Machine Learning

Machine learning

- See also Machine Learning on Coursera

```
# build with

pandoc -V geometry:margin=2.5cm -V title:"Machine Learning Notes" doc.md -o doc.pdf
```

Inputs and outputs

Example data *House Prices*

#	Size (sq. feet)	Num Bedrooms	Num. Floors	Age	Price
	x_1	x_2	x_3	x_4	y_1
1	2104	5	1	45	460
2	1416	3	2	40	232
3	1534	3	2	30	315
4	852	2	1	36	178
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
47	1321	1	2	10	300

Notation

- n: number of features, i.e. *dimension* of $x^{(i)}$
  - also denoted as D
- $x^{(i)}$: input (features) of i^{th} training example
- $x_j^{(i)}$: value of feature j in i^{th} training example
- see also https://nthu-datalab.github.io/ml/slides/Notation.pdf
  - K is dimension of a label y^i
  - $\mathbb{X}$ is set of training examples
  - N: size of $\mathbb{X}$
  - $\mathbb{F}$: hypothesis space of functions to be learnt, i.e. a model
  - $Cost[f]$: a cost functional $f \in \mathbb{F}$
  - (x', y'): a testing pair
  - $\hat{y}$: predicated label by a function f

Applied to Example

- $n = D = 4$
- $K = 1$
- $m = 47$

- $x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$
- $x_3^{(2)} = 2$

Linear Regression with Multiple Variables

Hypothesis h

- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$
  - Reads: Hypothesis parameterized by θ
  - For convenience: $x_0^{(i)} = 0$
- $x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$, $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$
- Vectorized Hypothesis: $h_\theta(x) = \theta^T x$

Cost function J

- A.k.a. **loss** or **objective** function
- Idea choose $\theta_0, \theta_1, \ldots, \theta_n$ so, that $h_\theta(x)$ is close to y for training examples (x, y)
- Cost function

$$J(\theta_0, \theta_1, \ldots, \theta_n) = J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \big(h_\theta(x^{(i)}) - y^{(i)}\big)^2$$

  - a.k.a *squared error function* J
- Goal: Minimize $J(\theta)$
- Gradient descent update step (*iteration*)

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

  - **Important**: Simultaneously update for every $j = 0, \ldots, n$
  - Learning rate α
- Gradient Descent can converge to a local minimum, even with a the learning rate α fixed
  - As we *approach a local minimum*, gradient descent will automatically take smaller steps, i.e. gradient $\frac{d}{d\theta_j} J(\theta_j)$ is smaller (becomes 0 when minimum is reached)
- "**Batch**" Gradient Decent: Each step of gradient decent uses **all** the **training examples**, i.e. $\sum_{i=1}^{m} \big((h_\theta(x^{(i)}) - y^{(i)}\big)$ operates on all samples
- Make sure gradient decent is working: plot number of iterations (x-axis) against $J(\theta)$ (y-axis), and $J(\theta)$ should **decrease after every iteration**
  - Convergence test: declare convergence, if $J(\theta)$ decreases by less than 10^{-3} in one iteration
  - For sufficiently small α, $J(\theta)$ should decrease on every iteration
- Choosing learning rate α
  - if α to too small: small convergence
  - is α is too large: $J(\theta)$ may not decrease on every iteration, and thus may not converge
  - To choose α, try $\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, \ldots$

Logistic Regression: Binary Classification

- Binary classification: $y \in \{0, 1\}$
  - 0 usually *negative class*, i.e. bening tumor (*gutartig*)
  - 1 usually *positive class*, i.e. malignant tumor (*bösartig*)

Hypothesis

- Want $0 \leq h_\theta(x) \leq 1$
- So

$$h_\theta(x) = g(\theta^T x)$$

  - with

$$g(z) = \frac{1}{1 + e^- z}$$

  - $g(z)$ is **logistic/sigmoid** function
- Interpretation of hypothesis output
  - $h_\theta(x)$ is estimated probability that $y = 1$ on input x
  - Example: if $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$ and $h_\theta(x) = 0.7 \rightarrow y = 1$, tell patient, that *70%* chance of tumor being malignant
  - $h_\theta(x) = P(y = 1|x; \theta)$, "probability that $y = 1$, given x, parameterized by θ"

Cost function

- How to choose parameters θ?
  - Training set

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

  - m examples:

$$x = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}, x_0 = 1, y \in \{0, 1\}$$

  - Hypothesis

$$h_\theta = \frac{1}{1 + e^{-\theta^T x}}$$

- Recap Linear regression:

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \big(h_\theta(x^{(i)}) - y^{(i)}\big)^2 \\ &= \frac{1}{m} \sum_{i=1}^{m} Cost\big(h_\theta(x^{(i)}), y^{(i)}\big) \end{aligned}$$

  - with

$$Cost\big(h_\theta(x^{(i)}), y^{(i)}\big) = \frac{1}{2} \big(h_\theta(x^{(i)}) - y^{(i)}\big)^2$$

  - Cost function $J(\theta)$ is a sum over the training set of the **cost term**
  - Simplified notation

$$Cost\big(h_\theta(x, y) = \frac{1}{2} \big(h_\theta(x) - y\big)^2$$

- Logistic regression cost function

$$Cost\big(h_\theta(x), y\big) = \begin{cases} -\log\big(h_\theta(x)\big), & \text{if } y = 1 \\ -\log\big(1 - h_\theta(x)\big), & \text{if } y = 0 \end{cases}$$

- **Note** $y = 0$ or $y = 1$ always (by definition)

- Simplified

$$Cost\big(h_\theta(x), y\big) = -y\log\big(h_\theta(x)\big) - (1-y)\log\big(1 - h_\theta(x)\big)$$

- To fit parameters θ:

$$\min_\theta J(\theta)$$

- To make a prediction given new x:
  - Output

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

  - reads $P(y = 1|x; \theta)$

Solving the problem of Overfitting (Regularization)

- If there are too many features (resulting in *high* polynomial of h_θ), the learned hypothesis may fit the training set very well, but fail to generalize new examples
- Options
  1. Reduce number of features
     - Manually select features to keep
     - Model selection
  2. Regularization
     - Keep all features, but reduce magnitude/values of parameters θ_j
     - Works well when we have a lot of features, each of which contributes a bit to predicting y
- Regularization: small values for parameters $\theta_0, \theta_1, \ldots, \theta_n$
  - *Simpler* hypothesis
  - Less prone to overfitting
- In regularized linear regression, we choose θ to minimize

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\big(\ h_\theta(x^{(i)}) - y^{(i)}\big)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

  - λ is **regularization parameter**
- λ penalizes θ_j values, i.e. forces them to stay *small*
- If λ is too high, e.g. $\lambda = 10^10$, then $h_\theta(x) = \theta_0$ and thus **underfits**