Deep Learning

1. Supervised Learning (Review 1)

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

Tue. 21.4.	(1)	1. Supervised Learning (Review 1)
Tue. 28.4.	(2)	2. Neural Networks (Review 2)
Tue. 5.5.	(3)	3. Regularization
Tue. 12.5.	(4)	4. Optimization
Tue. 19.5.	(5)	5. Convolutional Neural Networks
Tue. 26.5.	(6)	6. Recurrent Neural Networks
Tue. 2.6.	—	— *Pentecoste Break* —
Tue. 9.6.	(7)	7. Autoencoders
Tue. 16.6.	(8)	8. Generative Adversarial Networks
Tue. 23.6.	(9)	9. Recent Advances
Tue. 30.6.	(10)	10. Engineering Deep Learning Models
Tue. 7.7.	(11)	tbd.
Tue. 14.7.	(12)	Q & A

Outline

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*

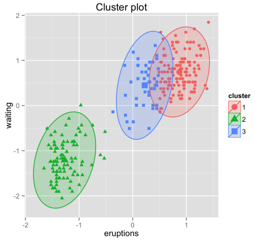


Figure 2: Clustering illustration, Courtesy of `www.sthda.com`

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`

Example: Covid 19 Early Warning System

- Physicians aim to develop an early warning system for Covid 19 infections that predicts if a person is likely to have caught Covid 19.
- They measure for many patients
  - their **temperature** over the day,
  - the number of other humans they have been in **contact** with over the day (measured by the number of smartphones that could be sensed via bluetooth),
  - self assessment for **headaches**, **lowered taste** and **lowered smell**,
  - outcome of a **Covid 19 viurs test** based on a blood sample.

Note: This is a fictitious use case.

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 $x^{\text{test}} \in \mathbb{R}^M$?
- Need a function $\hat{y}$ that predicts the target: $\hat{y}(x)$.
  - called **prediction model**
- When is such a function $\hat{y}$ a good function?
  - compare the observed ground truth y_n with the predictions $\hat{y}_n := \hat{y}(x_n)$
  - the closer they are, the better the model
  - How should we measure "close" ?

Difference to Ground Truth

- The quality of a prediction model $\hat{y}(x)$
  - Difference between the estimated target $\hat{y}$ and ground-truth target y
  - Defined by a function $\ell(y, \hat{y}) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ called **loss function**, e.g.,

$$\ell(y, \hat{y}) := (y - \hat{y})^2$$

  - The loss has to be minimized w.r.t. the parameters

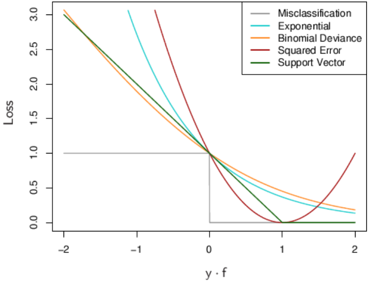


Figure 4: Loss types, (Hastie et al., 2009, The Elements of Statistical Learning)

The Supervised Learning Problem

Given

- a set $\mathcal{D}^{\text{train}} := \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \subseteq \mathbb{R}^M \times \mathbb{R}^O$ called **training data**, and
- a function $\ell : \mathbb{R}^O \times \mathbb{R}^O \to \mathbb{R}$ called **pairwise loss function**,

we want to estimate a function

$$\hat{y} : \mathbb{R}^M \to \mathbb{R}^O$$

called **model** s.t. for a set $\mathcal{D}^{\text{test}} \subseteq \mathbb{R}^M \times \mathbb{R}^O$ called **test set** the **test error**

$$\text{err}(\hat{y}; \mathcal{D}^{\text{test}}) := \frac{1}{|\mathcal{D}^{\text{test}}|} \sum_{(x,y) \in \mathcal{D}^{\text{test}}} \ell(y, \hat{y}(x))$$

is minimal.

Note: $\mathcal{D}^{\text{test}}$ has (i) to be from the same data generating process and (ii) not to be available during training.

The Supervised Learning Problem

- **classification**: $y_n \in \{0,1\}^O$ and there is exact one o with $y_{n,o} = 1$, otherwise **regression**.

