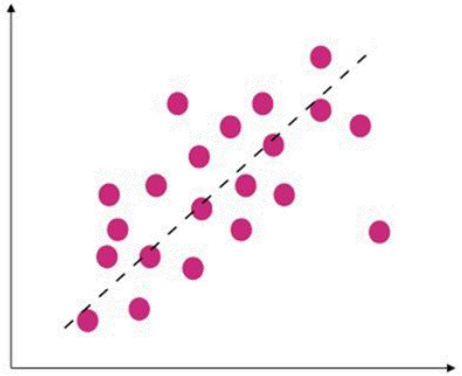
Classification Methods

MCV 97202 | Technion | Spring 2026

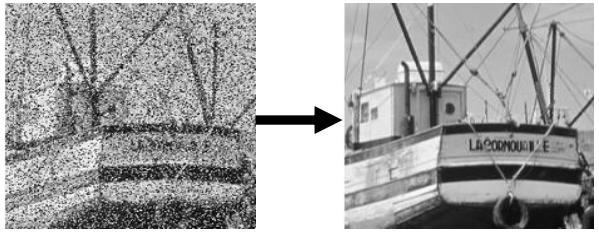
Based on DL4CV course, Weizmann Institute of Science

Supervised Learning

Regression

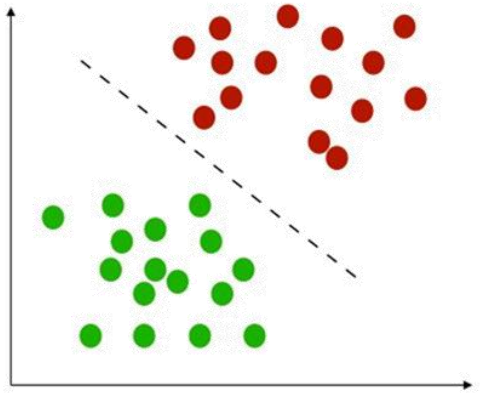


E.g. Image denoising



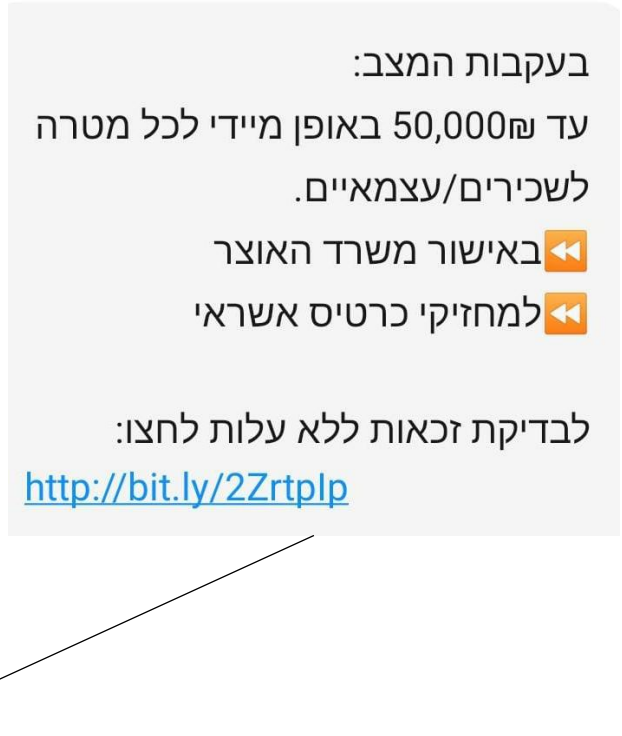
E.g. Object localization

Classification



E.g. Image classification

Binary Classification



Spam **Not Spam**

Binary Classification

- Samples: $x_i \in \mathcal{R}^d$, $\mathrm{y_i} \in \{0,1\}$
  - x_i - text message features, $y_i = \begin{cases} 1, & spam \\ 0, & not\ spam \end{cases}$
- Training Set: $\{x_i, y_i\}_{i=1}^m$
- Hypothesis: $h: \mathcal{R}^d \rightarrow \{0,1\}$, parameterized by θ

