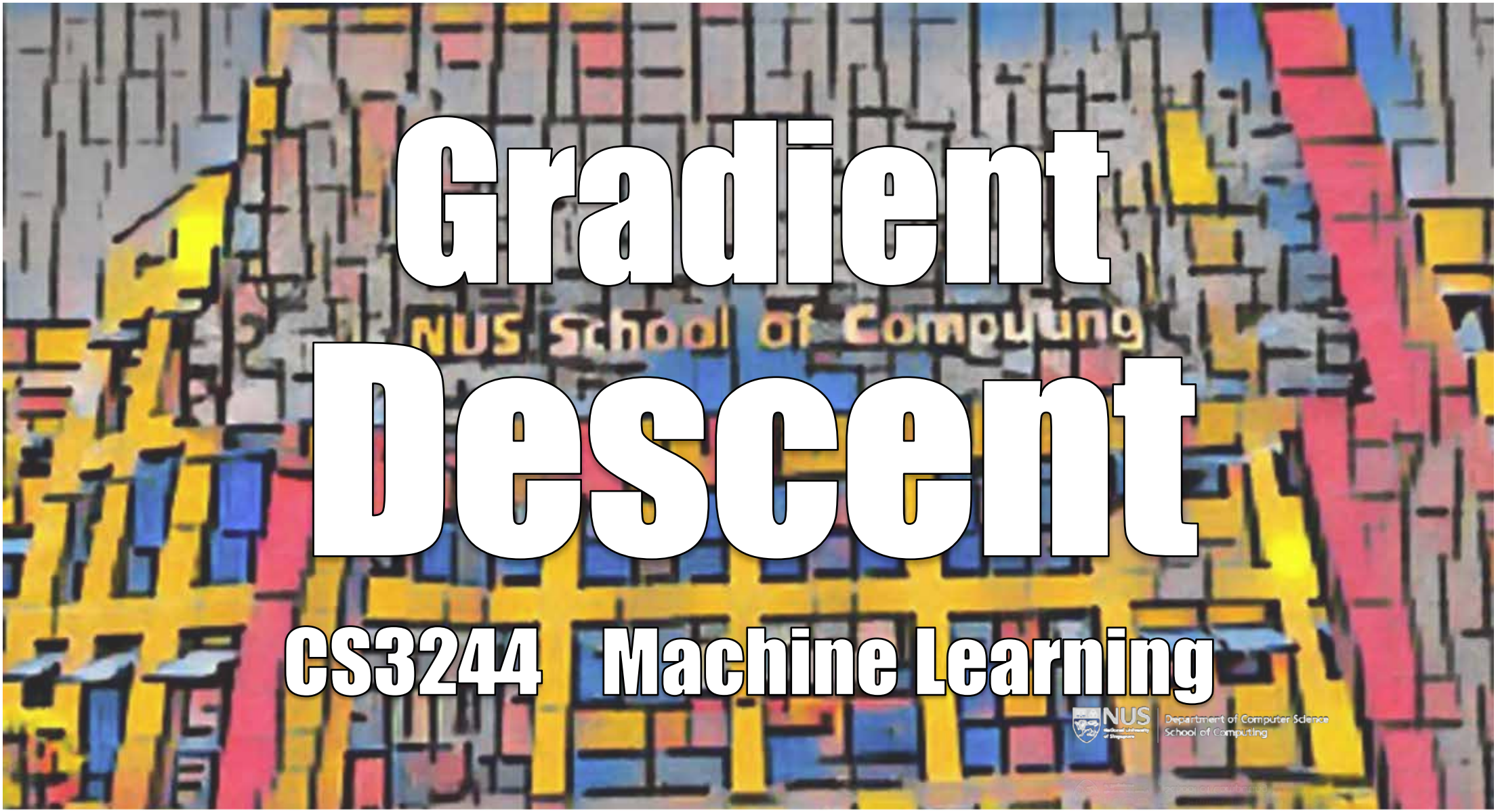
$$L_{train}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{j=1}^{m} ln(1 + e^{-y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}})$$

Iterative Solution

Compare to linear regression:

$$L_{train}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{j=1}^{m} \left(\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)} - y^{(j)}\right)^2$$

