Some Basics of Machine Learning

Qijia He, PhD student
Time: 10:00am – 11:00am, June 25, 2026

Slide Credits

1. *Introduction to Machine Learning*, 10-401, by Maria-Florina Balcan, Carnegie Mellon University: https://www.cs.cmu.edu/~ninamf/courses/401sp18/lectures.shtml

2. *Machine Learning*, CS4/5780, by Kilian Weinberger, Cornell University: https://www.cs.cornell.edu/courses/cs4780/2017sp/

3. *Intro to Machine Learning*, CS4/CS5780, by Wen Sun, Cornell University: https://www.cs.cornell.edu/courses/cs4780/2023fa/

4. Deep Learning for Computer Vision, CS231n, by Fei-Fei Li et. al., Standford University: https://cs231n.stanford.edu

Index

- Part I: Linear Classifier
- Part II: Linear Regression
- Part III: Gradient Descent (and Beyond)
- Part IV: Neural Networks

Part I: Linear Classifier

Classification

Name	Math	Grade
Alice	90	A
Bob	88	A
Eric	82	A
Lily	77	B
Dick	62	B
Lucy	**76**	**?**

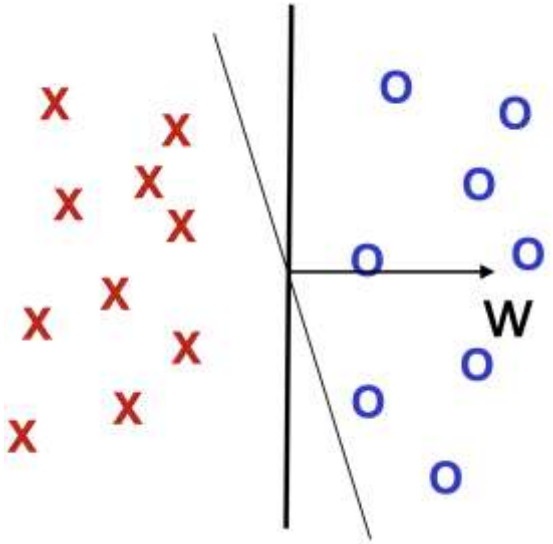
Name	Math	Biology	Art	History	Grade	Total Score
Alice	90	100	65	88	A	343
Bob	88	67	71	56	B	282
Eric	82	87	98	89	A	356
Lily	77	90	79	82	A	328
Dick	62	55	78	61	B	266
Lucy	**76**	**98**	**85**	**92**	**?**	341

KNN (K nearest neighbor)
Separator based Algorithms
Linear Classification

High level idea of Linear Classification:
Treat each subject score as a feature, look at the expert's grades, and learn a weighted combination of those features so that your prediction best matches the expert's data.

Linear Separators

- Feature space $X \in \mathbb{R}^d$.
- Hypothesis class of linear decision boundary in $\mathbb{R}^d$.
  - $h(x) = \boldsymbol{W}^T x + w_0$, where $\boldsymbol{W} = (w_1, \dots, w_d) \in \mathbb{R}^d$
  - If $h(x) \geq 0$, then label x as +, otherwise label it as –



How could we efficiently "learn" the weight ($\boldsymbol{W}$) of those features?

- When observing a new data with label, if the ground truth label is positive but we give a wrong prediction (h(x)<0), we want the boundary to be higher (let $h(x) + +$, such that next time it can have a higher probability to be predicted as positive), same for negative ($h(x) - -$)
- How to increase/decrease $h(x)$? One way is to adjust the weight with $\boldsymbol{W_{t+1}} \leftarrow \boldsymbol{W_t} \pm x$

  Positive samples: $\boldsymbol{W}_{t+1}^T x \leftarrow \boldsymbol{W}_t^T x + x^T x = \boldsymbol{W}_t^T x + ||x||^2$

  Negative samples: $\boldsymbol{W}_{t+1}^T x \leftarrow \boldsymbol{W}_t^T x - x^T x = \boldsymbol{W}_t^T x - ||x||^2$

Linear Separators: Perceptron Algorithm

- Set $t = 1$, start with the all zero vector $\boldsymbol{W_1} = (0, \ldots, 0)$.
- Given example x, predict positive iff $\boldsymbol{W}_t^T x \geq 0$.
- On a mistake, update as follows:
  - Mistake on positive, then update $\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t + x$
  - Mistake on negative, then update $\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t - x$

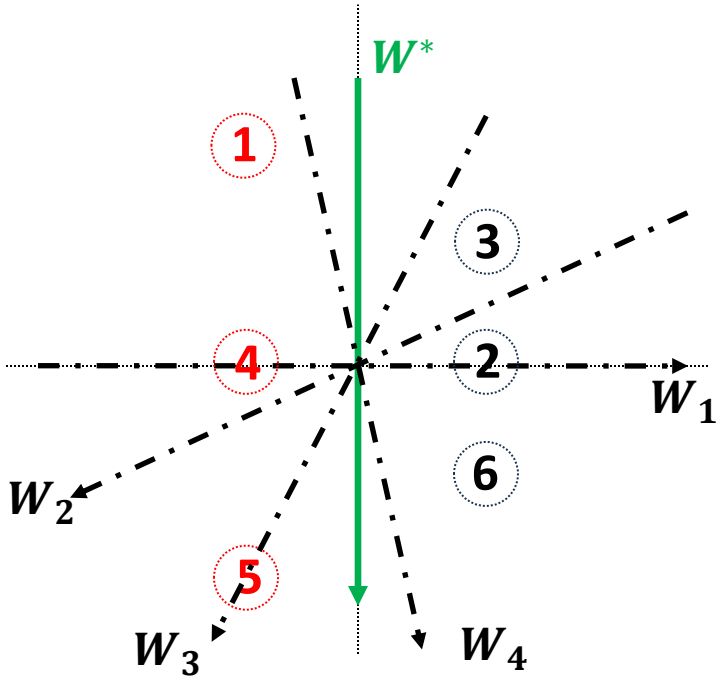
Natural ***greedy*** procedure:

- If true label of x is +1 and $\boldsymbol{W}_t$ incorrect on x, we have $\boldsymbol{W}_t^T x < 0$.
- By Perceptron Algorithm, we have $\boldsymbol{W}_{t+1}^T x \leftarrow \boldsymbol{W}_t^T x + x^T x = \boldsymbol{W}_t^T x + ||x||^2$.
- Then, there will be more chance that $\boldsymbol{W}_{t+1}$ classifies x correctly.
- Similar for mistakes on negative examples.

Perceptron Algorithm: Practice

Example: $W^* = (1, 0)$

1.	$(-1,2)$	-	$W_1 = (0,0)$	✗
2.	$(1,0)$	+	$W_2 = (1,-2)$	✓
3.	$(1,1)$	+	$W_2 = (1,-2)$	✗
4.	$(-1,0)$	-	$W_3 = (2,-1)$	✓
5.	$(-1,-2)$	-	$W_3 = (2,-1)$	✗
6.	$(1,-1)$	+	$W_4 = (3,1)$	✓



Algorithm:

- Set $t = 1$, start with the all zero vector $W_1 = (0, \dots, 0)$.
- Given example x, predict positive iff $W_t^T x \geq 0$.
- On a mistake, update as follows:
  - Mistake on positive, then update $W_{t+1} \leftarrow W_t + x$
  - Mistake on negative, then update $W_{t+1} \leftarrow W_t - x$

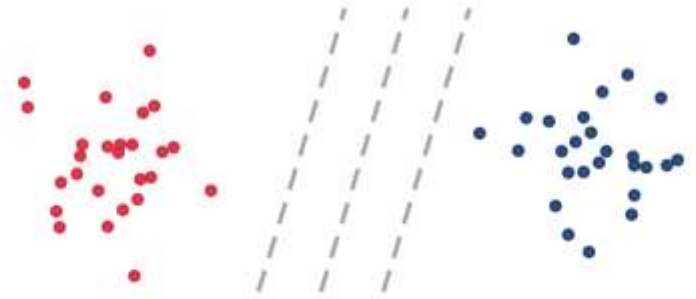
$W_1 = (0,0)$

$W_2 = W_1 - (-1,2) = (1,-2)$

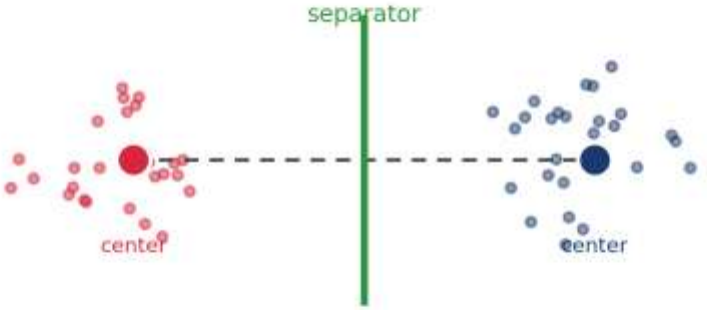
$W_3 = W_2 + (1,1) = (2,-1)$

$W_4 = W_3 - (-1,-2) = (3,1)$

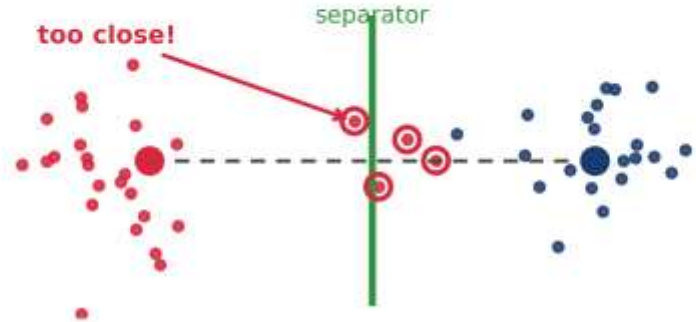
What is a good separator?



