Classification Methods

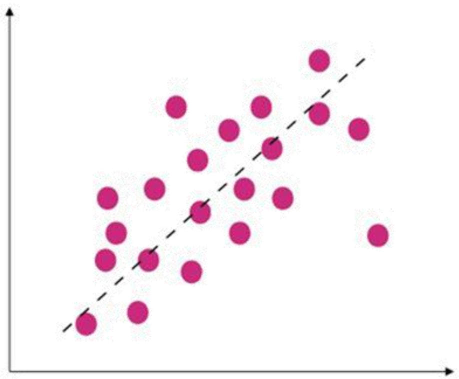
October 28th, 2021

Logistics - Reminder

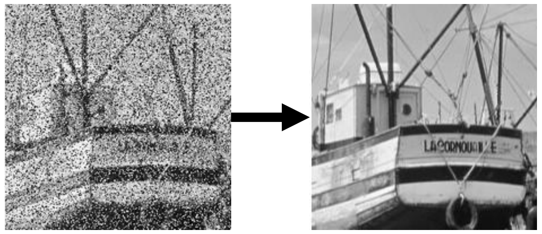
- Everything on course website / Moodle
- 4 credit points
- 2 hrs. lecture, 1 hr. tutorial (Tutorial covers new material)
- 4 homework assignments (60%) + Final Project (40%)
- All communication through Moodle
- Course mail: dl4cv.wis@gmail.com

Supervised Learning

Regression

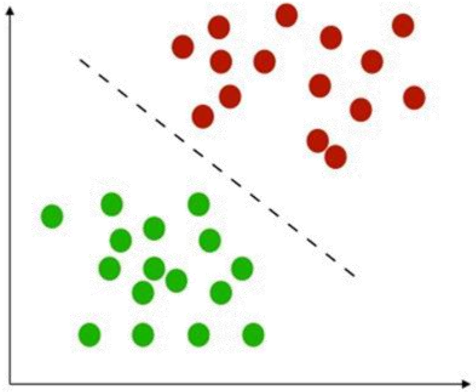


E.g. Image denoising



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

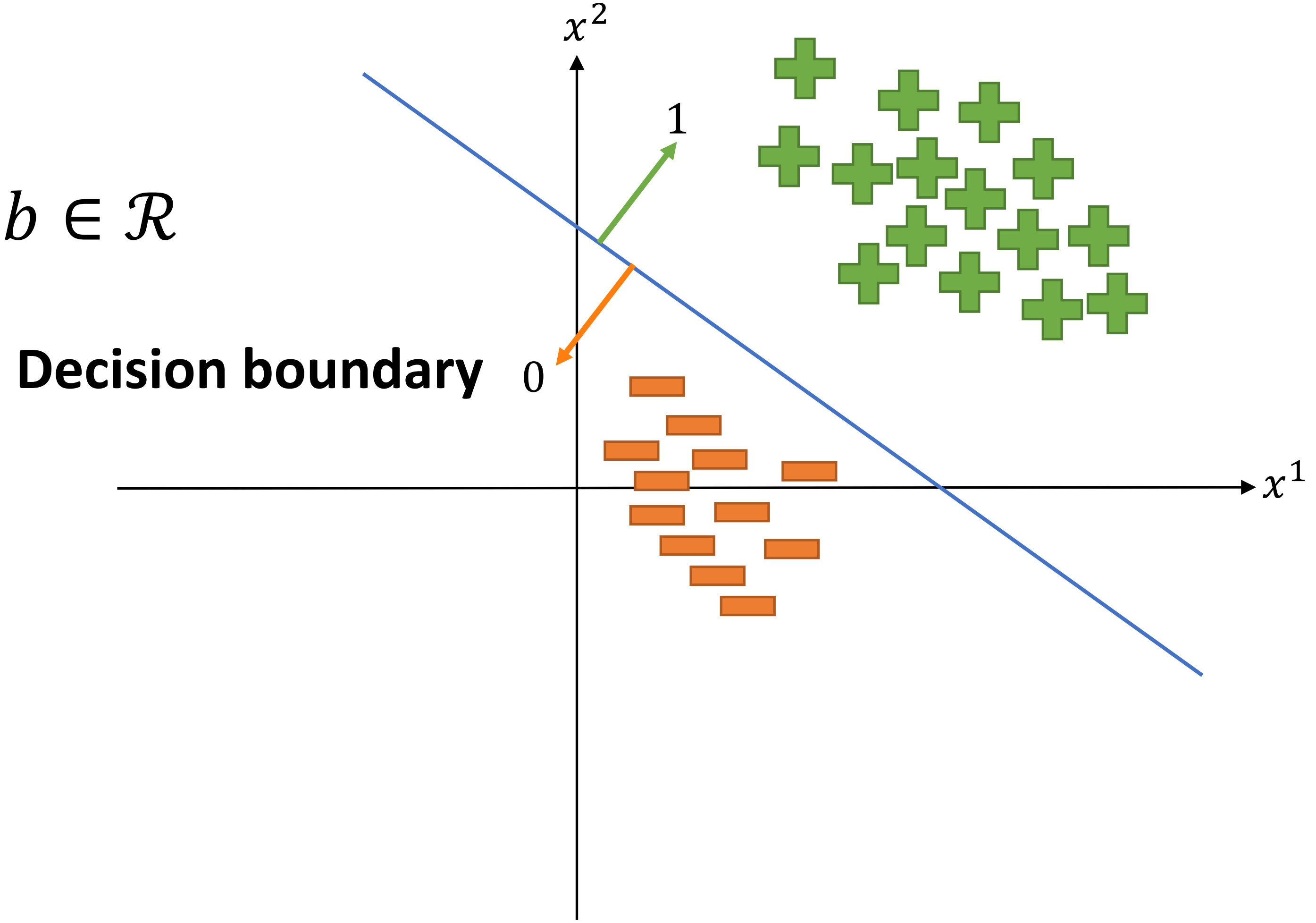
- Hypothesis: $h: \mathcal{R}^d \rightarrow \{0,1\}$, parameterized by θ

