Introduction to Supervised Learning

Dr. Josif Grabocka

ISMLL, University of Hildesheim

Deep Learning

Outline

Machine Learning

- A branch of Artificial Intelligence:
  - Learning to solve a task
  - Learn to correctly estimate a target variable
  - Use previous contextualized data to infer future variable's values
  - Context is expressed through features

Figure 1 : Face Recognition, Courtesy of `www.nec.com`

Supervised and Unsupervised Learning

- **Supervised** learning:
  - Data is labeled by an expert (ground-truth)
  - *Classification, Regression, Ranking*
- **Unsupervised** learning:
  - Data contain no explicit labels apart the context features
  - *Clustering, Dimensionality reduction, Anomaly/Outlier Detection*

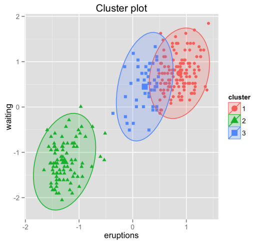


Figure 2 : Clustering illustration, Courtesy of `www.sthda.com`

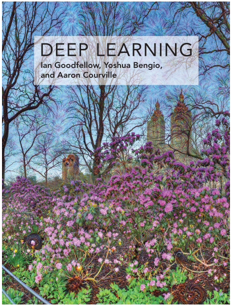
Deep Learning ...

- ... refers to a family of supervised and unsupervised methodologies involving:
  - Neural Network (NN) architectures
  - Specialized architectures, e.g. CNN, ...
  - Novel regularizations, e.g. Dropout, ...
  - Large-scale optimization approaches, e.g. GPU-s, ...

Figure 3 : Illustration of a neural network, Courtesy of www.extremetech.com

Course Description

- *Course name*: Deep Learning, *Course code*: 3107
- *Credits*: 6, *SWS*: 2
- *Location*: A102, *Time*: Wednesday 10:00 - 12:00 c.t.
- *Book*: "Deep Learning" by Ian Goodfellow, Yoshua Bengio and Aaron Courville, MIT Press 2016, Online: `www.deeplearningbook.org`



Outline

Example

- For N existing bank customers and $M = 23$ features, i.e. given $x \in \mathbb{R}^{N \times 23}$ and ground truth $y \in \{0,1\}^N$

$y_:$	Default credit card payment (Yes = 1, No = 0)
$x_{:,1}$	Amount of the given credit (NT dollar)
$x_{:,2}$	Gender (1 = male; 2 = female).
$x_{:,3}$	Education (1=graduate; 2=univ.; 3 = high school; 4 = others).
$x_{:,4}$	Marital status (1 = married; 2 = single; 3 = others).
$x_{:,5}$	Age (year)
$x_{:,6} - x_{:,11}$	Past Delays (-1=duly, ..., 9=delay of nine months)
$x_{:,12} - x_{:,17}$	Amount of bill statements
$x_{:,18} - x_{:,23}$	Amount of previous payments

Table 1 : Yeh, I. C., & Lien, C. H. (2009).

- Goal: Estimate the default of a new $(N+1)$-th customer, i.e. given $x_{N+1,:} \in \mathbb{R}^{23}$, estimate $y_{N+1} = ?$

Estimating the Target Variable

- Given a training data of N recorded instances, composed of
  - features variables $x \in \mathbb{R}^{N \times M}$ and
  - target variable $y \in \mathbb{R}^N$.
- Predict the target variable of a future instance $x^{test} \in \mathbb{R}^M$?
- Need to have a function $f(x)$ that predicts the target $\hat{y} := f(x)$
  - Known as "**Prediction Model**"
- How to find a good function? Answer:
  - Parametrize through learn-able parameters θ as $f(x, \theta)$
  - Learn parameters θ using the training data
  - *But, according to which criteria should we learn θ?*

Difference to Ground Truth

- The quality of a prediction model $f(x, \theta)$
  - Difference between the estimated target $\hat{y}$ and ground-truth target y
  - Defined as a loss function $\mathcal{L}(y, \hat{y}) : \mathbb{R} \times \mathbb{R} \leftarrow \mathbb{R}$
  - The term loss is used for minimization tasks, e.g. regression
- Note: sometimes a maximization of $\mathcal{L}(y, \hat{y})$ is needed