There's a lot of ways to separate the two clusters…



Keeping the center of each cluster as far from the separator as possible?



Not a good idea…

We introduce the concept of margin

What Is the Margin?

- The **margin** is the distance from the line to the closest data point — the width of that buffer.
- A bigger margin means more breathing room, and a more confident boundary.

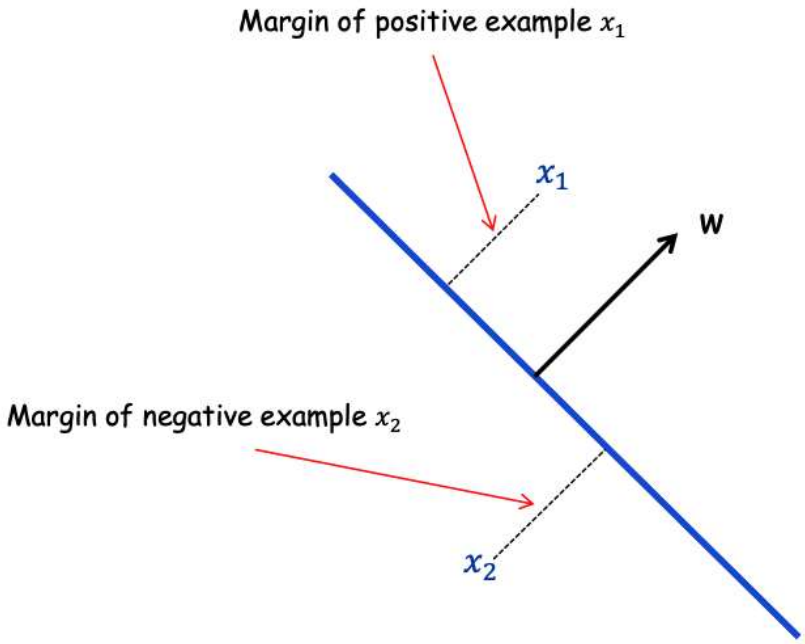
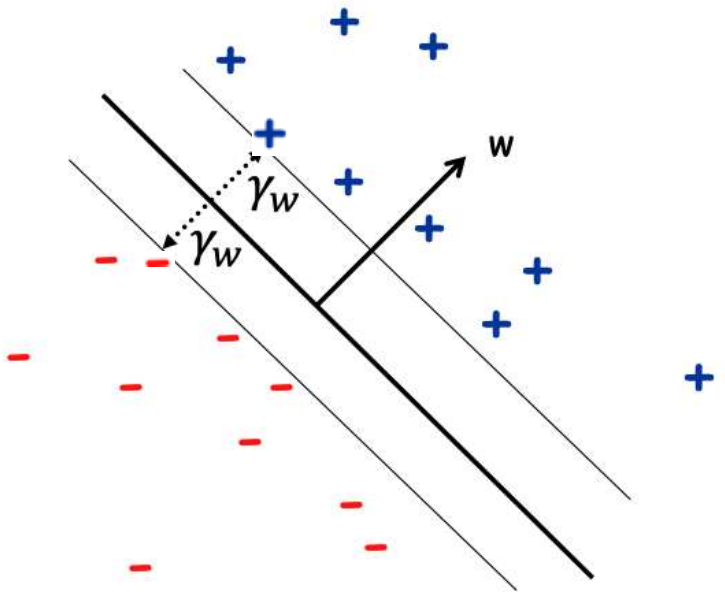


Figure from Balcan (2018)

Support Vector Machines (SVM)

- Many different lines can separate the two groups — but they are not equally good.
- A **Support Vector Machine (SVM)** picks the line with the **largest margin**: the one sitting in the widest possible "street" between the two groups.
- The few points that just touch the edges of the street are the **support vectors** — they alone decide where the line goes.

- Input: $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$;
- Find: some $\boldsymbol{W}$ and maximum γ where:
  - $\left|\left|\boldsymbol{W}\right|\right| = 1$
  - For all $i \in \{1, \dots, m\}, y_i \boldsymbol{W}^T x_i \geq \gamma$
- Output: maximum margin separator over S.



Part II: Linear Regression

From Discrete to Continuous Labels

Classification:

Sports
Science
News

X = Document

Y = Topic

Regression:

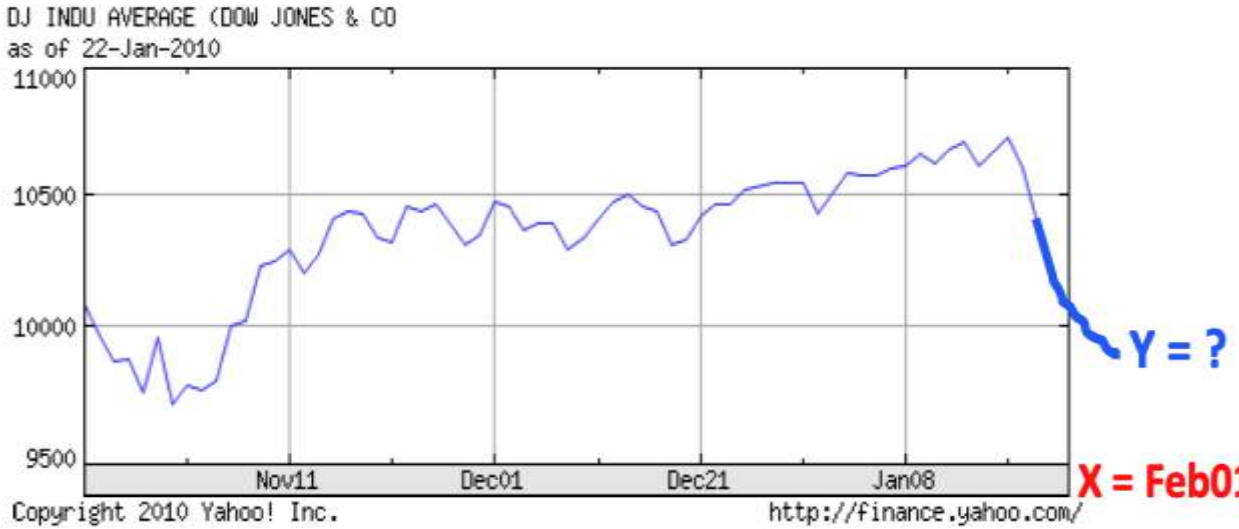


Figure from Balcan (2018)

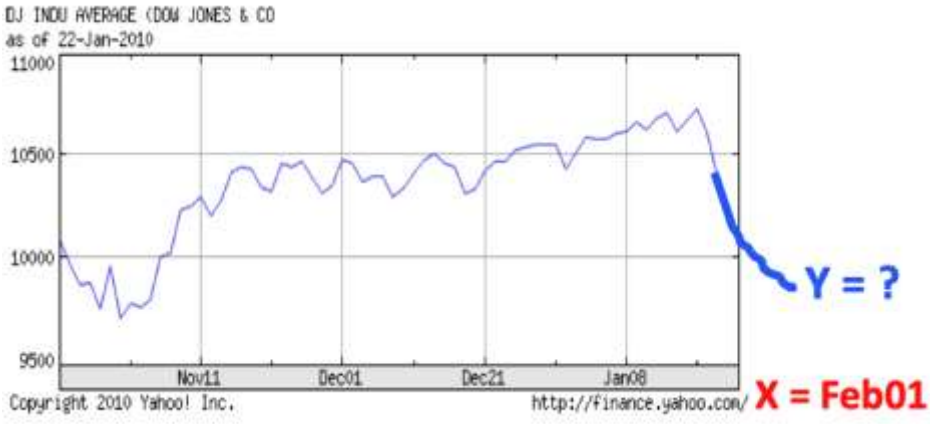
Supervised Learning

Goal: Construct a predictor $f: X \rightarrow Y$ to minimize a risk (error measure) err(f) .

X = Document

Sports
Science
News

Y = Topic



Classification:

$$\text{err}(f) = P(f(X) \neq Y)$$

Probability of error

Regression:

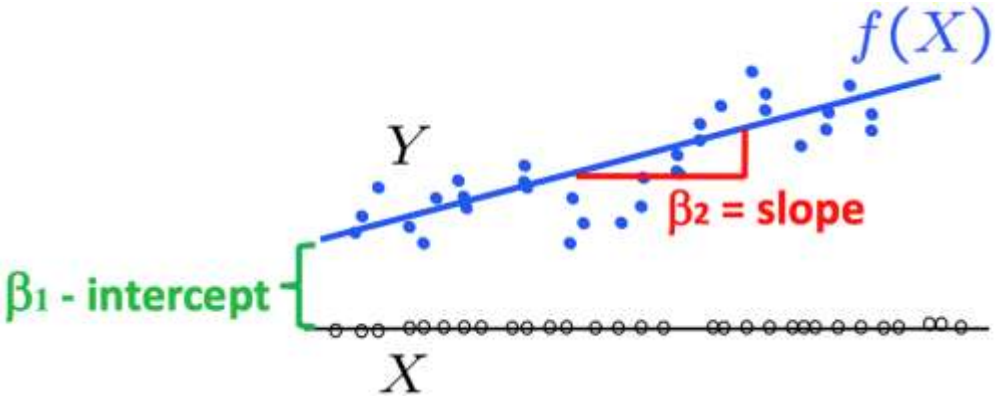
$$\text{err}(f) = E[(f(X) - Y)^2)]$$

Mean Squared Error

Figure from Balcan (2018)

Linear Regression

- Unit-variate case: $f(X) = \beta_1 + \beta_2 X$.



