Some Basics of Machine Learning

Hongbo Li, Postdoc Scholar
Time: 2:00pm – 3:00pm, June 10, 2025

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

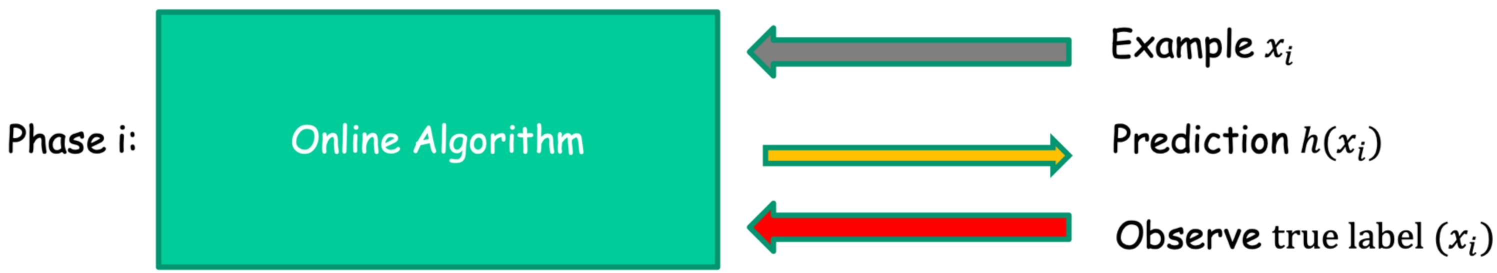
- Part I: Linear Classifier
- Part II: Gradient Descent (and Beyond)
- Part III: Linear Regression
- Part IV: Convolutional Neural Networks (CNNs)

Part I: Linear Classifier

Online Learning Model

- Examples arrive sequentially.

For i=1, 2, …, :

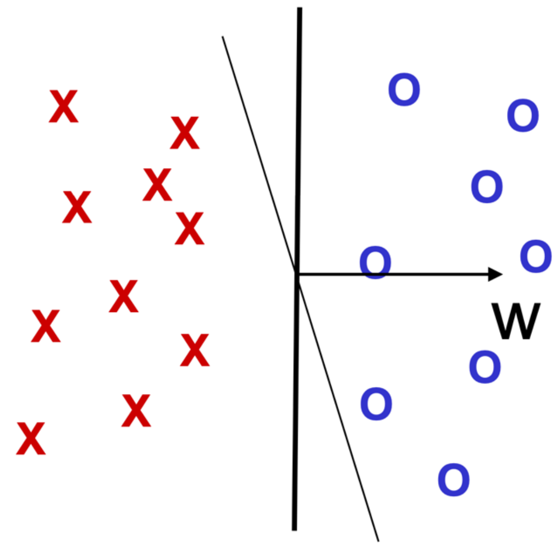


- Objective: Minimize the number of mistakes.
- Applications:
  - Email classification (distribution of both spam and regular mail changes over time).
  - Recommendation systems.
  - …

Figure from Balcan (2018)

Linear Separators

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



Trick: Without loss of generality $w_0 = 0$.

Proof: Can simulate a non-zero threshold with a dummy input feature x_0 that is always set up to 1.

- $x = (x_1, \ldots, x_d) \rightarrow \tilde{x} = (x_1, \ldots, x_d, 1)$
- $\boldsymbol{W}^T x + w_0 \geq 0$ iff $\widetilde{\boldsymbol{W}}^T \tilde{x} \geq 0$, where $\widetilde{\boldsymbol{W}} = (w_1, \ldots, w_d, w_0)$

Figure from Balcan (2018)

Linear Separators: Perceptron Algorithm

- Set $\mathrm{t} = 1$, start with the all zero vector $\boldsymbol{W}_{\mathbf{1}} = (0, \dots, 0)$.
- 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 $\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t - x$

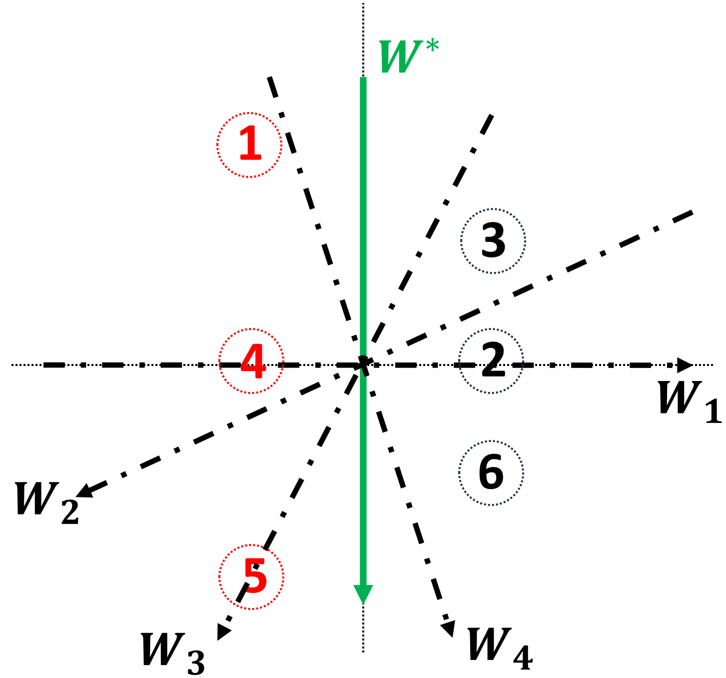
Natural ***greedy*** procedure:

- 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 $W_{t+1} \leftarrow W_t - x$

$$W_1 = (0,0)$$
$$W_2 = W_1 - (-1,2) = (1,-2)$$
$$W_3 = W_2 + (1,1) = (2,-1)$$
$$W_4 = W_3 - (-1,-2) = (3,1)$$

Geometric Margin

Definition: The margin of example x w.r.t a linear separator $\boldsymbol{W}$ is the distance from x to the plane $\boldsymbol{W}^T x = 0$ (or the negative if on wrong side)