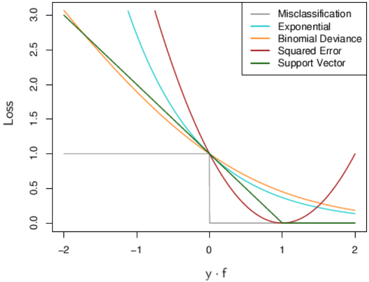


Figure 4 : Loss types, (Hastie et al., 2009, The Elements of Statistical Learning)

Archetype of a Machine Learning Method

- *Data dimensions*: N instances having M features
- *Features*: $x \in \mathbb{R}^{N \times M}$ and *Target*: $y \in \mathbb{R}^N$
- A *prediction model*: having parameters $\theta \in \mathbb{R}^K$ is $f : \mathbb{R}^M \times \mathbb{R}^K \to \mathbb{R}$

$$\hat{y}_n := f(x_n, \theta)$$

- *Loss function*: $\mathcal{L}(y_n, \hat{y}_n) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
- *Regularization*: $\Omega(\theta) : \mathbb{R}^K \to \mathbb{R}$
- *Objective function*:

$$\underset{\theta}{\operatorname{argmin}} \sum_{n=1}^{N} \mathcal{L}(y_n, \hat{y}_n) + \Omega(\theta)$$

Outline

Prediction Models - I

- Linear Model
  - $\hat{y}_n = \theta_0 + \theta_1 x_{n,1} + \theta_2 x_{n,2} + \cdots + \theta_M x_{n,M} = \theta_0 + \sum_{m=1}^{M} \theta_m x_{n,m}$

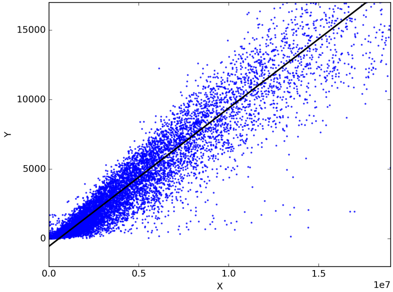


Figure 5 : Linear regression, $\theta = [-540, 0.001]$

Prediction Models - II

- Polynomial Regression
  - $\hat{y}_n = \theta_0 + \sum_{m=1}^{M} \theta_m x_{n,m} + \sum_{m=1}^{M} \sum_{m'=1}^{M} \theta_{m,m'} x_{n,m} x_{n,m'} + \dots$

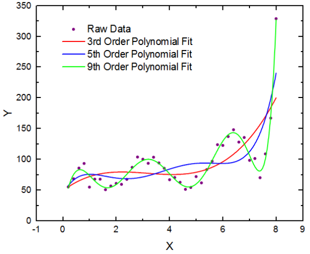


Figure 6 : Polynomial regression, Source: `www.originlab.com`

- Decision Trees
- Neural Networks

Decision Tree as a Prediction Model

A prediction model $\hat{y}_n := f(x_n, \theta)$ can be also a tree:

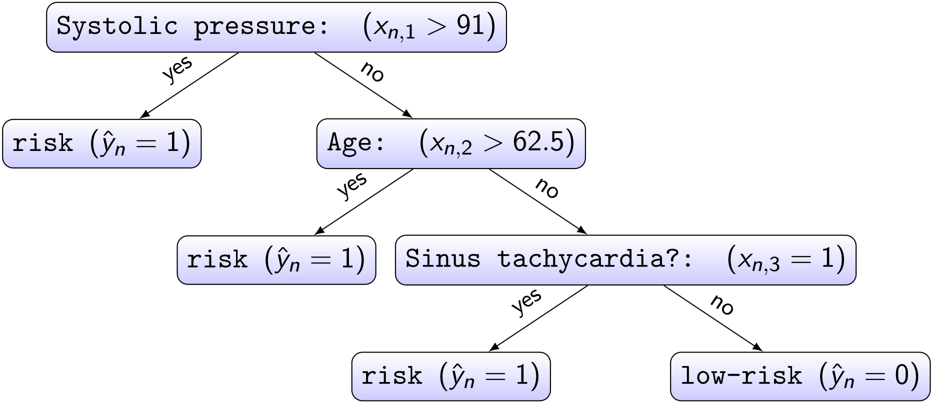
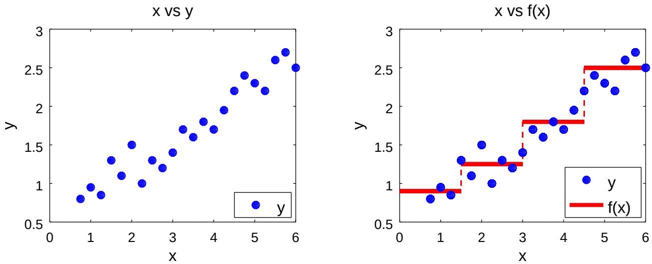
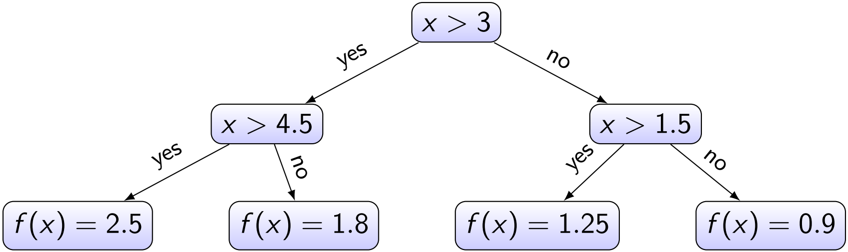


Figure 7 : San Diego Medical Center

Decision Tree as a Step-wise Function





Neural Network Model

- A neuron indexed i is a non-linear function $f_i(x, \theta_i)$
- If neuron i is connected to neuron j the model is $f_j(f_i(x, \theta_i), \theta_j)$

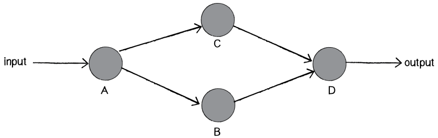


Figure 8 : One layer network, Courtesy of Shiffman 2010, The Nature of Code

$$\hat{y}_n := f_D(\theta_0 + \theta_{D1} f_C(f_A(x_n, \theta_A), \theta_C) + \theta_{D2} f_B(f_A(x_n, \theta_A), \theta_B))$$

Neural Network Regression

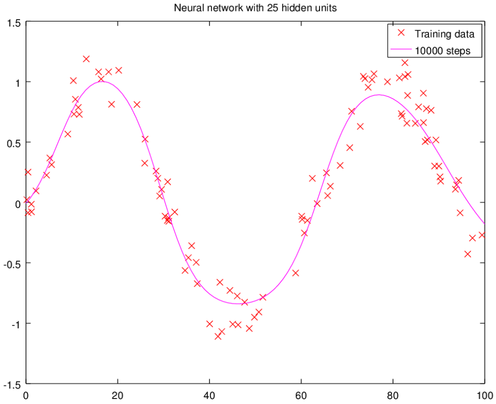


Figure 9 : Regression using Neural Network, Courtesy of `dungba.org`

Outline

Loss Functions

- Regression (target is real-values $y_n \in \mathbb{R}$)
  - Least-squares:

$$\mathcal{L}(y_n, \hat{y}_n) := (y_n - \hat{y}_n)^2$$

  - L1:

$$\mathcal{L}(y_n, \hat{y}_n) := |y_n - \hat{y}_n|$$

- Binary Classification $y_n \in \{0, 1\}$
  - Logistic loss:

$$\mathcal{L}(y_n, \hat{y}_n) := -y_n \log(\hat{y}_n) - (1 - y_n) \log(1 - \hat{y}_n)$$

  - Hinge loss:

$$\mathcal{L}(y_n, \hat{y}_n) := max(0, y_n \hat{y}_n)$$

Multi-class loss - Softmax

- Re-express targets $y_n \in \{1, \ldots, C\}$ as one-vs-all, i.e.