- Multi-variate case:

$$f(X) = f\left(X^{(1)}, \ldots, X^{(p)}\right) = \beta_1 X^{(1)} + \beta_2 X^{(2)} + \ldots + \beta_p X^{(p)}$$
$$= X\beta, \text{ where } X = \left[X^{(1)} \ldots X^{(p)}\right], \beta = [\beta_1 \ \ldots \beta_p]^T$$

- Least square estimator: $\hat{f} = \arg\min_{f \in F} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$, where F is the class of linear functions.

Figure from Balcan (2018)

Least Squares Estimator

- $\hat{f} = \arg\min_{f \in F} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$

- $\hat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (X_i\beta - Y_i)^2$

  $= \arg\min_{\beta} \frac{1}{n} (\mathbf{X}\beta - \mathbf{Y})^T (\mathbf{X}\beta - \mathbf{Y})$

$$\mathbf{X} = \begin{bmatrix} X_1 \\ \dots \\ X_n \end{bmatrix} = \begin{bmatrix} X_1^{(1)} & \dots & X_1^{(p)} \\ \vdots & \ddots & \vdots \\ X_n^{(1)} & \dots & X_n^{(p)} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} Y_1 \\ \dots \\ Y_n \end{bmatrix}.$$

Least Squares Estimator

- $\hat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{X}\beta - \mathbf{Y})^T(\mathbf{X}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$
- $J(\beta) = (\mathbf{X}\beta - \mathbf{Y})^T(\mathbf{X}\beta - \mathbf{Y})$
- $\frac{\partial J(\beta)}{\partial \beta}|_{\hat{\beta}} = 0$

- $(\mathbf{X}^T\mathbf{X})\hat{\beta} = \mathbf{X}^T\mathbf{Y}$
- If $\mathbf{X}^T\mathbf{X}$ is invertible,

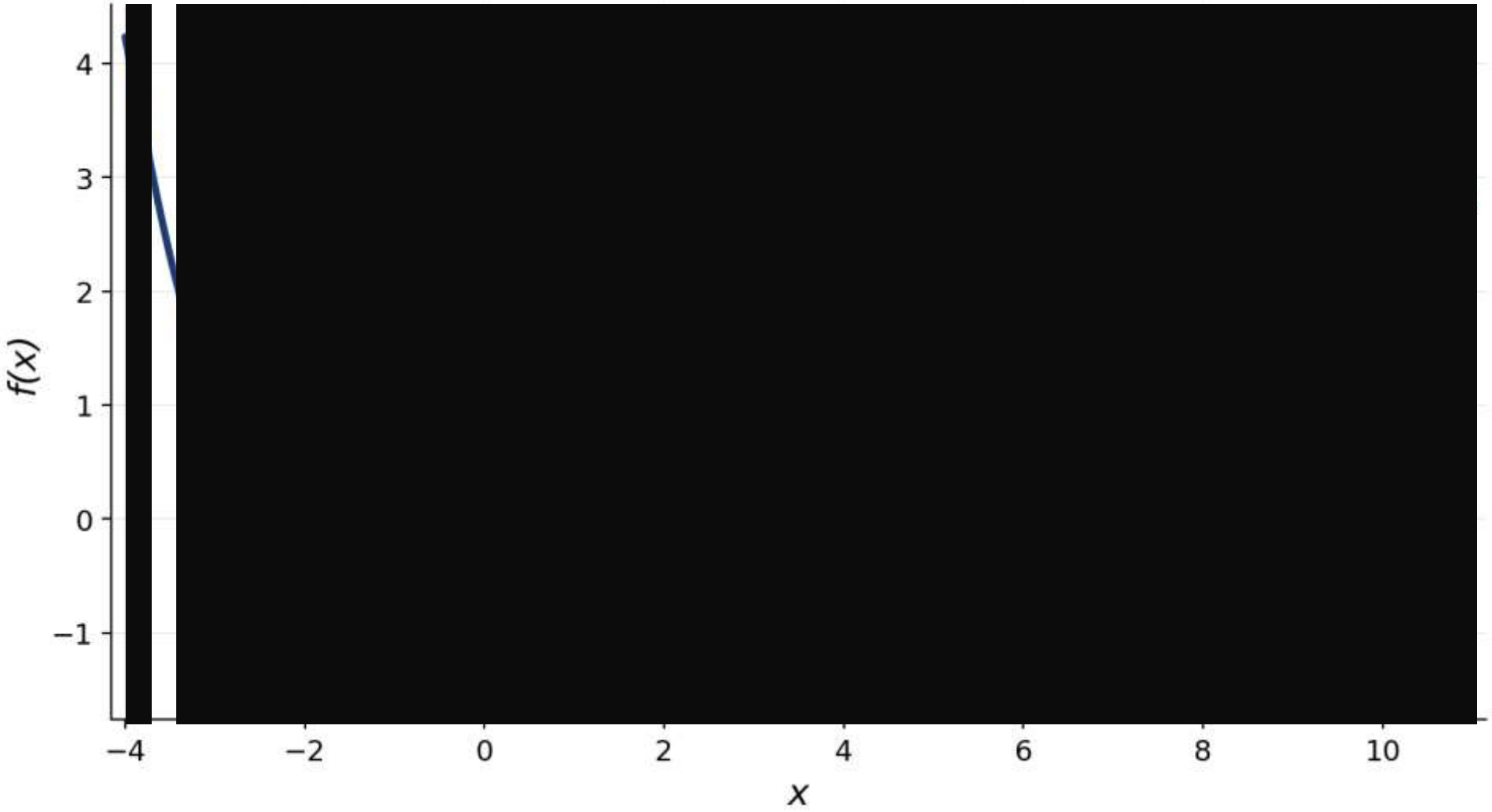
$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \qquad \hat{f}(\boldsymbol{X}) = \mathbf{X}\hat{\beta}$$

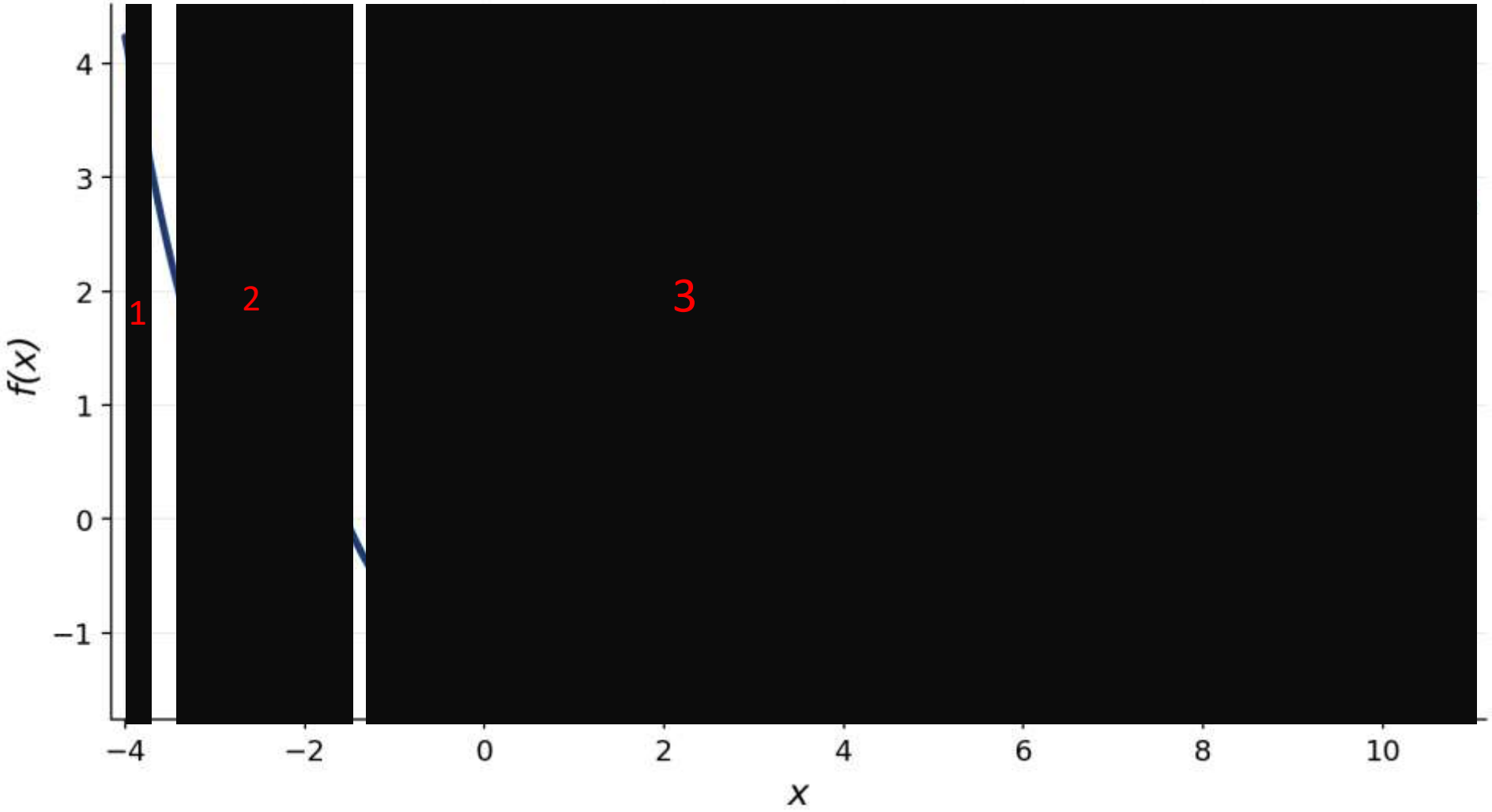
It is lucky that linear regression has such a nice closed form solution. However, for most ML problems, we cannot express the result in a mathematical form