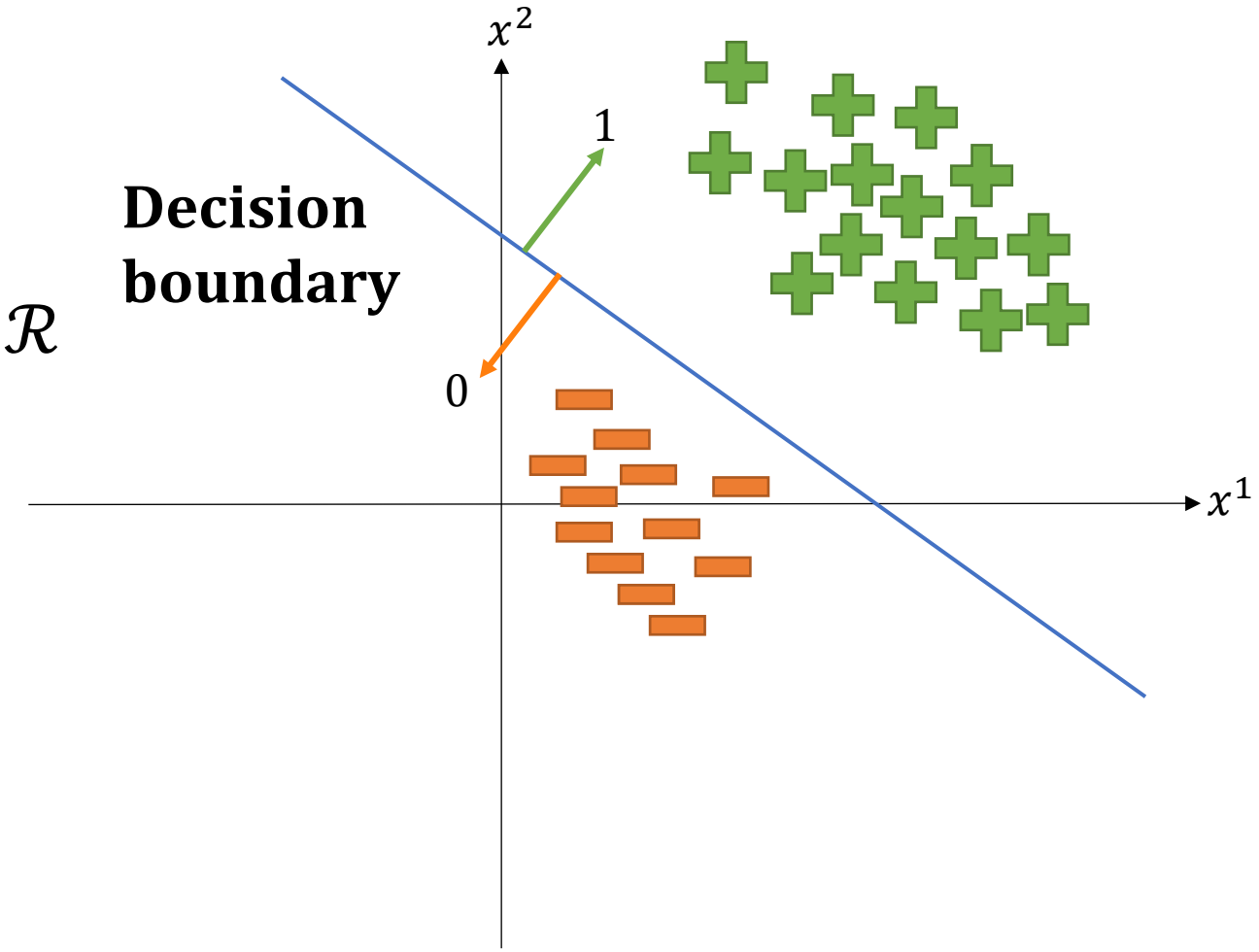
Linear Classifier

Indicator function

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

- Assume:

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



Linear Classifier

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

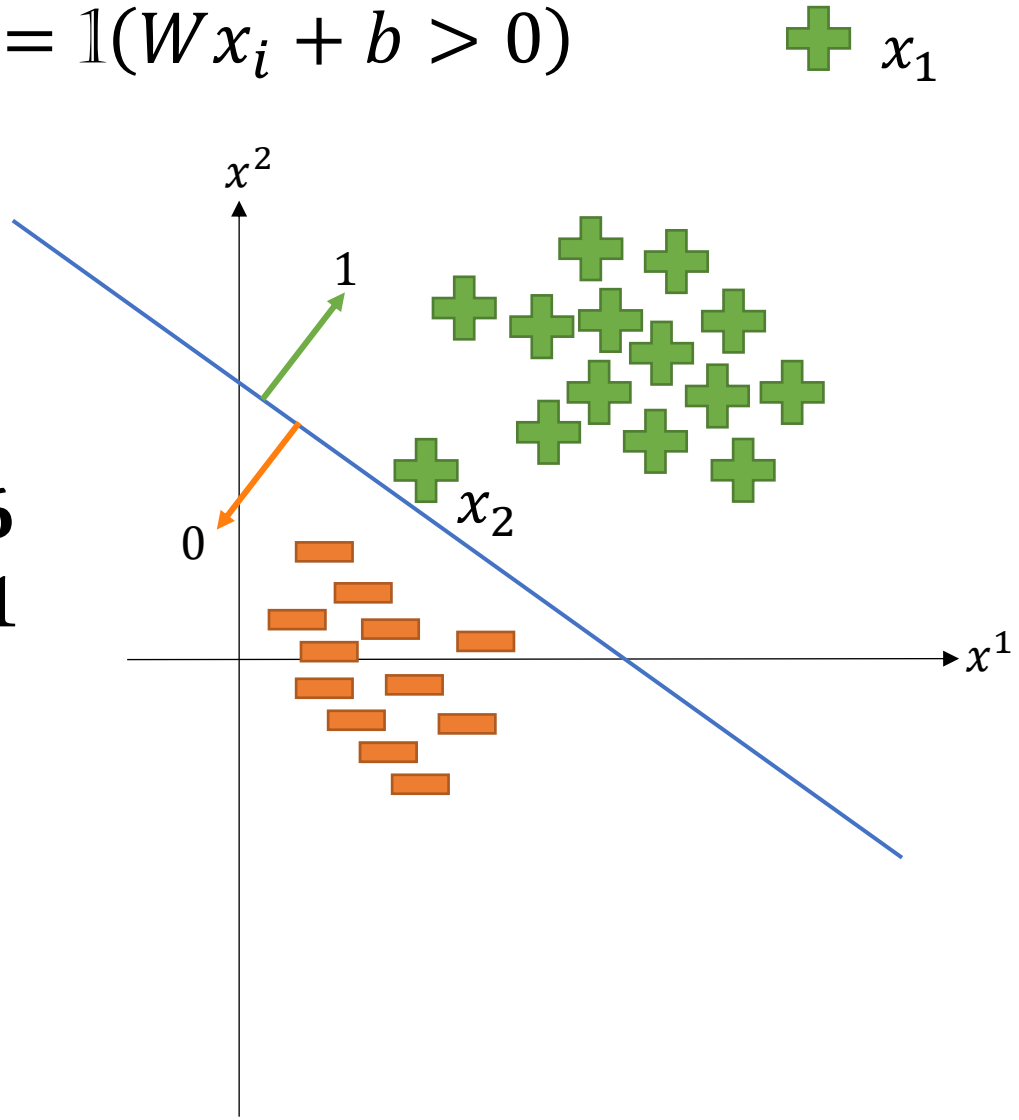
- How to find good W, b?

- 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)