$$y_{n,c} := \begin{cases} 1 & y_n = C \\ 0 & y_n \neq C \end{cases}$$

- Learn model parameters per class $\theta \in \mathbb{R}^{C \times K}$
- Estimations expressed as probabilities among classes

$$\hat{y}_{n,c} = \frac{e^{f(x_n,\theta_c)}}{\sum_{q=1}^{C} e^{f(x_n,\theta_q)}}$$

- Logloss:

$$\mathcal{L}(y_{n,:}, \hat{y}_{n,:}) := -\sum_{c=1}^{C} y_{n,c} \log(\hat{y}_{n,c})$$

Gradient Descent

Find the optimal parameters $\theta^* \in \mathbb{R}^K$ that minimize an objective function $\mathcal{F}$, given data $\mathcal{D} \in \bigcup_{j=1}^{J} \mathcal{D}_j$, i.e.:

$$\theta^* \quad := \quad \operatorname*{argmin}_{\theta} \quad \mathcal{F}(\mathcal{D}, \theta)$$

Algorithm 1: Gradient Descent Optimization

Require: Data $\mathcal{D} \in \bigcup_{j=1}^{J} \mathcal{D}_j$, Learning rate $\eta \in \mathbb{R}^+$, Iterations $\mathcal{I} \in \mathbb{N}^+$

Ensure: $\theta \in \mathbb{R}^K$

1: $\theta \sim \mathcal{N}(0, \sigma^2)$

2: **for** $1, \ldots, \mathcal{I}$ **do**

3: $\quad \theta \leftarrow \theta - \eta \frac{\partial \mathcal{F}(\mathcal{D}, \theta)}{\partial \theta}$

4: **return** θ

Stochastic Gradient Descent

Divide the objective function according to J data partitions $\mathcal{D} \in \bigcup_{j=1}^{J} \mathcal{D}_j$

$$\mathcal{F}(\mathcal{D}, \theta) \;:=\; \sum_{j=1}^{J} \mathcal{F}(\mathcal{D}_j, \theta) \;:=\; \sum_{j=1}^{J} \mathcal{F}_j$$

Algorithm 2: Stochastic Gradient Descent Optimization

Require: Data $\mathcal{D} \in \bigcup_{j=1}^{J} \mathcal{D}_j$, Learning rate $\eta \in \mathbb{R}^+$, Iterations $\mathcal{I} \in \mathbb{N}^+$

Ensure: $\theta \in \mathbb{R}^K$

1: $\theta \sim \mathcal{N}(0, \sigma^2)$
2: **for** $1, \ldots, \mathcal{I}$ **do**
3:     **for each** $j \in \{1, \ldots, J\}$ *in random order* **do**
4:         $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{F}_j}{\partial \theta}$
5: **return** θ

Outline

Overfitting, Underfitting

- Underfitting (High model bias): Unable to capture complexity
- Overfitting (High model variance): Capturing noise

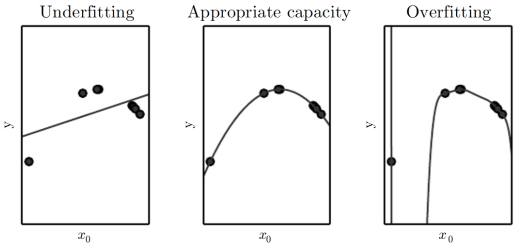


Figure 10 : Overfitting, Underfitting, Source: Goodfellow et al., 2016, Deep Learning

Capacity

- Expressiveness of a model
- Often expressed as the number of model parameters
- In Neural Networks is the number of neurons