- $x := (x_1 x_2 \ldots x_N)^T \in \mathbb{R}^{N \times M}$ **predictors** (aka **features**, **covariates**, **inputs**)
- $y := (y_1 y_2 \ldots y_N)^T \in \mathbb{R}^{N \times O}$ **targets** (aka **outputs**)

Loss Functions

- Regression (target is a real scalar $y_n \in \mathbb{R}$)
  - **quadratic loss** (aka **L2 loss**):

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

  - **absolute loss** (aka **L1 loss**):

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

- Binary Classification $y_n \in \{0, 1\}$
  - **logistic loss** (aka **binary logloss**):

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

  - **hinge loss**:

$$\ell(y_n, \hat{y}_n) := 2 \max(0, y_n + \hat{y}_n - 2 y_n \hat{y}_n)$$
$$\ell(y_n, \hat{y}_n) := \max(0, 1 - y_n \hat{y}_n), \quad \text{if } y_n, \hat{y}_n \in \{-1, +1\}$$

Multi-class logloss

- Re-express targets $y_n \in \{1, \ldots, C\}$ as **binary indicators** (aka **one-hot-encoding**) $y_n^{\text{new}} \in \{0,1\}^C$, i.e.

$$y_{n,c}^{\text{new}} := \begin{cases} 1, & \text{if } y_n = c \\ 0, & \text{else} \end{cases}$$

- **logloss** (aka **cross entropy**):

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

Example: Covid 19 Early Warning System

- Physicians aim to develop an early warning system for Covid 19 infections that predicts if a person is likely to have caught Covid 19.
- They measure for many patients
  - their **temperature** over the day,
  - the number of other humans they have been in **contact** with over the day (measured by the number of smartphones that could be sensed via bluetooth),
  - self assessment for **headaches**, **lowered taste** and **lowered smell**,
  - outcome of a **Covid 19 virus test** based on a blood sample.

Note: This is a fictitious use case.

Outline

Model Parameters

- How to find a good function / model $\hat{y}$?
  1. Parametrize functions through parameters θ as $\hat{y}(x; \theta)$ (**model class**, aka **type of model**)
  2. Find values for the parameters θ such that the model fits the training data well, i.e., has a low loss (**learning**)
     $\rightsquigarrow$ optimization problem w.r.t. the parameters.

Prediction Models - I

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

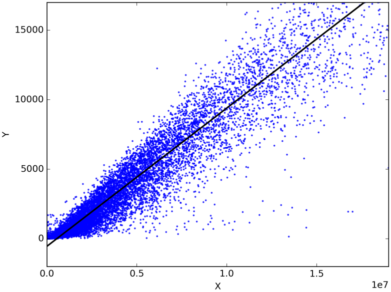
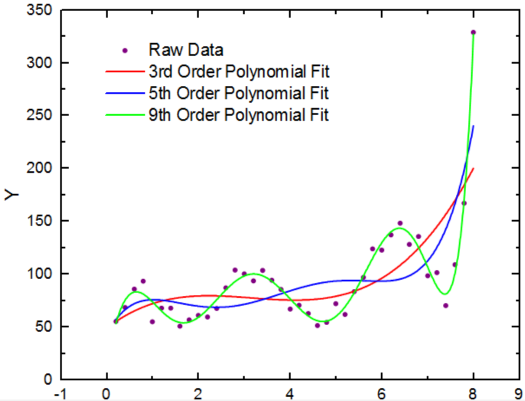


