Lecture 3:
- Classification with Logistic Regression
- Advanced optimization techniques
- Underfitting & Overfitting
- Model selection (Training- & Validation- & Testset)
CLASSIFICATION WITH LOGISTIC REGRESSION
Logistic Regression

- **Classification** and not regression
- Classification = recognition

Mogees and vibration classification: [https://www.youtube.com/watch?v=xv4hII_h10](https://www.youtube.com/watch?v=xv4hII_h10)
Action recognition: [https://www.youtube.com/watch?v=ajswsWVWQvY](https://www.youtube.com/watch?v=ajswsWVWQvY)
Logistic Regression

• „The“ default classification model
  • Binary classification
  • Extensions to multi-class later in the course

• Simple classification algorithm
  • Convex cost - unique local optimum
  • Gradient descent
  • No more parameter than with linear regre

• Interpretability of parameters

• Fast evaluation of hypothesis for making predictions
LOGISTIC REGRESSION

Hypothesis
Example (step function hypothesis)

<table>
<thead>
<tr>
<th>i</th>
<th>Tumor size (mm)</th>
<th>Malignant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.3</td>
<td>0 (N)</td>
</tr>
<tr>
<td>2</td>
<td>5.1</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>3</td>
<td>1.4</td>
<td>0 (N)</td>
</tr>
<tr>
<td>4</td>
<td>6.3</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>5</td>
<td>5.3</td>
<td>1 (Y)</td>
</tr>
</tbody>
</table>

Tumor size \( (x) \)

\[ h_\theta(x) = 0 \quad \text{for} \quad x < 4.2 \]

\[ h_\theta(x) = 1 \quad \text{for} \quad x \geq 4.2 \]

"labeled data"
**Example (logistic function hypothesis)**

<table>
<thead>
<tr>
<th>i</th>
<th>Tumor size (mm)</th>
<th>Malignant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.3</td>
<td>0 (N)</td>
</tr>
<tr>
<td>2</td>
<td>5.1</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>3</td>
<td>1.4</td>
<td>0 (N)</td>
</tr>
<tr>
<td>4</td>
<td>6.3</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>5</td>
<td>5.3</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Hypothesis:** Tumor is malignant with probability $p$

**Classification:**
- if $p < 0.5$: 0
- if $p \geq 0.5$: 1
Example (logistic function hypothesis)

<table>
<thead>
<tr>
<th>( i )</th>
<th>Tumor size (mm)</th>
<th>Malignant ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.3</td>
<td>0 (N)</td>
</tr>
<tr>
<td>2</td>
<td>5.1</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>3</td>
<td>1.4</td>
<td>0 (N)</td>
</tr>
<tr>
<td>4</td>
<td>6.3</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>5</td>
<td>5.3</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

"labeled data"

Hypothesis: Tumor is malignant with probability \( p \)

Classification: if \( p < 0.5 \): 0  
if \( p \geq 0.5 \): 1

\[
h_\theta(x) < 0.5 \quad h_\theta(x) \geq 0.5
\]
Logistic (Sigmoid) function

\[ \sigma(z) = \frac{1}{1 + \exp(-z)} \]

- \( \sigma(-3) \approx 0.05 \)
- \( \sigma(0) = 0.5 \)
- \( \sigma(3) \approx 0.95 \)

- Advantages over step function for classification:
  - Differentiable → (gradient descent)
  - Contains additional information (how certain is the prediction?)
Logistic regression hypothesis (one input)

\[ h_\theta(x) = \sigma(z) = \sigma(\theta_0 + \theta_1 \cdot x) \]
## Classification with multiple inputs

### Table

<table>
<thead>
<tr>
<th>i</th>
<th>Tumor size (mm)</th>
<th>Age</th>
<th>Malignant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.3</td>
<td>25</td>
<td>0 (N)</td>
</tr>
<tr>
<td>2</td>
<td>5.1</td>
<td>62</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>3</td>
<td>1.4</td>
<td>47</td>
<td>0 (N)</td>
</tr>
<tr>
<td>4</td>
<td>6.3</td>
<td>39</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>5</td>
<td>5.3</td>
<td>72</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Diagram

![Diagram showing classification with multiple inputs](image-url)
Multiple inputs and logistic hypothesis

1. Reduce point in high-dimensional space to a scalar \( z \)
2. Apply logistic function

<table>
<thead>
<tr>
<th>( i )</th>
<th>Tumor size (mm)</th>
<th>Age</th>
<th>Malignant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.3</td>
<td>25</td>
<td>0 (N)</td>
</tr>
<tr>
<td>2</td>
<td>5.1</td>
<td>62</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>3</td>
<td>1.4</td>
<td>47</td>
<td>0 (N)</td>
</tr>
<tr>
<td>4</td>
<td>6.3</td>
<td>39</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>5</td>
<td>5.3</td>
<td>72</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ h_\theta(x) = \sigma(z) \]

