# Classification Methods

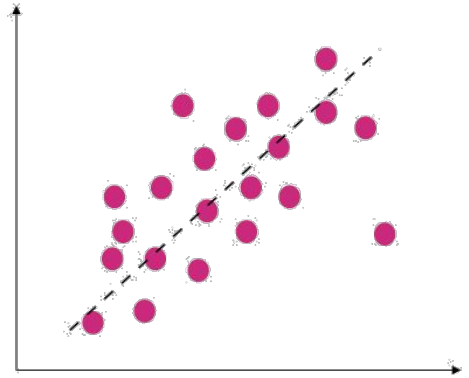
December 13th, 2023

# Logistics - Reminder

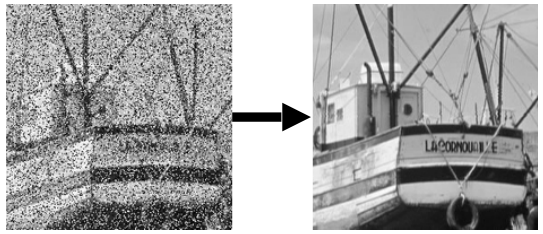
- Everything on course website / Moodle
- 4 credit points
- 2 hrs. lecture, 2 hr. tutorial (Tutorial covers new material)
- 4 homework assignments + Final Project
- All communication through Moodle

# Supervised Learning

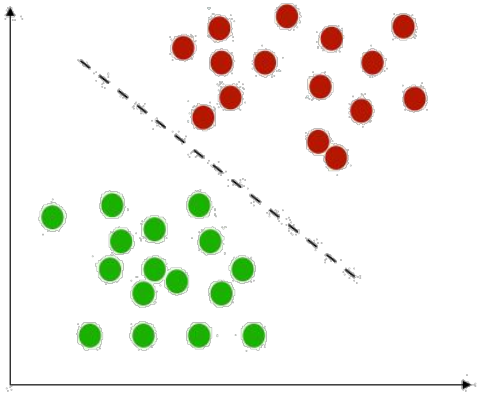
Regression $\{x_i, y_i\}_{i=1}^{m}$ Classification



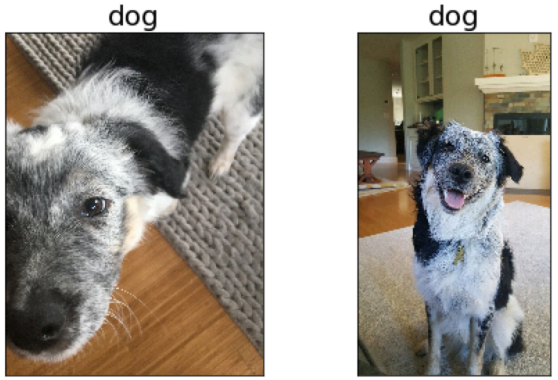
E.g. Image denoising



E.g. Object localization

E.g. Image classification

# Binary Classification

New email

בעקבות המצב:
עד ₪50,000 באופן מיידי לכל מטרה לשכירים/עצמאיים.
⏪באישור משרד האוצר
⏪למחזיקי כרטיס אשראי

לבדיקת זכאות ללא עלות לחצו:
http://bit.ly/2ZrtpIp

**Spam**

**Not Spam**

# Binary Classification

- Samples: $x_i \in \mathscr{R}^d,\ y_i \in \{0, 1\}$
  - $x_i$ - text email features, $y_i = \begin{cases} 1, & spam \\ 0, & not\ spam \end{cases}$
- Training Set: $\left\{x_i,\ y_i\right\}_{i=1}^{m}$
- Hypothesis: $h: \mathscr{R}^d \rightarrow \{0, 1\}$, parameterized by $\theta$

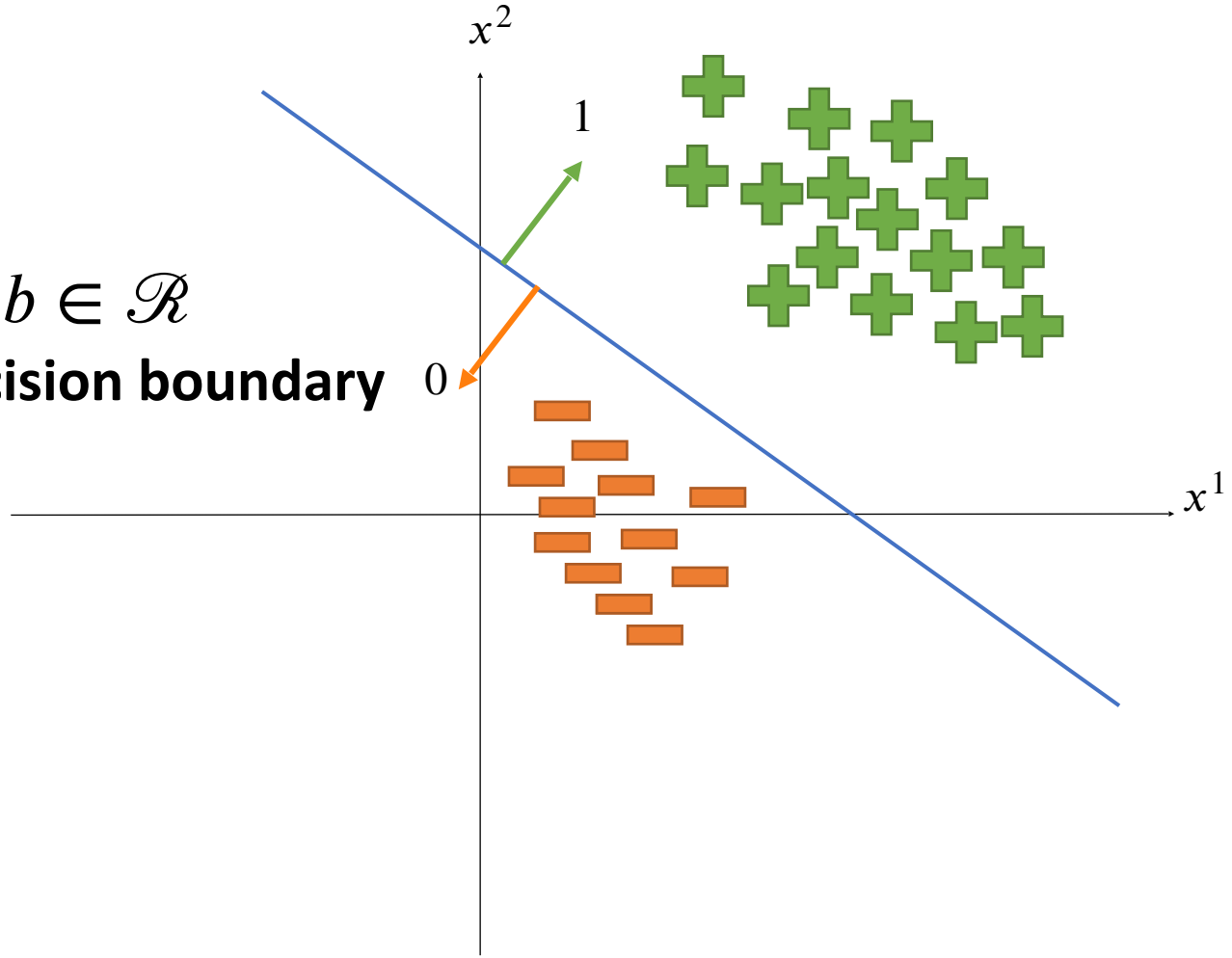
# Linear Classifier

Indicator function

- $\theta = [W, b], h_{W,b}(x_i) = 1(Wx_i + b > 0)$

- Assume:

$$x_i = \begin{pmatrix} x_i^1 \\ x_i^2 \end{pmatrix} \in \mathscr{R}^2,\ \mathrm{W} \in \mathscr{R}^{1\times 2},\ b \in \mathscr{R}$$