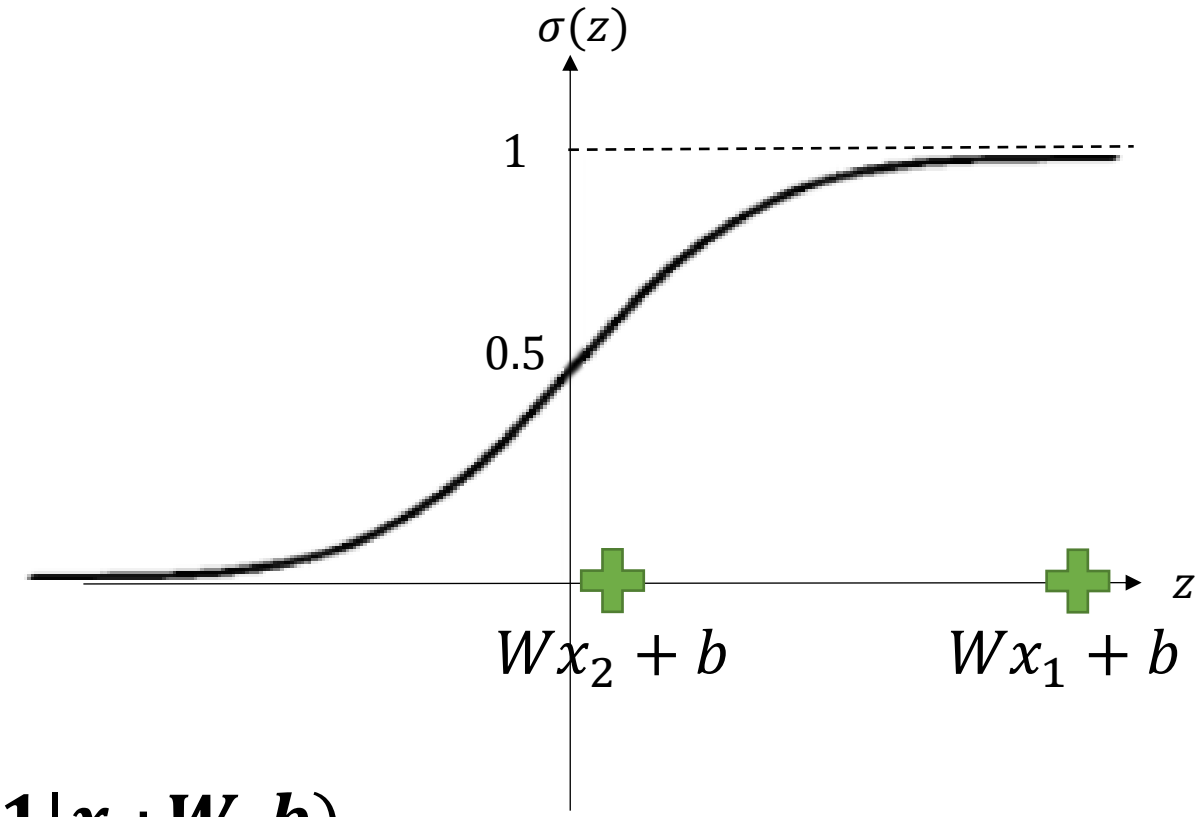
- Interpretable results
- Linear decision boundary
- Easy to find W, b

Sigmoid (Logistic Function)

$Wx + b \rightarrow$ probability

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

Now, our final prediction:
$h_{W,b}(x_i) = \mathbb{1}(\sigma(z) > 0.5)$



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

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

Sigmoid (Logistic Function)

Final prediction:

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

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

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

Predicting highest probability event

Logistic Regression

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

Sigmoid (Logistic Function)

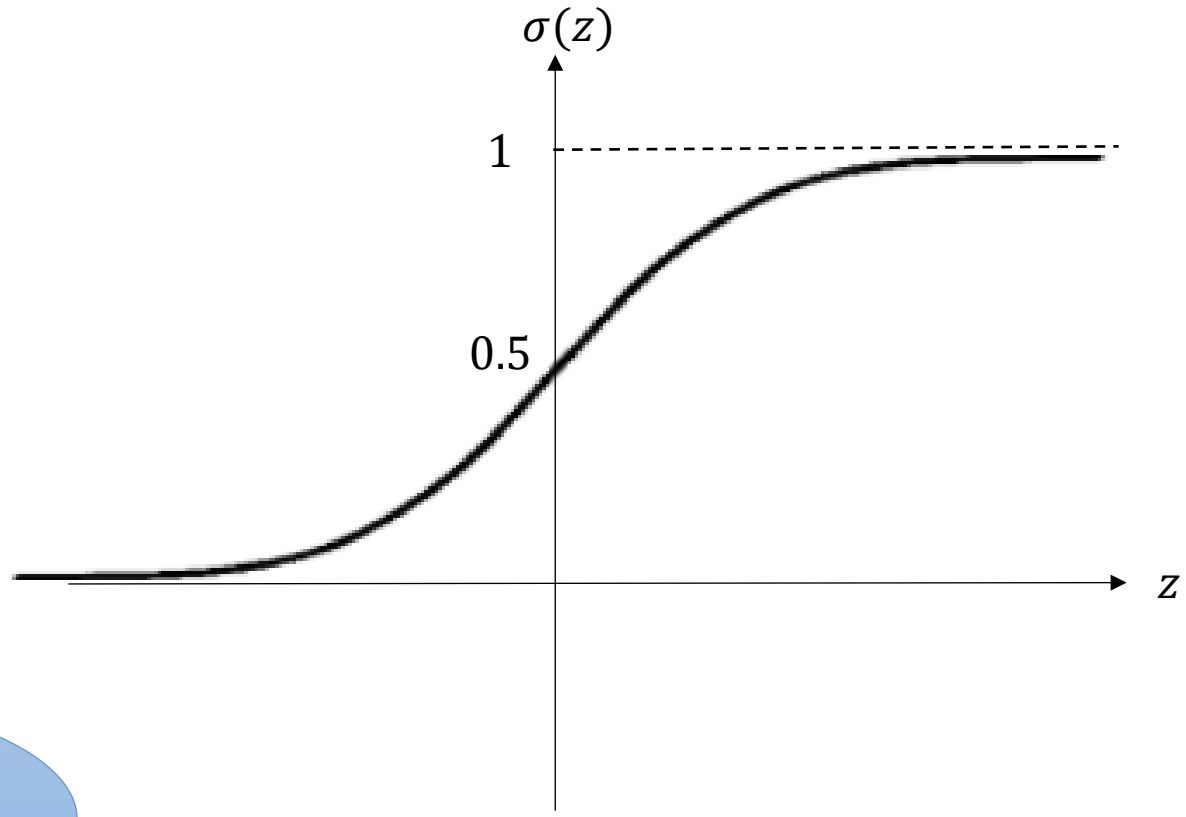
$$h_{W,b}(x_i) = \mathbb{1}(\sigma(z) > 0.5)$$