Linear Classifier

Indicator function

- $\theta = [W, b], h_{W,b}(x_i) = \mathbb{1}(Wx_i + b > 0.5)$

- 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

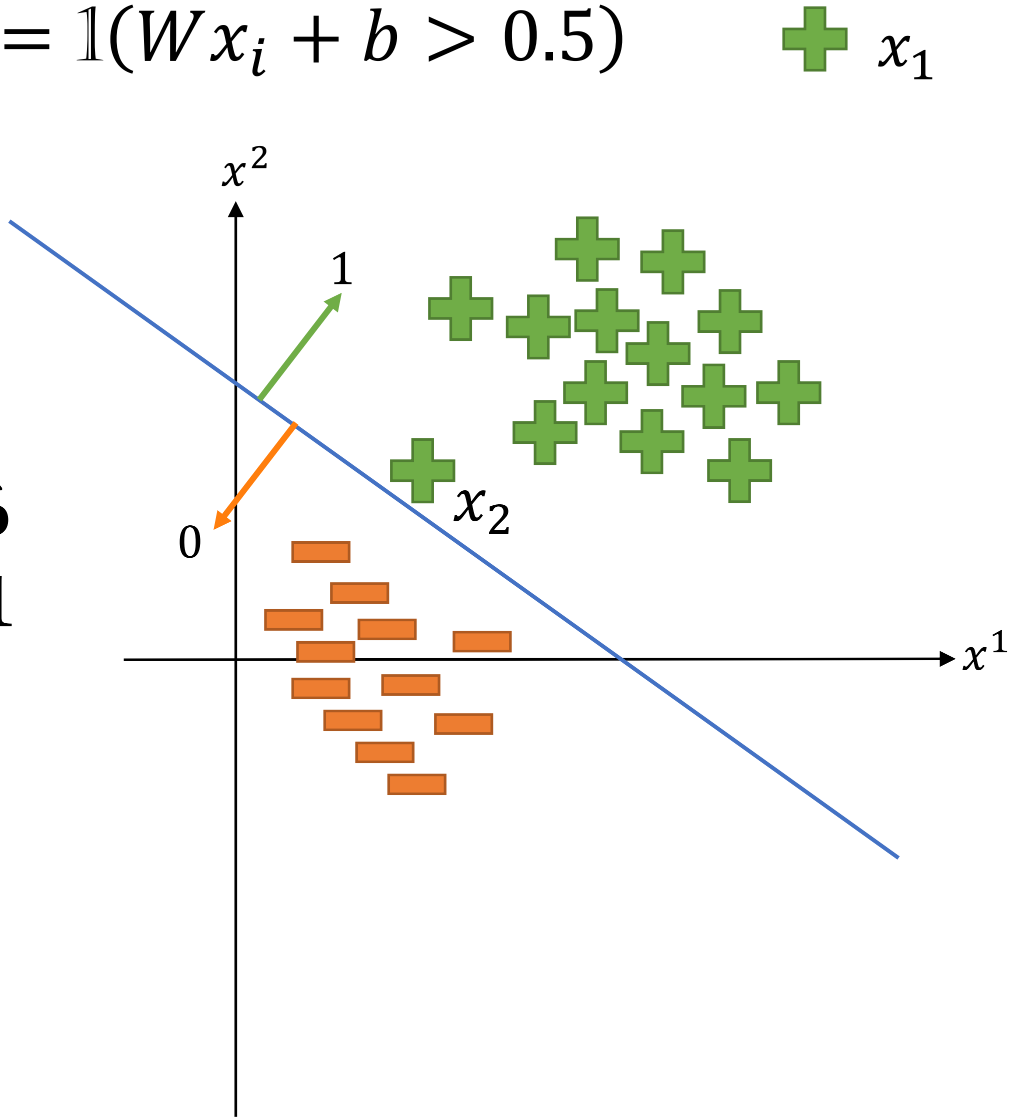
$$h_\theta(x_i) = \mathbb{1}(Wx_i + b > 0.5)$$

