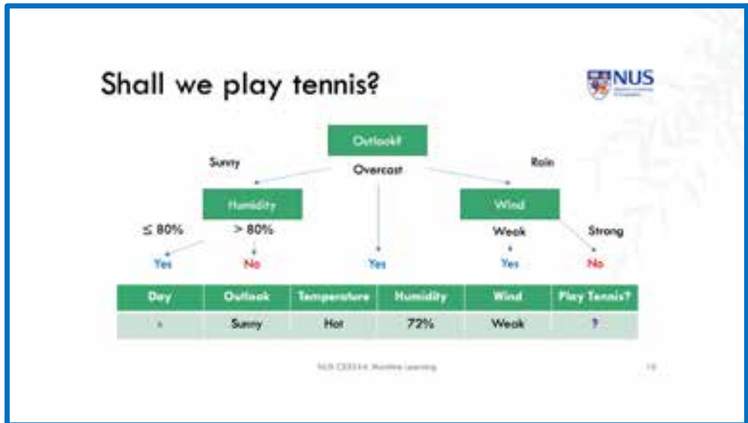
Linear Classifiers and Logistic Regression

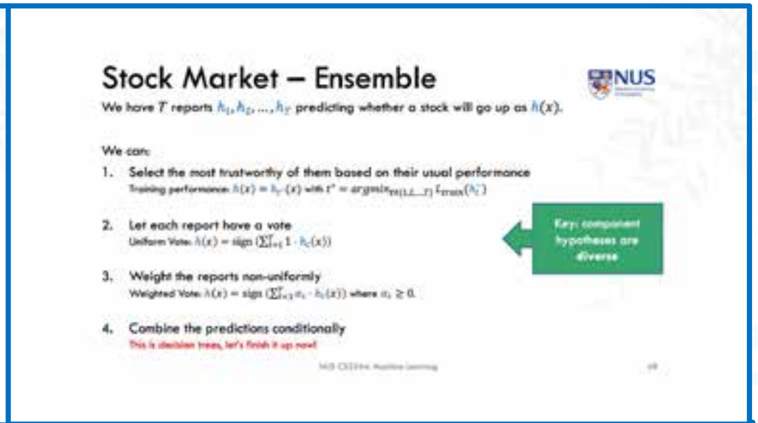
4A

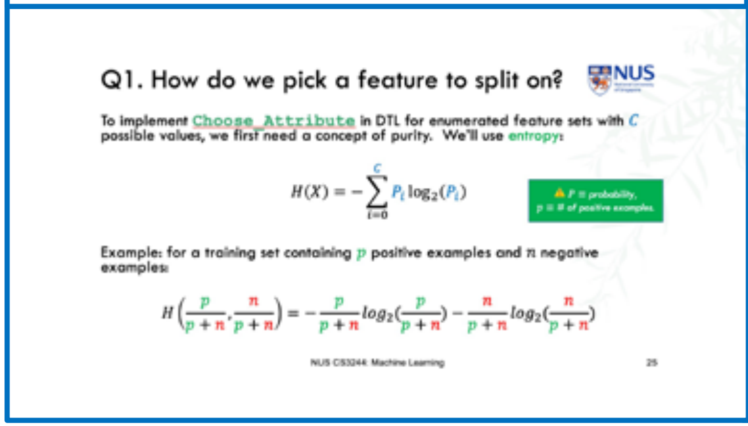
CS 3244
Machine Learning

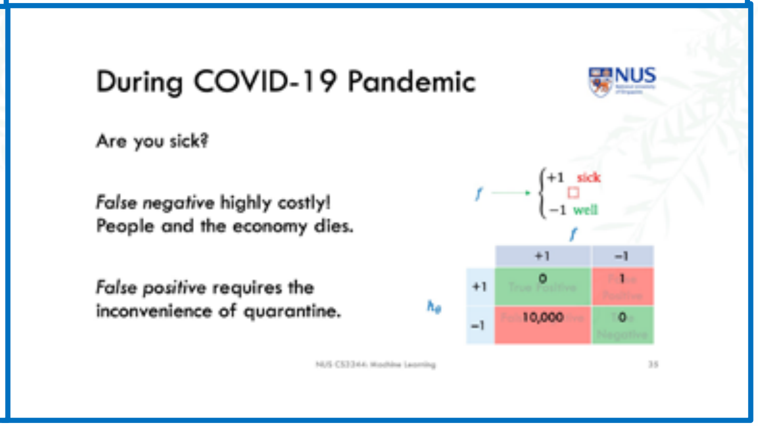
NUS | Computing

Recap from Week 03









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
- Non-linear Mappings

Curse of Dimensionality

CS3244 Machine Learning

NUS National University of Singapore
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”?

Photo by silvana amicone on Unsplash

In high dimensions:

The number of points needed to keep the same density grows exponentially.

The distance between arbitrary points is similar, making it hard to distinguish.

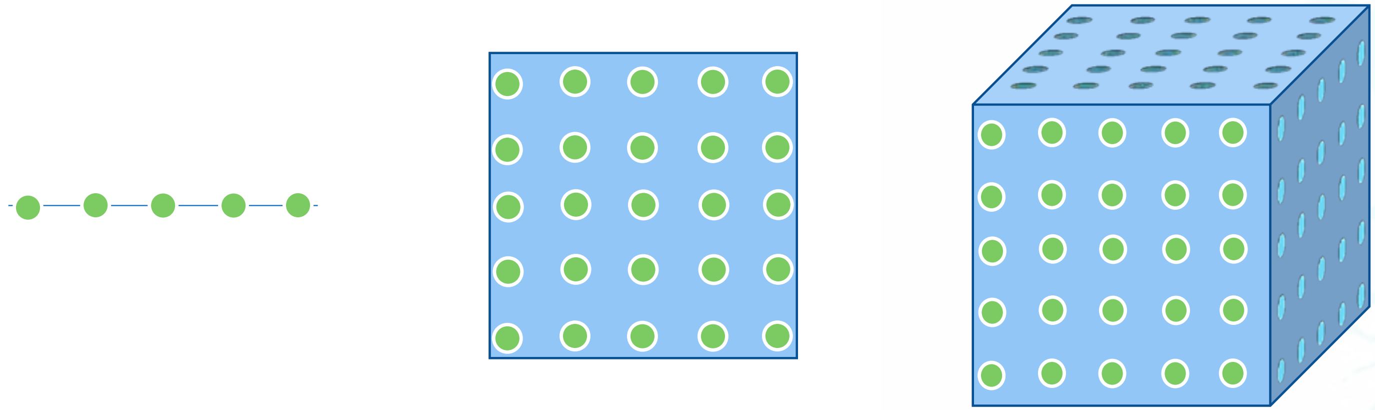
Sparsity with high dimensions

$m = 5$
$n = 1$

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 w/ Colab

Let's Go! (5 minutes)