\( \sigma(z) \) is the decision boundary,
\( p=0.8, \) class 1

\( \sigma(z) = \begin{cases} 
1 & \text{if } z > 0 \\
0 & \text{otherwise}
\end{cases} \)
## Classification with multiple inputs

1. Reduce point in high-dimensional space to a scalar $z$
2. Apply logistic function

$$h_\theta(x) = \sigma(z)$$

### Table

<table>
<thead>
<tr>
<th>$i$</th>
<th>Tumor size (mm)</th>
<th>Age</th>
<th>Malignant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.3</td>
<td>25</td>
<td>0 (N)</td>
</tr>
<tr>
<td>2</td>
<td>5.1</td>
<td>62</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>3</td>
<td>1.4</td>
<td>47</td>
<td>0 (N)</td>
</tr>
<tr>
<td>4</td>
<td>6.3</td>
<td>39</td>
<td>1 (Y)</td>
</tr>
<tr>
<td>5</td>
<td>5.3</td>
<td>72</td>
<td>1 (Y)</td>
</tr>
</tbody>
</table>

### Diagram

- $p=0.999$, class 1

\[ \text{Tumor size (x1)} \quad \text{Age (x2)} \]
Logistic regression hypothesis

1. Reduce high-dimensional input \( \mathbf{x} \) to a scalar

\[
z = \mathbf{x}^T \mathbf{\theta} \\
= \theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n
\]

2. Apply logistic function

\[
h_\theta(\mathbf{x}) = \sigma(\mathbf{x}^T \mathbf{\theta}) \\
= \sigma(\theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n)
\]

3. Interpret output \( h_\theta(\mathbf{x}) \) as probability and predict class:

\[
\text{Class} = \begin{cases} 
0 & \text{if } h_\theta(\mathbf{x}) < 0.5 \\
1 & \text{if } h_\theta(\mathbf{x}) \geq 0.5
\end{cases}
\]
LOGISTIC REGRESSION

Cost function
Logistic regression cost function

- How well does the hypothesis \( h_\theta(x) = \sigma(x^T \theta) \) fit the data?
Logistic regression cost function

- **Probabilistic model**: $y$ is 1 with probability:

$$h_{\theta}(x) = \sigma(x^T \theta)$$
Logistic regression cost function

- **Probabilistic model:** \( y \) is 1 with probability \( p(x, y=1) = h_\theta(x) = \sigma(x^T \theta) \)

The parameters should maximize the **likelihood** of the data

\[
\max_\theta \log p(X = (x_1 \ldots x_n), y = (y_1, \ldots y_n) | \theta)
\]

If data points are independants

\[
\max_\theta \sum_i \log p(x_i, y_i | \theta)
\]

Separating positive and negative examples

\[
\max_\theta \sum_{y_i=1} \log p(x_i, 1 | \theta) + \sum_{y_i=0} \log p(x_i, 0 | \theta)
\]

\[
\sigma(x^T \theta) \quad 1 - \sigma(x^T \theta)
\]
Logistic regression cost function

• How well does the hypothesis \( h_\theta(x) = \sigma(x^T \theta) \) fit the data?
• “Cost” for predicting probability \( p \) when the real value is \( y \):

\[
\text{Cost}(p, y) = \begin{cases} 
- \log(1 - p) & \text{if } y = 0, \\
- \log(p) & \text{if } y = 1.
\end{cases}
\]

• Mean over all training examples:

\[
J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)}).
\]
Multiple inputs and logistic hypothesis

- How well does the hypothesis $h_\theta(x) = \sigma(x^T \theta)$ fit the data?

![Graphs showing predictions and actual values for $y = 0$ and $y = 1$.](image)

- Prediction: 0.1  
  Actual value $y$: 0

- Prediction: 0.6  
  Actual value $y$: 0

- Prediction: 0.98  
  Actual value $y$: 1
Comparison cost functions

**Linear regression**

Cost\((h, y) = (h - y)^2\)

**Logistic regression**

\[
\text{Cost}(p, y) = \begin{cases} 
-\log(1 - p) & \text{if } y = 0 \\
-\log(p) & \text{if } y = 1 
\end{cases}
\]

Mean over all training examples:

\[
J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})
\]
Why not mean squared error (MSE) again?