- 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

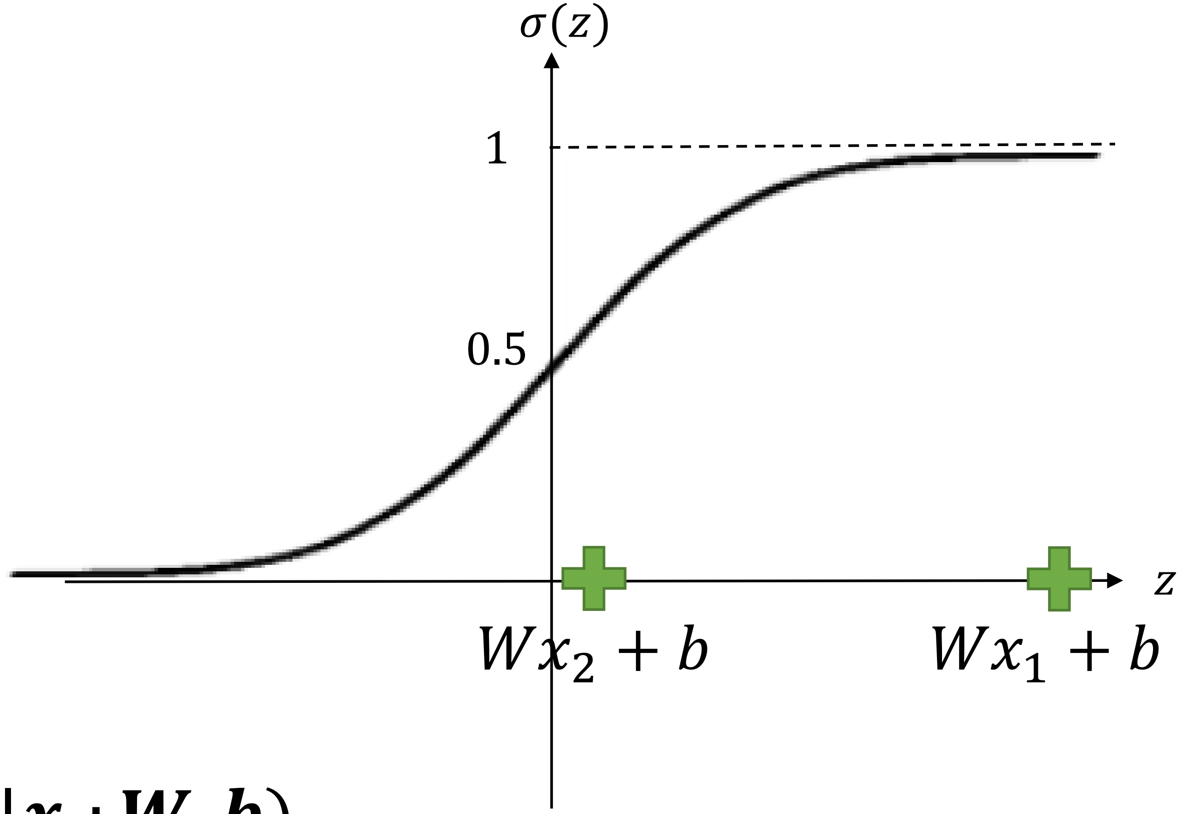
- Easy to find W, b

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

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

Sigmoid (Logistic Function)