Figure 5: Linear regression, $\theta = [-540, 0.001]$

Prediction Models - II

- Polynomial Regression

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



Decision Tree as a Prediction Model

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

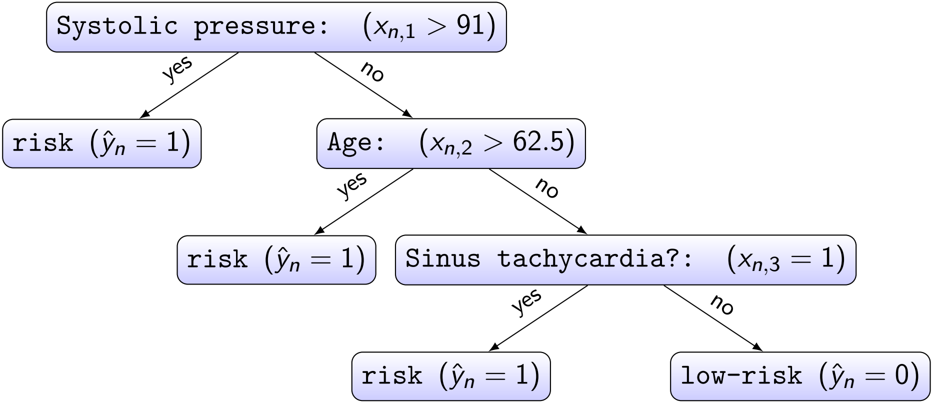
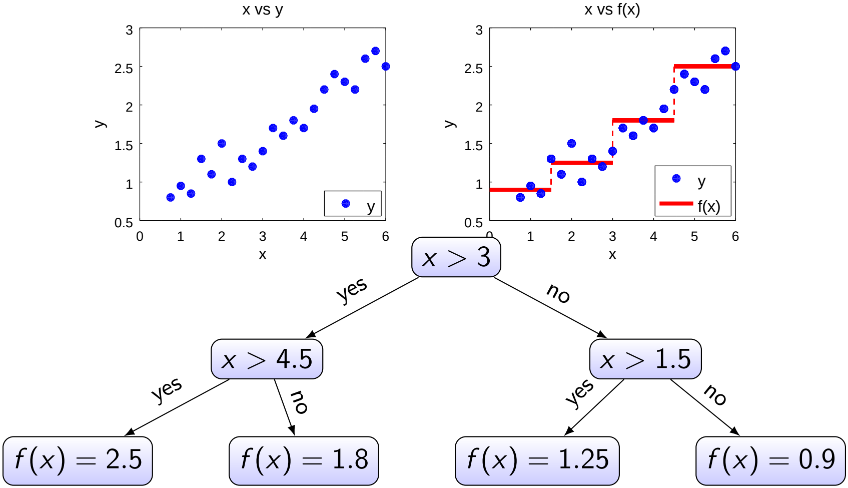


Figure 7: San Diego Medical Center

Decision Tree as a Step-wise Function



Neural Network Model

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

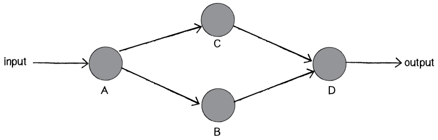


Figure 8: One layer network, Courtesy of Shiffman 2010, The Nature of Code

$$\hat{y}_n := g_D(\theta_0 + \theta_{D,1} g_C(g_A(x_n, \theta_A), \theta_C) + \theta_{D,2} g_B(g_A(x_n, \theta_A), \theta_B))$$

Neural Network Regression

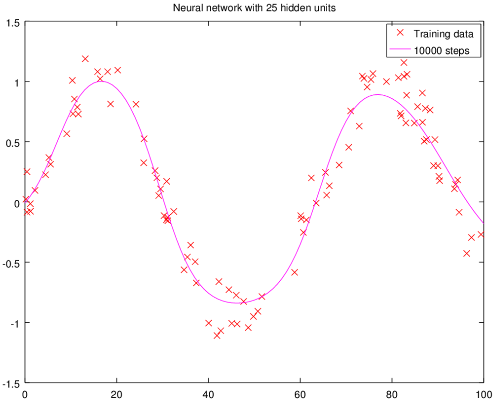


Figure 9: Regression using Neural Network, Courtesy of `dungba.org`

Outline

Gradient Descent (basic version)

learn-gd($f : \mathbb{R}^P \rightarrow \mathbb{R}, \mathcal{D}^{\text{train}}, \sigma^2 \in \mathbb{R}^+, \mu, i_{\max} \in \mathbb{N}$):
  $\theta \sim \mathcal{N}(0, \sigma^2)$
  for $i = 1, \ldots, i_{\max}$:
    $\theta := \theta - \mu_i \cdot \nabla f(\theta; \mathcal{D}^{\text{train}})$
  return θ

- f objective function (as function in the parameters θ)
- $\mathcal{D}^{\text{train}}$ training data
- σ^2 parameter initialization variance
- μ step size schedule
- $i_{\max}$ maximal number of iterations