- **MSE** with logistic hypothesis is **non-convex** (many local minima)
- Logistic regression **is convex** (unique minimum)
- Cost function can be derived from statistical principles („maximum likelihood“)
LOGISTIC REGRESSION
Learning from data
Minimizing the cost via gradient descent

- Gradient descent

\[
\theta_j := \theta_j - \eta \cdot \frac{\partial}{\partial \theta_j} J(\theta)
\]

(simultaneous update for \(j=0 \ldots n\))

- Gradient of logistic regression cost:

\[
\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}_j
\]

"error" "input"

(for \(j=0: x_0^{(i)} = 1\))
Linear to non-linear features

\[ x_1 = \text{Tumor Size}, \quad x_2 = \text{Age} \]

\[ h_\theta(x) = \sigma(-10 + 2 \cdot x_1 + 0.05 \cdot x_2) \]
Linear to non-linear features

\[ \phi_1 = \text{Tumor Size}, \ \phi_2 = \text{Age}, \ \phi_3 = \text{Tumor Size}^2, \]
\[ \phi_4 = \text{Age}^2, \ \phi_5 = \text{Tumor Size} \cdot \text{Age}, \ldots \]

\[ h_\theta(\phi) = \sigma(-3 + 1.2 \cdot \phi_1 + 0.07 \cdot \phi_2 - 0.9 \cdot \phi_3 + \ldots) \]
Decision boundaries

linear decision boundary

\[ h_\theta(x) = \sigma(-10 + 2 \cdot x_1 + 0.05 \cdot x_2) \]

non-linear decision boundary

\[ h_\theta(\phi) = \sigma(-3 + 1.2 \cdot \phi_1 + 0.07 \cdot \phi_2 - 0.9 \cdot \phi_3 + \ldots) \]

Decision boundary is a property of hypothesis, not of data!
Linear vs. Logistic Regression

Linear Regression

- Regression
- Hypothesis $h_\theta(x) = x^T \theta$
- Cost for one training example:
  \[
  \text{Cost}(h, y) = (h - y)^2
  \]

- Gradient
  \[
  \frac{\partial}{\partial \theta_j} J(\theta) = \frac{2}{m} \sum_{i=1}^{m} \underbrace{(h_\theta(x^{(i)}) - y^{(i)})}_{\text{error}} \cdot x_j^{(i)} \underbrace{x_j^{(i)}}_{\text{input}}
  \]
- Analytical:
  \[
  \theta^* = \left( X^T X \right)^{-1} X^T y
  \]

Logistic Regression

- Binary classification (!)
- Hypothesis $h_\theta(x) = \sigma(x^T \theta)$
- Cost for one training example:
  \[
  \text{Cost}(p, y) = \begin{cases} 
  -\log(1-p) & \text{if } y = 0 \\
  -\log(p) & \text{if } y = 1 
  \end{cases}
  \]

- Gradient
  \[
  \frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{(h_\theta(x^{(i)}) - y^{(i)})}_{\text{error}} \cdot x_j^{(i)} \underbrace{x_j^{(i)}}_{\text{input}}
  \]
- No analytical solution!
GRADIENT DESCENT TRICKS, AND MORE ADVANCED OPTIMIZATION TECHNIQUES

For linear regression, logistic regression, ....
GD trick #1: feature scaling

- Feature scaling and mean normalization
  - Bring all features into a similar range

- E.g.: shift and scale each feature to have mean 0 and variance 1
  \[ x_j \leftarrow \frac{x_j - \mu_j}{\sigma_j} \]
  \[ \phi_j \leftarrow \frac{\phi_j - \mu_j}{\sigma_j} \]
  (when using non-linear features)

- Do not apply to constant feature \( x_0 / \phi_0 \)!

- Typically leads to much faster convergence
GD trick #2: monitoring convergence

- Diagnose typical issues with Gradient Descent:

- Slow convergence (increase learning rate?)
- Oscillations (decrease learning rate)
- Divergence (decrease learning rate)
GD trick #3: adaptive learning rate

- At each iteration
  - Compare cost function value $J(\theta)$ before and after Gradient Descent update

- If cost increased:
  - Reject update (go back to previous parameters)
  - Multiply learning rate $\eta$ by 0.7 (for example)

- If cost decreased:
  - Multiply learning rate $\eta$ by 1.02 (for example)

Often eliminates slow convergence and divergence issues
A black box view of gradient descent

- Write code for computing the cost and its gradient

\[
\theta_j := \theta_j - \eta \cdot \frac{\partial}{\partial \theta_j} J(\theta)
\]

- Stop when \( \left\| \nabla J \right\| < 10^{-4} \)
More advanced optimization methods

• Gradient methods = order 1  ➔  Newton methods = order 2

  - Need **Hessian matrix** or approximations
  - Avoid choosing a learning rate
  - Conjugate gradient, BFGS, L-BFGS, ...

• Tricky to implement (numerical stability, etc.)
  - Use available toolbox / library implementations!
    ```
    scipy.optimize.minimize
    ```
  - **Only use** when fighting for performance
EVALUATION OF HYPOTHESIS

Training and test set
Training and Test set

- **Training set**: used by learning algorithm to fit parameters and find a hypothesis.