$$\sigma(z) > 0.5$$

$$\Leftrightarrow z > 0$$

$\Rightarrow$ Linear decision boundary

$$z = Wx + b$$



Logistic Regression

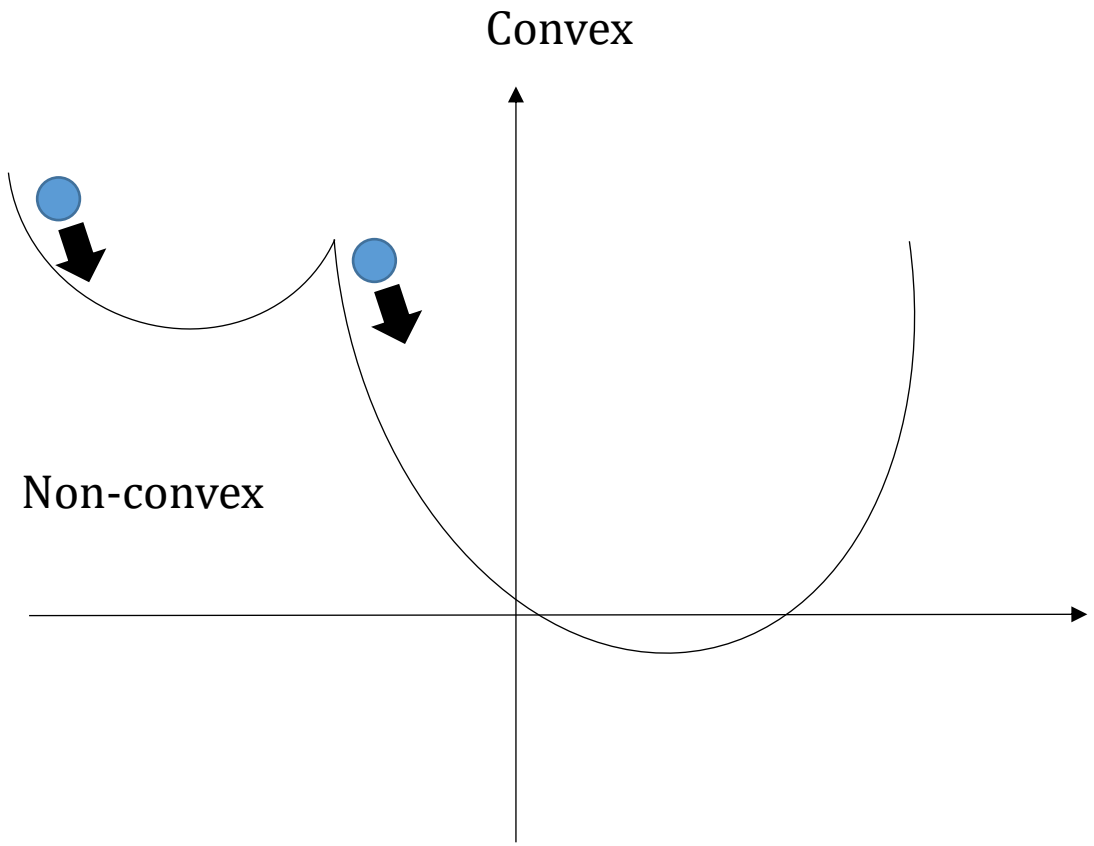
- Interpretable results
- Linear decision boundary
- Easy to find W, b

How to find good W, b?

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

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

- Use gradient descent to update W, b



Cross-Entropy Loss

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

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

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

	$\boldsymbol{y_i = 0}$	$\boldsymbol{y_i = 1}$
$\hat{\mathbf{y}}_\mathbf{i} \to \mathbf{0}$	$\mathcal{L}_\mathrm{i} \to 0$	$\mathcal{L}_\mathrm{i} \to \infty$
$\hat{\mathbf{y}}_\mathbf{i} \to \mathbf{1}$	$\mathcal{L}_\mathrm{i} \to \infty$	$\mathcal{L}_\mathrm{i} \to 0$

Visual Example

Finaly:

- Penalizes wrong predictions ✓
- Convex w.r.t W, b ✓

Cross-Entropy Loss - Intuition

Information as "Surprise": $\log \frac{1}{p(x)} = -\log(p)$

- Low probability → High surprise
- High probability → Low surprise

Entropy = Average Surprise

$$H(p) = E_{x\sim p}[-\log p\,(x)] = -\sum p(x)\log p\,(x)$$

Average surprise when you know the **true probabilities p**

Cross-Entropy Loss - Intuition

Cross-Entropy $H(p, q)$:

Average surprise when you **DONT** know the **true probabilities p.**

- In reality $p(x)$ but we have only $q(x)$ - an approximation of $p(x)$:

$$H(p, q) = E_{x \sim p}[-\log q\,(x)] = -\sum p(x) \log q\,(x)$$

(Average surprise when reality is p, but you believe q)

- Minimal when $p = q$.

True label p	Prediction q	H(p,q)
[1, 0]	[0.9, 0.1]	0.105 (low — good model)
[1, 0]	[0.1, 0.9]	2.303 (high — bad model)

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:

$$\sigma(W x_i + b) = \hat{y_i} = q(y_i = 1 | x_i ; W, b) \approx p(y_i = 1 | x_i)$$

- Note: $q(y_i = 0 | x_i ; W, b) = 1 - q(y_i = 1 | x_i ; W, b) = 1 - \hat{y_i}$

Cross-Entropy Loss - Intuition

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

- During training - **known label**:

$$p(y_i = 1|x_i) = \begin{cases} 1, y_i = 1 \\ 0, y_i = 0 \end{cases} = \begin{cases} y_i, y_i = 1 \\ y_i, y_i = 0 \end{cases} = y_i$$
$$\Rightarrow p(y_i = 0|x_i) = 1 - y_i$$

- Finally:

$$H(p,q) = -p(y_i = 1|x_i) \cdot \log(\widehat{y_i}) - p(y_i = 0|x_i) \cdot \log(1 - \widehat{y_i})$$
$$= y_i \cdot (-\log \widehat{y_i}) + (1 - y_i) \cdot (-\log(1 - \widehat{y_i}))$$
$$= \mathcal{L}_i$$

How to find good W, b?