Final prediction:

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

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

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

Predicting highest probability event

Logistic Regression

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

Sigmoid (Logistic Function)

[redacted]

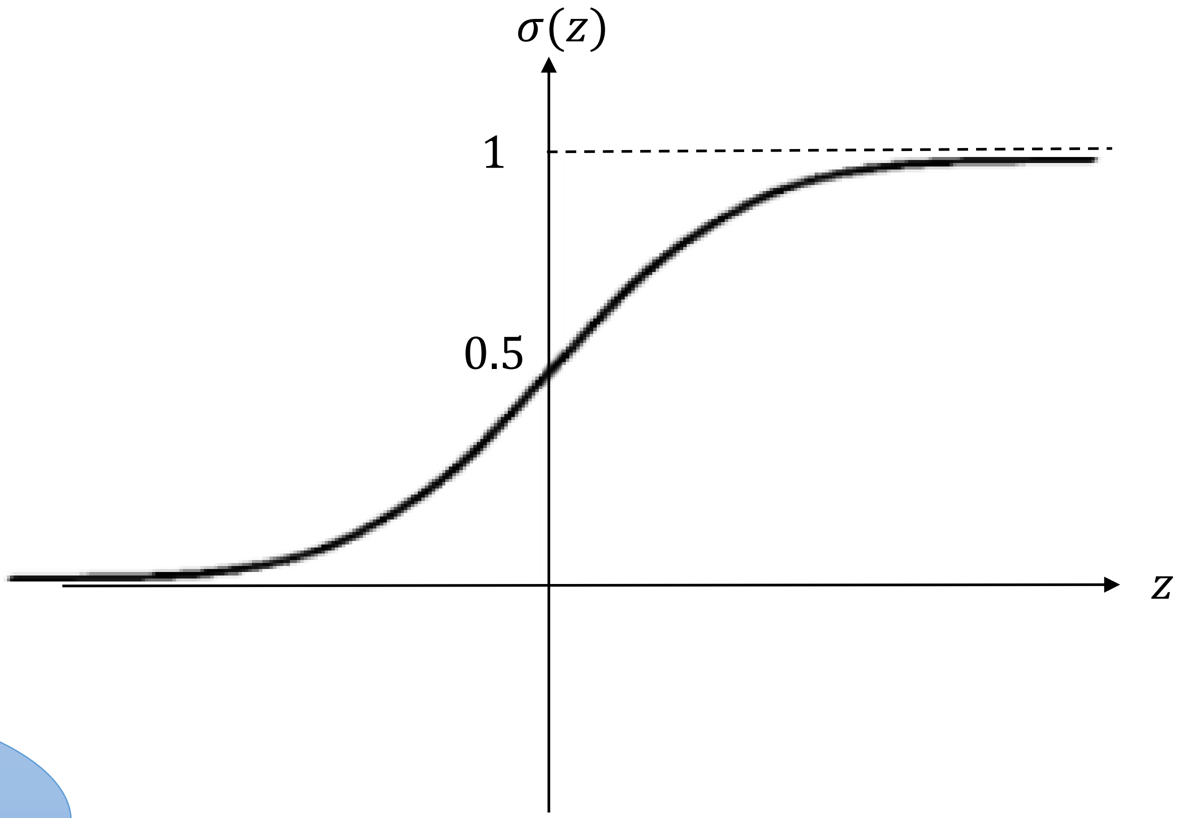
$$h_{W,b}(x_i) = \mathbb{1}(\sigma(z) > 0.5)$$