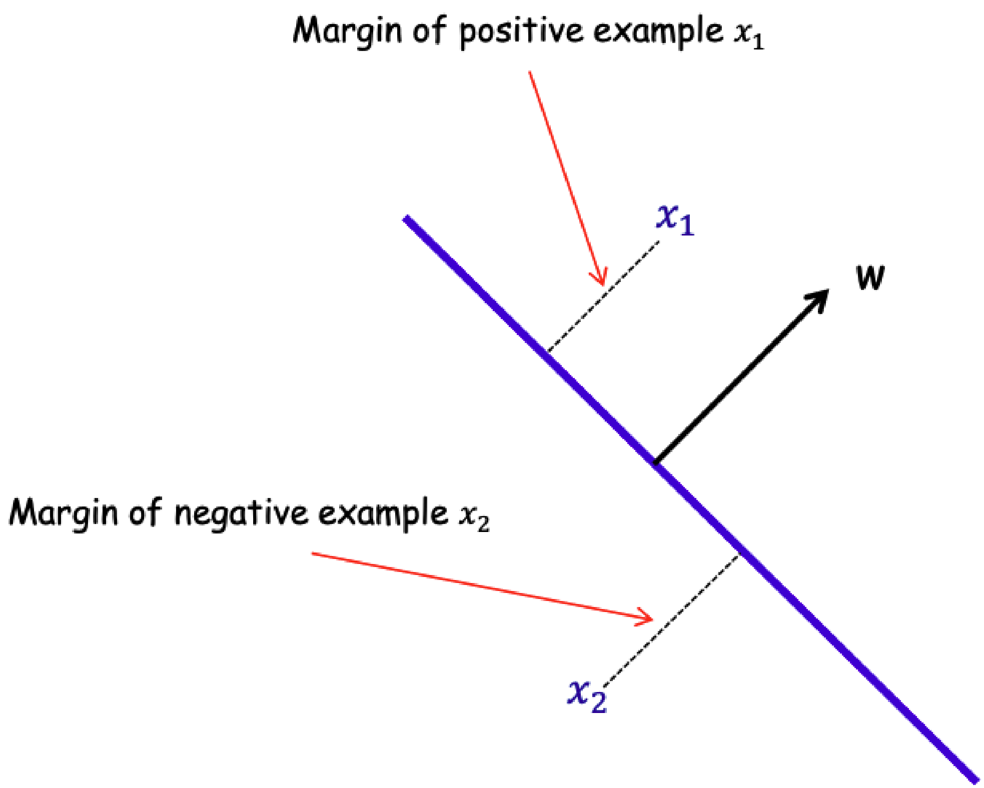


Figure from Balcan (2018)

Geometric Margin

Definition: The margin of example x w.r.t a linear separator $\boldsymbol{W}$ is the distance from x to the plane $\boldsymbol{W}^T x = 0$ (or the negative if on wrong side)

Definition: The margin γ_W of a set of example S w.r.t a linear separator $\boldsymbol{W}$ is the smallest margin over points $x \in S$.

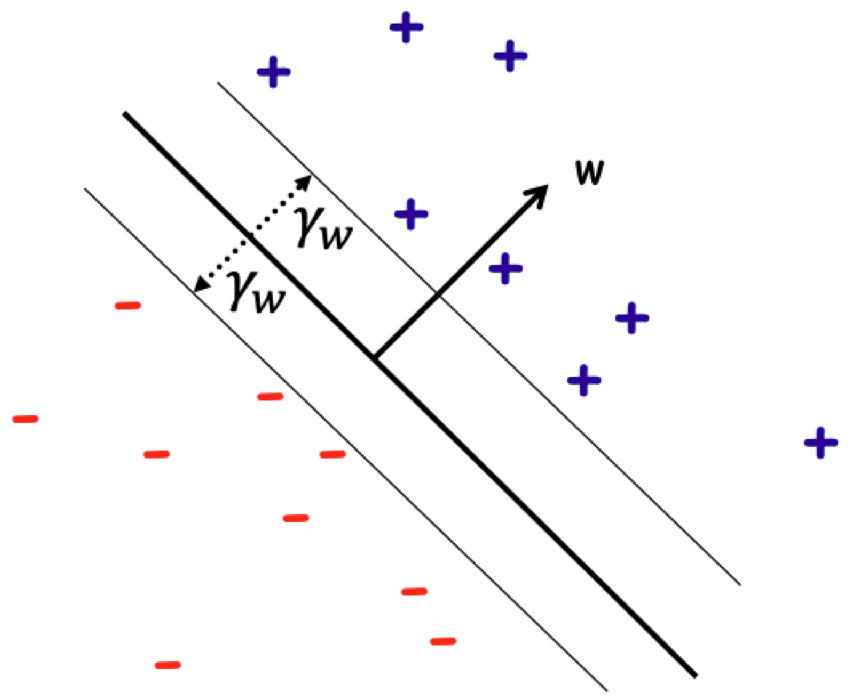


Figure from Balcan (2018)

Geometric Margin

Definition: The margin of example x w.r.t a linear separator $\boldsymbol{W}$ is the distance from x to the plane $\boldsymbol{W}^T x = 0$ (or the negative if on wrong side)

Definition: The margin γ_W of a set of example S w.r.t a linear separator $\boldsymbol{W}$ is the smallest margin over points $x \in S$.

Definition: The margin γ of a set of example S is the maximum γ_W over all linear separators w.

Q: Why is it defined to be the maximum γ_W?

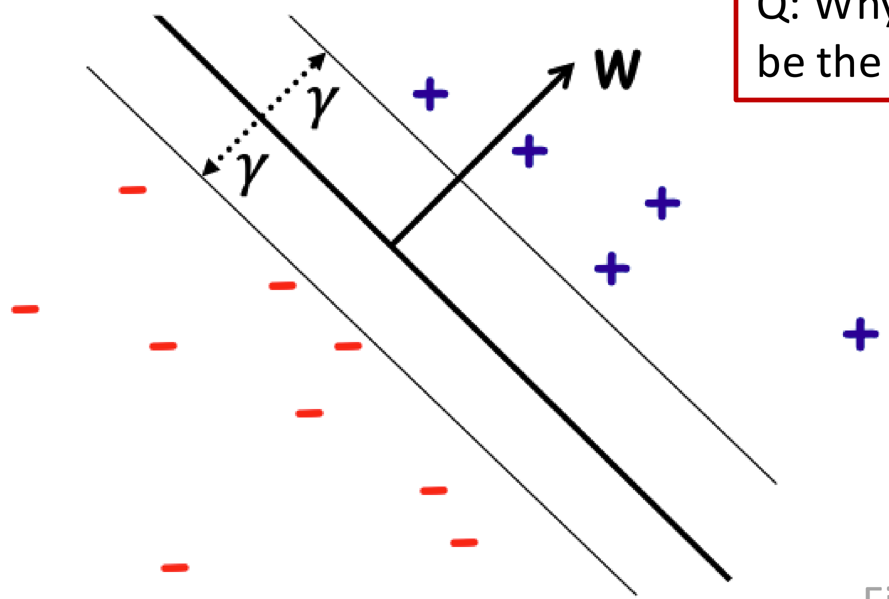


Figure from Balcan (2018)

Perceptron: Mistake Bound

Guarantee: If data has margin γ and all points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