# Linear Classifier

$$h_\theta(x_i) = 1(Wx_i + b > 0)$$

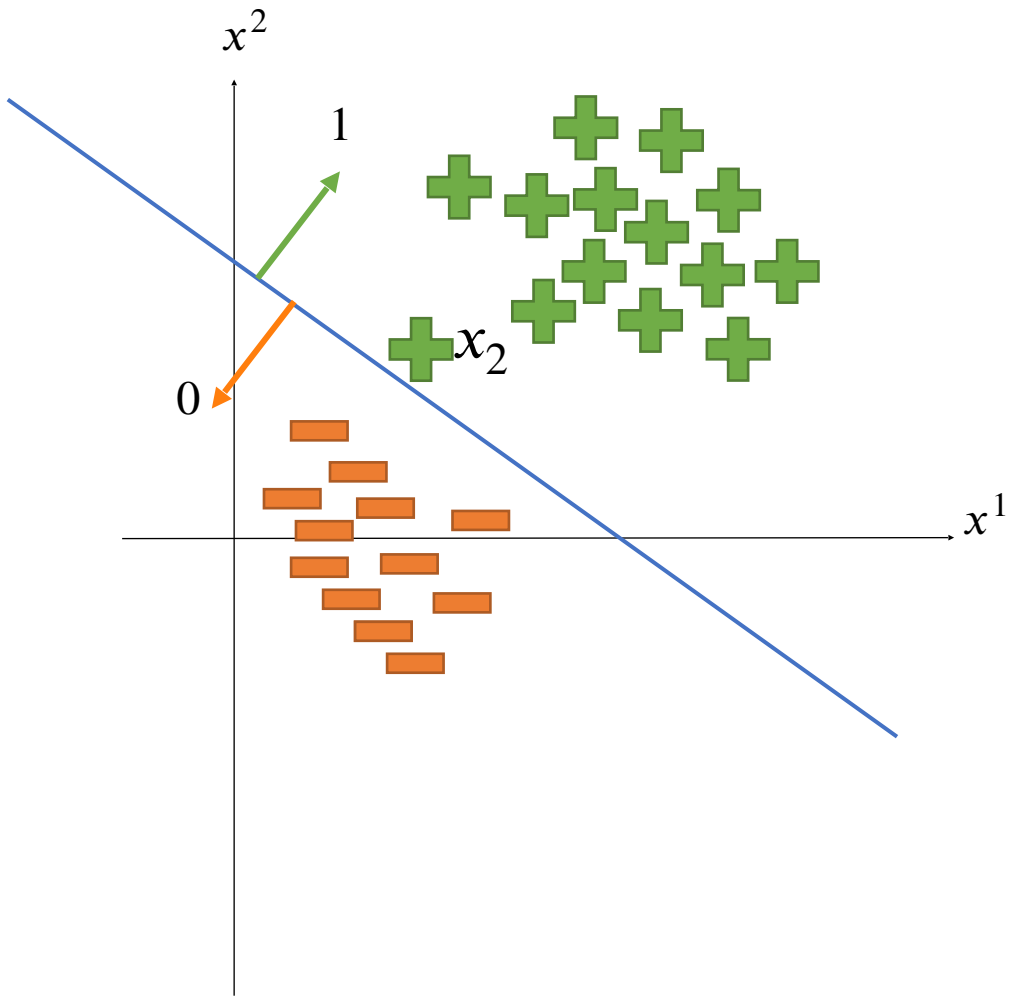
- How to find good $W, b$?

- How to interpret the results?

$Wx_1 + b = \mathbf{1000}$
$\Rightarrow h_{W,b}(x_1) = 1$
**For sure!**

$Wx_2 + b = \mathbf{0.6}$
$\Rightarrow h_{W,b}(x_2) = 1$
**Possibly…**

- Only intuitive, not measureable



# Logistic Regression (Classification)

- Interpretable results
- Linear decision boundary
- Easy to find $W, b$

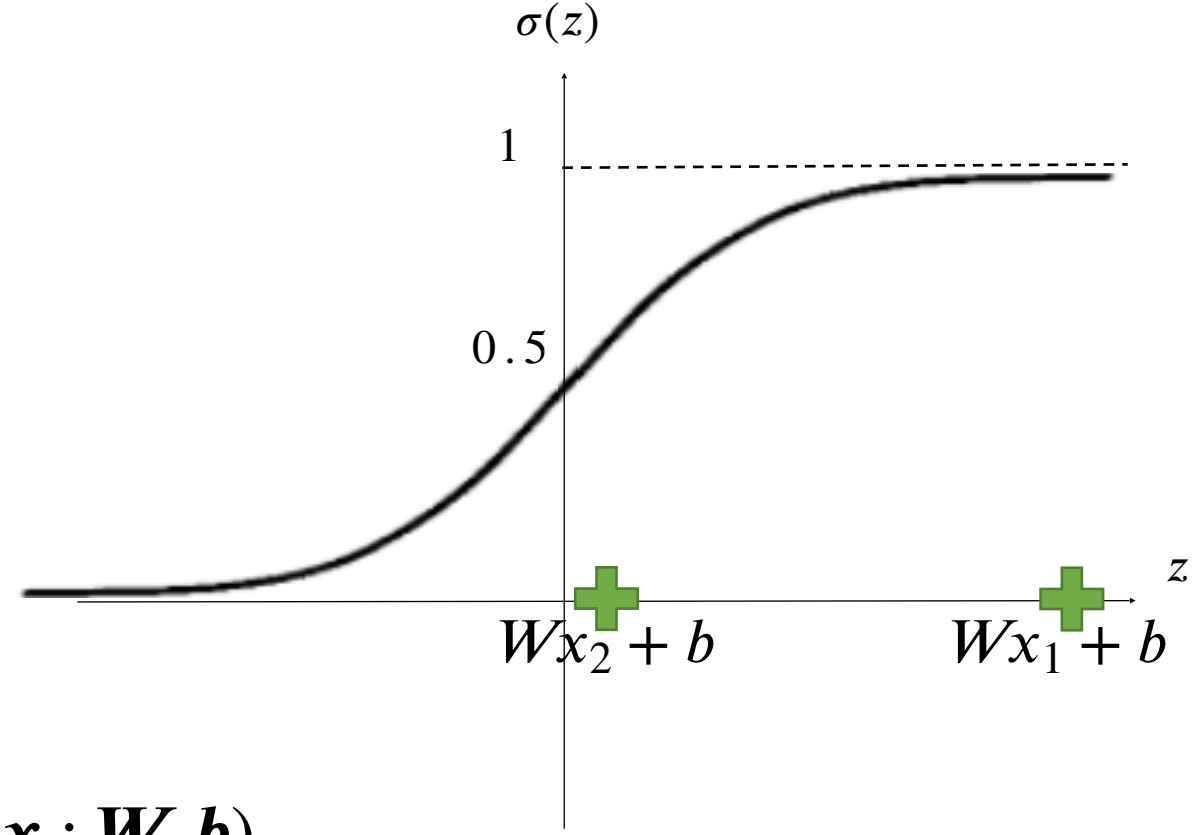
# Sigmoid (Logistic Function)

$Wx + b \rightarrow$ probability

$$z = Wx + b \Rightarrow \sigma(z) = \frac{1}{1 + e^{-z}}$$

Now, our final prediction:

$$h_{W,b}(x_i) = 1\left(\sigma(z) > 0.5\right)$$




Interpretation: $\boldsymbol{\sigma(z) = \mathbf{Pr}(y_i = 1 \,|\, x_i; W, b)}$

Note: $1 - \sigma(z) = \mathbf{Pr}(y_i = 0 \,|\, x_i; W, b)$

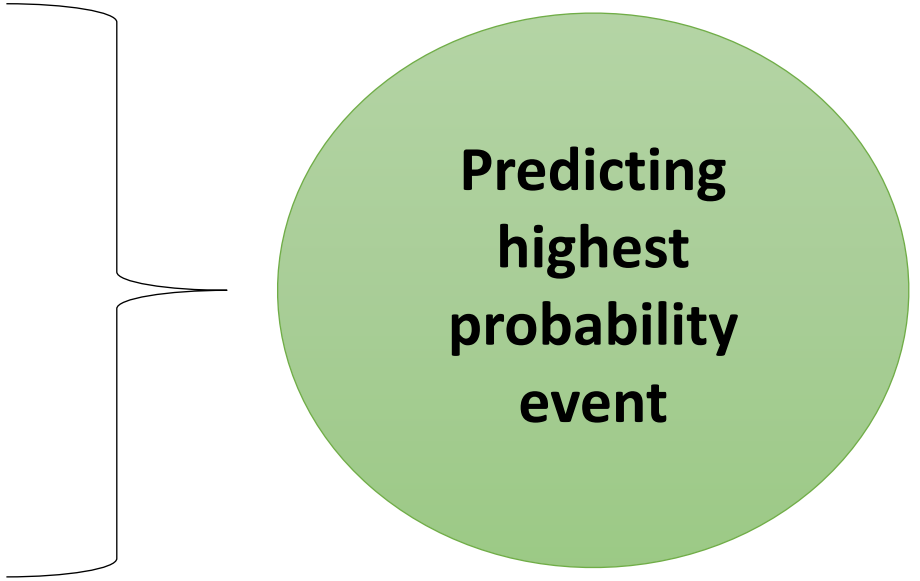
# Sigmoid (Logistic Function)