$\sigma(z) > 0.5$

$\Leftrightarrow z > 0$

$\Rightarrow$ Linear decision boundary

$z = Wx + b$



Logistic Regression

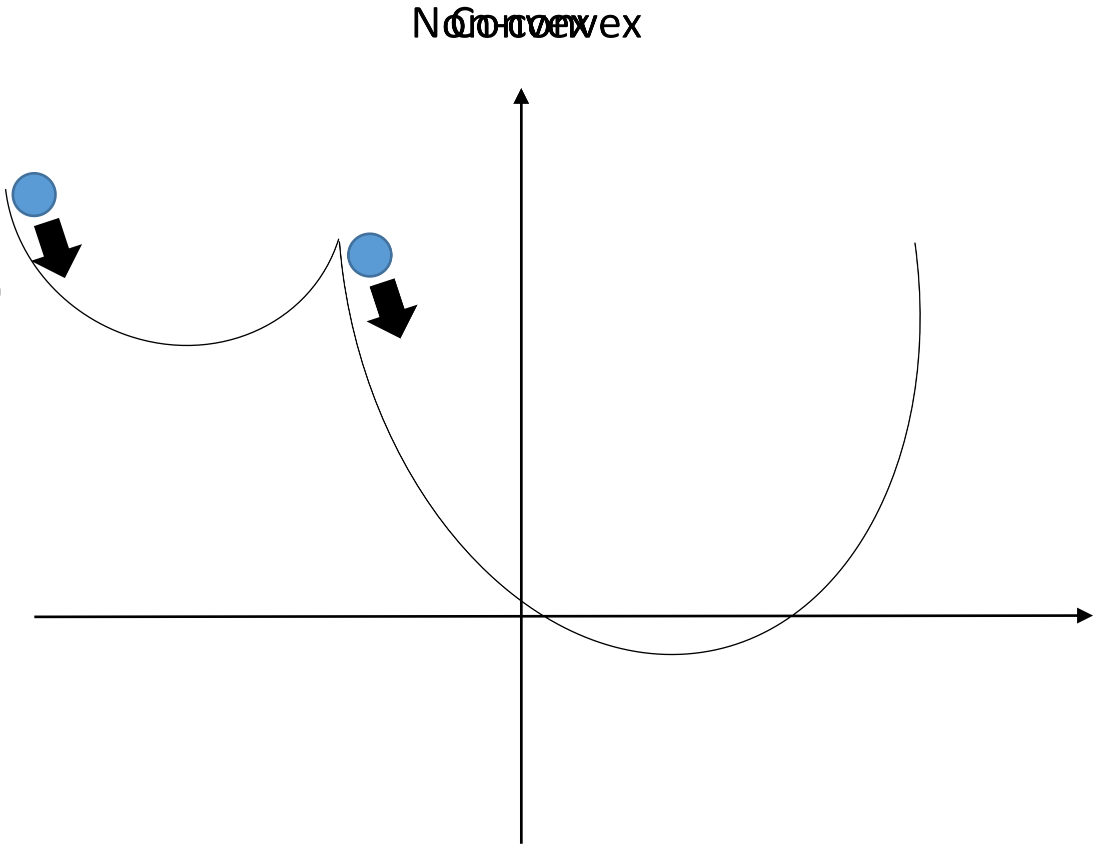
- Interpretable results

- 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: $\mathcal{L} \rightarrow \infty$
  - Correct prediction, small penalty: $\mathcal{L} \rightarrow 0$
- Use gradient descent to update W, b



Cross-Entropy Loss

$\Pr[y_i = 1|x_i]$

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

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

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

Cross-Entropy Loss

$\Pr[y_i = 1|x_i]$

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

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

	$\boldsymbol{y_i = 0}$	$\boldsymbol{y_i = 1}$
$\boldsymbol{\widehat{y}_i \rightarrow 0}$	$\mathcal{L}_i \rightarrow 0$	$\mathcal{L}_i \rightarrow \infty$
$\boldsymbol{\widehat{y}_i \rightarrow 1}$	$\mathcal{L}_i \rightarrow \infty$	$\mathcal{L}_i \rightarrow 0$

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

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

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

Find prediction

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

Calculate Loss

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

Update weights

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

Update bias

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

[redacted]

Learning Algorithm — Training set — Hypothesis

$$A(S) = h$$

Hypothesis class $\mathcal{H} = \{h_1, h_2 \dots\}$

Loss $\mathcal{L}$

Optimization method

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

Cross-entropy loss

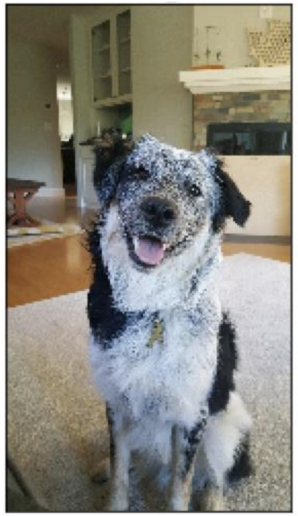
SGD

Logistic Regression - Summary

Probability Regression

- **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

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 **Maybe** **Not Spam**

Multi Class Classification

[redacted]