(Normalized margin: multiplying all points by 100*, or dividing all points by* 100*, doesn't change the number of mistakes: algo is invariant to scaling.)*

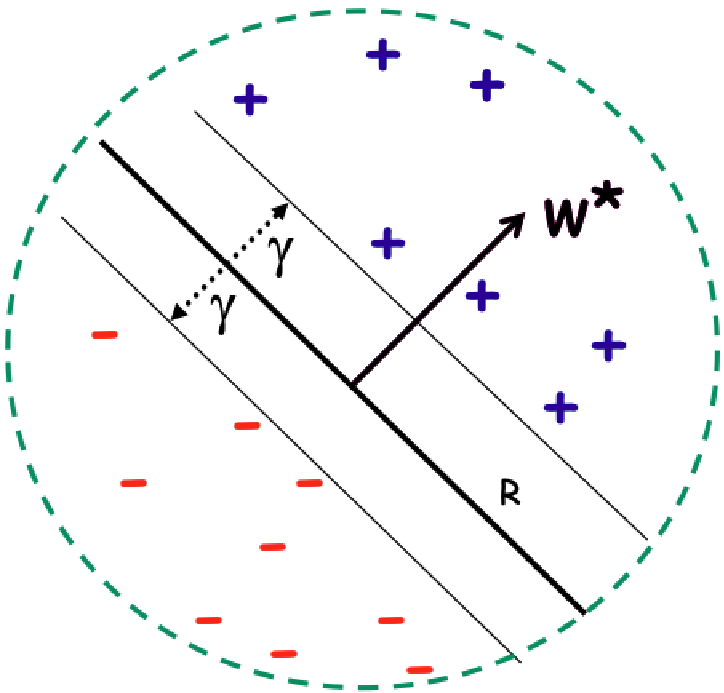


Figure from Balcan (2018)

Perceptron: Mistake Bound

Guarantee: If data has margin γ and all points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

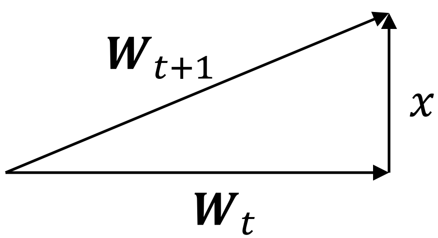
Update rule:

- Mistake on positive: $\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t + x$
- Mistake on negative: $\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t - x$

Proof:

Idea: analyze $\boldsymbol{W}_t^T \boldsymbol{W}^*$ and $||\boldsymbol{W}_t||$, where $\boldsymbol{W}^*$ is the max-margin separator with $||\boldsymbol{W}^*|| = 1$.

- Claim 1: $\boldsymbol{W}_{t+1}^T \boldsymbol{W}^* \geq \boldsymbol{W}_t^T \boldsymbol{W}^* + \gamma$. (because $\boldsymbol{W}_t^T x \geq \gamma$)
- Claim 2: $||\boldsymbol{W}_{t+1}||^2 \leq ||\boldsymbol{W}_t||^2 + R^2$. (by Pythagorean Theorem)



After M mistakes:

$\boldsymbol{W}_{t+1}^T \boldsymbol{W}^* \geq \gamma M$ (by Claim 1)

$||\boldsymbol{W}_{M+1}|| \leq R\sqrt{M}$ (by Claim 2)

$\boldsymbol{W}_{t+1}^T \boldsymbol{W}^* \leq ||\boldsymbol{W}_{M+1}||$ (since $\boldsymbol{W}^*$ is unit length)

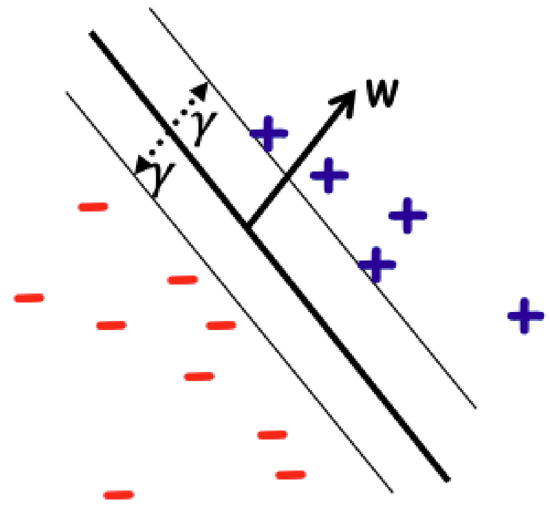
In summary, $\gamma M \leq \boldsymbol{W}_{t+1}^T \boldsymbol{W}^* \leq ||\boldsymbol{W}_{M+1}|| \leq R\sqrt{M}$. So, $M \leq (R/\gamma)^2$.

Margin Important Theme in ML

- If large margin, the number of mistakes Perceptron makes is small (independent on the dimension of the ambient space)

- Large margin can help prevent overfitting.

- **Support Vector Machines (SVMs)** directly optimize for the maximum margin separator:
  - Input: $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$;
  - Find: some $\boldsymbol{W}$ and maximum γ where:
    - $||\boldsymbol{W}|| = 1$
    - For all $i \in \{1, \dots, m\}, y_i \boldsymbol{W}^T \geq \gamma$
  - Output: maximum margin separator over S.



Part II: Gradient Descent (and Beyond)

Taylor Expansion

- How can you minimize a function $l(\boldsymbol{W})$ if you don't know much about it?
- The trick is to assume it is much simpler than it really is.
- This can be done with Taylor's approximation.
- Given a small norm $||s||_2$(i.e., $\boldsymbol{W}+s$ is very close to $\boldsymbol{W}$), we can approximate the function $l(\boldsymbol{W}+s)$ by its first and second derivatives:

$$l(\boldsymbol{W}+s) \approx l(\boldsymbol{W}) + g(\boldsymbol{W}) \cdot s \text{ (Gradient Descent)}$$

$$l(\boldsymbol{W}+s) \approx l(\boldsymbol{W}) + g(\boldsymbol{W}) \cdot s + \frac{1}{2} s^T \cdot H(\boldsymbol{W}) \cdot s \ (Newton's\ Method)$$

Here $g(x) = \nabla l(\boldsymbol{W})$ is the gradient and $H(x) = \nabla^2 l(\boldsymbol{W})$ is the Hessian of l.