Final prediction:

$$h_{W,b}(x_i) = 1\left(\sigma(z) > 0.5\right)$$

$$\sigma(z) = \mathbf{Pr}(y_i = 1 \mid x_i; W, b)$$

$$1 - \sigma(z) = \mathbf{Pr}(y_i = 0 \mid x_i; W, b)$$



# Logistic Regression

- Interpretable results
- Linear decision boundary
- Easy to find $W, b$

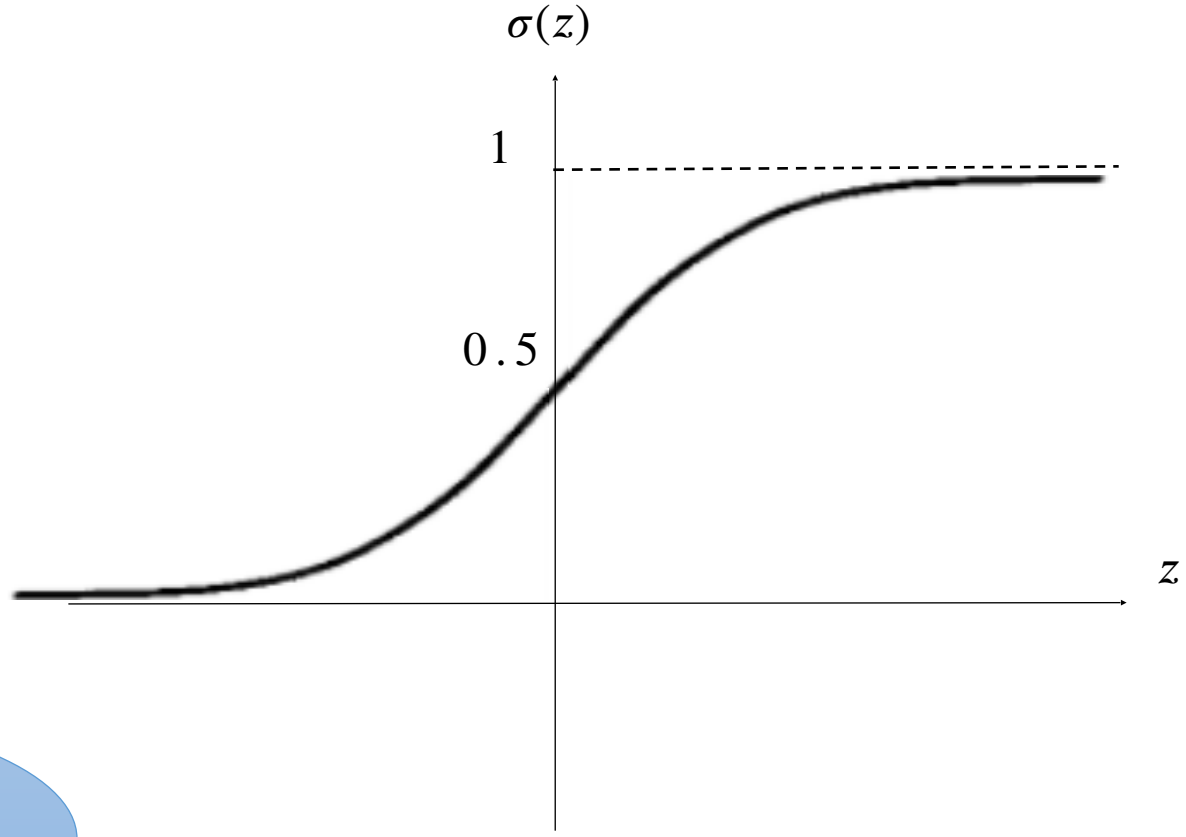
# Sigmoid (Logistic Function)

$$h_{W,b}(x_i) = 1(\sigma(z) > 0.5)$$

$$\sigma(z) > 0.5$$
$$\iff z > 0$$

$\Rightarrow$ Linear decision boundary

$$z = Wx + b$$

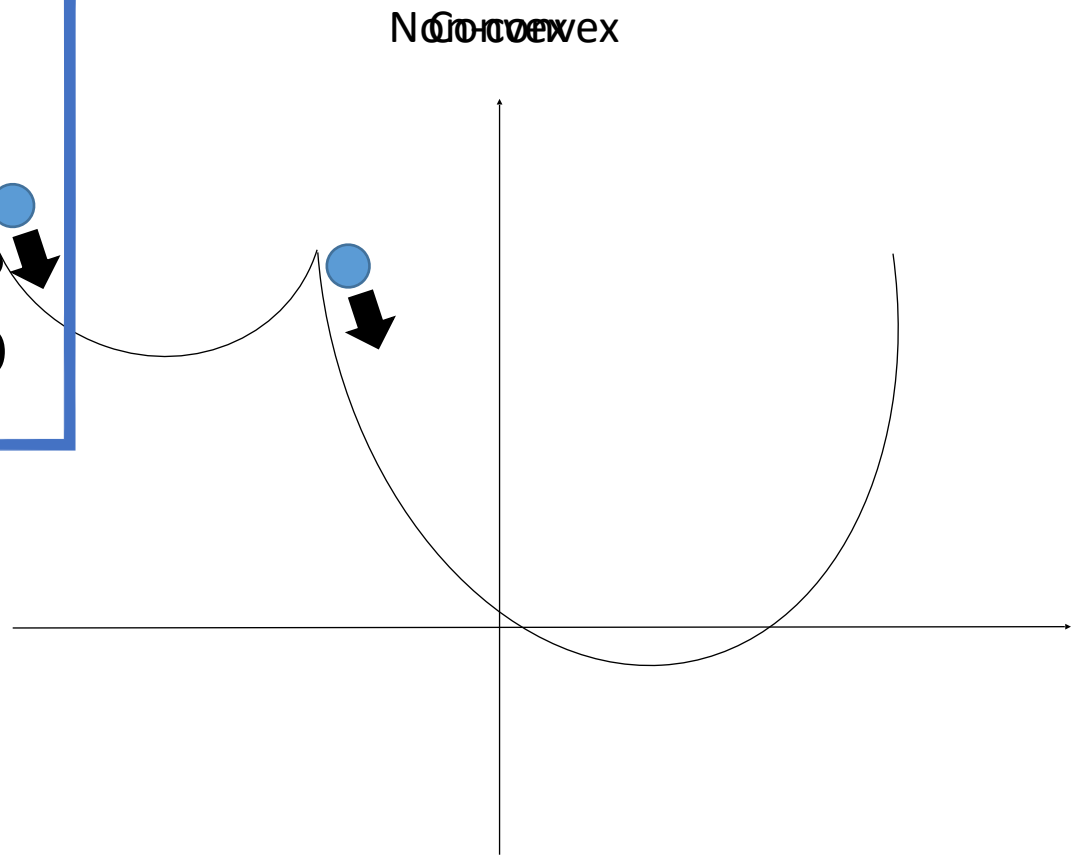


# Logistic Regression

- Interpretable results
- Linear decision boundary
- Easy to find $W, b$

# How to find good $W, b$?

- Define a **convex** cost function $\mathscr{L}$
- $\mathscr{L}$ penalizes errors
  - Wrong prediction, large penalty: $\mathscr{L} \rightarrow \infty$
  - Correct prediction, small penalty: $\mathscr{L} \rightarrow 0$
- Use gradient descent to update $W, b$



# Cross-Entropy Loss

$\Pr[y_i = 1 \mid x_i]$

- **Prediction**: $\hat{y}_i = \sigma(z) = \sigma(Wx + b)$
- **Label (GT)**: $y_i$

$$\mathscr{L}_i = y_i \cdot (-\log \hat{y}_i) + (1 - y_i) \cdot (-\log(1 - \hat{y}_i))$$

- Note that $y_i \in \{0, 1\}, \quad \hat{y}_i \in (0, 1)$
- Is this cost function good?