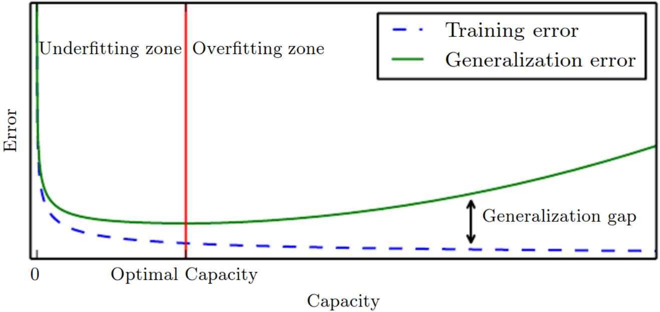


Figure 11 : Capacity, Source: Goodfellow et al., 2016, Deep Learning

Regularization

- Fights overfitting
- Penalize the parameter values

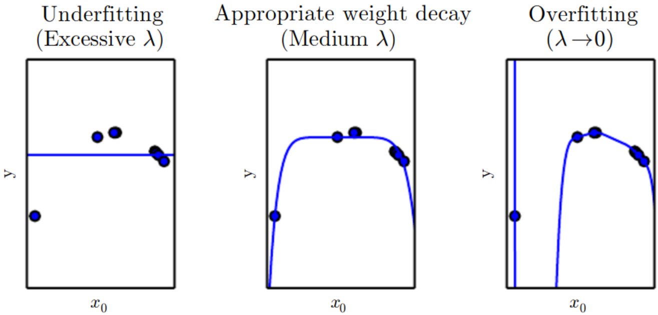


Figure 12 : Regularizing a polynomial regression, Source: Goodfellow et al., 2016, Deep Learning

Outline

Generative Model

- Considering a linear model

$$\hat{y} = \theta_0 + \sum_{m=1}^{M} \theta_m x_m$$

- Assume the error in predicting the ground truth y_n is normally distributed

$$\epsilon | x \sim \mathcal{N}(0, \sigma^2)$$

- In other words, the models generates estimations

$$\hat{y} \sim \mathcal{N}\left(\theta_0 + \sum_{m=1}^{M} \theta_m x_m, \sigma^2\right)$$

Maximum Likelihood Estimation

- Let $\hat{p}(y|x, \theta)$ be the probability density function for the target y given features x and parameters θ
- The likelihood of observing the target $y \in \mathbb{R}^N$ is

$$L(\theta) = \prod_{n=1}^{N} \hat{p}(y_n \,|\, x_n, \theta)$$

- What values of θ make our observed target more likely to occur?
- Aim: **Estimate** the θ-s which **maximize** the **likelihood**.

Maximum Likelihood Estimation - II

- Remember

$$\log(a\, b) = \log(a) + \log(b)$$
$$\max_\theta\ g(\theta) = \max_\theta\ \log(g(\theta))$$

- Taking the logarithm of the likelihood

$$\log \prod_{n=1}^{N} \hat{p}(y_n \mid \theta) = \sum_{n=1}^{N} log(\hat{p}(y_n \mid \theta))$$

- Assuming $\hat{p}$ is normally distributed we derive the log-likelihood:

$$\log L(\theta) = \sum_{n=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\hat{\sigma}}} e^{-\frac{(y_n - \hat{y}_n)^2}{2\hat{\sigma}^2}}\right)$$

Maximum Likelihood Estimation - III

- Deriving further:

$$
\begin{aligned}
\log L(\theta) &= \sum_{n=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\hat{\sigma}}} e^{-\frac{(y_n - \hat{y}_n)^2}{2\hat{\sigma}^2}}\right) \\
&= \sum_{n=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\hat{\sigma}}}\right) + \log\left(e^{-\frac{(y_n - \hat{y}_n)^2}{2\hat{\sigma}^2}}\right)
\end{aligned}
$$

- Omitting the constant term above with respect to the parameters θ:

$$
\begin{aligned}
\operatorname*{argmax}_{\theta} \log L(\theta) &\approx \operatorname*{argmax}_{\theta} \frac{1}{2\hat{\sigma}^2} \sum_{n=1}^{N} -\left(y_n - \left(\theta_0 + \sum_{m=1}^{M} \theta_m x_m\right)\right)^2 \\
&\approx \operatorname*{argmin}_{\theta} \sum_{n=1}^{N} \left(y_n - \left(\theta_0 + \sum_{m=1}^{M} \theta_m x_m\right)\right)^2
\end{aligned}
$$