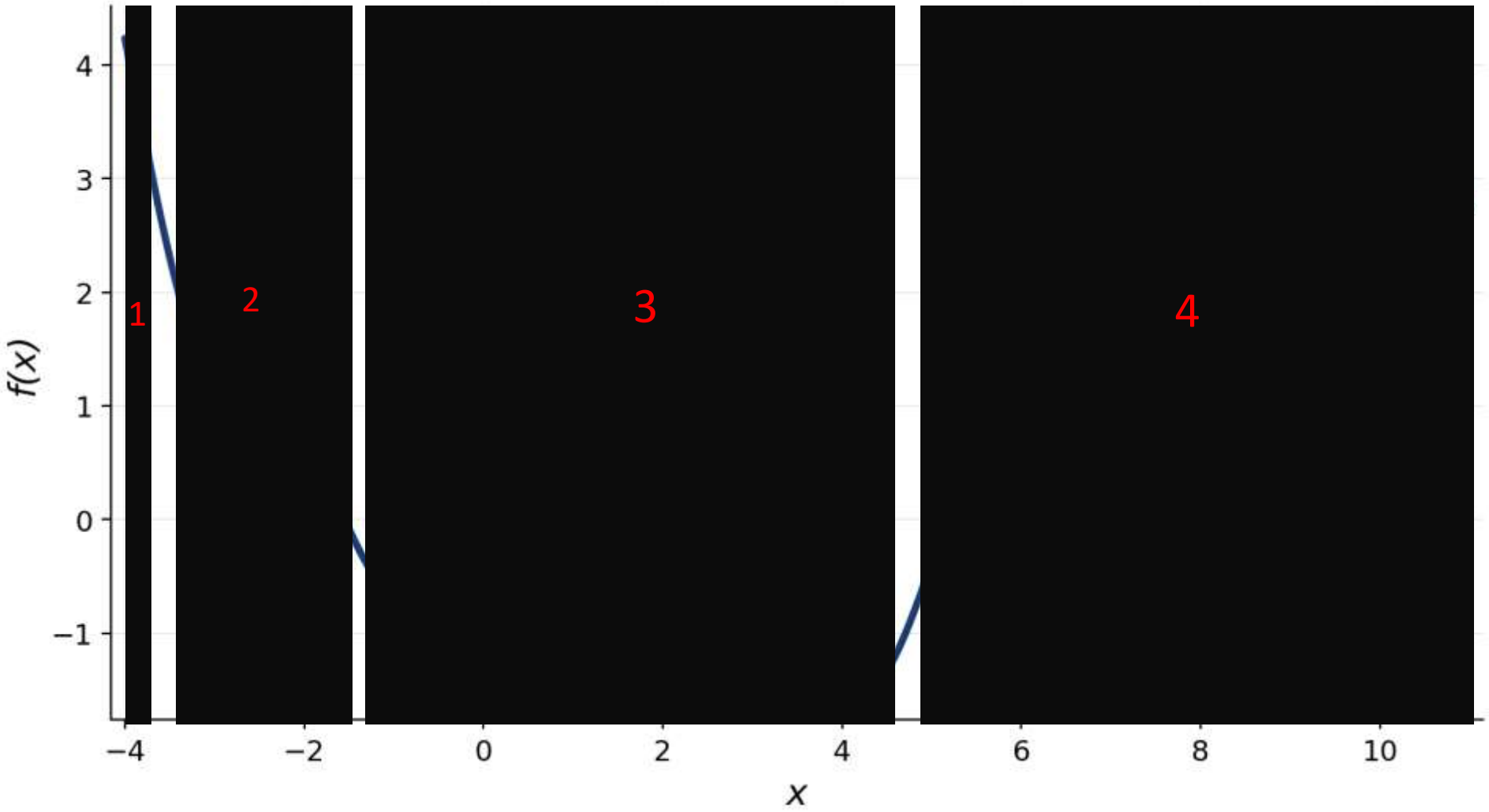
Most of time, The model is like a black box: We can calculate the predicted value given X, taking derivative, but can not find solution for $\frac{\partial J(\beta)}{\partial \beta}|_{\hat{\beta}} = 0$.

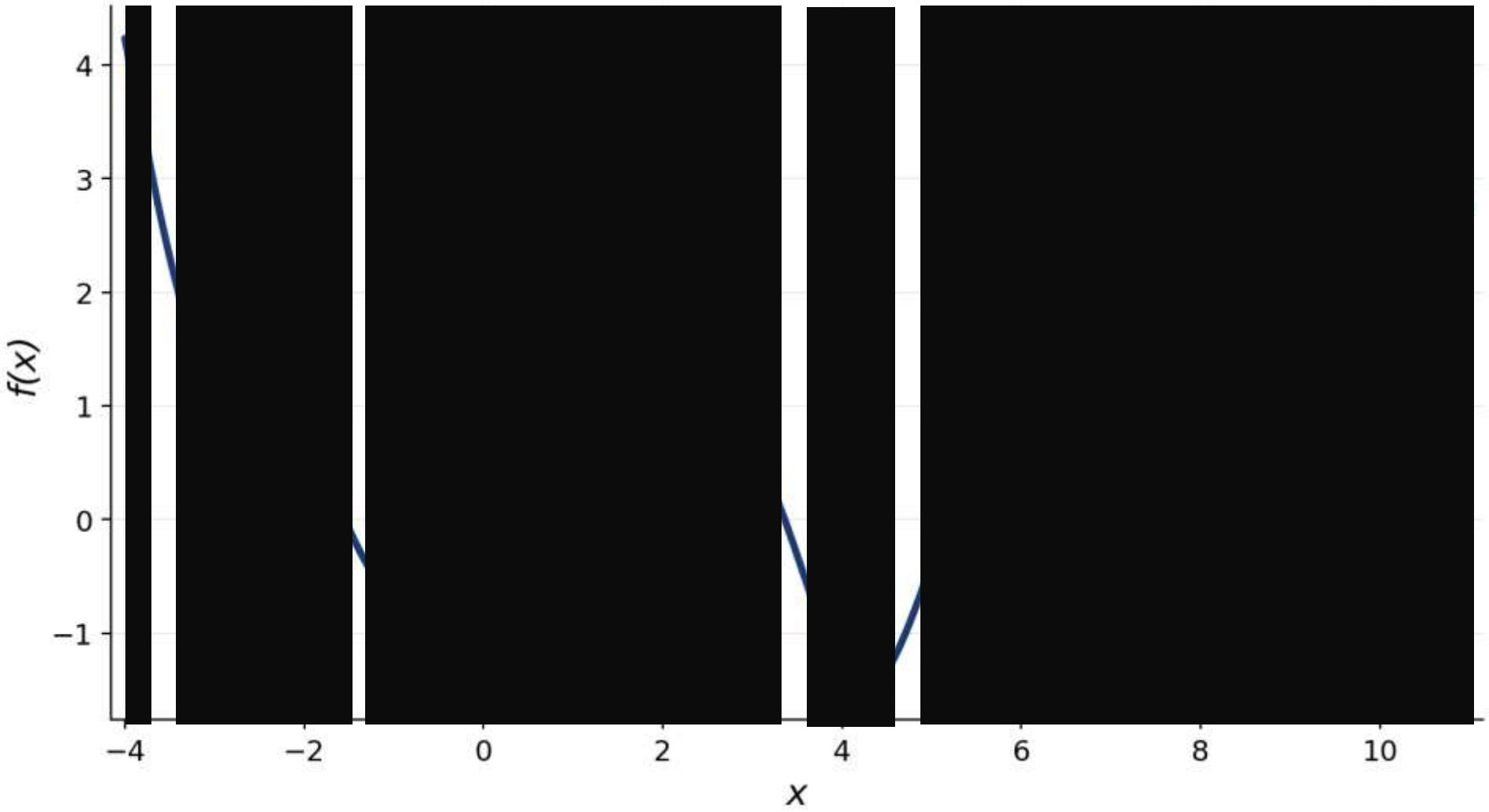
Part III: Gradient Descent (and Beyond)