# Cross-Entropy Loss

$$\mathscr{L}_i = y_i \cdot (-\log \hat{y}_i) + \left(1 - y_i\right) \cdot \left(-\log\left(1 - \hat{y}_i\right)\right)$$

| | $y_i = 0$ | $y_i = 1$ |
|---|---|---|
| $\hat{y}_i \to 0$ | $\mathscr{L} \to 0$ | $\mathscr{L} \to \infty$ |
| $\hat{y}_i \to 1$ | $\mathscr{L} \to \infty$ | $\mathscr{L} \to 0$ |

# Cross-Entropy Loss

- **Prediction**: $\hat{y}_i = \sigma(z) = \sigma(Wx + b)$
- **Label (GT)**: $y_i$

$$\mathscr{L}_i = y_i \cdot (-\log \hat{y}_i) + \left(1 - y_i\right) \cdot \left(-\log\left(1 - \hat{y}_i\right)\right)$$

- Note that $y_i \in \{0, 1\}, \quad \hat{y}_i \in (0, 1)$
- Is this cost function good?
  - Convex w.r.t $W, b$ ✓
  - Penalizes wrong predictions ✓

# Cross-Entropy Loss - Intuition

- Entropy (information theory) – measures “uncertainty”:
  - Uncertain – all events have the same probability
  - Certain – only one event possible
- Formally: $H(X) = -\sum_{x \in X} p(x) \log\, p(x)$
  - Expected number of bits to represent an event
- Cross-entropy: have ( ıt
- How many bits are n(
- Formally: $H(p, q) =$
  - Minimal when $q = p$

# Cross-Entropy Loss - Intuition

$$H(p,q) = -\sum_{x \in X} p(x) \log\, q(x)$$

- How is it related to us?
- Want to learn $W, b$, such that:
  $\sigma(Wx_i + b) = \hat{y}_i = q(y_i = 1 \mid x_i ; W$
- Note: $q(y_i = 0 \mid x_i ; W, b) = 1 - q(y$

# Cross-Entropy Loss - Intuition

$$H(p,q) = -\sum_{x \in X} p(x) \log\, q(x)$$

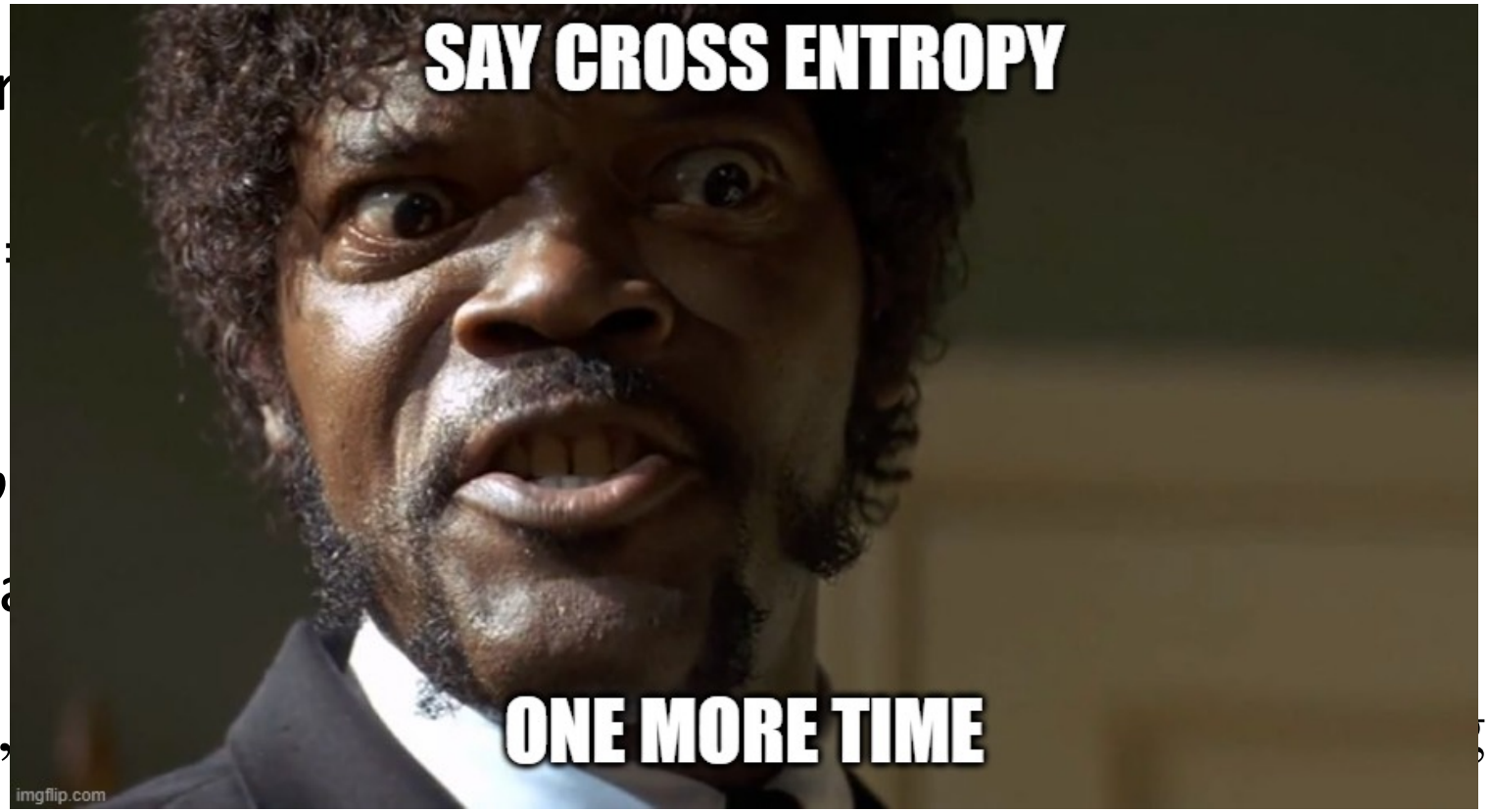
- Dur

$$p(y_i =$$

$$\Rightarrow p$$

- Fina

$$H(p,$$




$$\left(1 - \hat{y}_i\right)$$

$$= y_i \cdot (-\log \hat{y}_i) + (1 - y_i) \cdot \left(-\log\left(1 - \hat{y}_i\right)\right)$$

$$= \mathscr{L}_i$$

# How to find good $W, b$?

- Define a **convex** cost function $\mathscr{L}$

- $\mathscr{L}$ penalizes errors
  - Wrong prediction, large penalty: $\mathscr{L} \to \infty$
  - Correct prediction, small penalty: $\mathscr{L} \to 0$

- Use gradient descent to update $W, b$

# Gradient Descent – Single Sample

$$x_i \in \mathbb{R}^{d\times 1}, y_i \in \{0,1\}$$

Find prediction

$$W \in \mathbb{R}^{1\times d},\ b \in \mathbb{R} \Rightarrow \hat{y}_i \in (0,1))$$