Closed-form Solution; Iteration also possible

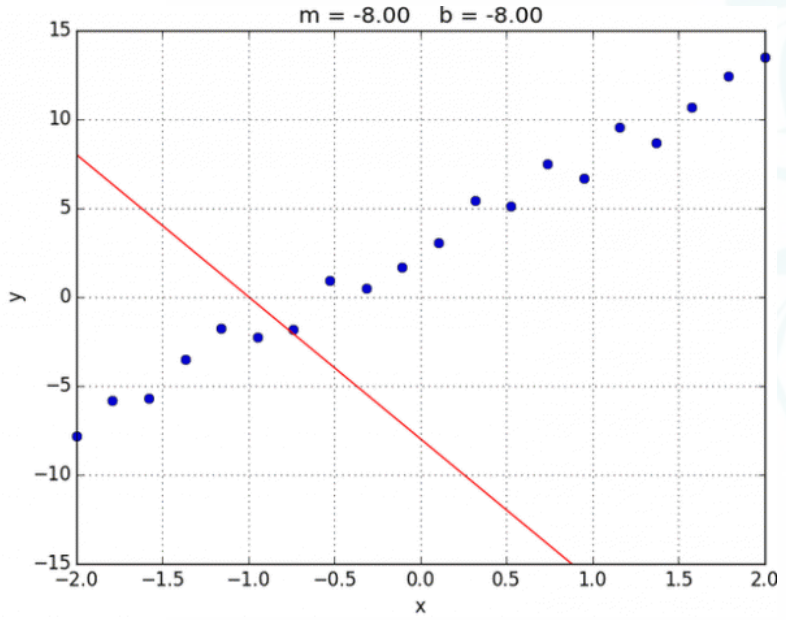

Gradient
Descent
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing

# Climbing up (down) one step at a time



Photo credits: Wikimedia

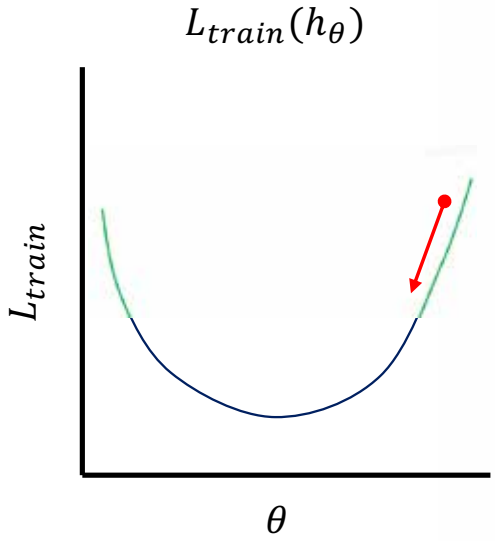
# Univariate Linear Regression: Salary to predict Credit Line