- **Test set**: independent data set, used after learning to estimate the performance of the hypothesis on new (unseen) test examples.

Regression example

- E.g. 70% randomly chosen examples from dataset are training examples, the remaining 30% are test examples. Must be disjoint subsets!
Training and Test set workflow

Training set → Learning algorithm → „Hypothesis“ $h$ → Testing → Test set

Training error (cost) $J(\theta)$

Test error (cost) $J_{test}(\theta)$
Linear regression training vs. test error

\[ J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left( h_{\theta} \left( x^{(i)} \right) - y^{(i)} \right)^2 \]

\[ J_{\text{test}}(\theta) = \frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \left( h_{\theta} \left( x^{(i)}_{\text{test}} \right) - y^{(i)}_{\text{test}} \right)^2 \]
Classification Training / Test set

Training set

Test set
Logistic regression training vs. test error

\[ J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \]

\[ J_{\text{test}}(\theta) = -\frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \text{Cost}(h_\theta(x^{(i)}_{\text{test}}), y^{(i)}_{\text{test}}) \]
UNDERFITTING AND OVERTFITTING
Polynomial regression under-/overfitting

\[ \text{MSE}_{\text{train}} = 0.22, \text{MSE}_{\text{test}} = 0.29 \]
Polynomial regression under-/overfitting

MSE_train = 0.01, MSE_test = 0.01

- training
- test
- polynomial fit degree 5
Polynomial regression under-/overfitting

MSE_{\text{train}}=0.00, \quad \text{MSE}_{\text{test}}=3.17
Polynomial regression under-/overfitting

MSE_{train}=0.22, MSE_{test}=0.29

underfitting

MSE_{train}=0.01, MSE_{test}=0.01

just right

MSE_{train}=0.00, MSE_{test}=3.17

overfitting
Logistic regression with polynomial terms

Training set

Test set
Logistic regression with polynomial terms

Terms up to power 1

decision boundary

Terms up to power 2
Logistic regression with polynomial terms

Training Error: 0.17

Test Error: 0.44

Terms up to power 20
underfitting

overfitting

Training Error: 0.33
Test Error: 0.32

Training Error: 0.17
Test Error: 0.44

Training Error: 0.19
Test Error: 0.19

„just right“
Under-/ and Overfitting

Model complexity
(e.g. degree of polynomial terms)

Training error (cost)
Test error (cost)

underfitting

overfitting

„just right“
Under- and Overfitting

- **Underfitting:**
  - Model is **too simple** (often: too few parameters)
  - **High training error, high test error**

- **Overfitting**
  - Model is **too complex** (often: too many parameters relative to number of training examples)
  - Low training error, **high test error**

- **In between:**
  - Model has „right“ complexity
  - Moderate training error
  - **Lowest test error**
How to deal with overfitting

- Use **model selection** to automatically select the right model complexity
- Use **regularization** to keep parameters small (*other lecture*…)

- Collect more data
  (often not possible or inefficient)

- Manually throw out features which are unlikely to contribute
  (often hard to guess which ones, potentially throwing out the wrong ones)

- Change features vectors, use pre-processing
  (often not possible or inefficient, time consuming)
MODEL SELECTION
Training, Validation and Test sets
Model selection

- Selection of learning algorithm and "hyperparameters" (model complexity) that are most suitable for a given learning problem

![Graph showing training and test error versus model complexity]
Idea

- Try out different learning algorithms/variants
  - Vary degree of polynomial
  - Try different sets of features
  - ...

- Select variant with best predictive performance
Training, Validation, Test set

- **Training set**: used by learning algorithm to fit parameters and find a hypothesis for each learning algorithm/variant.

- **Validation set**: used to estimate predictive performance of each learning algorithm/variant. The hypothesis with lowest validation error (cost) is selected.

- **Test set**: independent data set, used after learning and model selection to estimate the performance of the final (selected) hypothesis on new (unseen) test examples.

E.g. 60/20/20 % randomly chosen examples from dataset. Must be disjoint subsets!
Training/Validation/Test set workflow

For example:
degree of polynome=1, 5, and 15
Training/Validation/Test set workflow

For example:
Linear regression, Polynomial regression, Artificial Neural Network
Some questions…

• Logistic regression is a method for … regression/classification?
• Logistic regression hypothesis?
• What’s the cost function used for logistic regression?
• Is it convex or non-convex?
• What does „adaptive learning rate“ mean in the context of gradient descent?
• How to evaluate a hypothesis?
• What is under-/overfitting?
• What is model selection?
• What are training, validation and test sets?
• How does model selection work (procedure)?
What is next?

• Neural Networks:
  • Perceptron
  • Feedforward Neural Network
  • Backpropagation