Calculate Loss

$$+ (1-y_i)\cdot\left(-\log\left(1-\hat{y}_i\right)\right)$$

Update weights

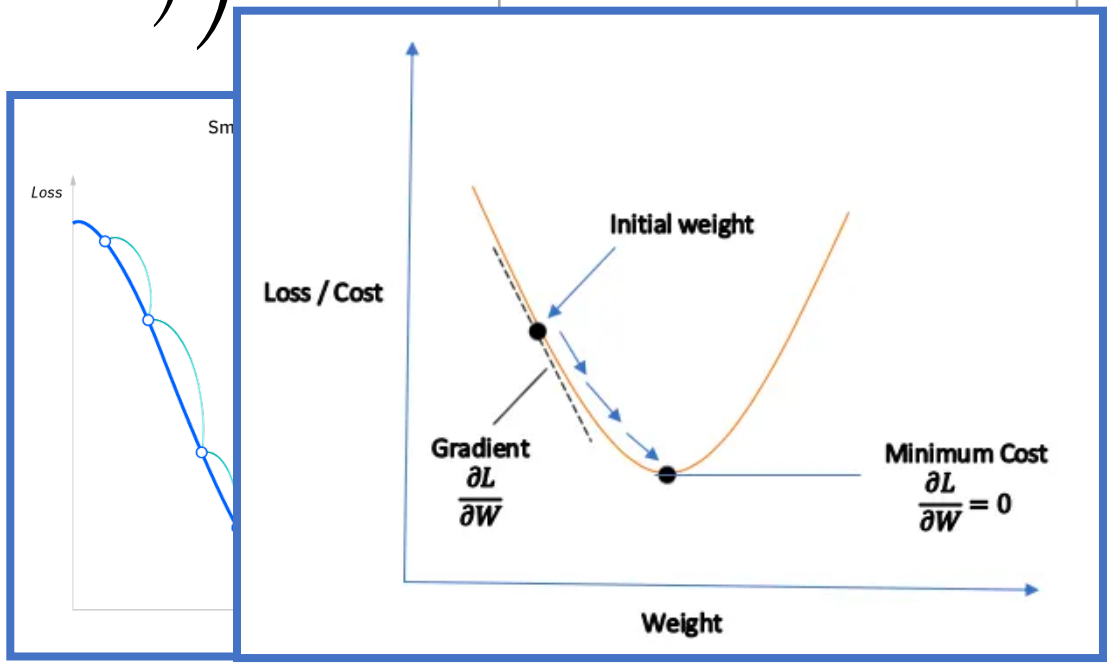
$$W \leftarrow W - \alpha \frac{\partial \mathcal{L}_i}{\partial W} = W - \alpha \cdot x_i^T(\hat{y}_i - y_i)$$

Update bias

$$b \leftarrow b - \alpha \frac{\partial \mathcal{L}_i}{\partial b} = b - \alpha \cdot (\hat{y}_i - y_i)$$

Find prediction

Calculate Loss



# Stochastic Mini-Batch Gradient Descent

Repeat until convergence:

Sample a mini batch $\{x_i, y_i\}_{i=1}^{m}$:

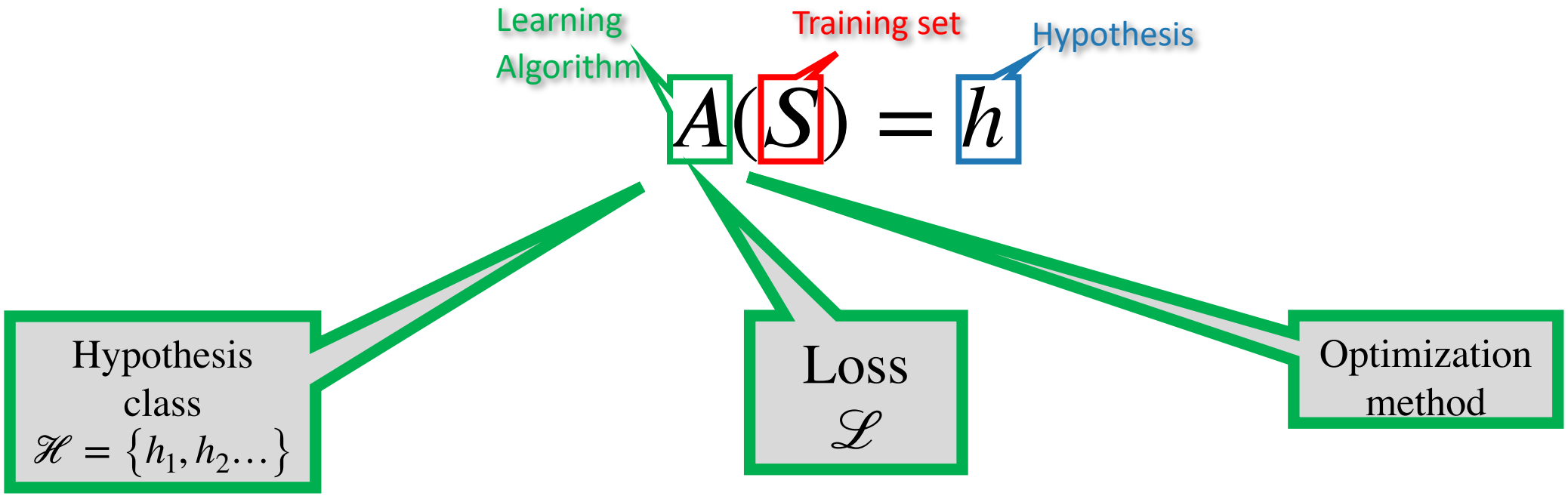
Calculate $\mathscr{L}_i$ for each pair

$$\mathscr{L} = \frac{1}{m}\sum_i \mathscr{L}_i$$

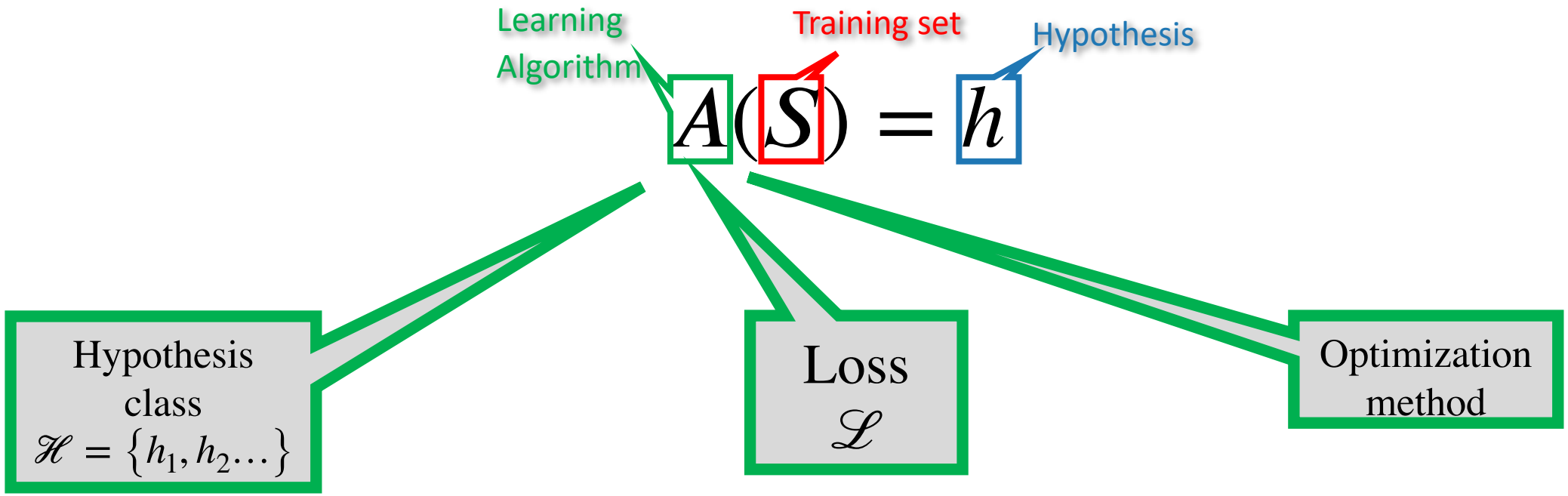
$$W \leftarrow W - \alpha \cdot \frac{\partial \mathscr{L}}{\partial W}$$

$$b \leftarrow b - \alpha \cdot \frac{\partial \mathscr{L}}{\partial b}$$

# Supervised Learning



# Supervised Learning – Logistic Regression




$\mathcal{H} = \{h_{W,b}(x_i) = 1(\sigma(Wx_i + b) > 0.5)\}$

Cross-entropy loss

SGD

# Logistic Regression - Summary

- **Binary** classification
- **Linear** $(Wx + b)$
- **Sigmoid** for interpretable (probability) output
- **Predict** highest probability event
- **Learning:**
  - **Cross-entropy** loss (convex, penalizes mistakes)
  - **Gradient descent** to update weights and bias