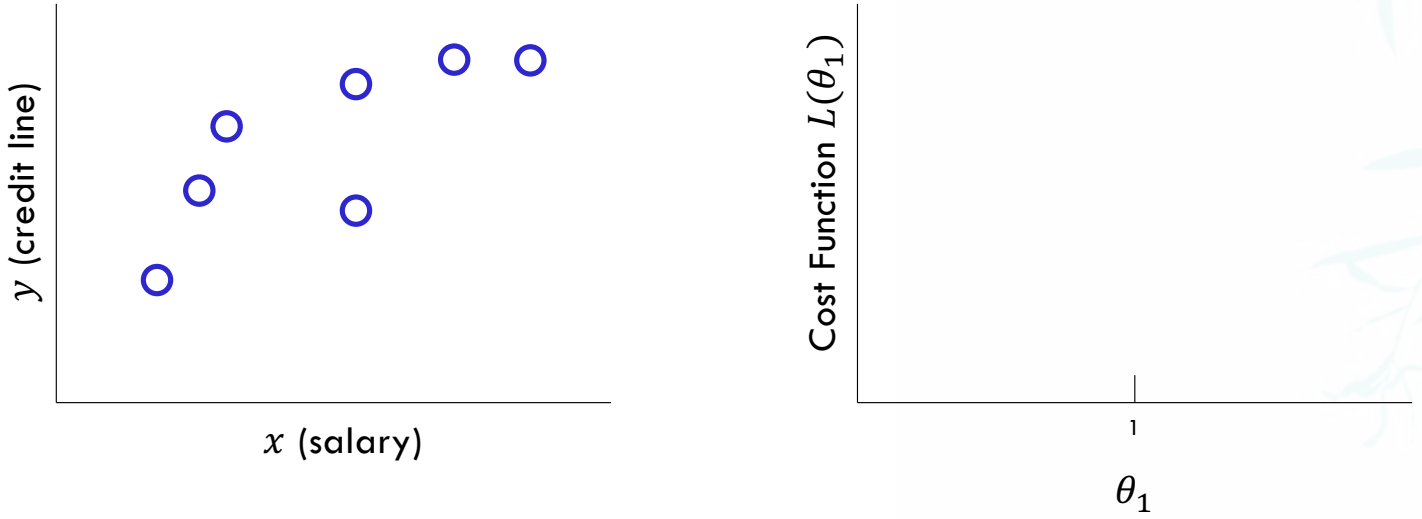

Credits: Alykhan Tejani's Medium Post

# Our $L_{train}$ only has one valley

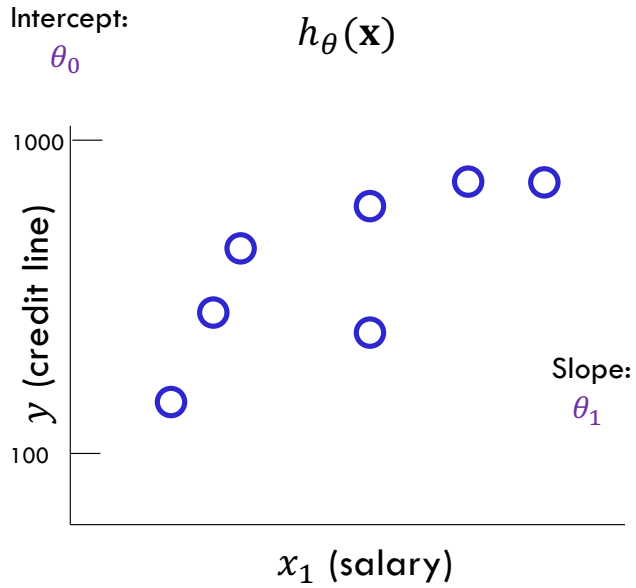


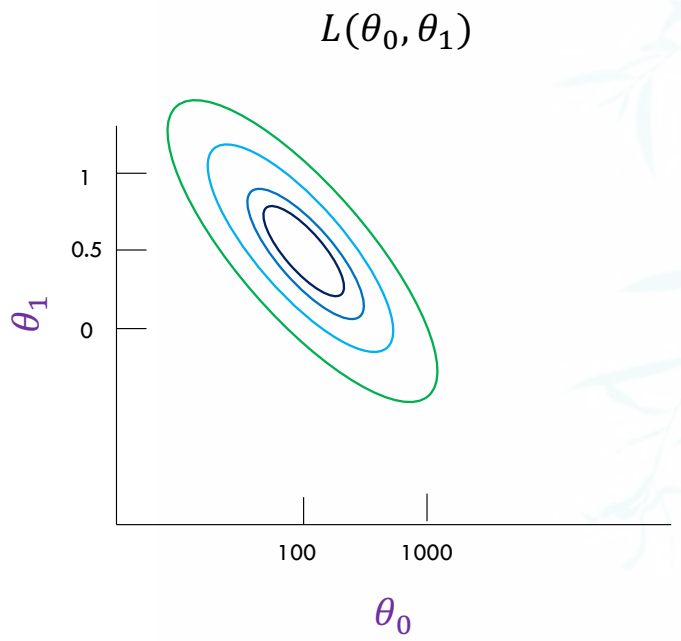

… Because $L_{train}$ is a convex function of $\theta$.

