EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATHEMATISCH-
NATURWISSENSCHAFTLICHE FAKULTÄT

Deep Neural Networks

Chapter 5: Machine Learning Basics

5.1 Learning Algorithms

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- A machine learning (ML) algorithm is an algorithm that is able to learn from data.
- A definition (Mitchell 1997): A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
- Machine learning is interesting because understanding machine learning entails developing our understanding of the principles that underlie intelligence.
- Machine learning tasks are usually described in terms of how the machine learning system should process an example, a vector $\boldsymbol{x} \in \mathbb{R}^n$ of features x_i.

5.1.1 Machine Learning Tasks

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Some of the most important ML tasks are:

- Classification: here, the computer program is asked to specify which of k categories some input belongs to.
  To solve this task, the learning algorithm is usually asked to produce a function $f : \mathbb{R}^n \rightarrow \{1, \ldots, k\}$.
  When $y = f(\boldsymbol{x})$, the model assigns an input described by vector $\boldsymbol{x}$ to a category identified by numeric code y.
- An example classification task is object recognition.
- Modern object recognition is best accomplished with deep learning. (Krizhevsky *et al.*, 2012; Ioffe and Szegedy, 2015)

Machine Learning Tasks

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Classification with missing inputs: the computer program is not guaranteed that every measurement in its input vector will always be provided.
- When some of the inputs may be missing, the learning algorithm must learn a *set* of functions. Each function corresponds to classifying x with a different subset of its inputs missing.
- This kind of situation arises frequently in medical diagnosis, because many kinds of medical tests are expensive or invasive.
- One way to efficiently define such a large set of functions is to learn a probability distribution over all of the relevant variables, then solve the classification task by marginalizing out the missing variables.

Machine Learning Tasks

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Regression: In this type of task, the computer program is asked to predict a numerical value given some input. To solve this task, the learning algorithm is asked to output a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$
- This type of task is similar to classification, except that the format of the output is different.
- An example is the prediction of the expected claim amount that an insured person will make, or the prediction of future prices of securities.

Machine Learning Tasks

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Transcription: In this type of task, the machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe it into discrete, textual form.
  For example, in optical character recognition, the computer is shown a photograph with an image of text and is asked to return this text as a sequence of characters.

↳ difference: already structured

- Machine translation: Here the input already consists of a sequence of symbols in some language, and the computer program must convert this into a sequence of symbols in another language.

Machine Learning Tasks

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Structured output tasks: These tasks involve any task $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ where the output $\boldsymbol{y} \in \mathbb{R}^m$ is a vector (or other data structure containing multiple values) with important relationships between the different elements.
- One example is parsing - mapping a natural language sentence into a tree that describes its grammatical structure and tagging nodes of the trees as being verbs, nouns, or adverbs, and so on.
- Another example is pixel-wise segmentation of images, where the computer program assigns every pixel in an image to a specific category. x categorization
- (This category is broad and subsumes the transcription and translation tasks described above, and many more)

Machine Learning Tasks

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Anomaly detection: Here the computer program sifts through a set of events or objects, and flags some of them as being unusual or atypical. An example of an anomaly detection task is credit card fraud detection.
- Imputation of missing values: Here the ML algorithm is given a new example $\boldsymbol{x} \in \mathbb{R}^n$, but with some entries x_i of $\boldsymbol{x}$ missing. The algorithm must provide a prediction of the values of the missing entries.
- Denoising: Here the ML algorithm is given a *corrupted example* $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ obtained by an unknown corruption process. The learner must predict the clean example $\boldsymbol{x} \in \mathbb{R}^n$ from its corrupted version $\tilde{\boldsymbol{x}}$, or more generally predict the conditional probability distribution $p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$.