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

Gradient Descent – Single Sample

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

Find prediction	$\hat{y}_i = \sigma(Wx_i + b)\ (W \in \mathcal{R}^{1\times d}, b \in \mathcal{R} \Rightarrow \hat{y}_i \in [0,1])$
Calculate Loss	$\mathcal{L}_i = y_i \cdot (-\log \hat{y}_i) + (1 - y_i) \cdot (-\log(1 - \hat{y}_i))$
Update weights	$W \leftarrow W - \alpha \cdot \frac{\partial \mathcal{L}_i}{\partial W} = W - \alpha \cdot x_i^T (\hat{y}_i - y_i)$
Update bias	$b \leftarrow b - \alpha \cdot \frac{\partial \mathcal{L}_i}{\partial b} = b - \alpha \cdot (\hat{y}_i - y_i)$

Stochastic Mini-Batch Gradient Descent

Repeat until convergence:

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

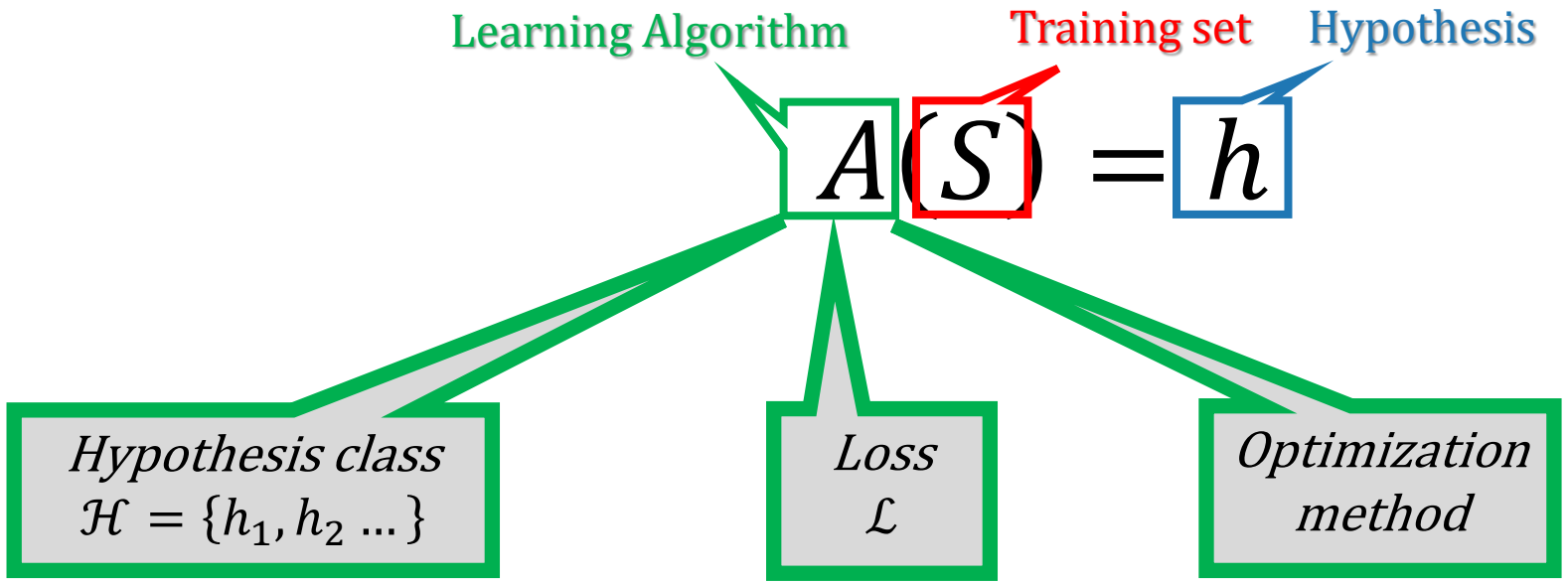
Calculate $\mathcal{L}_i$ for each pair

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

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

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

Supervised Learning – Logistic Regression

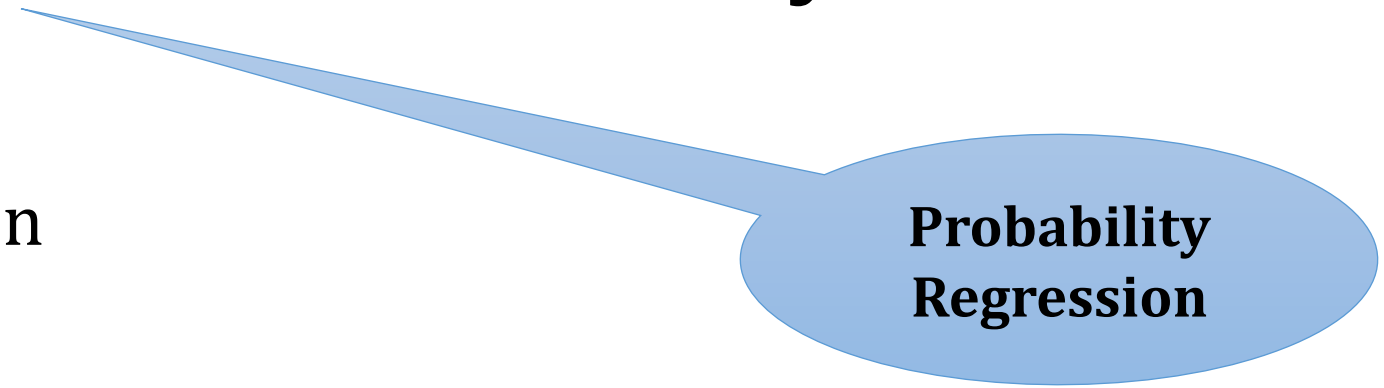


$$\mathcal{H} = \left\{h_{W,b}(x_i) = \mathbb{1}(\sigma(Wx_i + b) > 0.5)\right\}$$