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 $\eta > 0$.
- It is straight-forward to prove that in this case $l(\boldsymbol{W} + s) < l(\boldsymbol{W})$:

$$\begin{aligned} &l(\boldsymbol{W} + (-\eta \cdot g(\boldsymbol{W}))) \\ &\approx l(\boldsymbol{W}) - \eta \cdot g(\boldsymbol{W})^T g(\boldsymbol{W}) \\ &< l(\boldsymbol{W}) \end{aligned}$$

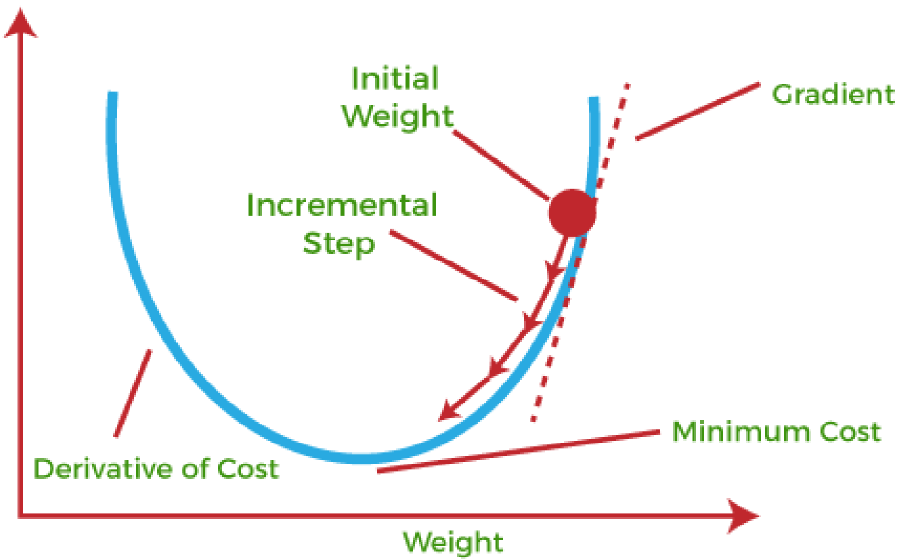
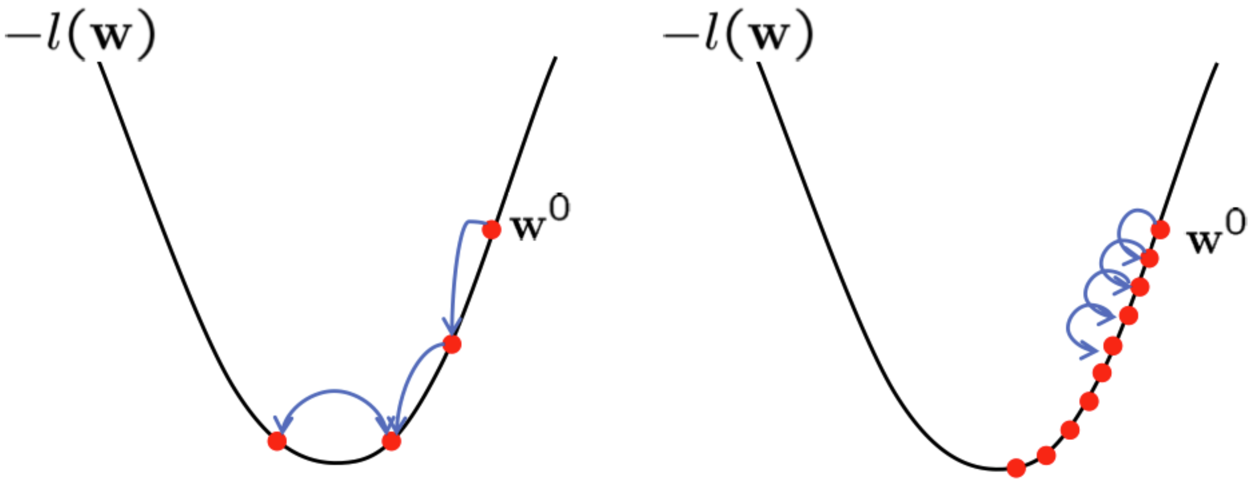


Figure from Kharkar (2023)

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

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

Stochastic GD (SGD)

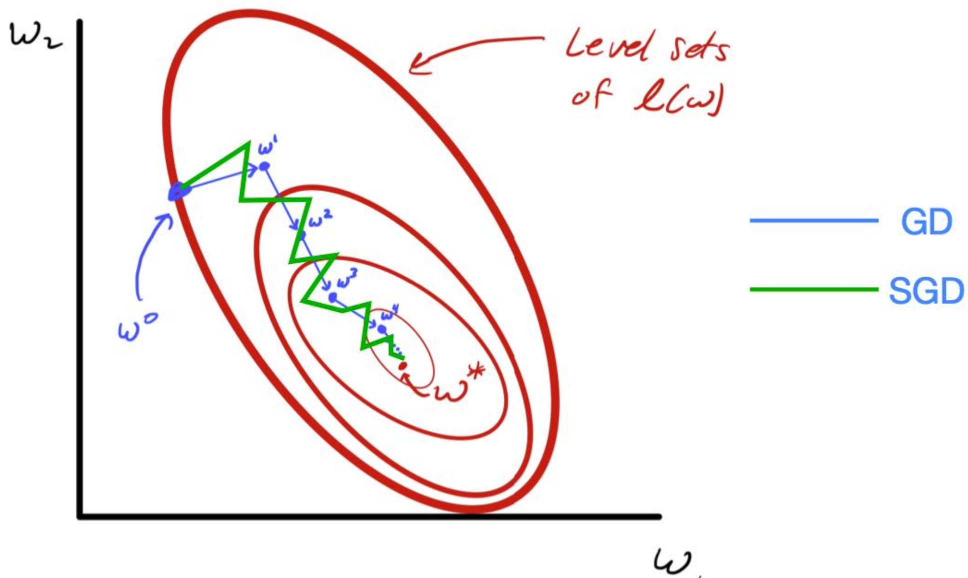
- Goal: minimize $l(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} l(x_i, y_i; \boldsymbol{W})$

- 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): $\boldsymbol{W}^{\boldsymbol{t}+\boldsymbol{1}} = \boldsymbol{W}^{\boldsymbol{t}} - \eta\tilde{g}^t$

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)