Machine Learning Tasks

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Density estimation: Here the ML algorithm is asked to learn a function $p : \mathbb{R}^n \rightarrow \mathbb{R}$, where $p(\boldsymbol{x})$ is a probability density function (if $\boldsymbol{x}$ is continuous) or a probability mass function (if $\boldsymbol{x}$ is discrete) on the space of examples.
- The algorithm needs to learn the structure of the data it has seen. It must know where examples cluster tightly.
- Many tasks described above require the ML algorithm to capture the structure of $p(\boldsymbol{x})$ implicitly. If density estimation yields a probability distribution $p(\boldsymbol{x})$, we can use it to solve the missing value imputation task: If x_i is missing and all of the other values $\boldsymbol{x}_{-i}$, are given, then the distribution over it is given by $p(x_i \mid \boldsymbol{x}_{-i})$.
- In practice, often the required operations on $p(\boldsymbol{x})$ for density estimation are computationally intractable.

5.1.2 The Performance Measure P

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- We often measure the accuracy of the model. This is the proportion of examples for which the model produces the correct output.
- We can equivalently measure the error rate, the proportion of examples for which the model produces an incorrect output.
- We often refer to the error rate as the expected 0-1 loss.
- Usually we are interested in how well the ML algorithm performs on data that it has not seen before. We therefore evaluate these performance measures using a test set of data that is separate from the data used for training the machine learning system.

5.1.3 The Experience, E

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Machine learning algorithms can be broadly categorized as unsupervised or supervised by what kind of experience they have during the learning process
- Unsupervised learning algorithms see a dataset with many features, then learn useful properties of the structure of this dataset. In deep learning, we usually want to learn the entire PD that generated a dataset, whether explicitly as in density estimation or implicitly for tasks like synthesis or denoising.
- Supervised learning algorithms experience a dataset containing features, but each example is also associated with a label or target. For example, the Iris dataset is annotated with one of the 3 species of each iris plant.

Supervised vs. Unsupervised Learning

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Unsupervised learning involves observing several examples of a random vector x, and attempting to implicitly or explicitly learn the probability distribution $p(\mathrm{x})$, or some interesting properties of that distribution.
- Supervised learning involves observing several examples of a random vector x and an associated value or vector y, and learning to predict y from x, usually by estimating $p(y|\mathrm{x})$.
- The term supervised learning originates from the view that the target y is provided by an instructor or teacher who shows the machine learning system what to do.
- In unsupervised learning, there is no instructor, and the algorithm must make sense of the data itself.

Supervised vs. Unsupervised Learning

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Many machine learning technologies can be used to perform both supervised and unsupervised learning.
- E.g., the chain rule of probability states that for a vector $\mathrm{x} \in \mathbb{R}^n$, the joint distribution can be decomposed as

$$p(\mathrm{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

  This means we can solve the unsupervised problem of modeling $p(\mathrm{x})$ by splitting it into n supervised problems.
- Alternatively, we can solve the supervised learning problem of learning $p(y|\mathrm{x})$ by using unsupervised learning to learn the joint distribution $p(\mathrm{x}, y)$ and inferring

$$p(y \mid \mathrm{x}) = \frac{p(\mathrm{x}, y)}{\sum_{y'} p(\mathrm{x}, y')}$$

- Reinforcement learning algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences.
- We will not treat them in this class. See Sutton and Barto (1998)

- A design matrix is a common way to describe a dataset. It is a matrix containing a different example in each row. Each column of the matrix holds a different feature. For instance, the Iris dataset contains 150 examples with four features each. This means we can represent the dataset with a design matrix $\boldsymbol{X} \in \mathbb{R}^{150\times 4}$, where $X_{i,1}$ is the first feature of plant i, etc.

5.1.4 Example: Linear Regression

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- We present an example of a simple machine learning algorithm: linear regression.
- The goal is to build a system that can take a vector $\boldsymbol{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}$ as output.
- In linear regression, the output is a linear function of the input. Let $\hat{y}$ be the value that our model predicts for y. We define the output to be

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x}$$

  where $\boldsymbol{w} \in \mathbb{R}^n$ is a vector of parameters.
- Parameters are values that control the behavior of the system. We can think of w as a set of weights that determine how each feature affects the prediction.