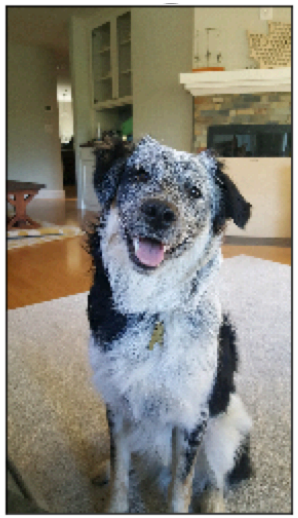
**Probability Regression**

# Multi Class Classification

New email


**Spam** **Maybe** **Not Spam**

# Multi Class Classification



# Multi Class Classification

- $C$ classes
- Samples: $x_i \in \mathscr{R}^d,\ y_i \in [C]$
  - $x_i$ - text email features, $y_i = \begin{cases} 2, & maybe \\ 1, & spam \\ 0, & not\ spam \end{cases}$
- Training Set: $\{x_i,\ y_i\}_{i=1}^{m}$
- Hypothesis: $h: \mathscr{R}^d \rightarrow [\boldsymbol{C}]$, parameterized by $\theta$

# Softmax Classifier

- Generalizing Logistic Regression, **binary → multi-class**
- Begin with **Linear Transformation** $(z = Wx + b)$

| **Logistic Regression** | **Softmax classifier** |
|---|---|
| $W \in \mathbb{R}^{1\times d}, b \in \mathbb{R}$ | $W \in \mathbb{R}^{C\times d}, b \in \mathbb{R}^{C}$ |

- Apply **non-linearity** to transform $z$ to $\hat{y}$ (probabil

| Sigmoid (logistic function) | Softmax |
|---|---|

- Final prediction – pick the event with **highest probability**

| $1(\hat{y} > 0.5)$ | $\text{argmax}(\hat{y})$ |
|---|---|

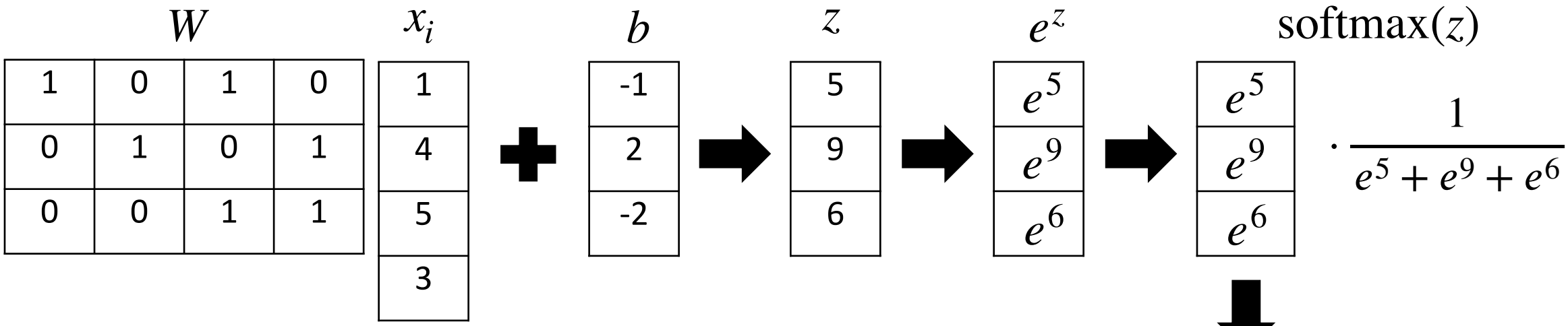
Similar Learning:
1. Cross-entropy loss
2. SGD

$C$ dimensional $z$

$z^j$ = $j$'th class score

# Softmax Function - Example

Given a text $x_i$: **Not Spam (0), Spam(1)** or **Maybe (2)**




$\hat{y}_i$ - **Probability distribution**

$\hat{y}_i$ $[j]$ **- Predicted probability that the class is** $j$

$\hat{y}_i$

| | |
|---|---|
| 0.02 | 2% Not Spam |
| **0.93** | **93% Spam!** |
| 0.05 | 5% Maybe |

# Softmax Function - Formally

Converting $z$ to probability distribution over the classes $C$:

$$z = Wx + b,\ z \in \mathscr{R}^C$$

$$\Rightarrow f(z)^i = \frac{e^{z^i}}{\sum_{j \in [C]} e^{z^j}}$$

$$\Rightarrow \hat{y} = \left[ f(z)^0, \ldots, f(z)^{C-1} \right], \sum_{j \in C} \hat{y}^j = 1$$

# Softmax Classifier

- Generalizing Logistic Regression, **binary → multi-class**

Similar Learning:
1. Cross-entropy loss
2. SGD

- Begin with **Linear Transformation** $(z = Wx + b)$

| **Logistic Regression** | **Softmax classifier** |
|---|---|
| $W \in \mathbb{R}^{1\times d}, b \in \mathbb{R}$ | $W \in \mathbb{R}^{C\times d}, b \in \mathbb{R}^{C}$ |

- Apply **non-linearity** to transform $z$ to $\hat{y}$ (probability)

| Sigmoid (logistic function) | Softmax |
|---|---|

- Final prediction – pick the event with **highest probability**

| $1(\hat{y} > 0.5)$ | $\text{argmax}(\hat{y})$ |
|---|---|

# Cross-Entropy Loss – Softmax Classifier

$$H(p,q) = -\sum_{x \in X} p(x) \log\, q(x)$$

$p(y_i = c \mid x)$ – Ground Truth (known), $\hat{y}_i^{\,c} = q(y_i = c \mid x)$ – Our Prediction

| | |
|---|---|
| 0 | Spam |
| **1** | **Not spam!** |
| 0 | Maybe |

| | |
|---|---|
| 0.02 | 2% Spam |
| **0.93** | **93% Not spam!** |
| 0.05 | 5% Maybe |

$$H(p,q) = -\sum_{c \in \{0,1,2\}} p(y_i = c \mid x_i) \cdot \log\left(\hat{y}_i^{\,c}\right) = -\log\left(\hat{y}_i^{\,1}\right)$$

**Goal: Prediction should be closer to GT.**

# Cross-Entropy Loss – Softmax Classifier

- Generally, with $C$ classes, training example $\{x_i, l\}$

$$\mathscr{L}_i = H(p,q) = -\sum_{c \in C} p(y_i = c \,|\, x_i) \cdot \log\left(\hat{y}_i^c\right)$$

$$= -log \frac{e^{z^l}}{\sum_{j \in [C]} e^{z[j]}}$$

$$p(y_i = l \,|\, x_i) = 1$$

- Wish to make the loss **smaller** (log argument larger):
  - **Increase numerator** → higher confidence of class $l$
  - **Decrease denominator** → higher confidence it's not other classes!

# Softmax Classifier

- Generalizing Logistic Regression, **binary → multi-class**

Similar Learning:
1. Cross-entropy loss
2. SGD

- Begin with **Linear Transformation** $(z = Wx + b)$

| **Logistic Regression** | **Softmax classifier** |
|---|---|
| $W \in \mathbb{R}^{1\times d}, b \in \mathbb{R}$ | $W \in \mathbb{R}^{C\times d}, b \in \mathbb{R}^{C}$ |

- Apply **non-linearity** to transform $z$ to $\hat{y}$ (probability)

| Sigmoid (logistic function) | Softmax |
|---|---|

- Final prediction – pick the event with **highest probability**

| $1(\hat{y} > 0.5)$ | $\text{argmax}(\hat{y})$ |
|---|---|

# Gradient Descent – Logistic Regression

$$x_i \in \mathscr{R}^{d\times 1}, y_i \in \{0,1\}$$

$$\hat{y}_i = \sigma(Wx_i + b)\ (W \in \mathscr{R}^{1\times d},\ b \in \mathscr{R} \Rightarrow \hat{y}_i \in [0,1])$$

Find prediction

$$\mathscr{L}_i = y_i \cdot \left(-\log \hat{y}_i\right) + \left(1 - y_i\right) \cdot \left(-\log\left(1 - \hat{y}_i\right)\right)$$

Calculate Loss

$$W \leftarrow W - \alpha \cdot \frac{\partial \mathscr{L}_i}{\partial W} = W - \alpha \cdot x_i^T(\hat{y}_i - y_i)$$

Update weight

$$b \leftarrow b - \alpha \cdot \frac{\partial \mathscr{L}_i}{\partial b} = b - \alpha \cdot (\hat{y}_i - y_i)$$