# Cost Function – only slope $L(\theta_1)$



# Cost Function with intercept $L(\theta_0, \theta_1)$



# Iterative method: gradient descent

General method for nonlinear optimization

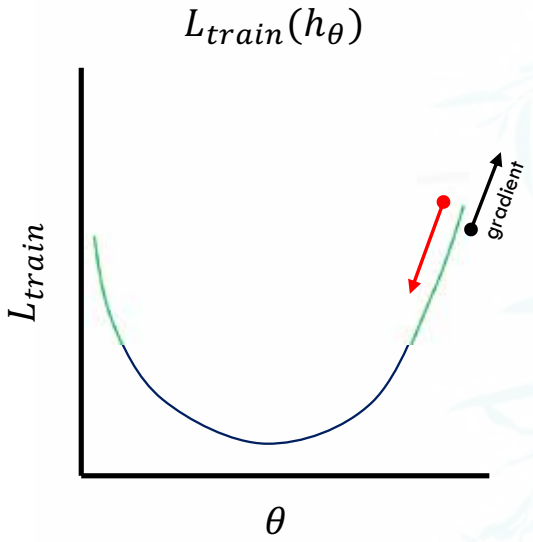
Start at $\boldsymbol{\theta}(0)$; take a step in the opposite direction of the steepest slope

Notation: $\theta(t) \equiv$ set of parameters at iteration $t$

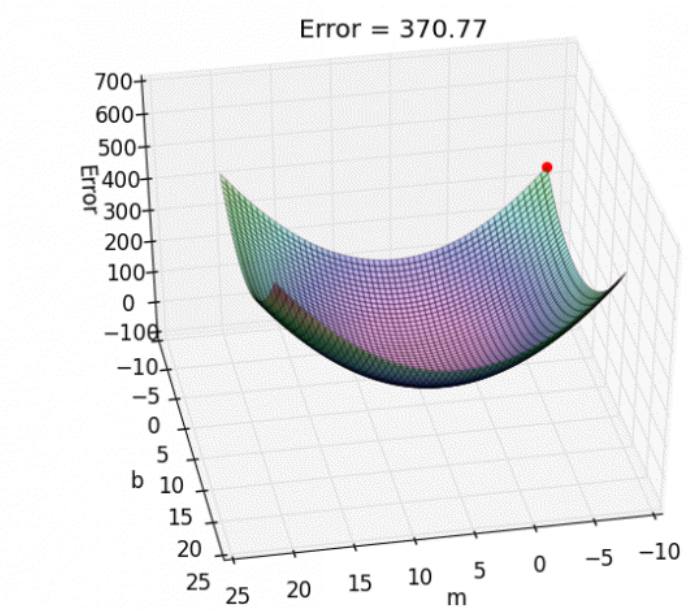
Fixed step size $\eta$ : $\boldsymbol{\theta}(1) = \boldsymbol{\theta}(0) + \eta \mathbf{v}$

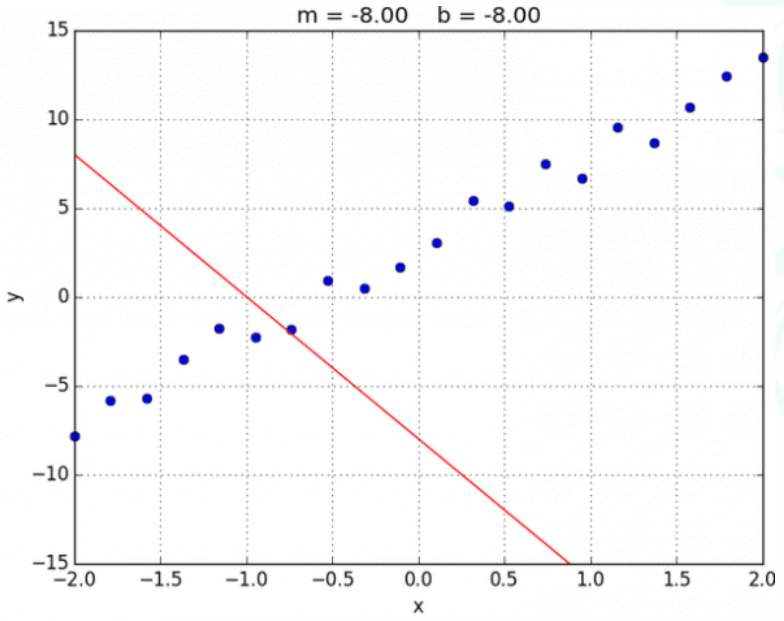
We get to pick the direction of $\mathbf{v}$.
What is the best direction to pick?