Stochastic Gradient Descent (basic version)

learn-sgd$(f : \mathbb{R}^P \rightarrow \mathbb{R}, \mathcal{D}^{\text{train}}, \sigma^2 \in \mathbb{R}^+, \mu, i_{\max} \in \mathbb{N}, B \in \mathbb{N})$:
  $\theta \sim \mathcal{N}(0, \sigma^2)$
  for $i = 1, \ldots, i_{\max}$:
    $\mathcal{D}^{\text{batch}} \sim \mathcal{D}^{\text{train}}$ draw B instances uniformly at random
    $\theta := \theta - \mu_i \cdot \nabla f(\theta; \mathcal{D}^{\text{batch}})$
  return θ

f	objective function (as function in the parameters θ)
$\mathcal{D}^{\text{train}}$	training data
σ^2	parameter initialization variance
μ	step size schedule
$i_{\max}$	maximal number of iterations
B	minibatch size

Outline

Overfitting, Underfitting

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

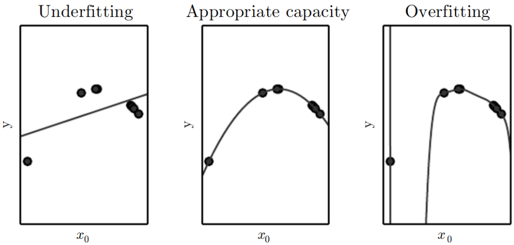


Figure 10: Overfitting, Underfitting, Source: Goodfellow et al., 2016, Deep Learning

Capacity

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

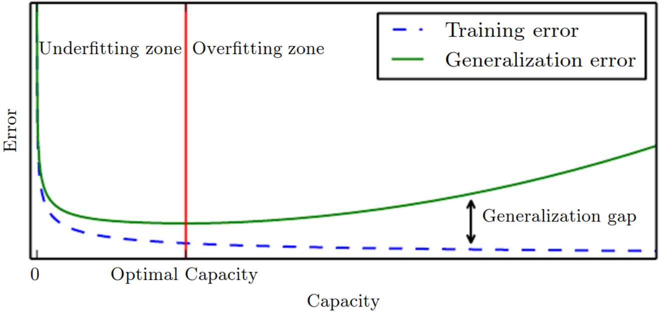


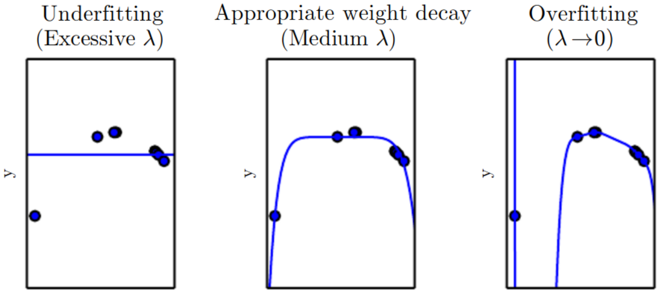
Figure 11: Capacity, Source: Goodfellow et al., 2016, Deep Learning

Regularization

- Fights overfitting
- e.g., combine loss and a penality for large parameter values into an **objective function** f:

$$f(\theta; x, y) := \ell(y, \hat{y}(x)) + \lambda\Omega(\theta), \quad \Omega(\theta) := \sum_{p=1}^{P} \theta_p^2$$

- minimze objective function f (not just the loss)



Outline

Bayesian Regression

- Consider a regression model that does not only yield
  - the most likely target values $\hat{y}_n := \hat{y}(x_n)$, but also
  - how the model believes this value could vary across different observations of x_n (its own uncertainty)
- Considering a linear model:

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

- Assume the uncertainty ϵ is normally distributed

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

- In other words, the model estimates not just a single value (point estimation), but a whole distribution of possible values:

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

Maximum Likelihood Estimation

- Let $p(y|x,\theta)$ 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

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

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

Maximum Likelihood Estimation - II

- Remember

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

- Taking the logarithm of the likelihood

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

- Assuming p is normally distributed we derive the log-likelihood:

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

Maximum Likelihood Estimation - III

- Deriving further:

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

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

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

$$\arg\max_{\theta} \log L(\theta) \approx \arg\max_{\theta} \frac{1}{2\hat{\sigma}^2} \sum_{n=1}^{N} - \left(y_n - \left(\theta_0 + \sum_{m=1}^{M} \theta_m x_m \right) \right)^2$$

$$\approx \arg\min_{\theta} \sum_{n=1}^{N} \left(y_n - \left(\theta_0 + \sum_{m=1}^{M} \theta_m x_m \right) \right)^2$$

Outline

Character of the Lecture

This is an advanced lecture:

- I will assume good knowledge of Machine Learning I.
  - but I will review major concepts in the first two sessions.
- Slides will contain major keywords, not the full story.
- For the full story, you need to read the referenced chapters in one of the books.