http://www.comp.nus.edu.sg/~cs3244/AY2122S1/04.colab.html

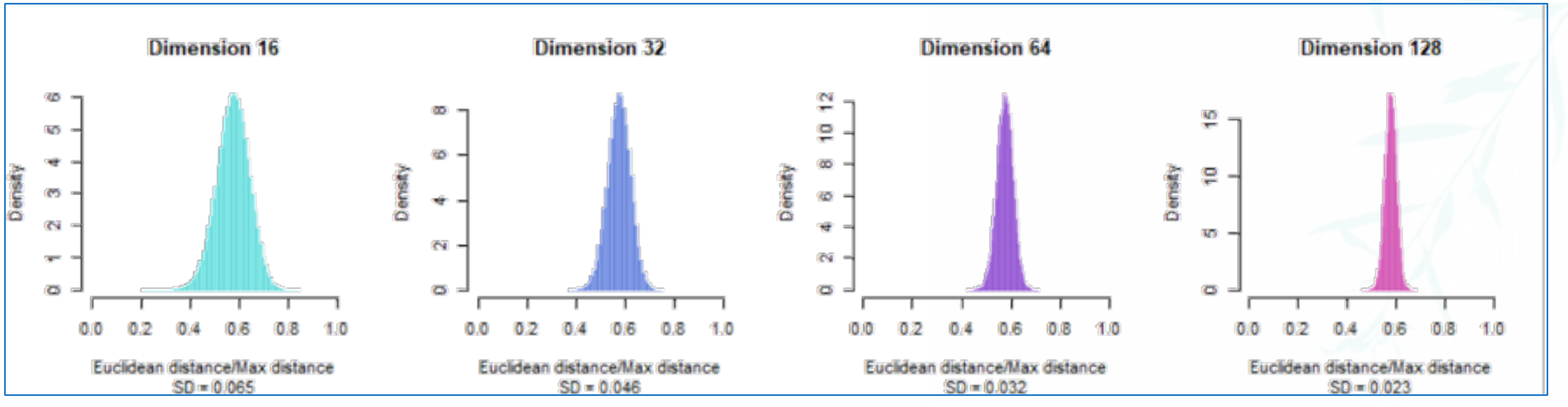
Standard Deviation σ grows slowly

```
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

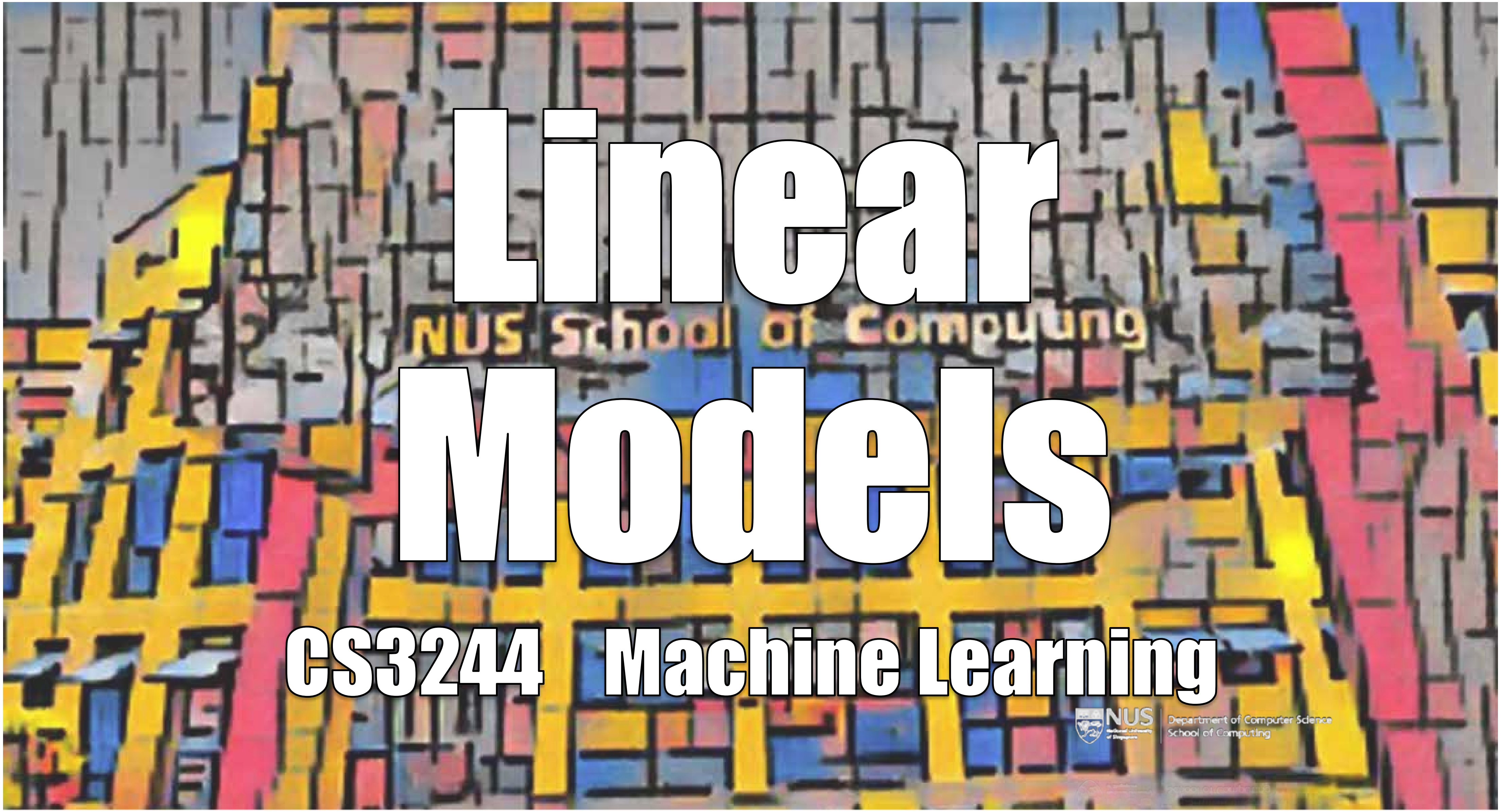
Most points end up far away; Euclidean distance becomes less meaningful

- High dimensional hypercube is most all “edge” space, far from the center of the hypercube

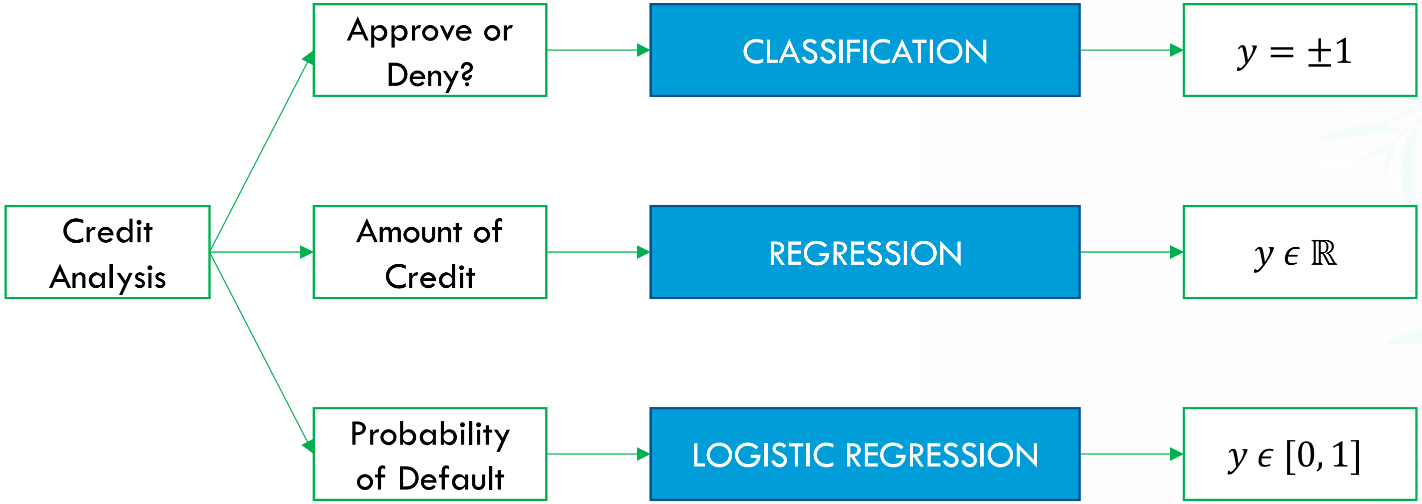
Choose or design an appropriate distance metric

Linear Models

CS3244 Machine Learning



Three learning problems

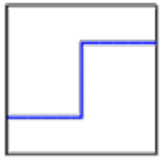


Linear models are the fundamental models.

The linear model is the first model to try.

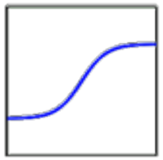
The linear signal

$$s = \boldsymbol{\theta}^{\top}\mathbf{x}$$



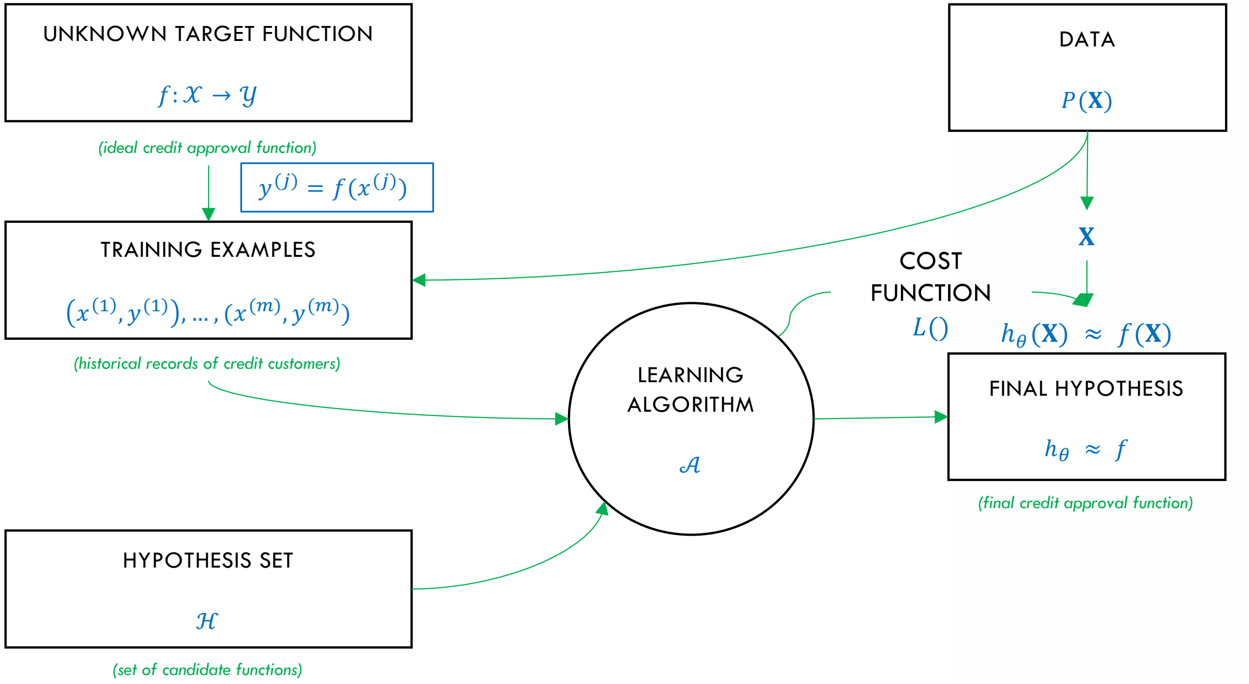
$\text{sign}\,(\boldsymbol{\theta}^{\top}\mathbf{x})$ $\quad y = \pm 1$

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



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

Recap: The Learning Diagram

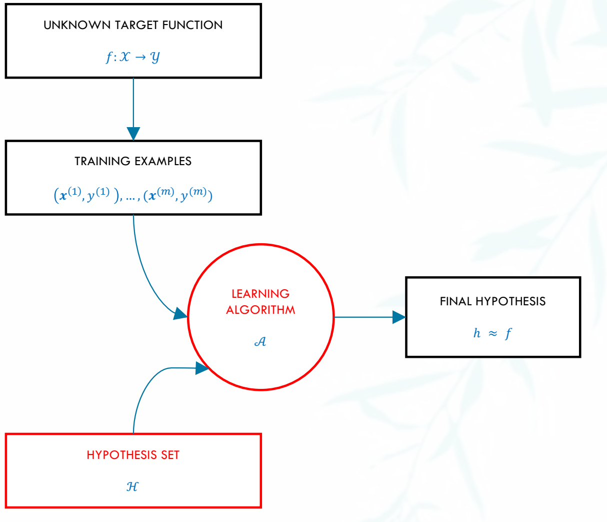


Choosing a model

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 θ.

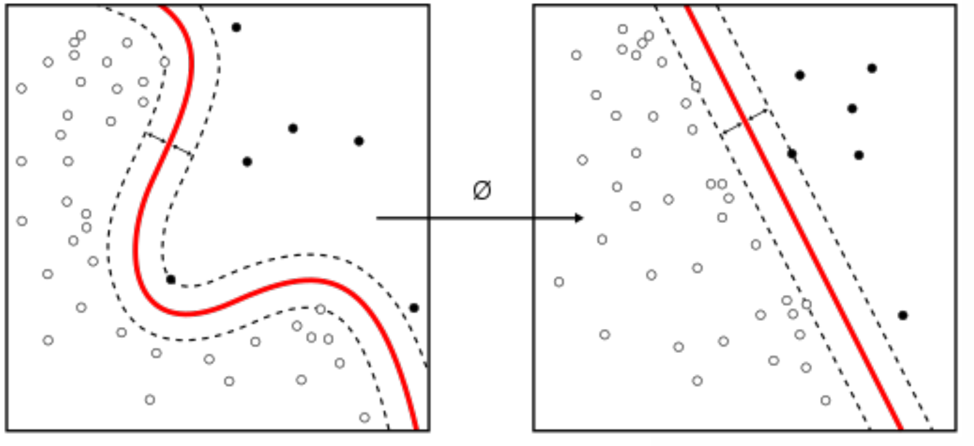
i.e., the learning algorithm $\mathcal{A}$ selects $h_\theta \in \mathcal{H}$.



$\mathcal{H}$ 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 θ, 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

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

$[\![test]\!]$ here means return 1 if $test$ is true, 0 otherwise.

Normalize x:

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

The 'bias weight' θ_0 corresponds to the threshold. θ_1 and θ_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.

Credit Approval

Do we extend credit to this person?

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

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 $\mathbf{x}$?

🥇 🥈

$$\begin{bmatrix} \theta_0 \\ \cdots \\ \theta_n \end{bmatrix} \times \begin{bmatrix} x_0 & \cdots & x_n \end{bmatrix} =$$

$$\begin{bmatrix} \theta_0 & \cdots & \theta_n \end{bmatrix} \times \begin{bmatrix} x_0 \\ \cdots \\ x_n \end{bmatrix} =$$

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)

(Recall

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