Cross-entropy loss

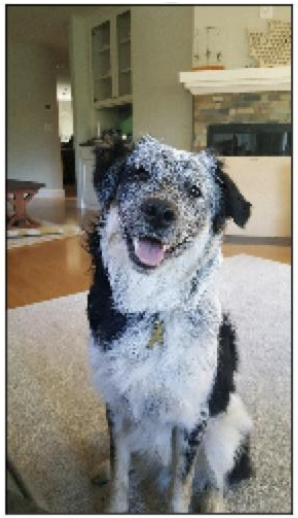
SGD

Logistic Regression - Summary

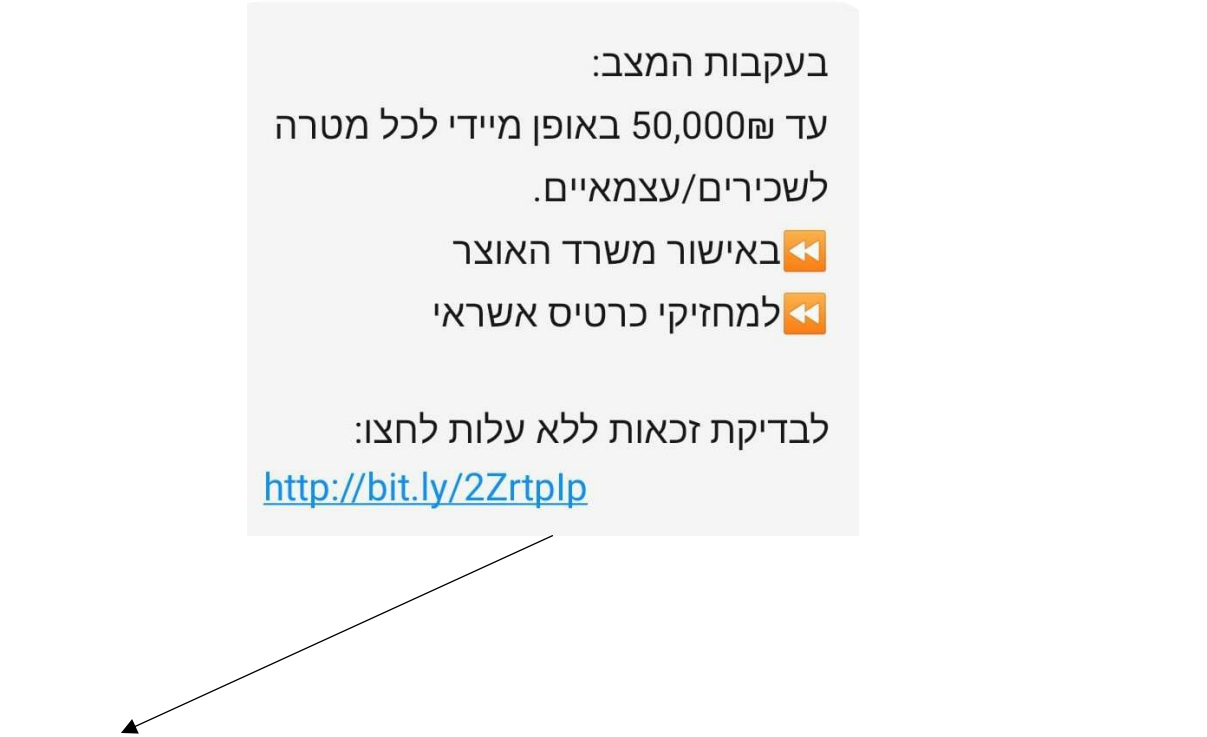


- **Binary** classification
- **Linear** $(Wx + b)$
- **Sigmoid** for interpretable (probability) output
- **Predict** highest probability event
- **Learning:**
  - **Cross-entropy** loss (convex, penalizes mistakes)
  - **Gradient descent** to update weights and bias

Multi Class Classification



Multi Class Classification



Multi Class Classification

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

Softmax Classifier

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

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

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

C dimensional z
z^j = j'th class score

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

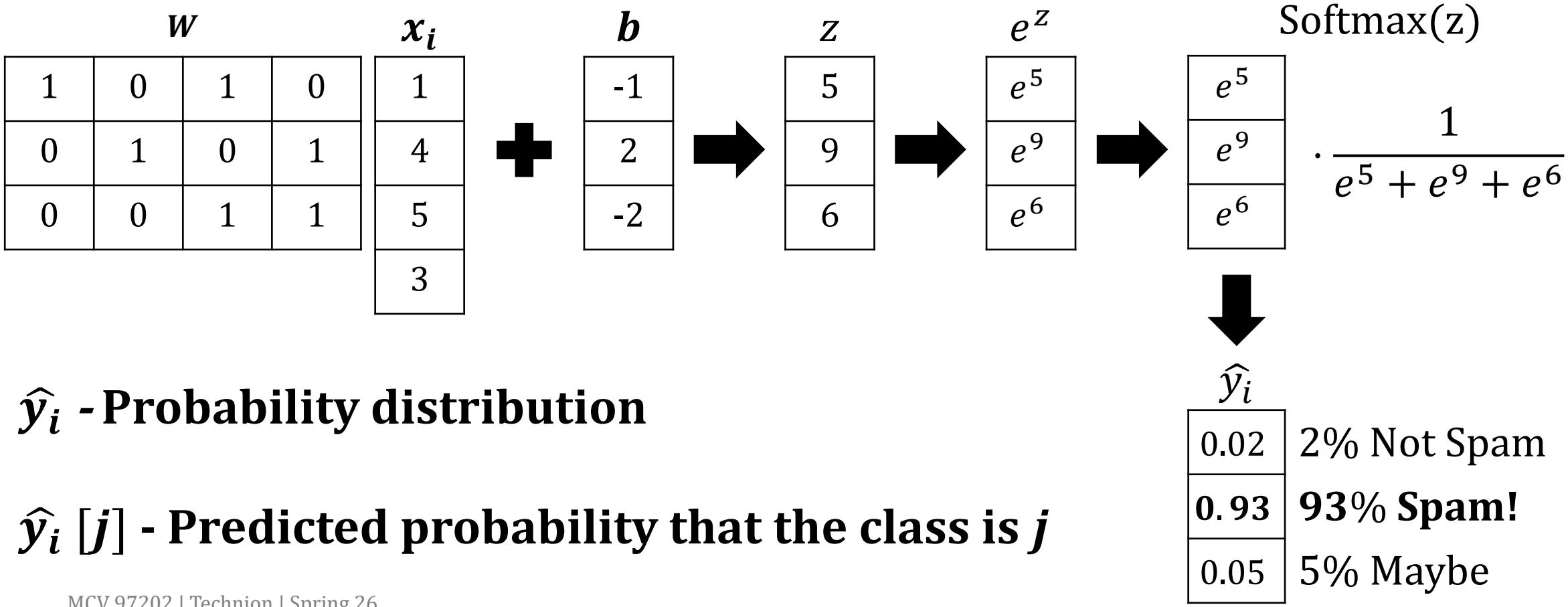
Sigmoid (Logistic function)	Softmax

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

$\mathbb{1}(\hat{y} > 0.5)$	$argmax(\hat{y})$

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*

Softmax Function - Formally

Converting z to probability distribution over the classes C:

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

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

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

Softmax Classifier

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

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

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

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

Sigmoid (Logistic function)	Softmax

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

$\mathbb{1}(\hat{y} > 0.5)$	$argmax(\hat{y})$

Softmax Classifier

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

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

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

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

Sigmoid (Logistic function)	Softmax

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

$\mathbb{1}(\hat{y} > 0.5)$	$argmax(\hat{y})$

Cross-Entropy Loss – Softmax Classifier

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

$p(y_i = c|x)$ – Ground Truth (known), $\widehat{y_i}^c = q(y_i = c|x)$ – Our Prediction

$p(y_i = c\|x_i)$	
0	Spam
1	**Not spam!**
0	Maybe

$\widehat{y_i}$	
0.02	2% Spam
0.93	**93% Not spam!**
0.05	5% Maybe

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

Goal: Prediction should be closer to GT.