Part III: Linear Regression

From Discrete to Continuous Labels

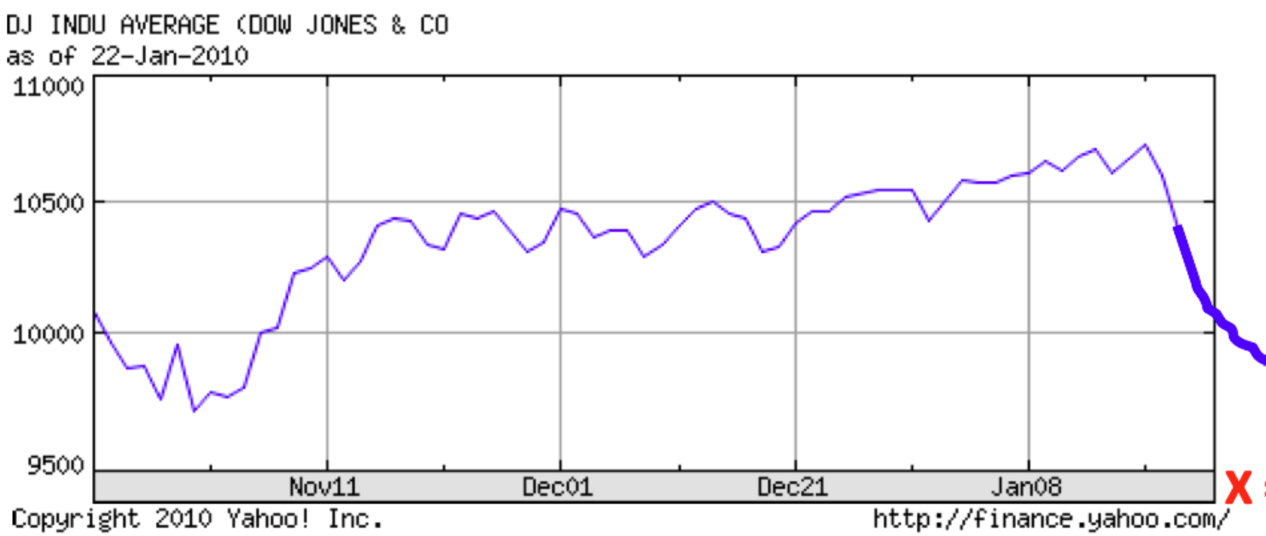
Classification:

Sports
Science
News

X = Document

Y = Topic

Regression:



Y = ?

X = Feb01

Figure from Balcan (2018)

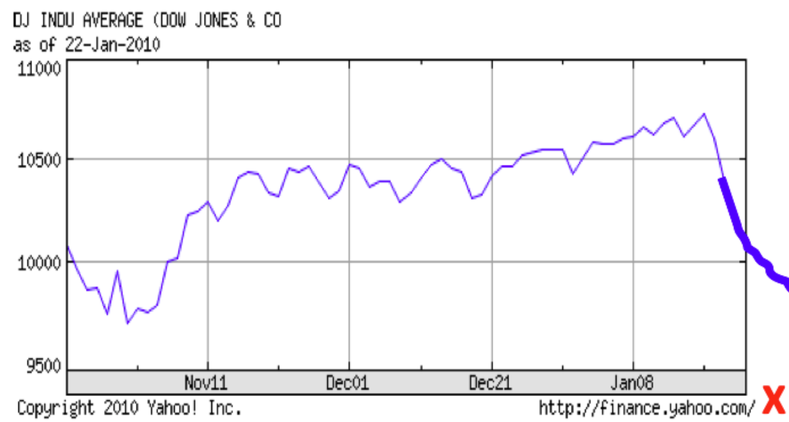
Supervised Learning

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

Sports
Science
News

X = Document

Y = Topic



Y = ?

X = Feb01

Classification:

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

Probability of error

Regression:

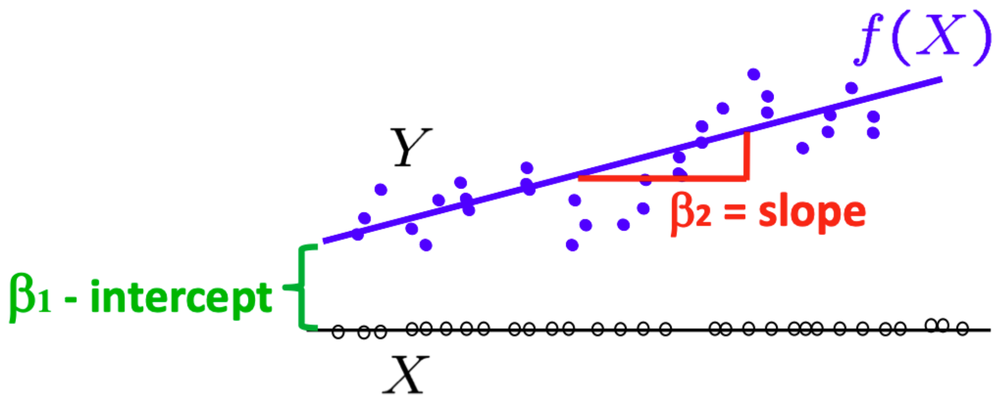
$$\text{err}(f) = E[(f(X) - Y)^2)]$$

Mean Squared Error

Figure from Balcan (2018)

Linear Regression

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



- Multi-variate case:
$$f(X) = f\left(X^{(1)}, \dots, X^{(p)}\right) = \beta_1 X^{(1)} + \beta_2 X^{(2)} + \dots + \beta_p X^{(p)}$$
$$= X\beta, \text{ where } X = \left[X^{(1)} \dots X^{(p)}\right], \beta = [\beta_1 \ \dots \beta_p]^T$$

- 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

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

$\Downarrow$

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

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

Least Squares Estimator

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

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

$$\Downarrow$$

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

- If $\mathbf{X}^T\mathbf{X}$ is invertible,

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

Least Squares Estimator: Verification

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

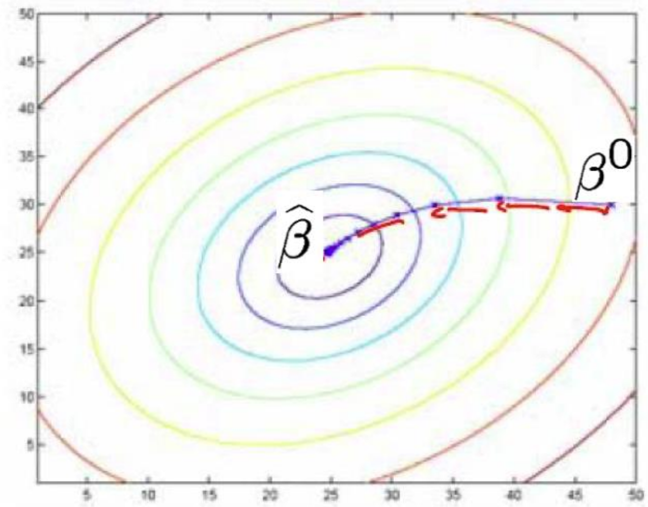
$$\mathbf{X}^T(\hat{f}(\mathbf{X}) - \mathbf{Y}) = \mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\mathbf{Y}^T - \mathbf{X}\mathbf{Y}^T = \mathbf{0}.$$

Revisiting Gradient Descent

- Even when $\mathbf{X}^T\mathbf{X}$ is invertible, might be computationally expensive if $\mathbf{X}$ is huge.