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

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

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

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

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

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

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

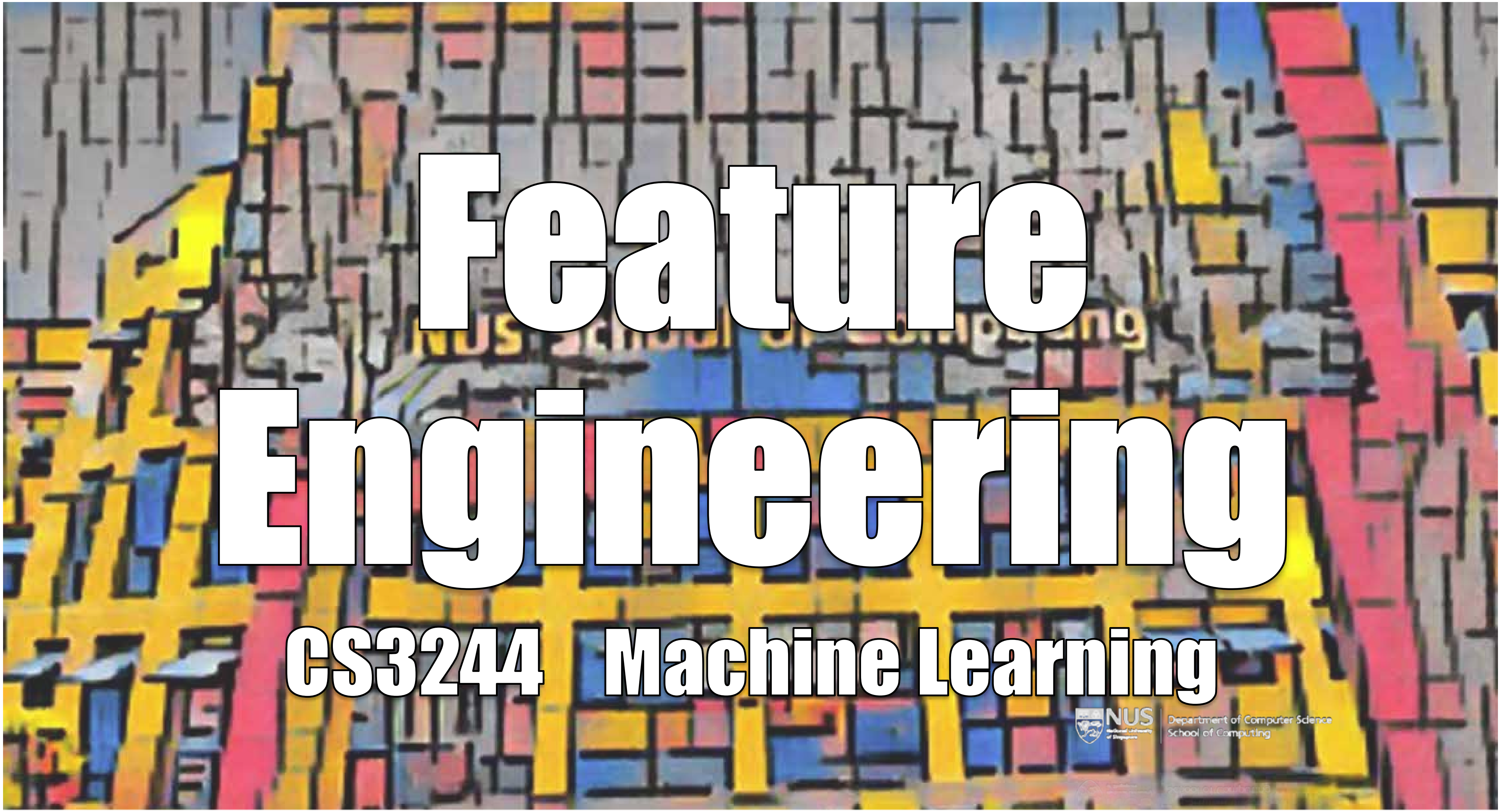
A simple $\mathcal{H}$: hyperplane with a threshold (bias)

For an input $\mathbf{x} = (x_1, x_2, \dots, x_n)$ *'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 $h \epsilon \mathcal{H}$ can be written as:

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


Feature
Engineering
CS3244 Machine Learning
NUS
Department of Computer Science
School of Computing

MNIST (recap)

Each digit is a 16x16 pixel gray-intensity image.

[-1 -1 -1 -1 -1 -1 -1 -0.63 0.86 -0.17 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.99 0.3 1 0.31 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.41 1 0.99 -0.57 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.68 0.83 1 0.56 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -0.94 0.54 1 0.78 -0.72 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0.1 1 0.92 -0.44 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.26 0.95 1 -0.16 -1 -1 -1 -0.99 -0.71 -0.83 -1 -1 -1 -1 -1 -0.8 0.91 1 0.3 -
0.96 -1 -1 -0.55 0.49 1 0.88 0.09 -1 -1 -1 -1 0.28 1 0.88 -0.8 -1 -0.9 0.14 0.97 1 1 1 0.99 -0.74 -1 -1 -0.95 0.84 1 0.32 -1 -1 0.35 1 0.65 -0.10 -0.18 1 0.98 -0.72 -1 -1 -0.63 1 1 0.07 -0.92 0.11
0.96 0.30 -0.88 -1 -0.07 1 0.64 -0.99 -1 -1 -0.67 1 1 0.75 0.34 1 0.70 -0.94 -1 -1 0.54 1 0.02 -1 -1 -1 -0.90 0.79 1 1 1 1 0.53 0.18 0.81 0.83 0.97 0.86 -0.63 -1 -1 -1 -1 -0.45 0.82 1 1 1 1 1 1 1 1
0.13 -1 -1 -1 -1 -1 -1 -0.48 0.81 1 1 1 1 1 1 0.21 -0.94 -1 -1 -1 -1 -1 -1 -1 -0.97 -0.42 0.30 0.82 1 0.48 -0.47 -0.99 -1 -1 -1 -1]

Input representation

Raw input $\mathbf{x} = (x_0, x_1, x_2, x_3, x_4, \dots, x_{256})$

Too many (257) parameters!

Linear model: $(\theta_0, \theta_1, \theta_2, \dots, \theta_{256})$

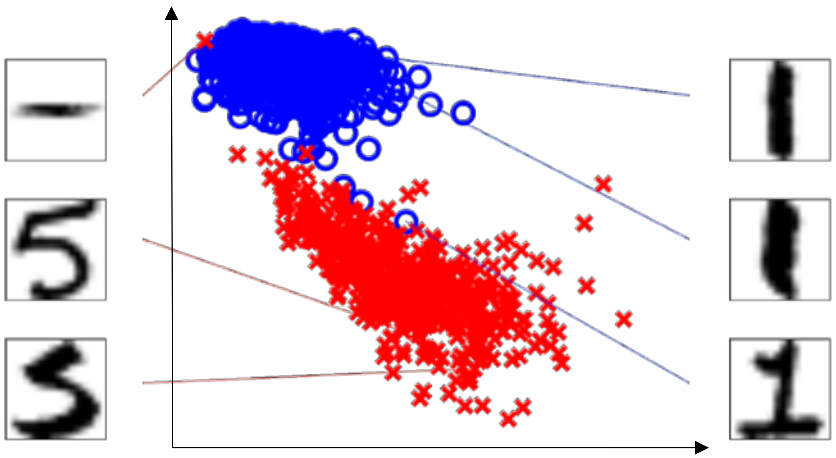
Features: extract useful information, e.g.,

Intensity and symmetry: $\mathbf{x} = (x_0, x_1, x_2)$

Linear model: $(\theta_0, \theta_1, \theta_2)$