Cross-Entropy Loss – Softmax Classifier

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

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

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

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

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

Softmax Classifier

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

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

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

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

Sigmoid (Logistic function)	Softmax

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

$\mathbb{1}(\hat{y} > 0.5)$	$argmax(\hat{y})$

Gradient Descent – Logistic Regression

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

$\hat{y_i} = \sigma(Wx_i + b)\ (W \in \mathcal{R}^{1\times d}, b \in \mathcal{R} \Rightarrow \hat{y_i} \in [0,1])$ — Find prediction

$\mathcal{L}_i = y_i \cdot (-\log \hat{y_i}) + (1 - y_i) \cdot (-\log(1 - \hat{y_i}))$ — Calculate Loss

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

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

Gradient Descent – Softmax Classifier

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

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

Find prediction

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

Calculate Loss

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

Update weight

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

Update bias

Gradient Descent – Softmax Classifier

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

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

Find prediction

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

Calculate Loss

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

Update weight

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

Update bias

Gradient Descent – Softmax Classifier

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

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

Find prediction

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

Calculate Loss

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

Update weight

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

Update bias

Gradient Descent – Softmax Classifier

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

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

Find prediction

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

Calculate Loss

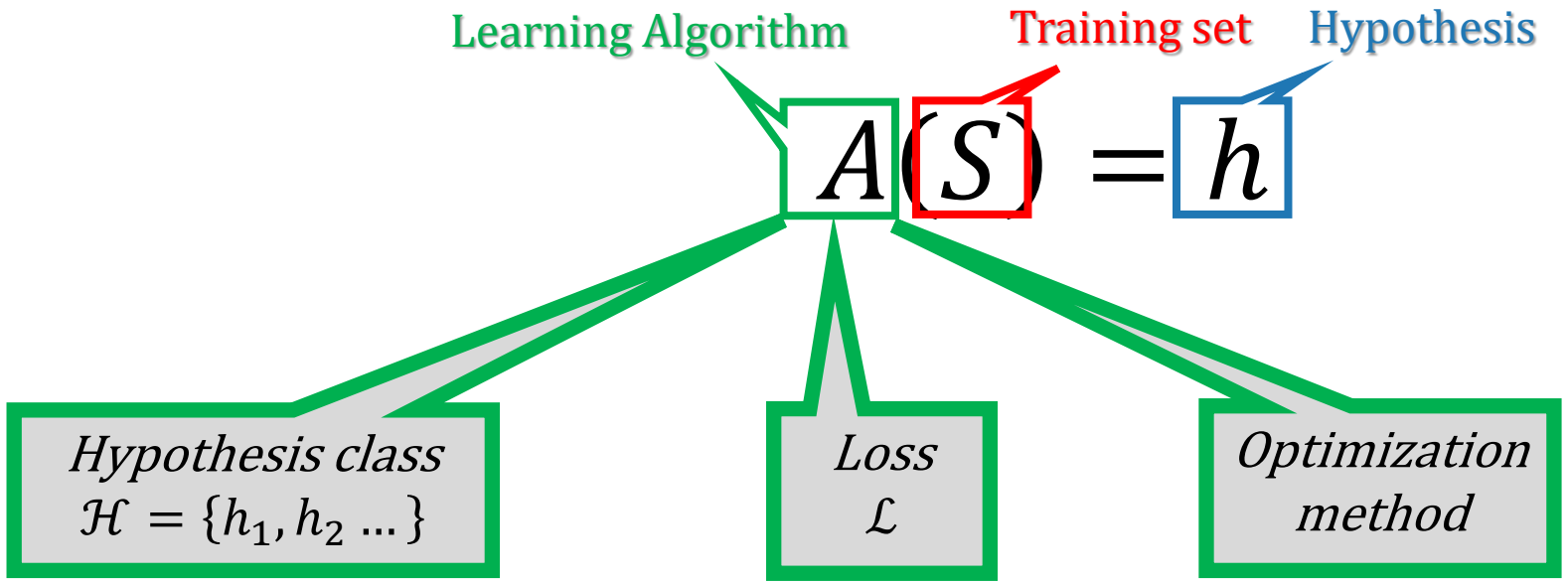
$$W \leftarrow W - \alpha \cdot \frac{\partial \mathcal{L}_i}{\partial W} = W - \alpha \cdot \boldsymbol{x_i^T}(\hat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{\delta}_{\boldsymbol{y_i}})$$

Update weight

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

Update bias

Supervised Learning – Softmax Classifier



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

Cross-entropy loss

SGD

Conclusion

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

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

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

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

Sigmoid (Logistic function)	Softmax

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

$\mathbb{1}(\hat{y} > 0.5)$	$argmax(\hat{y})$

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 $\mathcal{L}_i$ for each pair