The one to minimize $L_{train}$.




Gradient descent can minimize any smooth function.
Oh yeah!






Credits: Alykhan Tejani's Medium Post

# Formula for the direction $\mathbf{v}$

$$\Delta L_{train} = L_{train}\big(\boldsymbol{\theta}(t+1)\big) - L_{train}\big(\boldsymbol{\theta}(t)\big)$$

Taylor approximation

$f(x_1) = f(x_0) + \nabla f(x_0) \cdot |x_1 - x_0| + O(\|x_1 - x_0\|^2)$

$$= L_{train}(\boldsymbol{\theta}(t) + \eta\mathbf{v}) - L_{train}\big(\boldsymbol{\theta}(t)\big)$$

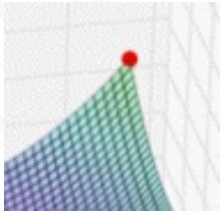
$$= \eta \nabla L_{train}\big(\boldsymbol{\theta}(t)\big)\mathbf{v} + O(\eta^2)$$

$$\geq -\eta \left\| \nabla L_{train}\big(\boldsymbol{\theta}(t)\big) \right\|$$

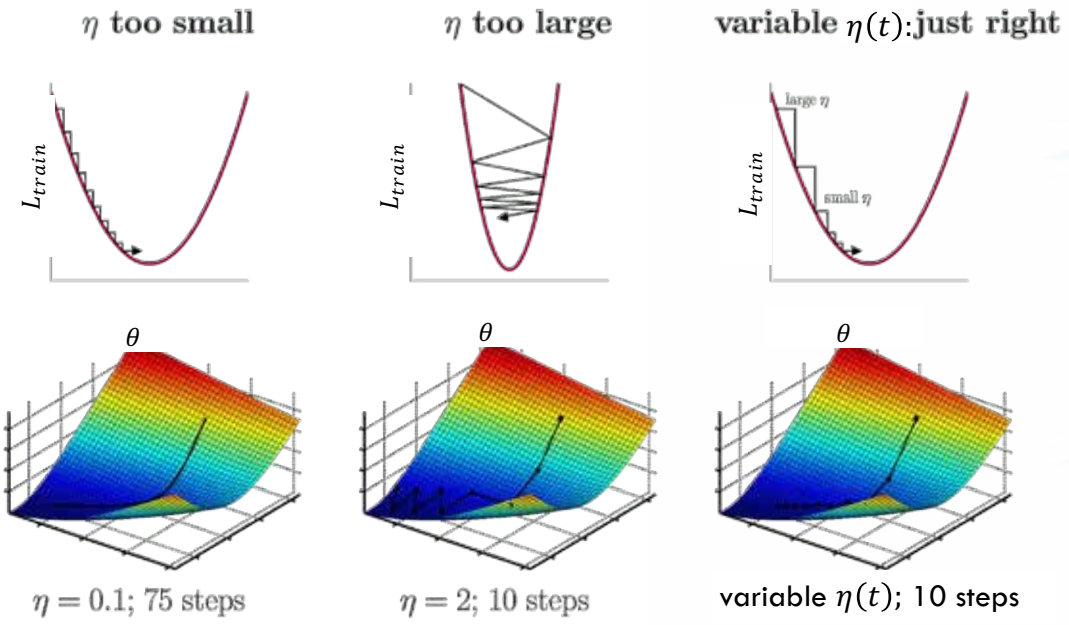
Notation: ∇ ("nabla") ≡ Gradient. Generalization of slope for higher dimensions

Create $v$ to be a unit vector:

$$\mathbf{v} = -\frac{\nabla L_{train}(\boldsymbol{\theta}(t))}{\|\nabla L_{train}(\boldsymbol{\theta}(t))\|}$$