Illustration of features

$\mathbf{x} = (x_0, x_1, x_2)$ x_1= intensity x_2= symmetry



Your Turn: which axes is which?

Linear Regression

CS3244 Machine Learning

NUS National University of Singapore | 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: $h_\theta(\mathbf{x}) = \sum_{i=0}^{n} \theta_i x_i = \theta^\top \mathbf{x}$
  $= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
- 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

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

$\mathbb{R}$-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:

The sweet spot: Plausible AND Convenient

$$L_{train}(h) = \underbrace{\frac{1}{m}\sum_{j=1}^{m}}_{\text{Pointwise Average}} \underbrace{(h_\theta(x^{(j)}) - y^{(j)})^2}_{\text{How bad is our prediction?}}$$

Illustration of linear regression

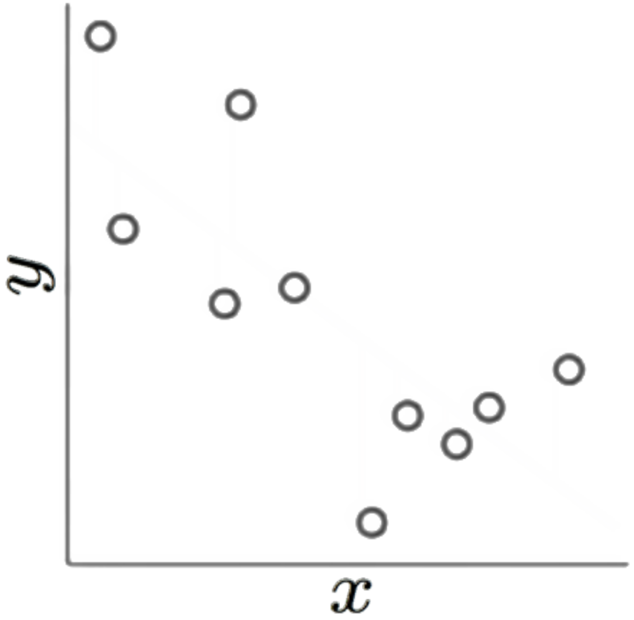
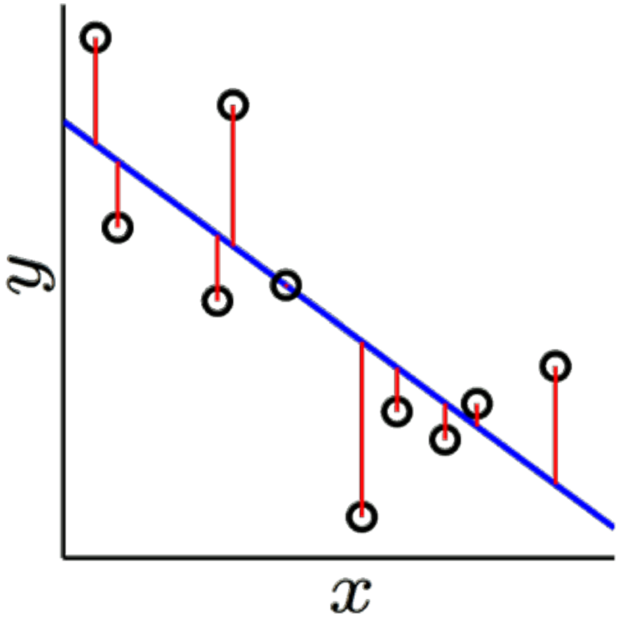
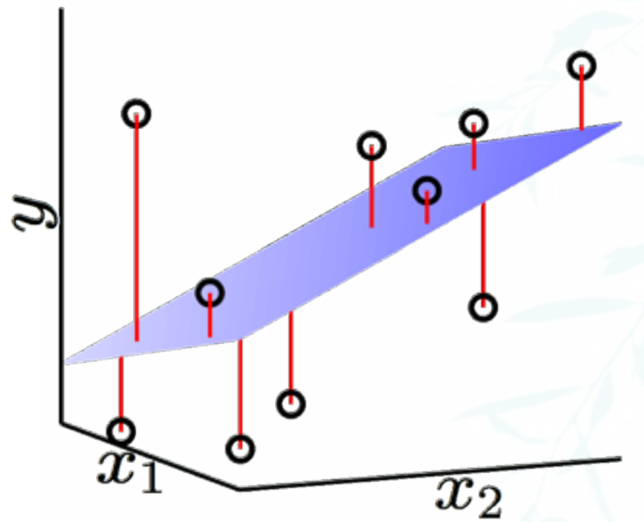


Illustration of linear regression





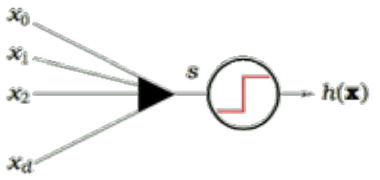
Hypothesis: $h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$


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

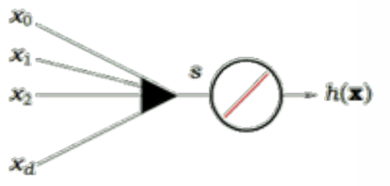
A third linear model

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

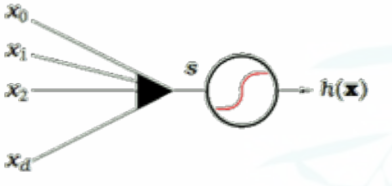
Linear classification



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:

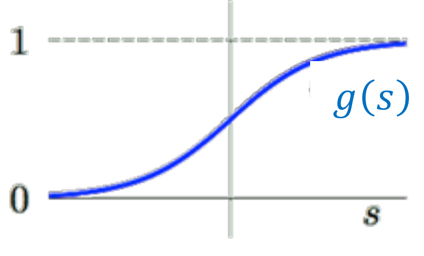
$$g(s) = \frac{exp^s}{1 + exp^s}$$

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

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

$\mathbf{x}^{(j)}$ – 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:

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

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,

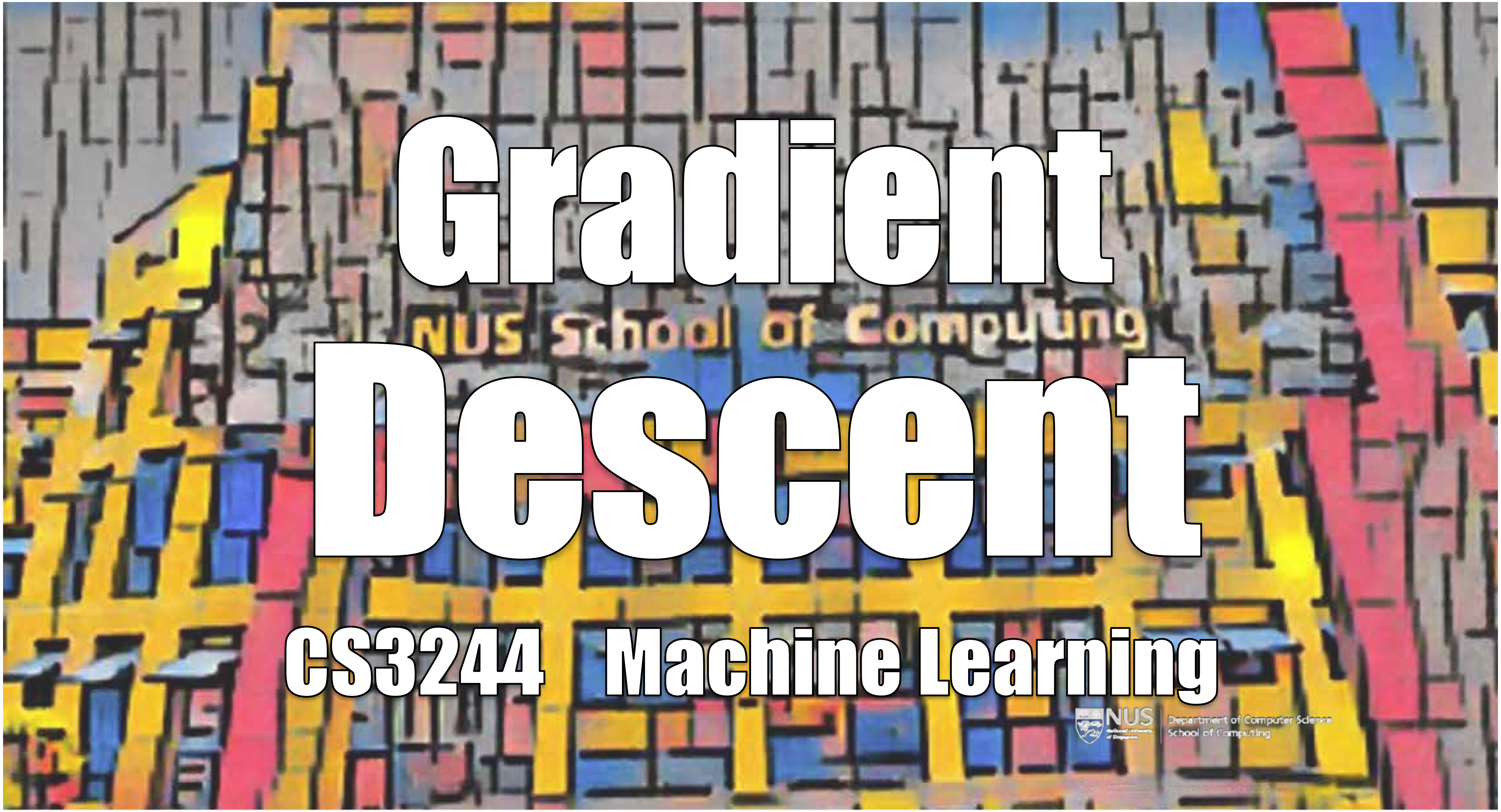
$$L_{train}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{j=1}^{m} ln(1 + exp^{-y^{(j)}\boldsymbol{\theta}^{\top}\mathbf{x}^{(j)}})$$

Iterative Solution

Compare to linear regression:

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

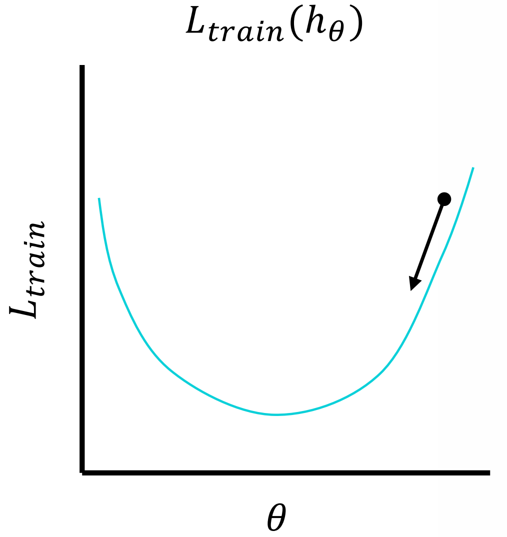
Closed-form solution; but iteration also possible


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

Climbing up (down) one step at a time