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

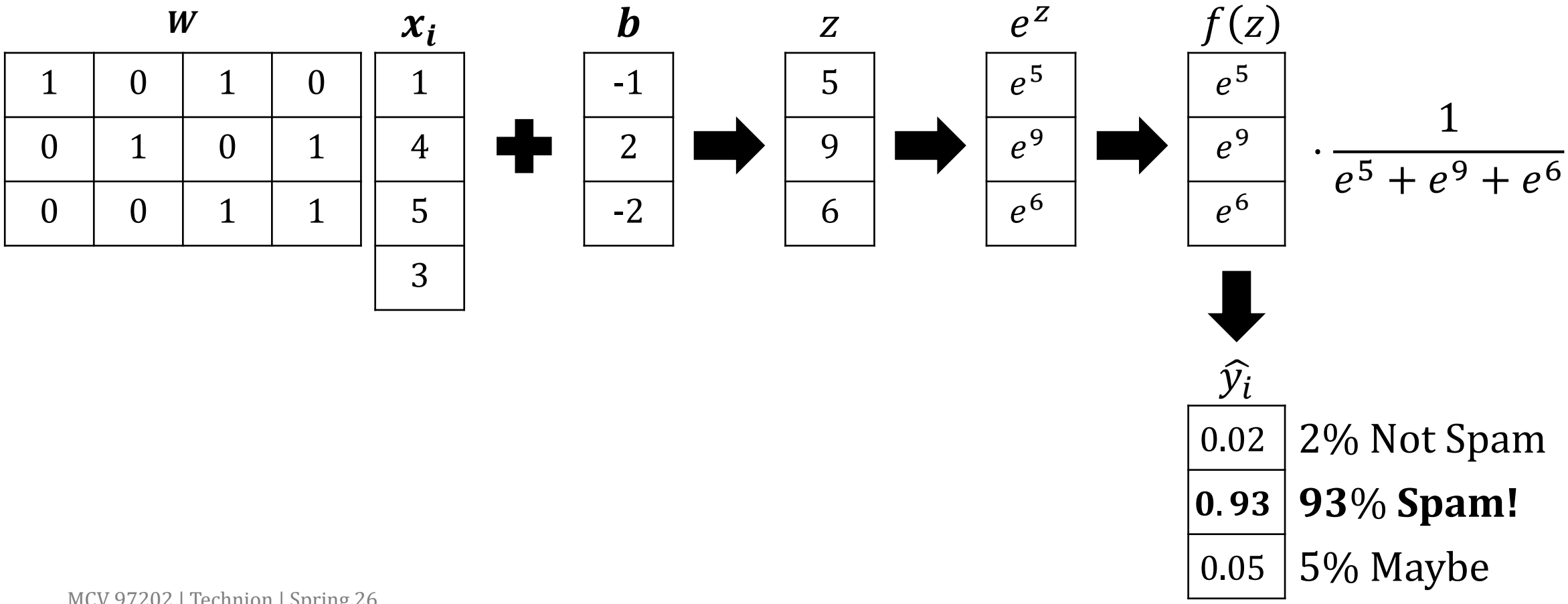
$$W \leftarrow W - \alpha \cdot \frac{\partial \mathcal{L}}{\partial W}$$

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

Not efficient sequentially

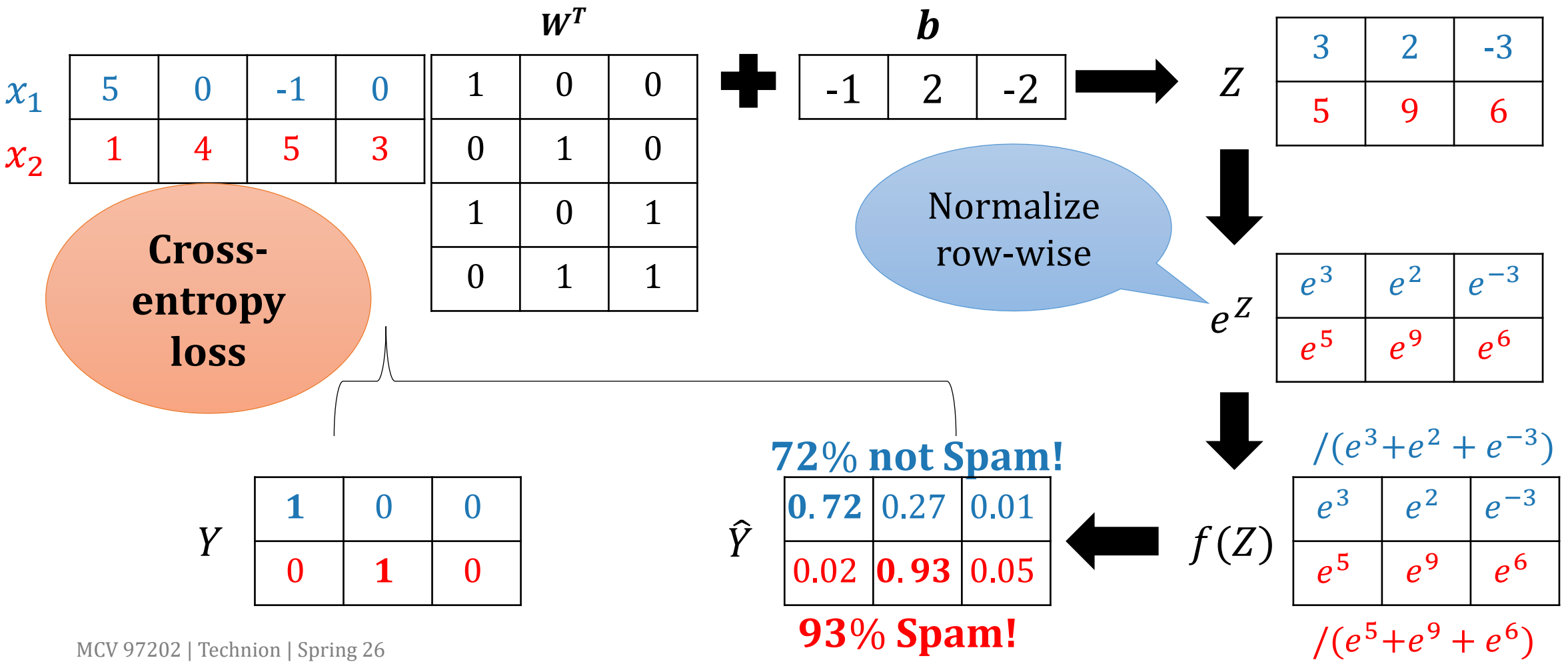
Softmax Function - Reminder

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



Softmax Classifier – Batched Example

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



Mini Batches - Formally

Define $X = [x_1, \dots, x_m]^T \in \mathcal{R}^{m \times d}$ - samples matrix

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

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

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

(b is added to each row)

Mini Batches - Formally

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

Softmax & Cross-entropy applied row-wise

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

$$\mathcal{L} = \frac{1}{m} \sum_{\mathrm{i}} \mathcal{L}_i$$

Reminder: Weights update (single-sample):

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

Weights update (mini-batch):

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

Numerical Stability

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