- 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

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

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

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

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

Softmax Function - Example

[redacted]

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

$$\overset{w}{\begin{bmatrix}1&0&1&0\\0&1&0&1\\0&0&1&1\end{bmatrix}}\overset{\boldsymbol{x_i}}{\begin{bmatrix}1\\4\\5\\3\end{bmatrix}} + \overset{\boldsymbol{b}}{\begin{bmatrix}-1\\2\\-2\end{bmatrix}} \rightarrow \overset{z}{\begin{bmatrix}5\\9\\6\end{bmatrix}} \rightarrow \overset{e^z}{\begin{bmatrix}e^5\\e^9\\e^6\end{bmatrix}} \rightarrow \overset{f(z)}{\begin{bmatrix}e^5\\e^9\\e^6\end{bmatrix}} \cdot \frac{1}{e^5+e^9+e^6}$$

$\downarrow$

$\hat{y}_i$

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

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

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

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

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

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

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

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

Sigmoid (Logistic function)	Softmax

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

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

Similar Learning:

1. Cross-entropy loss
2. SGD

Softmax Classifier

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

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

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

Sigmoid (Logistic function)	Softmax

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

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

Similar Learning:

1. Cross-entropy loss
2. SGD

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), $\hat{y}_i^{\,c} = q(y_i = c|x)$ – Our Prediction

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

$\hat{y}_i$	
0.02	2% Spam
0.93	**93% Not spam!**
0.05	5% Maybe

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

Goal: Prediction should be closer to GT.

Cross-Entropy Loss – Softmax Classifier

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

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

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

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

- 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

[redacted]

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

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

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

Sigmoid (Logistic function)	Softmax

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

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

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

Gradient Descent – Logistic Regression

[redacted]

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

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

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

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

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

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

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

Gradient Descent – Softmax Classifier

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

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

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

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

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

Find prediction

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

Calculate Loss

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

Update weight

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

Update bias

Gradient Descent – Softmax Classifier

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

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

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

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

Update weight

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

Update bias

Supervised Learning – Softmax Classifier

Learning Algorithm

Training set

Hypothesis

$$A(S) = h$$

Hypothesis class
$\mathcal{H} = \{h_1, h_2 \ldots\}$

Loss
$\mathcal{L}$

Optimization method

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

Cross-entropy loss

SGD

Conclusion

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

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

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

Sigmoid (Logistic function)	Softmax

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

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

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

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

[redacted]

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

w

1	0	1	0
0	1	0	1
0	0	1	1

$\boldsymbol{x_i}$

1
4
5
3

+

$\boldsymbol{b}$

-1
2
-2

→

z

5
9
6

→

e^z

e^5
e^9
e^6

→

$f(z)$

e^5
e^9
e^6

$$\cdot \frac{1}{e^5 + e^9 + e^6}$$

↓

$\hat{y}_i$

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

Softmax Classifier – Batched Example

spam

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

x_1	5	0	-1	0
x_2	1	4	5	3

$\boldsymbol{w}$

1	0	0
0	1	
1		
0		

➕ $\boldsymbol{b}$

-1	2	-2

➡ Z

3	2	-3
5	9	6

Normalize row-wise

e^Z

	e^9	e^6

$/(e^3+e^2+e^{-3})$

$f(Z)$

e^3	e^2	e^{-3}
e^5	e^9	e^6

$/(e^5+e^9+e^6)$

72% not Spam!

$\hat{Y}$

0.72	0.27	0.01
0.02	**0.93**	0.05

93% Spam!

Cross-entropy loss

Y

1	0	0
0	**1**	0

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

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

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

Reminder: Weights update (single-sample):

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

Weights update (mini-batch):

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

Similarly derive cross-entropy rule

Numerical Stability

- Softmax: e^z can be extremely large

- Cross-entropy: $\hat{y}$ possibly close to zero $\Rightarrow \log(\widehat{y_i}) \to -\infty$

- Many possible solutions


QUESTIONS?
imgflip.com