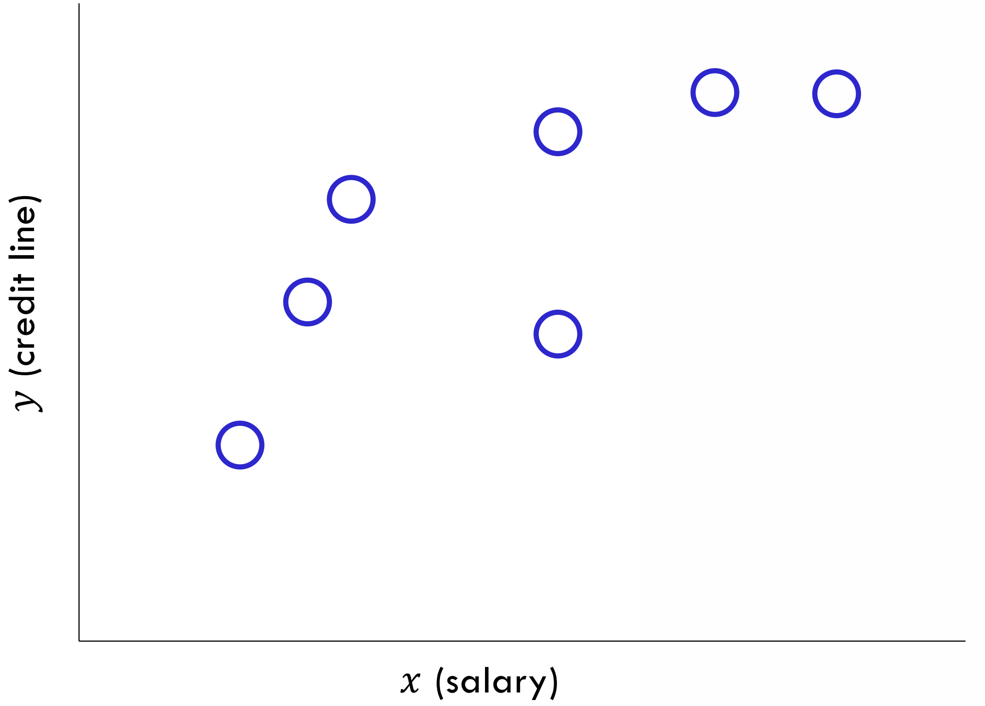
Photo credits: Wikimedia

Our L_{train} only has one valley

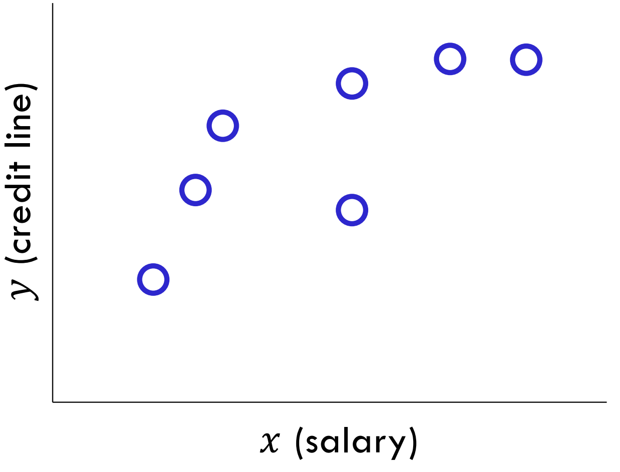


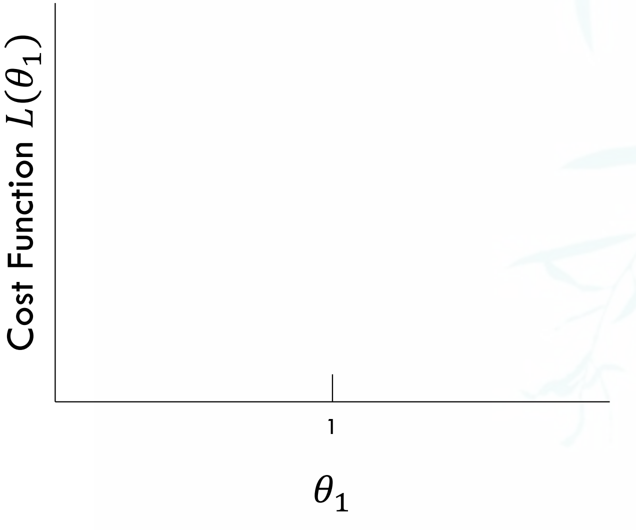
… Because L_{train} is a convex function of $\boldsymbol{\theta}$.

Univariate Linear Regression: Salary to predict Credit Line

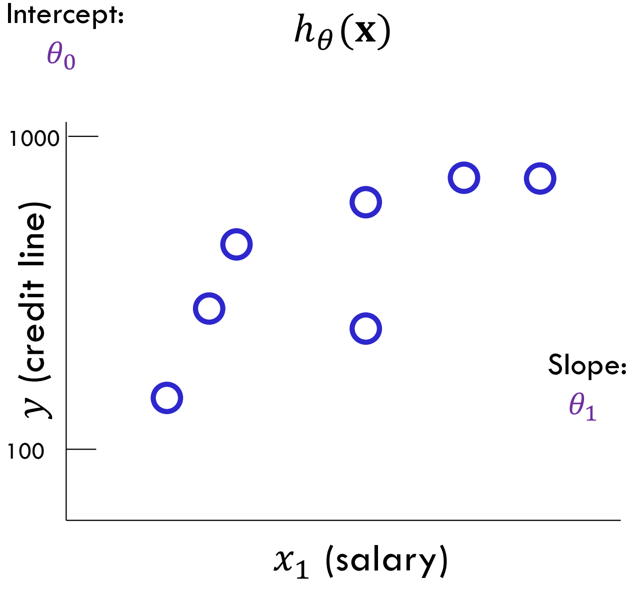


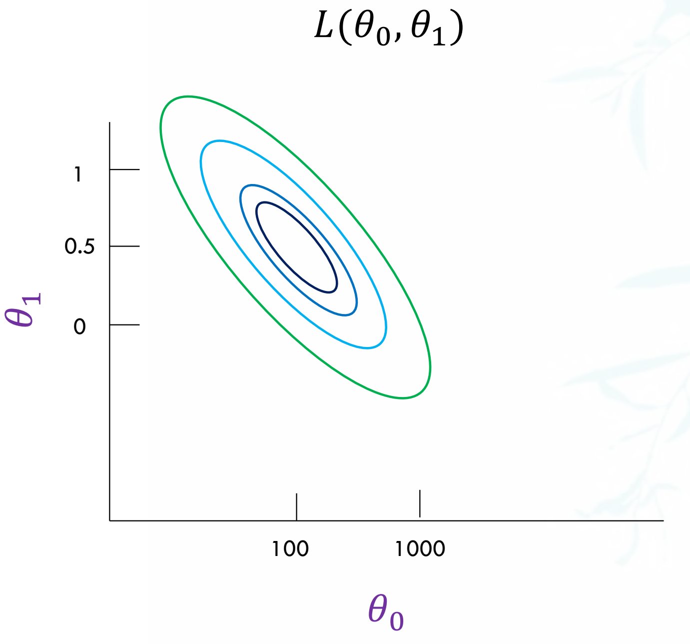
Ignoring the bias term





Cost Function $L(\theta_0, \theta_1)$





Iterative method: gradient descent

General method for nonlinear optimization

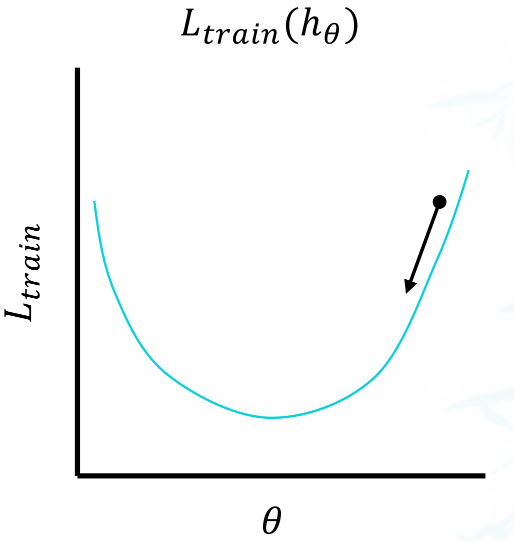
Start at $\theta(0)$; take a step along steepest slope

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

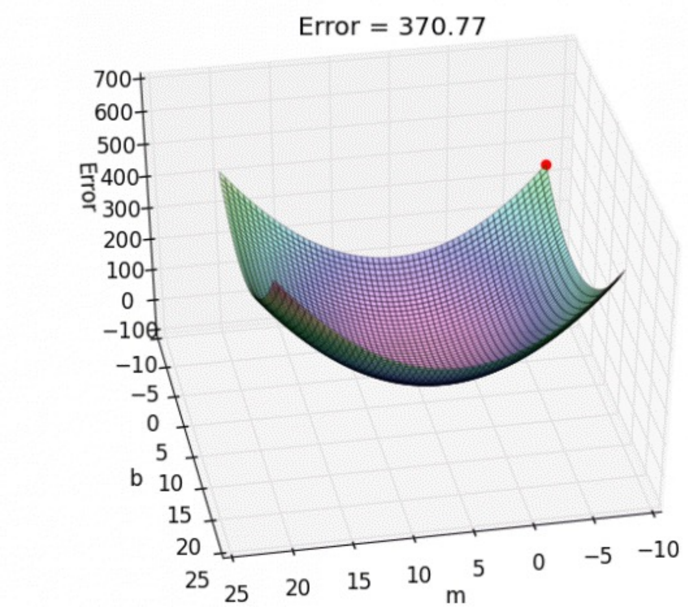
Fixed step size η : $\theta(1) = \theta(0) + \eta v$

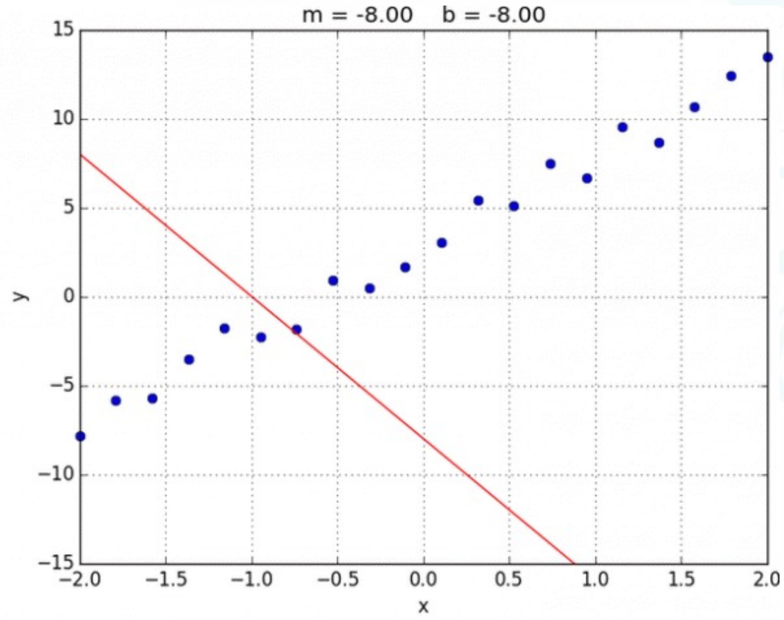
We get to pick the direction of v. What is the best direction to pick?

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 v

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

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

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

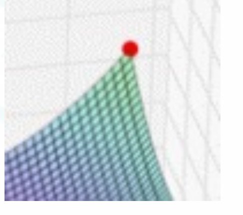
$$\geq -\eta \left\| \nabla L_{train}(\theta(t)) \right\|$$

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

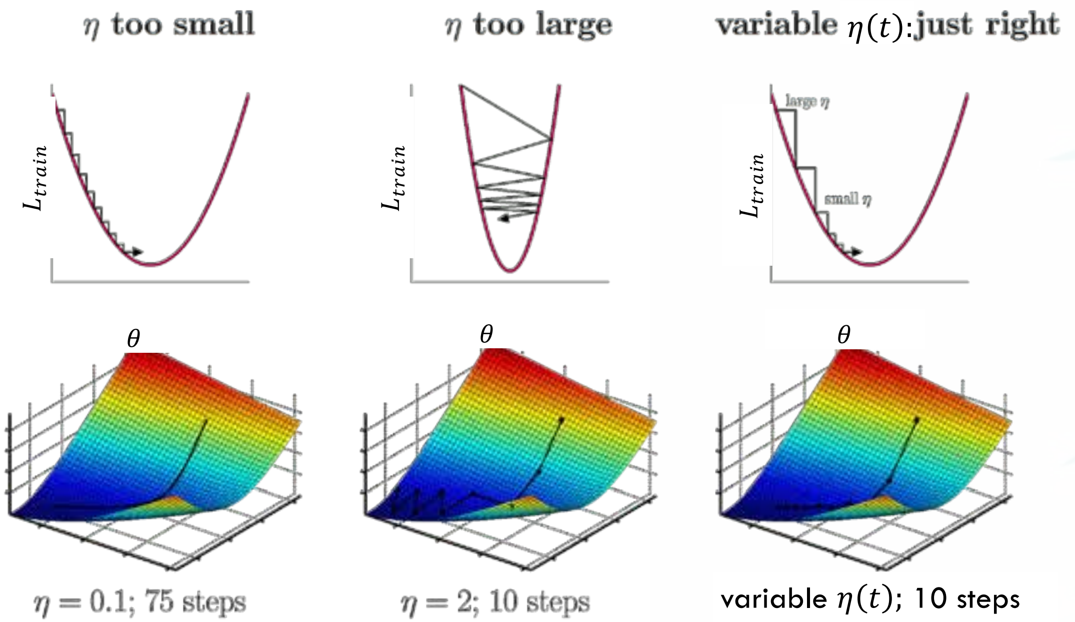
Notation: ∇ ("nabla") ≡ Gradient. Generalization of slope for higher dimensions

Create v to be a unit vector:

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



Goldilocks step size



η should increase with the slope

Varying step size, for free

Instead of

$$\Delta\theta = \eta v$$

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

Have

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

Fixed learning <u>rate</u> α instead of fixed learning <u>step</u> η

Logistic regression algorithm

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

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

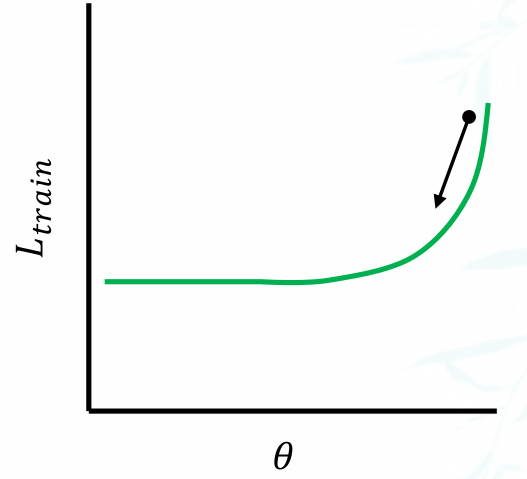
4. // Move in the direction $v(t) = -\nabla(t)$
   Update the weights $\theta(t+1) = \theta(t) - \alpha \nabla L_{train}$
5. Continue to next iteration, until it is time to stop
6. Return the final weights θ^*

Termination Condition

When to stop?

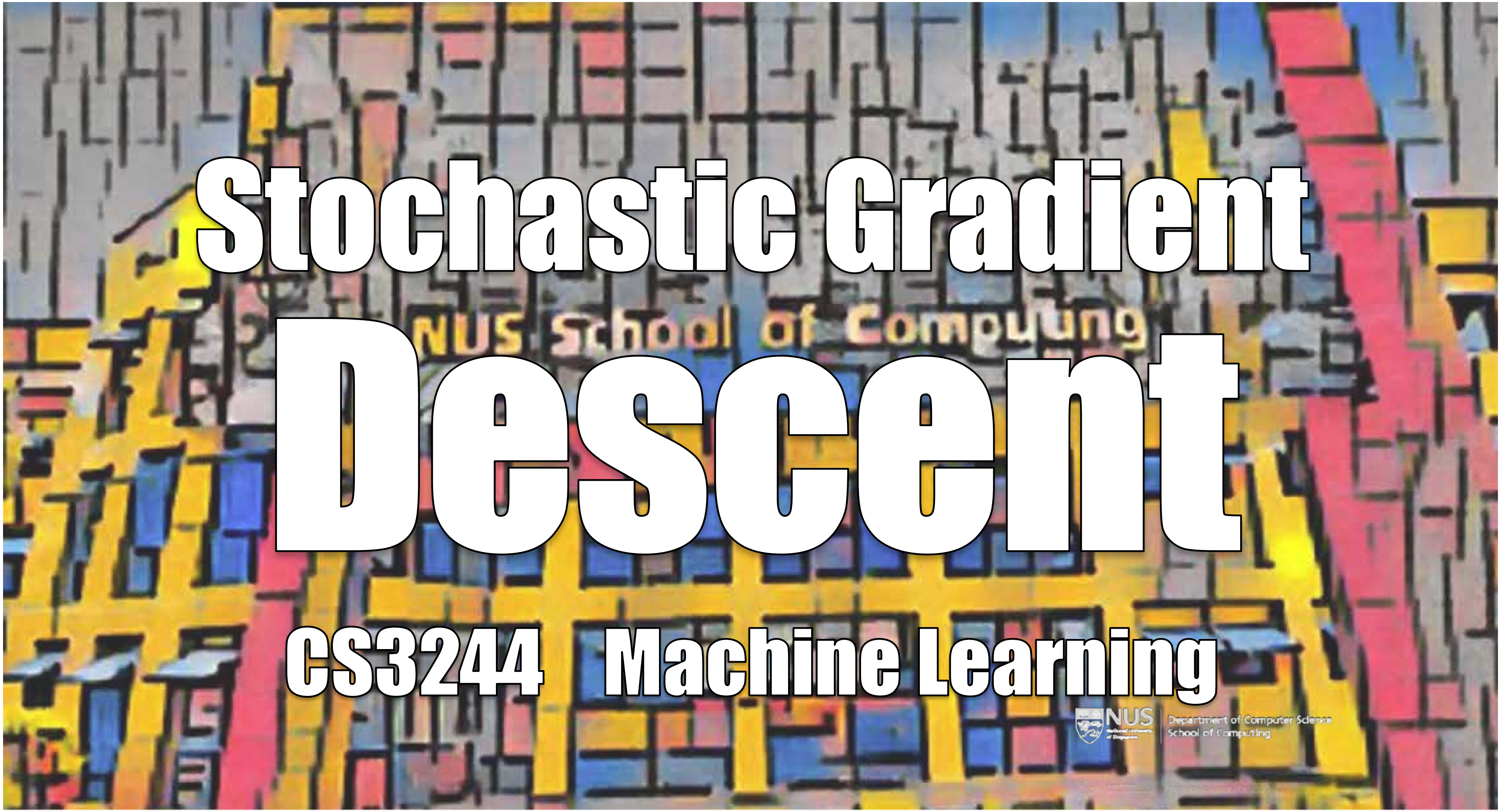