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

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



Q: What about Stochastic GD for linear regression?

Figure from Balcan (2018)

Part IV: Convolutional Neural Networks (CNNs)

Motivation of Convolution

- Suppose we track the location of a spaceship with a laser sensor. The laser sensor provides a single output $u(t)$, which is the position of the spaceship at second t.

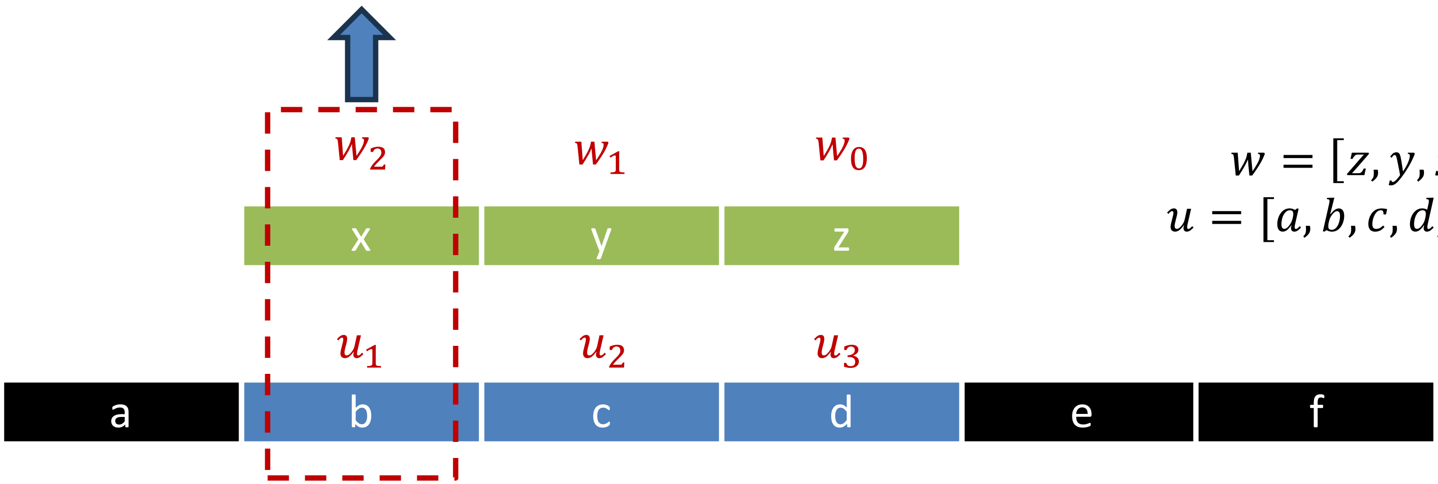
- Suppose sensor is noisy. To obtain a less noisy estimate of the spaceship's position, we average several measurements. More recent measurements are more relevant, so we use a weighted average that gives more weight to recent measurements.

- Use a weighting function $w(a)$, where a is the age of a measurement. If we apply such a weighted average operation at every moment, we obtain a new function s providing a smoothed estimate of the position of the spaceship:

$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a}$$

Illustration 1

$u_1 w_2 = xb$

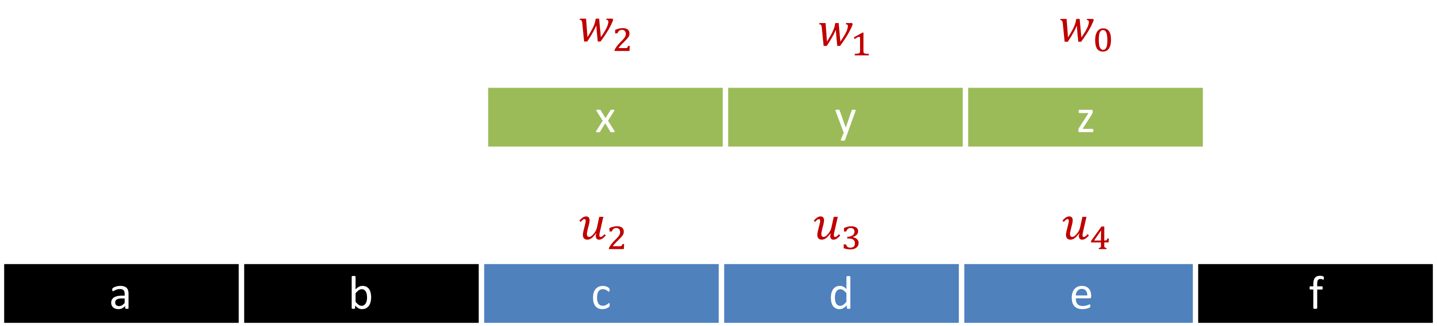


$$w = [z, y, x]$$
$$u = [a, b, c, d, e, f]$$

$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a} \Rightarrow s_3 = \sum_{a=1}^{3} u_a w_{3-a} = u_1 w_2 + u_2 w_1 + u_3 w_0$$
$$= xb + yc + zd$$

Q: What is s_3 here?

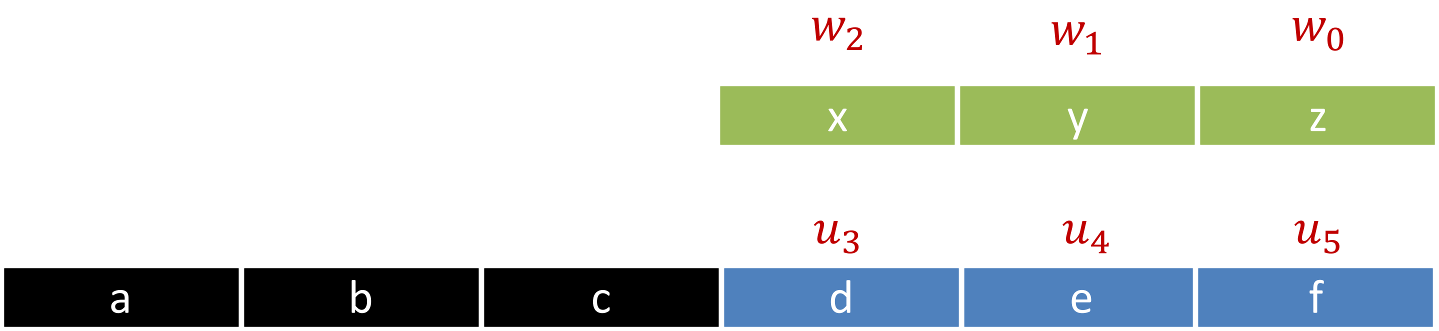
Illustration 1



$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a} \Rightarrow s_4 = \sum_{a=2}^{4} u_a w_{4-a} = u_2 w_2 + u_3 w_1 + u_4 w_0$$