# Goldilocks step size




$\eta$ should increase with the slope

# Varying step size, for free

Instead of

$$\Delta\boldsymbol{\theta} = \eta\mathbf{v}$$

$$= -\eta \frac{\nabla L_{train}(\boldsymbol{\theta}(t))}{\|\nabla L_{train}(\boldsymbol{\theta}(t))\|}$$

Have

$$\Delta\boldsymbol{\theta} = -\alpha L_{train}(\boldsymbol{\theta}(t))$$

Fixed learning rate $\alpha$ instead of fixed learning step $\eta$

# Logistic regression algorithm

1. Initialize the weights at $t = 0$ to $\boldsymbol{\theta}(0)$
2. Do
3. Compute the gradient

$$\nabla(t) = \nabla L_{train}(\boldsymbol{\theta}(t)) = -\frac{1}{m}\sum_{j=1}^{m}\frac{y^{(j)}\mathbf{x}^{(j)}}{1 + e^{y^{(j)}\boldsymbol{\theta}(t)^{\top}\mathbf{x}^{(j)}}}$$

Batch G.D.: true gradient computed over **all** points

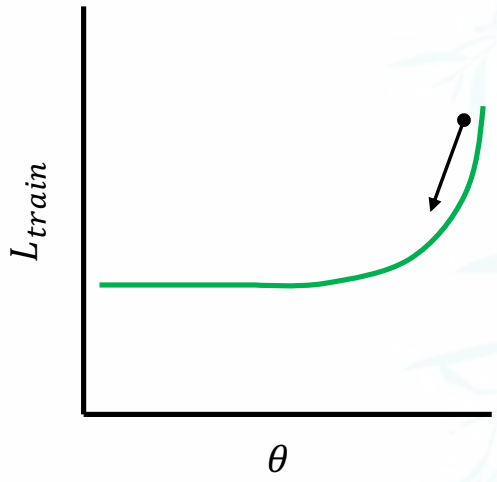
4. // Move in the direction $\mathbf{v}(t) = -\nabla(t)$
   Update the weights $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \alpha\nabla L_{train}$
5. Continue to next iteration, until it is time to stop
6. Return the final weights $\boldsymbol{\theta}^*$

# Termination Condition

When to stop?

Natural choice: *gradient* < *threshold*

But lots of flat regions in most spaces:




Instead, use criteria:

1. error change is small and/or;
2. error is small;
3. maximum number of iterations is reached.

# Stochastic Gradient Descent

A variation of GD that considers only the error on one data point.

Pick <u>one</u> $(\mathbf{x}^{(*)}, y^{(*)})$ at a time.

Apply GD to $l(h_\theta(\mathbf{x}^{(*)}), y^{(*)})$

"Average" Direction: $\mathbb{E}_*\left[-\nabla l(h_\theta(\mathbf{x}^{(*)}), y^{(*)})\right]$

$$= \frac{1}{m}\sum_{k=1}^{m} -\nabla l(h_\theta(\mathbf{x}^{(*)}), y^{(*)})$$
$$= -\nabla L_{train}$$

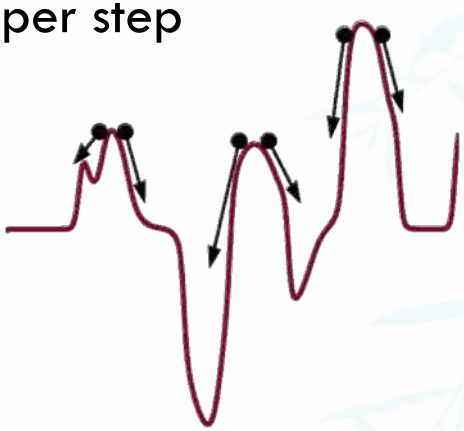
**Stochastic** Gradient Descent (SGD)