Natural choice: *gradient* < *threshold*

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

(Batch) Gradient Descent

Gradient Descent minimizes:

$$L_{train}(\theta) = \frac{1}{m}\sum_{j=1}^{m} \underbrace{l\left(h_\theta\left(\mathbf{x}^{(j)}\right), y^{(j)}\right)}_{\ln(1+e^{y^{(j)}\theta^\top \mathbf{x}^{(j)}})}$$

By iterative steps along $-\nabla L$:

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

$-\alpha\nabla L_{train}$ is based on **all** examples $(\mathbf{X}, y)$

This is **"batch"** Gradient Descent

Overly conservative?

In Zoom breakout or physical subgroups:

(5 mins): Suggest a way to speed up Gradient Descent

Hint: $-\alpha \nabla L_{train}$ is based on **all** examples $(\mathbf{X}, y)$

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

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)

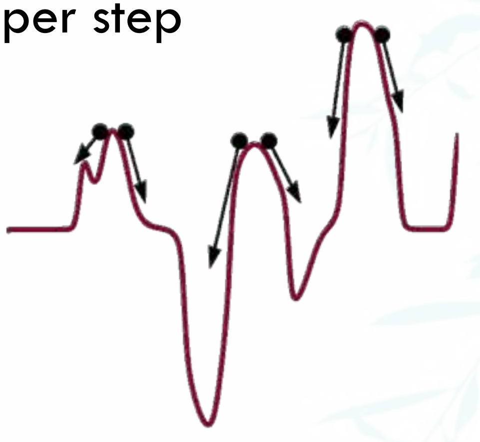
Photo credits: Wikipedia - Singapore Dollar

Benefits of SGD

1. Cheaper computation: Fraction $1/m$ cheaper per step

2. Stochastic: Helps escape local minima

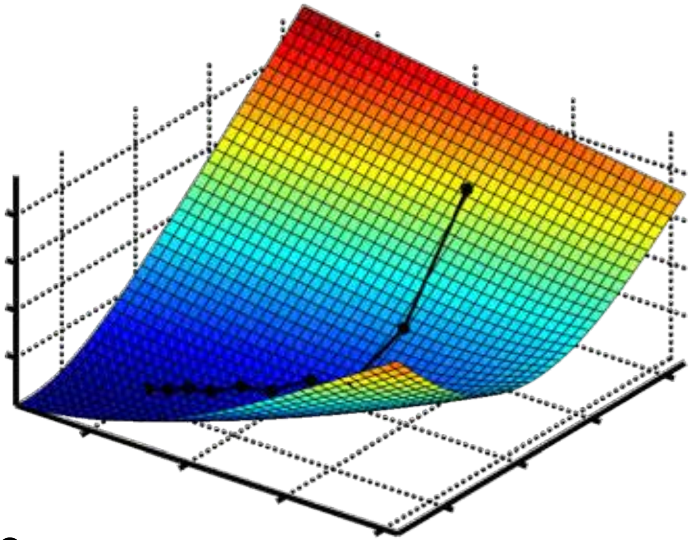
3. Simple



Rule of Thumb:
$\alpha = 0.1$ usually works!

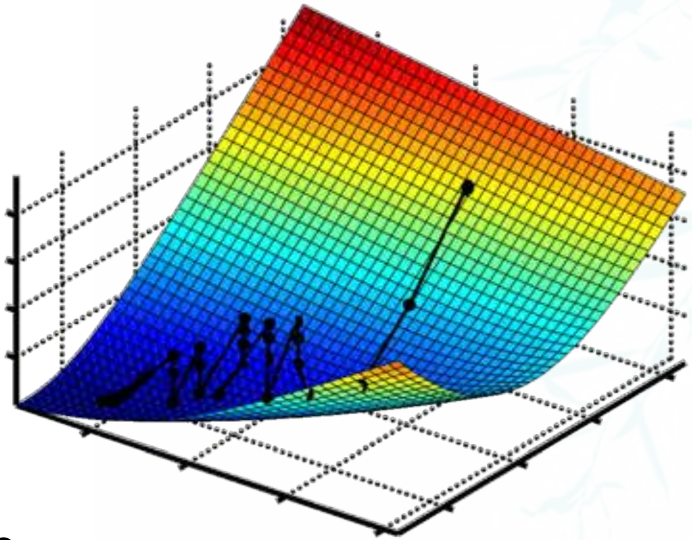
GD vs. SGD on $m = 10$

GD



10 steps

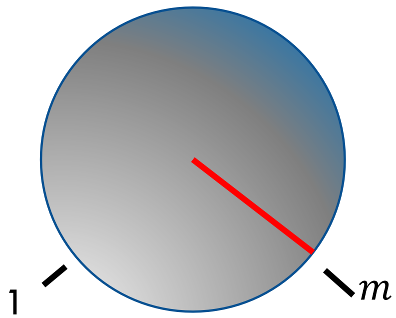
SGD



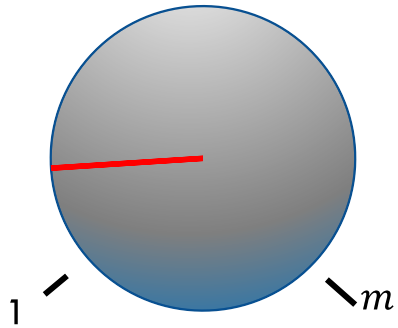
30 steps

Mini Batch Gradient Descent

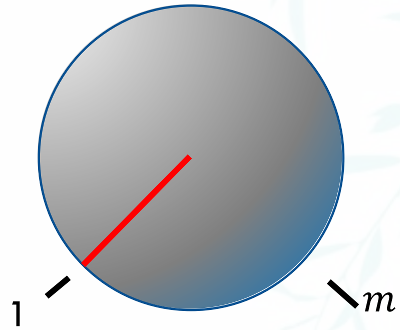
Gradient Descent



Mini Batch



Stochastic Gradient Descent



Batch size determined

Normal Equation for Linear Regression

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.

The expression for L_{train}

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

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

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

Minimizing L_{train}

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

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

Minimizing L_{train}

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

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

$$= (\mathbf{X\theta} - \mathbf{y})^\top(\mathbf{X\theta} - \mathbf{y})$$

$$= (\mathbf{X\theta})^\top\mathbf{X\theta} - (\mathbf{X\theta})^\top\mathbf{y}$$

$$-\mathbf{y}^\top\mathbf{X\theta} \quad + \mathbf{y}^\top\mathbf{y}$$