$$= xc + yd + ze$$

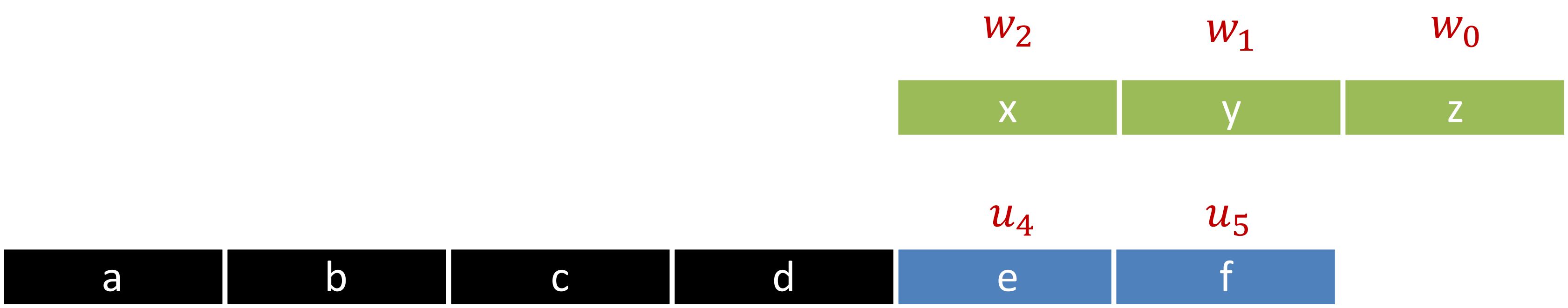
Illustration 1



$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a} \Rightarrow s_5 = \sum_{a=3}^{5} u_a w_{5-a} = u_3 w_2 + u_4 w_1 + u_5 w_0$$

$$= xd + ye + zf$$

Illustration 1



$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a}$$

$$s_6 = \sum_{a=4}^{5} u_a w_{6-a} = u_4 w_2 + u_5 w_1$$
$$= xe + yf$$

Illustration 1 as Matrix Multiplication

$$
\begin{bmatrix}
y & z & & & & \\
x & y & z & & & \\
 & x & y & z & & \\
 & & x & y & z & \\
 & & & x & y & z \\
 & & & & x & y
\end{bmatrix}
\qquad
\begin{bmatrix}
a \\ b \\ c \\ d \\ e \\ f
\end{bmatrix}
$$

Illustration 2: Two-Dimensional Case

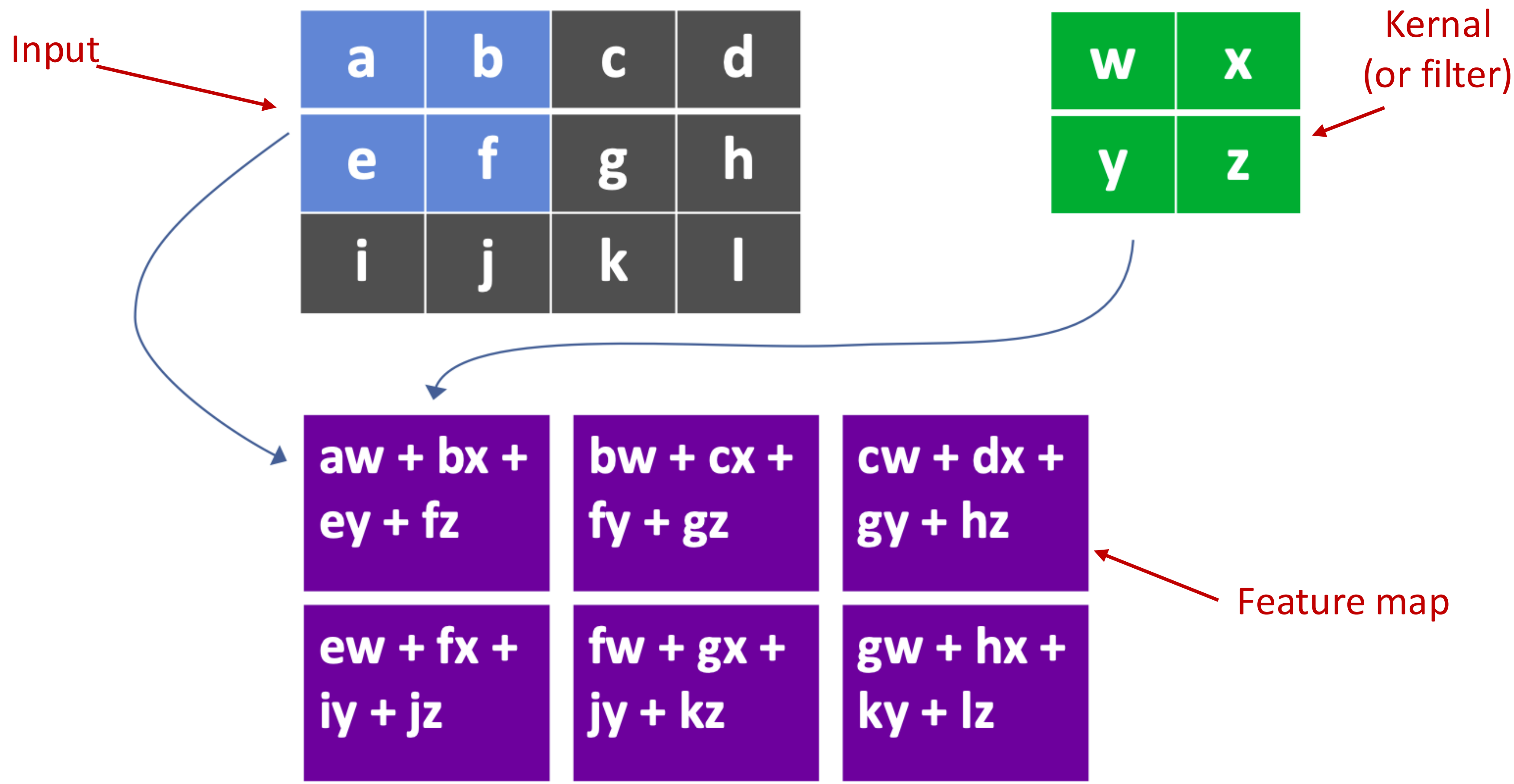
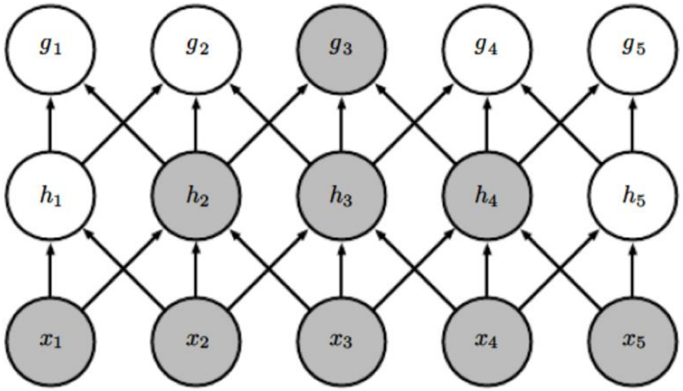