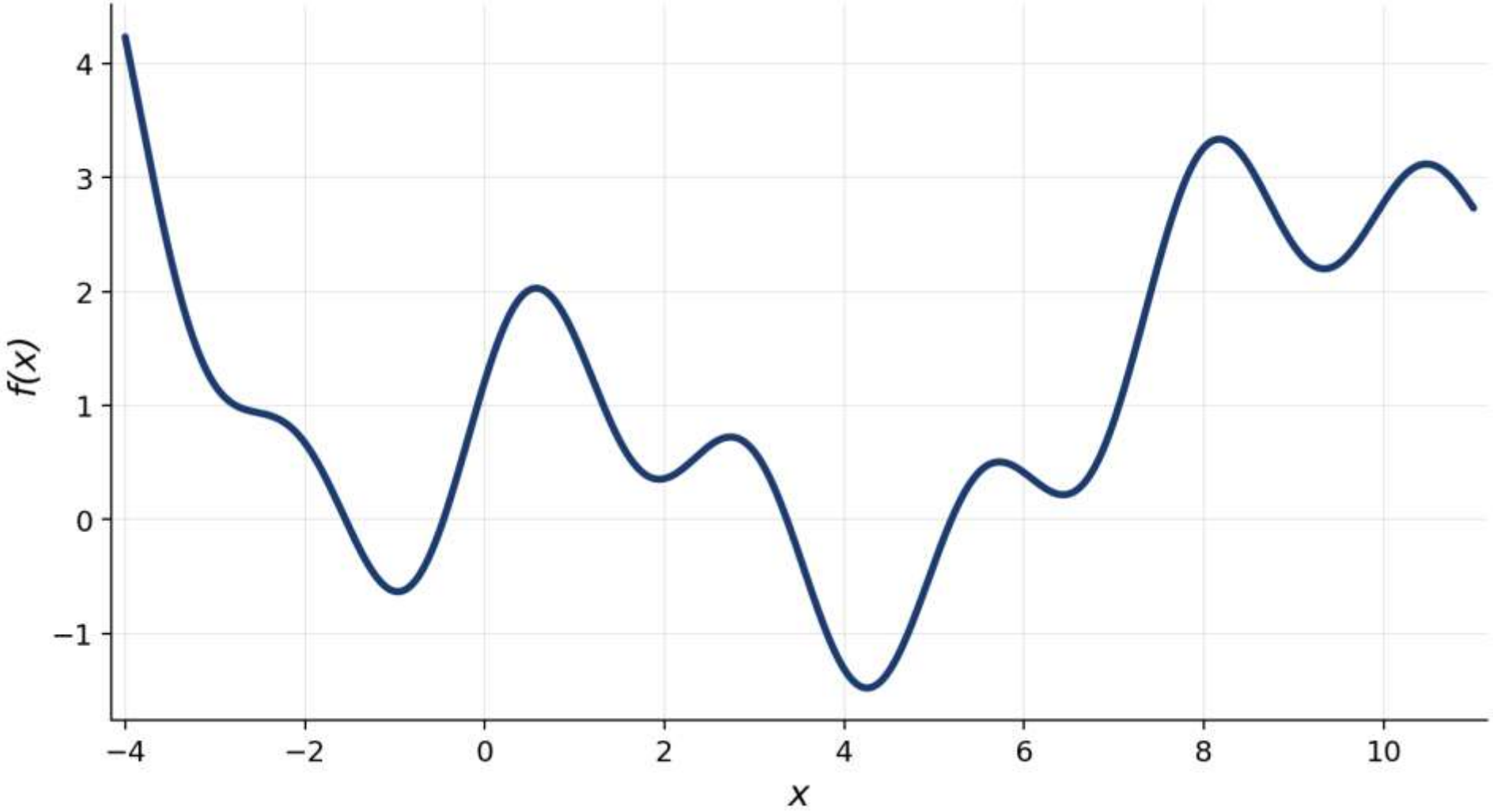

X
Model
???
y


4
3
2
1
0
−1
f(x)
−4
−2
0
2
4
6
8
10
x


1
2
3
f(x)
x
4
3
2
1
0
−1
−4
−2
0
2
4
6
8
10


1
2
3
4
f(x)
x
4
3
2
0
−1
−4
−2
0
2
4
6
8
10


4
3
2
1
0
−1
f(x)
−4
−2
0
2
4
6
8
10
x


4
3
2
1
0
−1
f(x)
−4
−2
0
2
4
6
8
10
x

Gradient Descent (GD)

- In GD, we only use the gradient (first order).
- We assume the function l around $\boldsymbol{W}$ is linear and behaves like $l(\boldsymbol{W}) + g(\boldsymbol{W}) \cdot s$.
- Our objective is to find a vector s that minimizes function l.
- In steepest descent we simply set

$$s = -\eta \cdot g(\boldsymbol{W})$$

for some small $\eta > 0$.

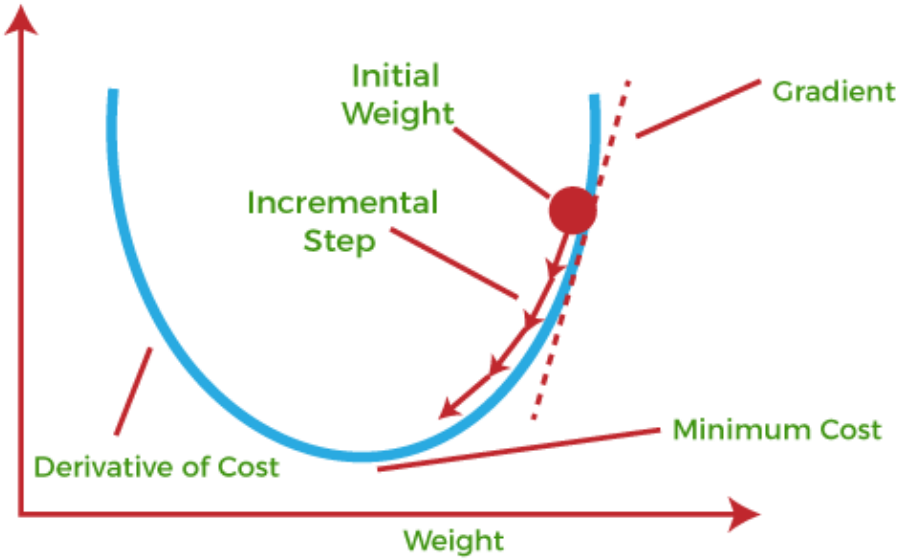
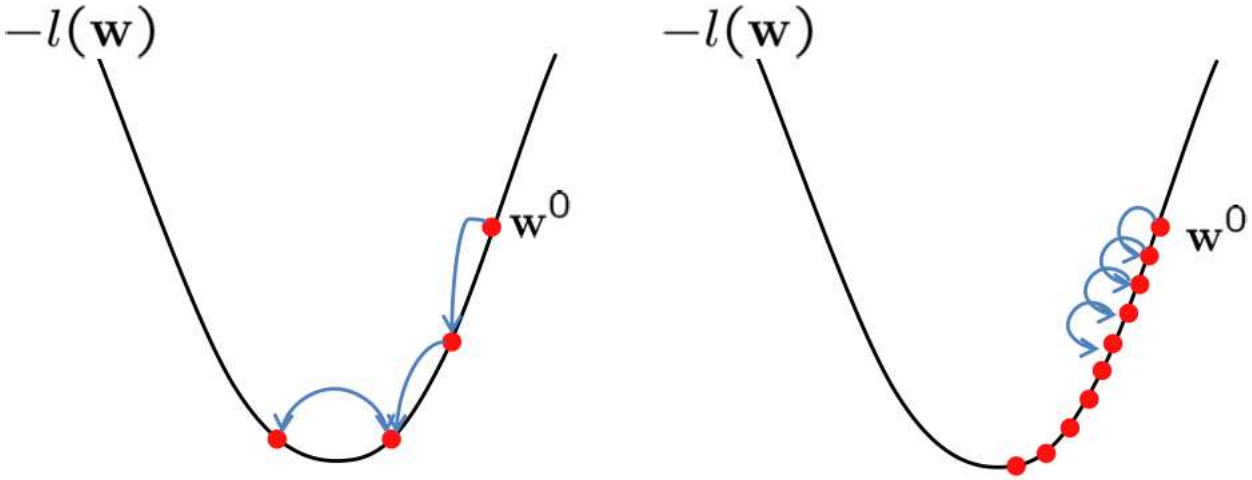


Figure from Kharkar (2023)

GD: Learning Rate

- Setting the learning rate $\eta > 0$ is a dark art.
  - Large $\eta \Longrightarrow$ Fast convergence but larger residual error $||\boldsymbol{W^{t+1}} - \boldsymbol{W^{t}}||_2$, with possible oscillations.
  - Small $\eta \Longrightarrow$ Slow convergence but small residual error.



- A safe (but sometimes slow) choice is to set $\eta = \frac{1}{t}$, which guarantees that it will eventually become small enough to converge.

Figure from Balcan (2018)

GD: In Practice

In ML, the loss we minimize typically has some special form, e.g.,

$$l(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + \exp(-y_i(\boldsymbol{W}^T x_i)))$$

Average over n data points

To compute the gradient $\nabla l(\boldsymbol{W})$, we need to enumerate all n training data points, which can be very slow!

Stochastic GD (SGD) to Rescue

In ML, the loss we minimize typically has some special form, e.g.,

$$l(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + \exp(-y_i(\boldsymbol{W}^T x_i)))$$

Average over n data points

Idea: Randomly sample a data point (x, y), use $\nabla l(x, y; \boldsymbol{W})$ to replace $\nabla l(\boldsymbol{W})$.

Stochastic GD (SGD)

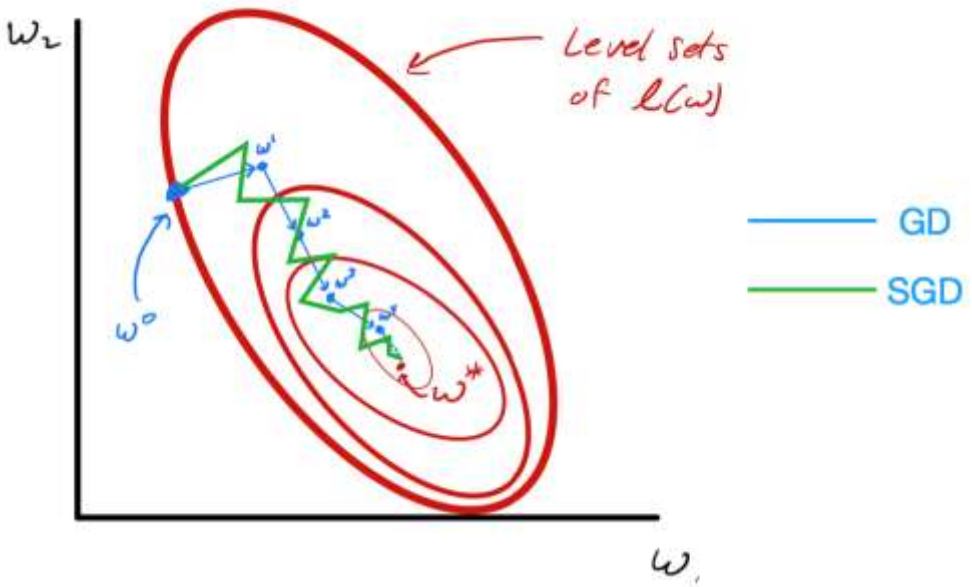
- Goal: minimize $l(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} l(x_i, y_i; \boldsymbol{W})$

- Initialize: $\boldsymbol{W}^0 \in \mathbb{R}^d$ randomly
- Iterate until convergence:
  1. Randomly sample a point (x_i, y_i) from the n data points
  2. Compute noisy gradient $\tilde{g}^t = \nabla l(x_i, y_i; \boldsymbol{W})|_{\boldsymbol{W}=\boldsymbol{W}^t}$
  3. Update (GD): $\boldsymbol{W^{t+1}} = \boldsymbol{W^t} - \eta \tilde{g}^t$

Why Can SGD Work?