$$= \mathbf{\theta}^\top\mathbf{X}^\top\mathbf{X\theta} - 2(\mathbf{X\theta})^\top\mathbf{y} + \mathbf{y}^\top\mathbf{y}$$

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

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

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

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

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

Jacobian: stack of gradients as partial derivatives.

Minimizing L_{train}

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

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

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

$$\frac{\partial L_{train}}{\partial \boldsymbol{\theta}} \equiv \mathbf{X}^\top\mathbf{X\theta} - \mathbf{X}^\top\mathbf{y} = \overleftarrow{\mathbf{0}}$$
$$\mathbf{X}^\top\mathbf{X\theta} = \mathbf{X}^\top\mathbf{y}$$
$$\boldsymbol{\theta} = \left((\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right)\mathbf{y}$$

Column of zeroes

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

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

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

$$\underbrace{\begin{bmatrix} \quad & \quad & \quad \end{bmatrix}}_{n+1 \times m} \quad \underbrace{\begin{bmatrix} \quad \\ \quad \\ \quad \end{bmatrix}}_{m \times n+1}$$

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

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

$$\underbrace{\begin{bmatrix} \underbrace{\left[\qquad\right]}_{(n+1)\times(n+1)} \end{bmatrix}^{-1} \underbrace{\left[\qquad\qquad\right]}_{(n+1)\times m}}_{(n+1)\times m}$$

Linear regression by Normal Equation

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

$$\mathbf{X} = \underbrace{\begin{bmatrix} -\mathbf{x}^{(1)^\top} - \\ -\mathbf{x}^{(2)^\top} - \\ -\mathbf{x}^{(3)^\top} - \\ \vdots \\ -\mathbf{x}^{(m)^\top} - \end{bmatrix}}_{\text{input data matrix}}, \quad y = \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix}}_{\text{target vector}}$$

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!

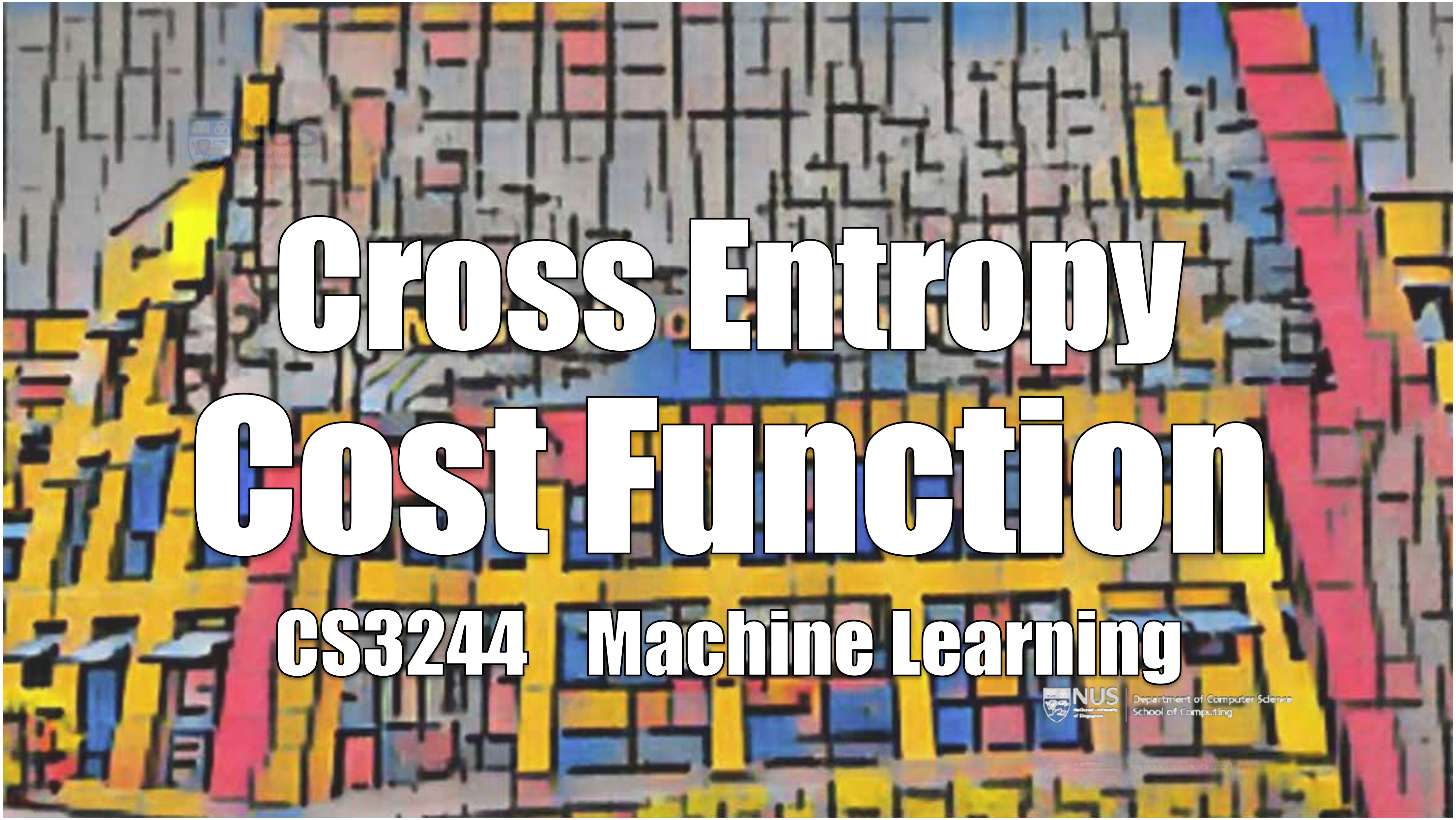
Linear Regression – Summary

Two methods for training

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

Your Turn: What are some reasons that $\mathbf{X}^\top\mathbf{X}$ might not be invertible?