Figure from Balcan (2018)

Advantages of CNNs

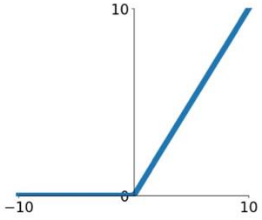


- Sparse interaction
    - Reduces memory requirements
    - Improves statistical efficiency
- Parameter sharing
    - The same kernel are used repeatedly
- Equivariant representations
    - transforming the input = transforming the output
    - Useful when care only about the existence of a pattern, rather than the location

Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville

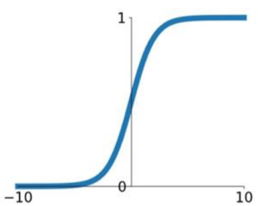
Other Layers: Activation Functions

- Activation functions determine whether a neuron is activated based on its input, effectively deciding whether the input is important for making a prediction.
- ReLU: $\sigma(\mathrm{x}) = \max(0, x)$



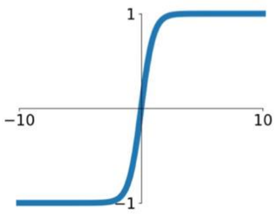
Q: Any other benefit?

- Sigmoid: $\sigma(\mathrm{x}) = \frac{1}{1+e^{-x}}$



Activation Functions introduce non-linearity, enabling the network to learn complex patterns and model intricate relationships within data.

- tanh: $\sigma(\mathrm{x}) = \tanh(\mathrm{x})$



Other Layers: Pooling Layer

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

1	3	0	7
2	4	8	3
5	1	9	0
3	4	1	8

4	

Other Layers: Pooling Layer

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

1	3	0	7
2	4	8	3
5	1	9	0
3	4	1	8

4	8

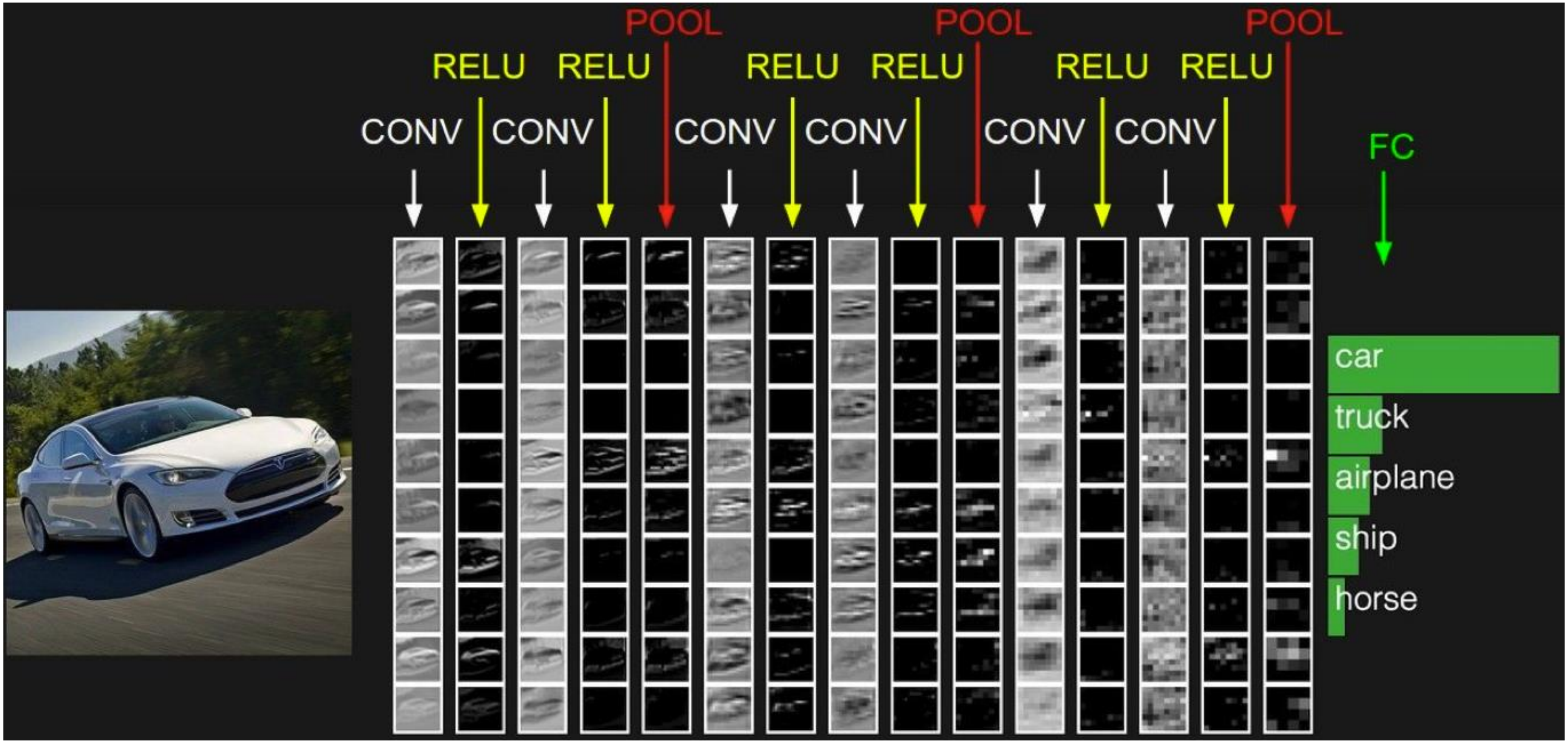
Other Layers: Pooling Layer

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

1	3	0	7
2	4	8	3
5	1	9	0
3	4	1	8

4	8
5	9

A Case of CNNs



Q: How to update the parameters of each layer?

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

EDGE Institute

धन्यवाद