Claim: the random noisy gradient is an unbiased estimate of the true gradient

$$\mathbb{E}[\nabla l(x_i, y_i; \boldsymbol{W})] = \frac{1}{n}\sum_{i=1}^{n} \nabla l(x_i, y_i; \boldsymbol{W}) = \nabla\left[\frac{1}{n}\sum_{i=1}^{n} l(x_i, y_i; \boldsymbol{W})\right] = \nabla l(\boldsymbol{W})$$



Q: Which one is faster?

A: SGD is slower in theory, but is often much faster in practice.

Figure from Sun (2022)

Revisiting Regression

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

- It might be computationally expensive if $\mathbf{X}$ is huge or $(\mathbf{X}^T\mathbf{X})^{-1}$ is not invertible.

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{X}\beta - \mathbf{Y})^T(\mathbf{X}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

- Gradient Descent since $J(\beta)$ is convex
  - Initialize: β^0
  - Update: $\beta^{t+1} = \beta^t - \frac{\eta}{2}\mathbf{X}^T \frac{\partial J(\beta)}{\partial \beta}\Big|_t$
    $= \beta^t - \eta\mathbf{X}^T(\mathbf{X}\beta^t - \mathbf{Y})$
  - Stop: when some criterion met, e.g., fixed # iterations, or $\frac{\partial J(\beta)}{\partial \beta}\Big|_t < \varepsilon$.

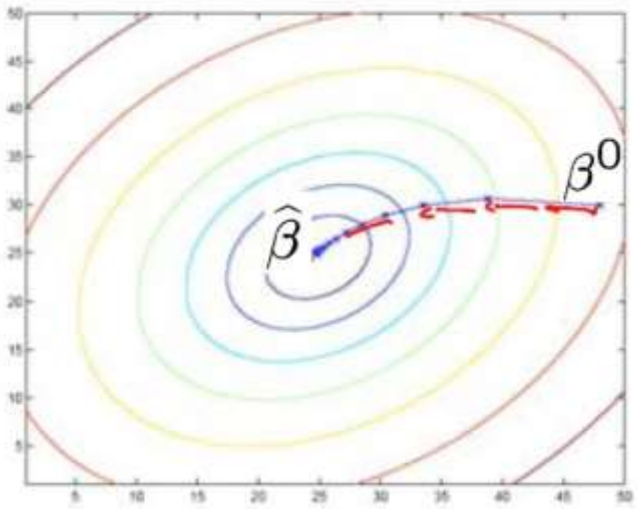


Figure from Balcan (2018)

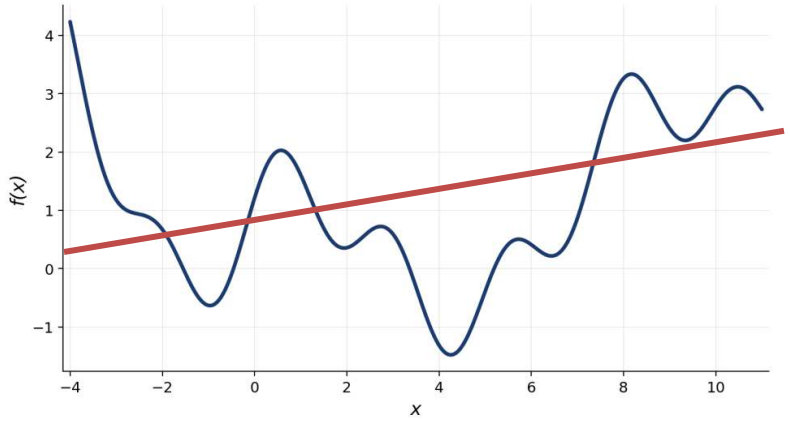
Part IV: Neural Networks

Neural Networks

Recall linear (regression) model:

$$\mathrm{Y} = \mathrm{Wx} + \mathrm{b}$$

Definitely not good for a non-linear model!



How about add some nonlinearity towards the linear feature?

$$\mathrm{Y} = \mathrm{f}(\mathrm{Wx} + \mathrm{b})$$

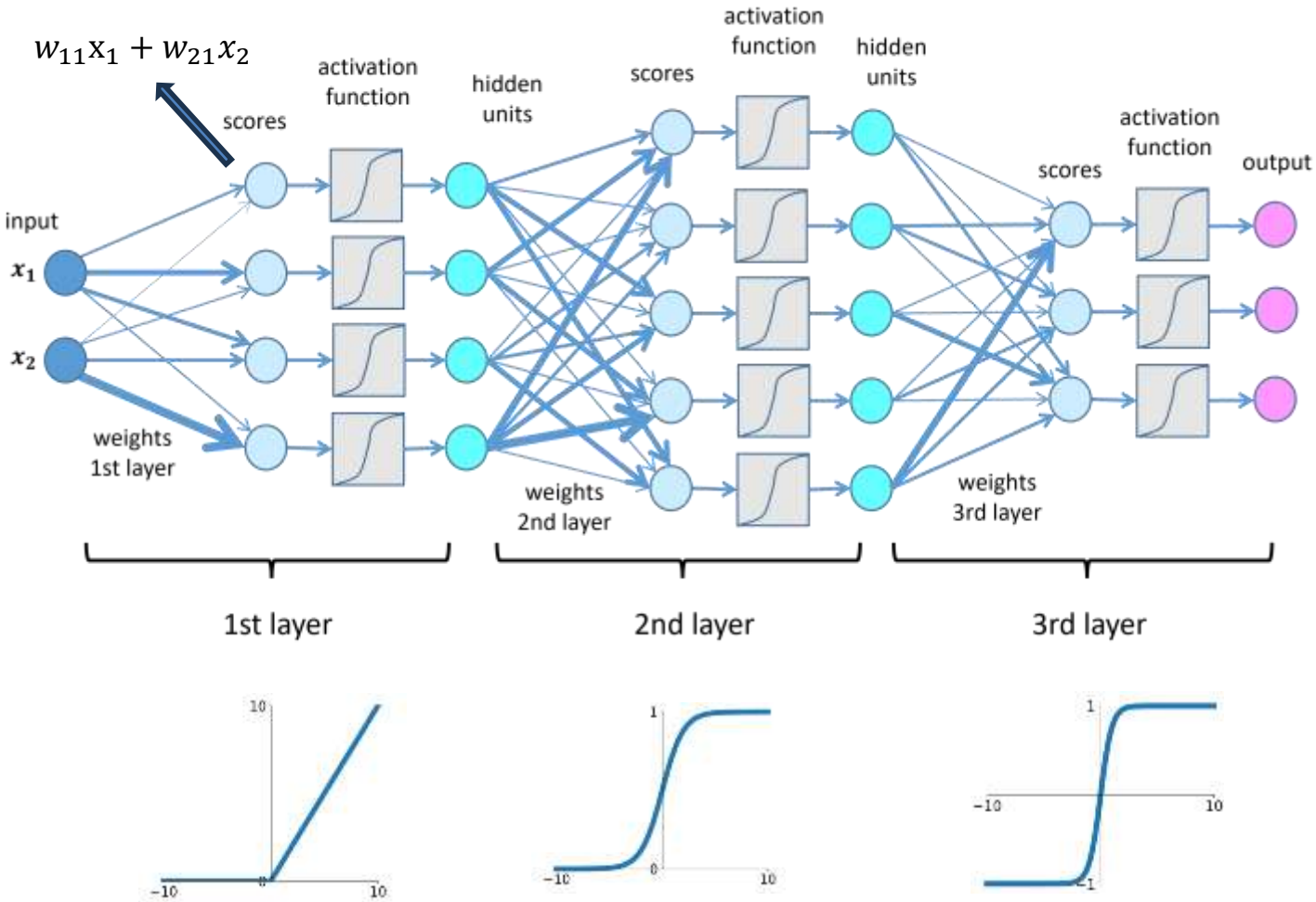
Where $\mathrm{f}(x)$ is a nonlinear function, i.e. e^x, $\frac{1}{1+\mathrm{e}^x}$

Perhaps one nonlinear function is not enough to capture the function space

Let's do more!