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



Cross Entropy Cost Function

CS3244 Machine Learning

NUS National University of Singapore | Department of Computer Science School of Computing

Logistic Regression Cost Function

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

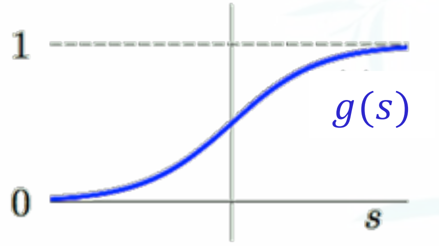
Plausible cost function based on likelihood:

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

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



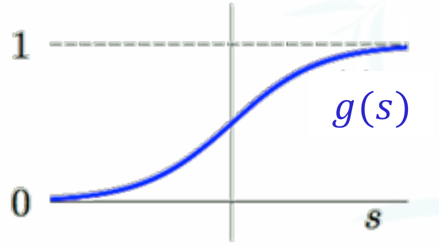
Substitute $h_\theta(\mathbf{x}) = g(\boldsymbol{\theta}^\top \mathbf{x})$,

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 $\mathbf{X} = \left(\mathbf{x}^{(1)}, y^{(1)}\right), \dots, \left(\mathbf{x}^{(m)}, y^{(m)}\right)$
where samples are i.i.d. is:

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

Maximizing the likelihood ≡ Minimizing cross entropy

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

Maximizing the likelihood ≡ Minimizing cross entropy

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

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

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

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

Maximizing the likelihood ≡ Minimizing cross entropy

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

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

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

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

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

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

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

"cross entropy" error

Maximizing the likelihood ≡ Minimizing cross entropy

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

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

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

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

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

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

Plug in

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

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

"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