Syllabus

Tue. 21.4.	(1)	1. Supervised Learning (Review 1)
Tue. 28.4.	(2)	2. Neural Networks (Review 2)
Tue. 5.5.	(3)	3. Regularization
Tue. 12.5.	(4)	4. Optimization
Tue. 19.5.	(5)	5. Convolutional Neural Networks
Tue. 26.5.	(6)	6. Recurrent Neural Networks
Tue. 2.6.	—	— *Pentecoste Break* —
Tue. 9.6.	(7)	7. Autoencoders
Tue. 16.6.	(8)	8. Generative Adversarial Networks
Tue. 23.6.	(9)	9. Recent Advances
Tue. 30.6.	(10)	10. Engineering Deep Learning Models
Tue. 7.7.	(11)	tbd.
Tue. 14.7.	(12)	Q & A

Exercises and Tutorials

- There will be a weekly sheet with 2 exercises handed out **each Wednessday**.

- Solutions to the exercises can be submitted until **next Wednessday noon, 12pm**

- Tutorials **Friday 12pm-2pm**, 1st tutorial next week, Fr. 24.04.

    - Plagiarism is strictly prohibited and leads to expulsion from the program.

- Register in Learnweb (Assignment submission) and LSF (Providing grades)

Deep Learning Exam

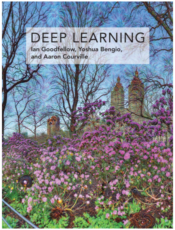
- Grade is dependent on two things:
  - Tutorials - 50%
  - End Exam (under video surveilance) - 50%
  - To pass - min 20% through tutorials, and 20% through exam and 50% in total

Exam and Credit Points

- The course gives 6 ECTS (2+2 SWS).
- The course can be used in
  - International Master in Data Analytics (mandatory)
  - IMIT MSc. / Informatik / Gebiet KI & ML
  - Wirtschaftsinformatik MSc / Informatik / Gebiet KI & ML & Wirtschaftsinformatik MSc / Wirtschaftsinformatik / Gebiet BI
  - as well as in all IT BSc programs.

Some Books

- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The Mit Press, Cambridge, Massachusetts, November 2016. ISBN 978-0-262-03561-3 www.deeplearningbook.org



Summary (1/2)

- **Deep Learning** aims to build machine learning models for a vast set of problems by constructing **deep neural networks**, i.e., neural networks with many layers (in the dozens and hundreds).

- **Supervised Prediction Problems** ask for a **model** $\hat{y}$ that predicts **targets** y for any **predictors** x, based on **data** on observed predictor/target pairs (x_n, y_n), s.t. for new **test data** (x, y) the **loss** between the true target y and the predicted target $\hat{y}(x)$ is minimal.

- There exist many different **types of models** to accomplish supervised prediction, i.e.,
  - linear models,
  - polynomimal models,
  - kernel models and support vector machines,
  - neural networks

  that can be fit to data by setting their **model parameters** (aka **weights**).

Summary (2/2)

- **Learning algorithms** are minimization algorithms that minimize a loss function of a model on the training data to fit the model to the data, e.g,
  - gradient descent,
  - stochastic gradient descent
- To **generalize to new data**, models should not fit the training data too closely (memorization), but pick up only the regularities / the signal of the data, not the noise, e.g., by
  - **structural regularization**: have only a limited number of model parameters.
  - **L2-regularization**: force the model parameters to be small.

Further Readings

- Goodfellow et al. 2016, ch. 5
- lecture Machine Learning, chapters 0, A.1, A.2 and A.3.

Acknowledgement: An earlier version of the slides for this lecture have been written by my former postdoc Dr Josif Grabocka.

References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The Mit Press, Cambridge, Massachusetts, November 2016. ISBN 978-0-262-03561-3.