Update bias

# Gradient Descent – Softmax Classifier

$$x_i \in \mathscr{R}^{d\times 1}, y_i \in \{\mathbf{0, 1, \ldots, C-1}\}$$

$$\hat{y}_i = \sigma(Wx_i + b)\ (W \in \mathscr{R}^{1\times d},\ b \in \mathscr{R} \Rightarrow \hat{y}_i \in [0, 1])$$

$$\mathscr{L}_i = y_i \cdot \left(-\log\hat{y}_i\right) + \left(1 - y_i\right) \cdot \left(-\log\left(1 - \hat{y}_i\right)\right)$$

$$W \leftarrow W - \alpha \cdot \frac{\partial \mathscr{L}_i}{\partial W} = W - \alpha \cdot x_i^T(\hat{y}_i - y_i)$$

$$b \leftarrow b - \alpha \cdot \frac{\partial \mathscr{L}_i}{\partial b} = b - \alpha \cdot (\hat{y}_i - y_i)$$

Find prediction

Calculate Loss

Update weight

Update bias

# Gradient Descent – Softmax Classifier

$$x_i \in \mathbb{R}^{d\times 1}, y_i \in \{0, 1, \ldots, C-1\}$$

$$\hat{y}_i = f(Wx_i + b) \; (W \in \mathbb{R}^{C\times d} \; b \in \mathbb{R}^{C}, \; \sum_{j\in C} \hat{y}_i[j] = 1)$$

$$\mathcal{L}_i = y_i \cdot \left(-\log \hat{y}_i\right) + \left(1 - y_i\right) \cdot \left(-\log\left(1 - \hat{y}_i\right)\right)$$

$$W \leftarrow W - \alpha \cdot \frac{\partial \mathcal{L}_i}{\partial W} = W - \alpha \cdot x_i^T(\hat{y}_i - y_i)$$

$$b \leftarrow b - \alpha \cdot \frac{\partial \mathcal{L}_i}{\partial b} = b - \alpha \cdot (\hat{y}_i - y_i)$$

Find prediction

Calculate Loss

Update weight

Update bias

# Gradient Descent – Softmax Classifier

$$x_i \in \mathcal{R}^{d\times 1}, y_i \in \{0, 1, \ldots, C-1\}$$

$$\hat{y}_i = f(Wx_i + b)\ (W \in \mathcal{R}^{C\times d}\ b \in \mathcal{R}^{C},\ \sum_{j\in C} \hat{y}_i[j] = 1)$$

$$\mathcal{L}_i = -\sum_{c\in C} p(y_i = c \mid x_i)\ \cdot log(f(z)\,[c])$$

$$W \leftarrow W - \alpha \cdot \frac{\partial \mathcal{L}_i}{\partial W} = W - \alpha \cdot x_i^T(\hat{y}_i - y_i)$$

$$b \leftarrow b - \alpha \cdot \frac{\partial \mathcal{L}_i}{\partial b} = b - \alpha \cdot (\hat{y}_i - y_i)$$

Find prediction

Calculate Loss

Update weight

Update bias

# Gradient Descent – Softmax Classifier

$$x_i \in \mathcal{R}^{d\times 1}, y_i \in \{0, 1, \ldots, C-1\}$$

$$\hat{y}_i = f(Wx_i + b) \; (W \in \mathcal{R}^{C\times d} \; b \in \mathcal{R}^{C} \; \sum \hat{y}_i = 1)$$

$$\mathcal{L}_i = -\sum_{c\in C} p(y_i = c \,|\, x_i) \cdot log(f(z)[c])$$

$$\delta_{y_i}[c] = \begin{cases} 1, & c = y_i \\ 0, & o.w. \end{cases}$$

$$W \leftarrow W - \alpha \cdot \frac{\partial \mathcal{L}_i}{\partial W} = W - \alpha \cdot \left(\hat{y}_i - \delta_{y_i}\right) x_i^T$$

$$b \leftarrow b - \alpha \cdot \frac{\partial \mathcal{L}_i}{\partial b} = b - \alpha \cdot (\hat{y}_i - \delta_{y_i})$$

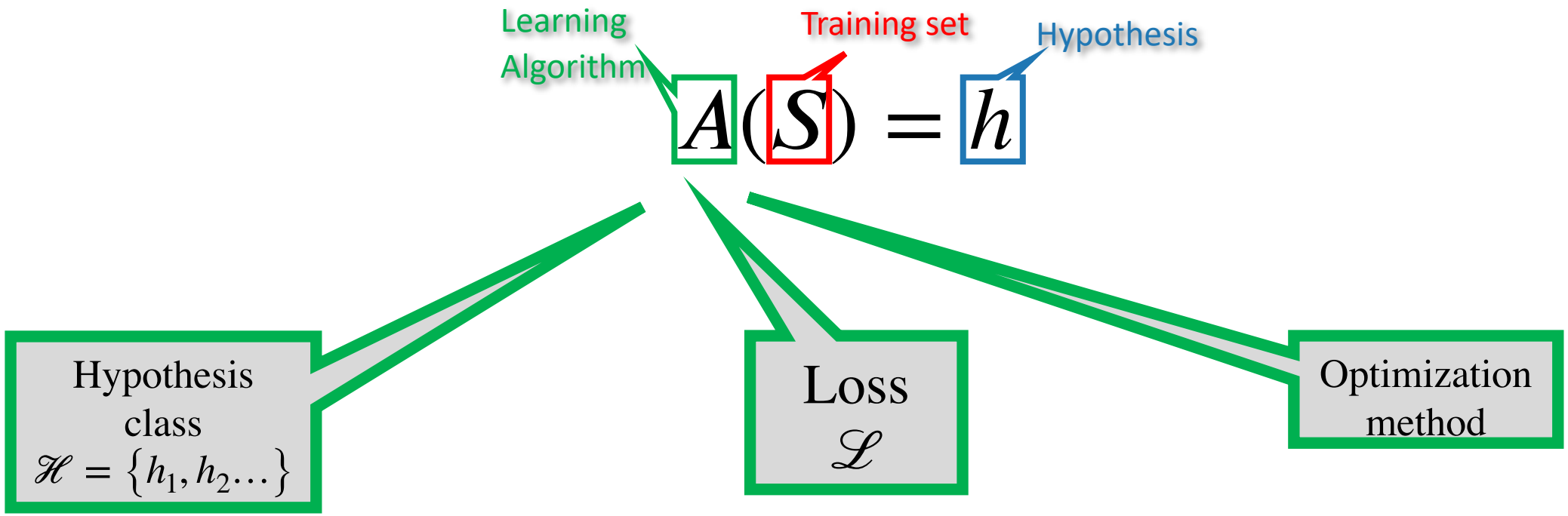
Find prediction

Calculate Loss

Update weight

Update bias

# Supervised Learning – Softmax Classifier




$\mathcal{H} = \{h_{W,b}(x_i) = \text{argmax}(f(Wx_i + b))\}$

Cross-entropy loss

SGD

# Conclusion

- Begin with **Linear Transformation** ($z = Wx + b$)**:**

| **Logistic Regression** | **Softmax classifier** |
|---|---|
| $W \in \mathbb{R}^{1\times d}, b \in \mathbb{R}$ | $W \in \mathbb{R}^{C\times d}, b \in \mathbb{R}^{C}$ |

Similar Learning:
1. Cross-entropy loss
2. SGD

- Apply **non-linearity** to transform $z$ to $\hat{y}$ (probability)

| Sigmoid (logistic function) | Softmax |
|---|---|

- Final prediction – the event with **highest probability**

| $1(\hat{y} > 0.5)$ | $\text{argmax}(\hat{y})$ |
|---|---|

# Pracitcal Considerations

- **Mini Batches**
- **Numerical Stability**

# Stochastic Mini-Batch Gradient Descent

Repeat until convergence:

Sample a mini batch $\{x_i, y_i\}_{i=1}^{m}$:

Calculate $\mathscr{L}_i$ for each pair

$$\mathscr{L} = \frac{1}{m}\sum_i \mathscr{L}_i$$

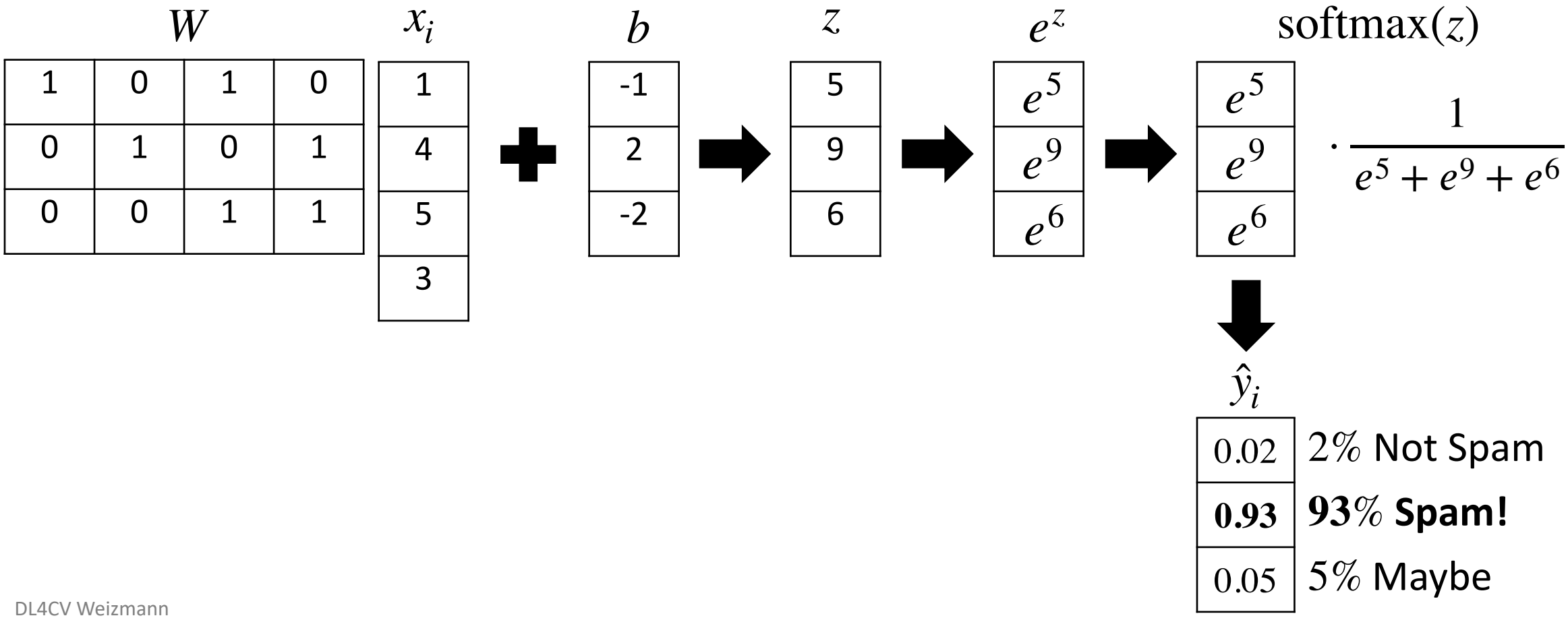
$$W \leftarrow W - \alpha \cdot \frac{\partial \mathscr{L}}{\partial W}$$

$$b \leftarrow b - \alpha \cdot \frac{\partial \mathscr{L}}{\partial b}$$

**Not efficient sequentially**

# Softmax Function - Reminder

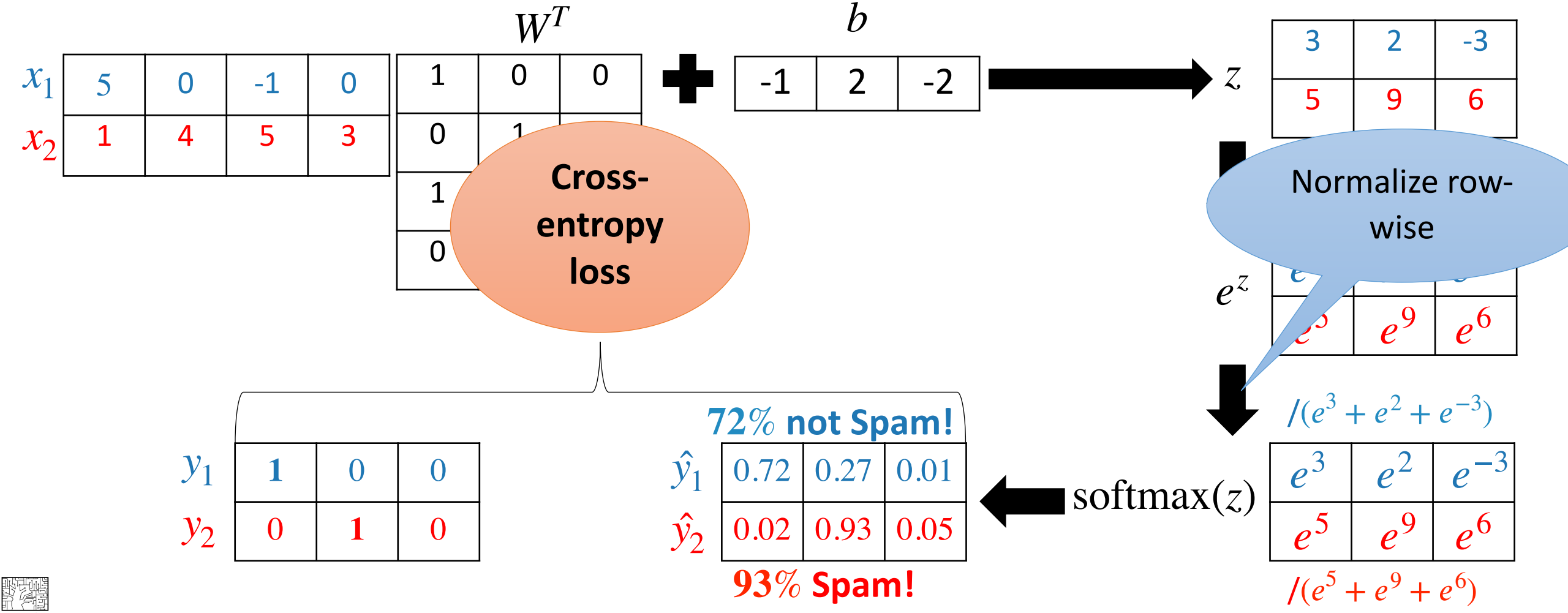
Given a text $x_i$: **Not Spam (0), Spam(1)** or **Maybe (2)**



# Softmax Classifier – Batched Example

spam

Given two pairs: $\{x_1, 0\}, \{x_2, 1\}$



# Mini Batches - Formally

Define $X = \left[x_1, \ldots, x_m\right]^T \in \mathscr{R}^{m \times d}$ - samples matrix

$Y \in \mathscr{R}^{m \times C}$ - GT labels matrix (Each row in $Y$ is one-hot vector)

Reminder: $W \in \mathscr{R}^{C \times d},\ b \in \mathscr{R}^{C}$

$$Z = \underbrace{XW^T}_{\in \mathscr{R}^{m \times c}} + b$$

($b$ is added to each row)

# Mini Batches - Formally

$$Z = \underbrace{XW^T}_{\in \mathscr{R}^{m \times c}} + b\ ,\ Z \in \mathscr{R}^{m \times C}$$

Softmax & Cross-entropy applied row-wise

$$\hat{Y} = f(Z)$$

$$\mathscr{L} = \frac{1}{m} \sum_i \mathscr{L}_i$$

Reminder: Weights update (single-sample):

$$W \leftarrow W - \alpha \cdot \frac{\partial \mathscr{L}_i}{\partial W} = W - \alpha \cdot \left( \hat{y}_i - \delta_{y_i} \right) x_i^T$$

Weights update (mini-batch):

$$W \leftarrow W - \alpha \cdot \frac{\partial \mathscr{L}}{\partial W} = W - \frac{\alpha}{m} \cdot \left( \hat{Y} - Y \right)^T X$$

Similarly derive cross-entropy rule

# Numerical Stability

$$f(z)^i = \frac{e^{z^i}}{\sum_{j\in[C]} e^{z^j}}$$

- Softmax: $e^z$ can be extremely large
- Cross-entropy: $\hat{y}$ possibly close to zero $\Rightarrow \log\left(\hat{y}_i\right) \to -\infty$
- Many possible solutions

$$H(p,q) = -\sum_{c\in[C]} p\left(y_i = c \middle| x_i\right) \cdot \log\left(\hat{y}_i^c\right)$$


QUESTIONS?
imgflip.com