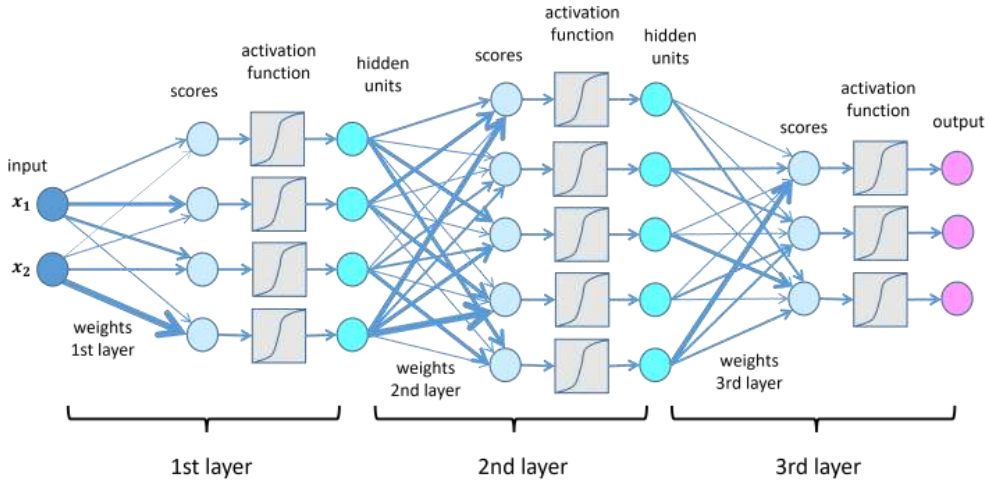
$$\mathrm{Y} = \mathrm{f}(\mathrm{Wx} + \mathrm{b})$$
$$\mathrm{Y} = \mathrm{f}(W\mathrm{f}(\mathrm{Wx} + \mathrm{b}) + \mathrm{b})$$
$$\mathrm{Y} = \mathrm{f}(W\mathrm{f}(W\mathrm{f}(\mathrm{Wx} + \mathrm{b}) + \mathrm{b}) + \mathrm{b})$$
$$\ldots$$

Neural Networks



- ReLU: $\sigma(x) = \max(0, x)$ Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$ tanh: $\sigma(x) = \tanh(x)$

Neural Networks



For each training steps:

1. We sample a batch of data (x1, x2)
2. Let the sampled data go through the Neural Networks, and get predictions (y1 y2)
3. We calculate the loss (e.g. MSE: $E[(f(X) - Y)^2)]$), and compute the gradients
4. Update the weights using Stochastic Gradient Descent

Language Task: Autoregressive Model

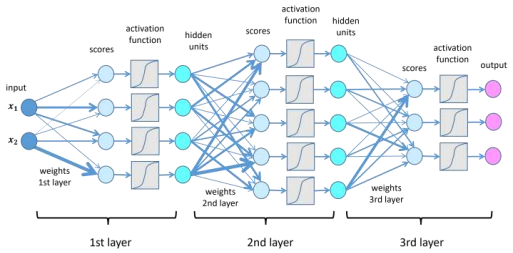
prefix

autoregressive language model (Encode and decode)

next-token distribution

Sampled token

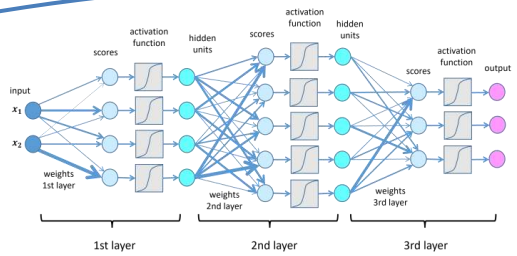
How are you



P(I'm|How are you)=0.6
P(good|How are you)=0.3
…

I'm

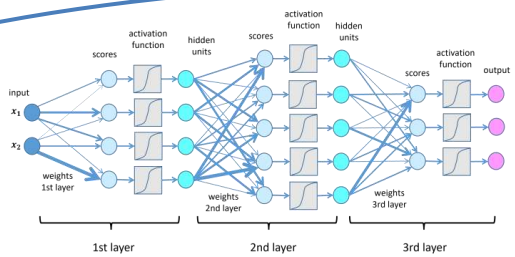
How are you. I'm



P(good|How are you. I'm)=0.5
P(fine|How are you. I'm)=0.45
…

fine

How are you I'm fine

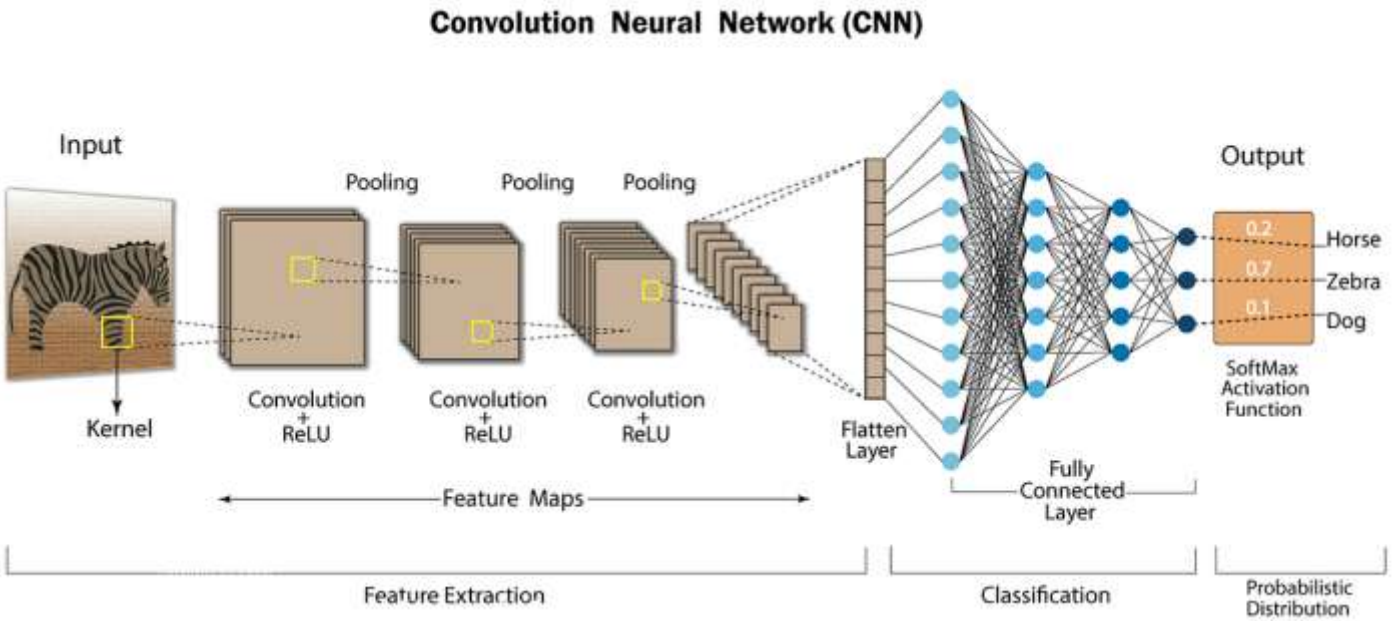


P(<EOS>|How are you. I'm fine)=0.8
P(Thank|How are you. I'm fine)=0.12
…

<EOS>

Final response: I'm fine ← External feedback: score=8

Vision task: Convolutional Neural Networks



Two new mechanisms:

1. Convolutional mechanism to capture the spatial information
2. Use Pooling to do dimensional reduction and extract key information