Photo credits: Wikipedia - Singapore Dollar

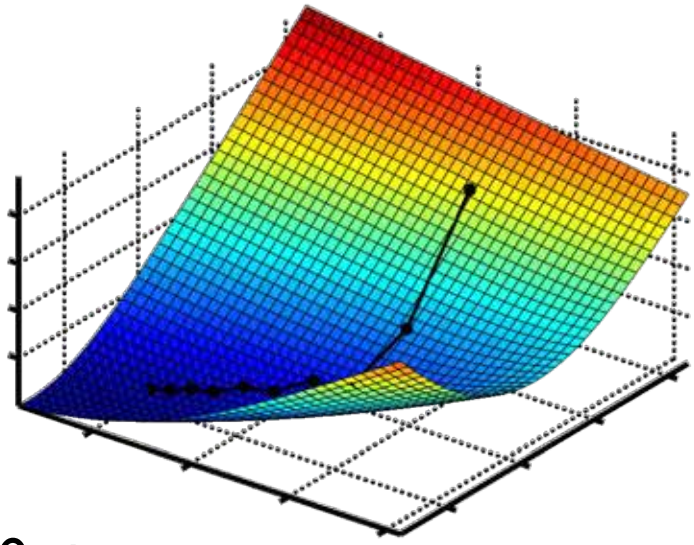
# Benefits of SGD

1. Cheaper computation: Fraction $1/m$ cheaper per step
2. Stochastic: Helps escape local minima
3. Simple



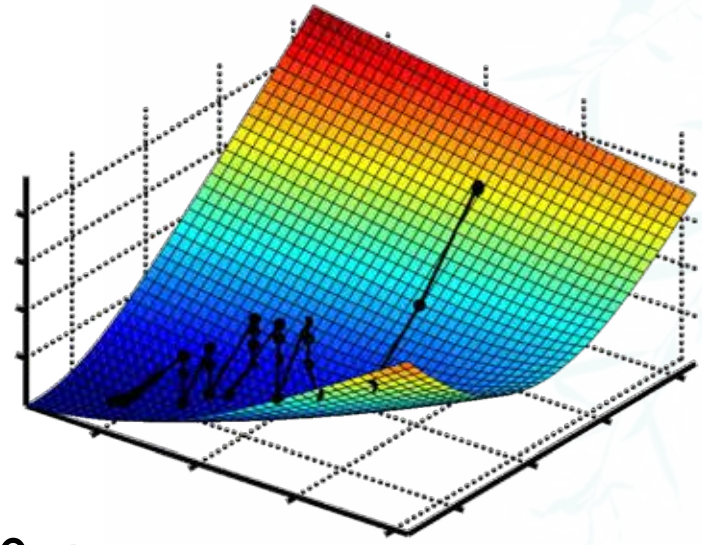
# GD vs. SGD on $m = 10$

GD



10 steps

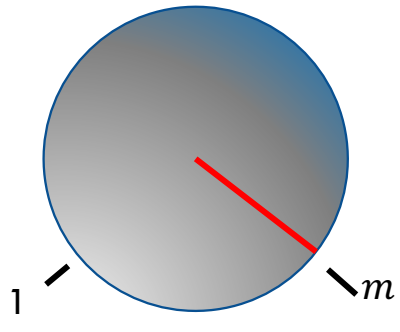
SGD



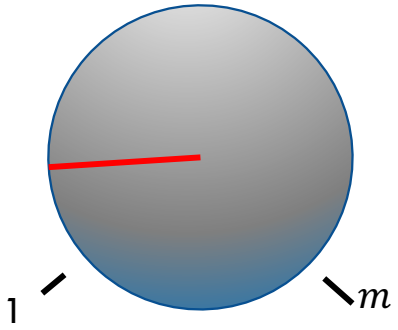
30 steps

# Mini Batch Gradient Descent

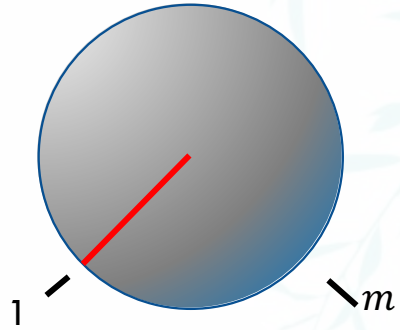
**Gradient Descent**




**Mini Batch**




**Stochastic Gradient Descent**




Pick a few $(\mathbf{x}^{(*)}, y^{(*)})$ at a time.