A Case of CNNs

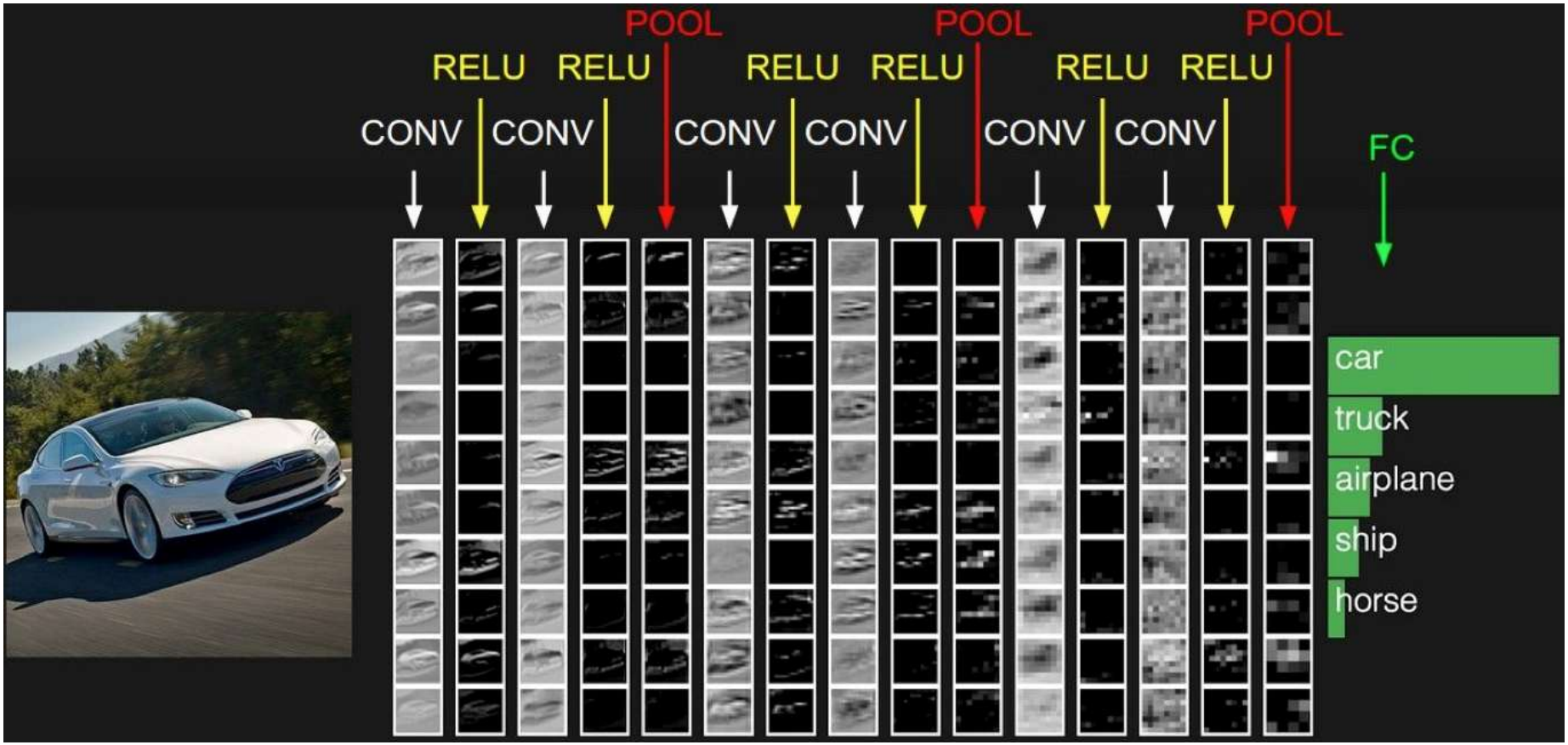


Figure from *Feifei Li & Andrej Karpathy (2016)*


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
bedankt
спасибо
dziękuję
go raibh maith agat
mochchakkeram
sukriya
kop khun krap
terima kasih
grazie
arigatō
dakujem
мерси
tapadh leat
dank je
takk
taiku
sagolun
mauruuru
hvala
감사합니다
ありがとう
ευχαριστώ
рахмат
Баярлалаа
faafetai lava
kiitos
dankie
dhanyavad
enkosi
mersi
barka
welalin
tack
misaotra
matondo
paldies
grazzi
blagodaram
vinaka
спасибі
хвала
manana
obrigada
chnorakaloutioun
gratias ago
gràcies
sulpáy
trugarez
tanemirt
rahmet
diolch
dhanyavadagalu
shukriya
mesi
najis tuke
kam sah hamnida
xiexie
sobodi
dekuji
mamnun
chokrane
koszönöm
asante
spas

धन्यवाद

Illustration: Two-Dimensional Case

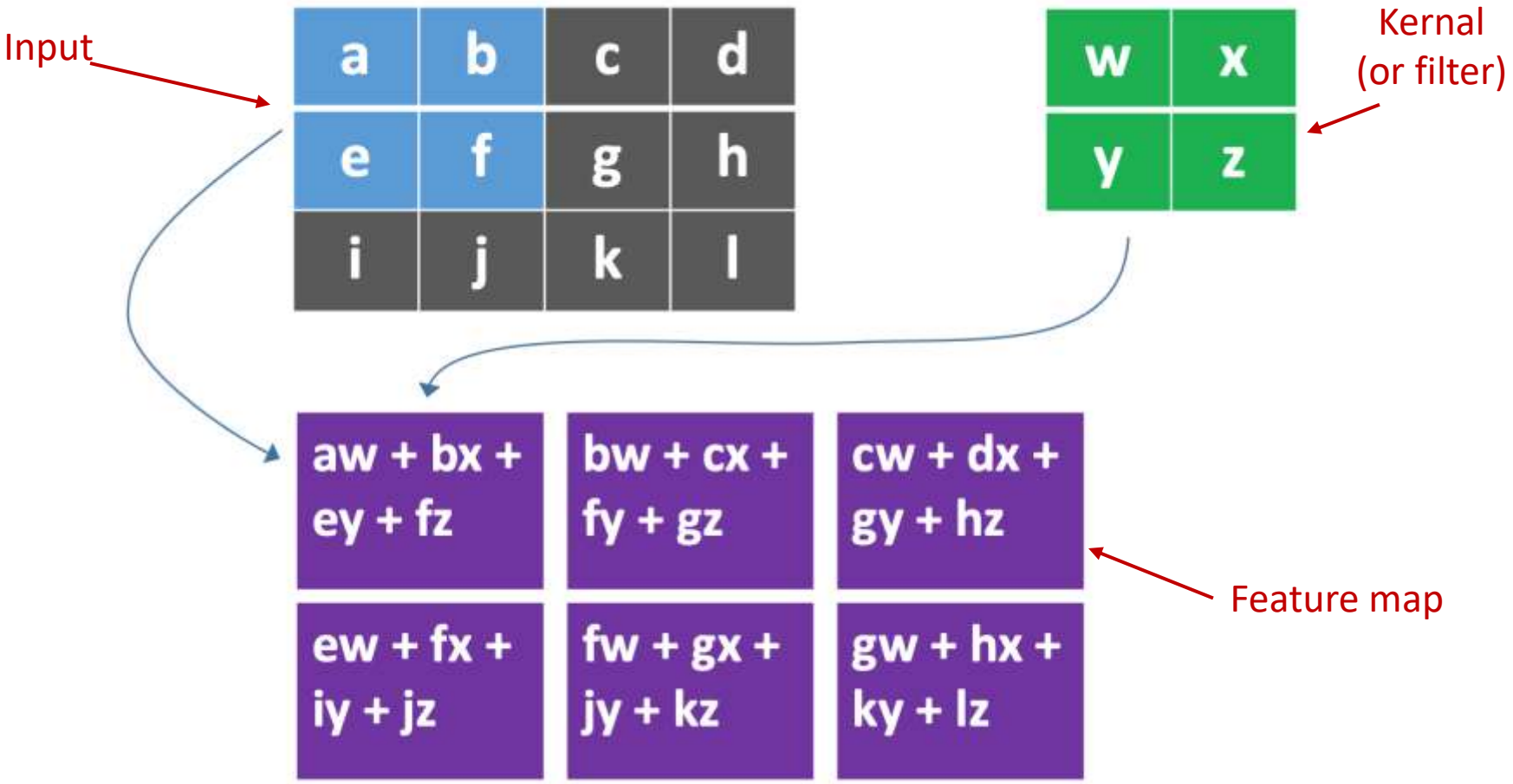
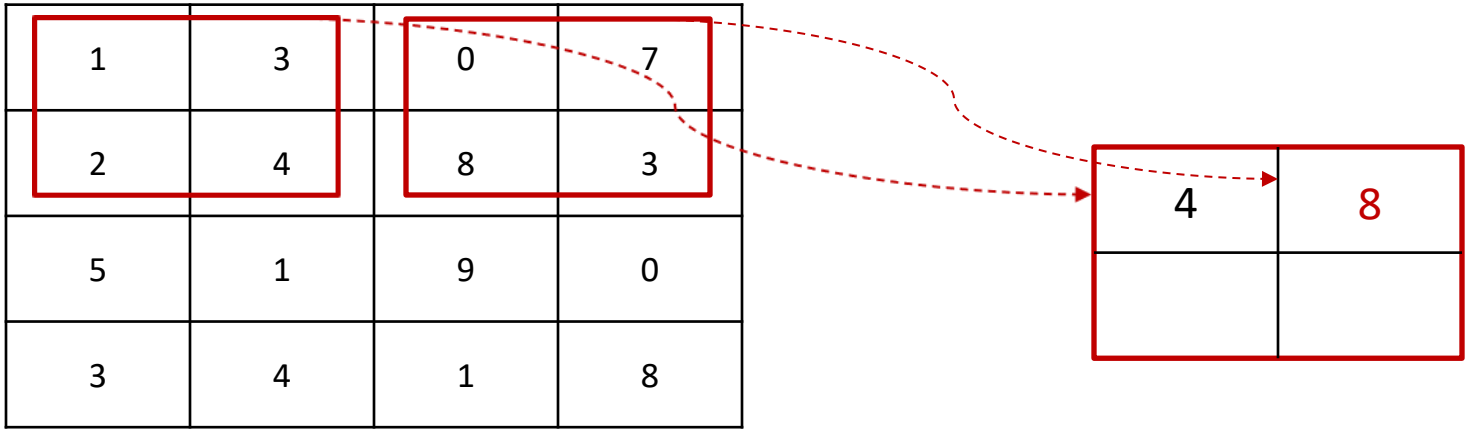


Figure from Balcan (2018)

Other Layers: Pooling Layer

- We use a pooling layer to downsize the inputs.
- For example, max pooling (2x2 filter and stride 2)

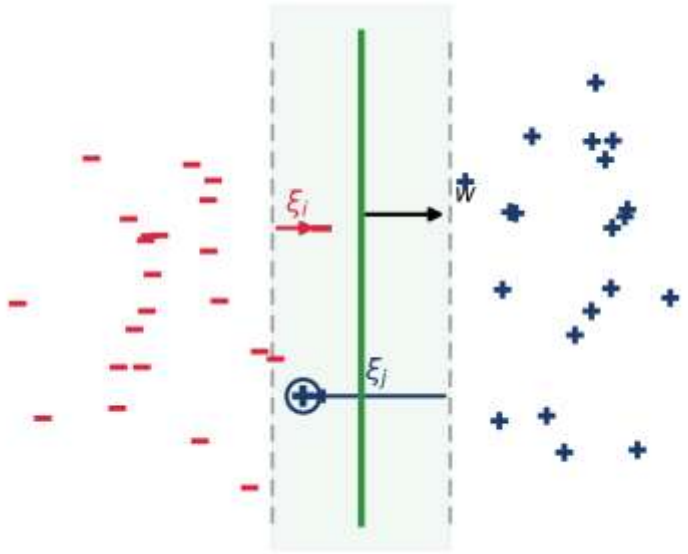


Cases where margin fails

We can't separate the two classes perfectly — so we allow a few mistakes, at a cost.

Soft margin: relax the constraint, penalize the slack

- Input: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$;
- Find: some $\boldsymbol{W}$ and maximum $\gamma - C \sum_{i=1}^{m} \xi_i$ where:
  - $\left|\left|\boldsymbol{W}\right|\right| = 1$
  - For all $i \in \{1, \ldots, m\}, y_i \boldsymbol{W}^T x_i \geq \gamma - \xi_i$
- Output: maximum margin separator over S.



- ξ measures how far a point breaks the margin ($\xi > 1 \Rightarrow$ misclassified).
- Penalty $C \sum_{i=1}^{m} \xi_i$ trades off: large C $\Rightarrow$ fewer mistakes, narrow margin; small C $\Rightarrow$ wider margin.