Example: Linear Regression

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- If w_i is positive, then increasing feature x_i increases the value of our prediction $\hat{y}$.
- If w_i is negative, then increasing feature x_i decreases the value of our prediction $\hat{y}$.
- If $|w_i|$ is large, then feature x_i has a large effect on $\hat{y}$.
- If $w_i = 0$, feature x_i it has no effect on the prediction.

- So our task T is to predict y from $\boldsymbol{x}$ by outputting $\hat{y} = \boldsymbol{w}^T \boldsymbol{x}$.
- We now must define our performance measure P.
- Suppose we have a design matrix $\boldsymbol{X}^{(\text{test})}$ of m example inputs that are not used for training, only for testing. We also have a vector $\boldsymbol{y}^{(\text{test})}$ of regression targets.

Example: Linear Regression

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- One way of measuring the performance of the model is to compute the mean squared error of the model on the test set.

$$MSE_{test} = \frac{1}{m}\sum_i (\hat{\boldsymbol{y}}^{(test)} - \boldsymbol{y}^{(test)})_i^2$$

- This error measure decreases to 0 when $\hat{\boldsymbol{y}}^{(test)} = \boldsymbol{y}^{(test)}$. We also see that

$$MSE_{test} = \frac{1}{m}\left\|(\hat{\boldsymbol{y}}^{(test)} - \boldsymbol{y}^{(test)})\right\|_2^2$$

- So the error increases whenever the Euclidian distance between the predictions and the targets increases.

Example: Linear Regression

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- We now must design an algorithm that will improve the weights w in a way that reduces MSE_{test} when the algorithm learns by observing a training set $\left(\boldsymbol{X}^{(train)}, \boldsymbol{y}^{(train)}\right)$
- One way to do this is to minimize the mean squared error on the training set, $\text{MSE}_{\text{train}}$.
- To minimize $\text{MSE}_{\text{train}}$, we can simply solve for where its gradient is **0**, by the following derivations:

$$\nabla_{\boldsymbol{w}} MSE_{train} = 0$$

$$\nabla_{\boldsymbol{w}} \frac{1}{m} \left\| \hat{\boldsymbol{y}}^{(train)} - \boldsymbol{y}^{(train)} \right\|_2^2 = 0$$

$$\frac{1}{m} \nabla_{\boldsymbol{w}} \left\| \boldsymbol{X}^{(train)} \boldsymbol{w} - \boldsymbol{y}^{(train)} \right\|_2^2 = 0$$

Example: Linear Regression

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- By writing the Euclidian norm as vector product we get

$$\nabla_w \left((\boldsymbol{X}^{(train)} \boldsymbol{w} - y^{(train)})^T (\boldsymbol{X}^{(train)} \boldsymbol{w} - y^{(train)}) \right) = 0$$

$$\nabla_w (\boldsymbol{w}^T \boldsymbol{X}^{(train)T} \boldsymbol{X}^{(train)} \boldsymbol{w} - 2\boldsymbol{w}^T \boldsymbol{X}^{(train)T} y^{(train)} + y^{(train)T} y^{(train)}) = 0$$

$$2\boldsymbol{X}^{(train)T} \boldsymbol{X}^{(train)} \boldsymbol{w} - 2\boldsymbol{X}^{(train)T} y^{(train)} = 0$$

$$\boldsymbol{X}^{(train)T} \boldsymbol{X}^{(train)} \boldsymbol{w} = \boldsymbol{X}^{(train)T} y^{(train)}$$

$$\boldsymbol{w} = (\boldsymbol{X}^{(train)T} \boldsymbol{X}^{(train)})^{-1} \boldsymbol{X}^{(train)T} y^{(train)}$$

- The system of equations whose solution is above is known as the normal equations. See also Fig. 5.1

A Linear Regression Problem

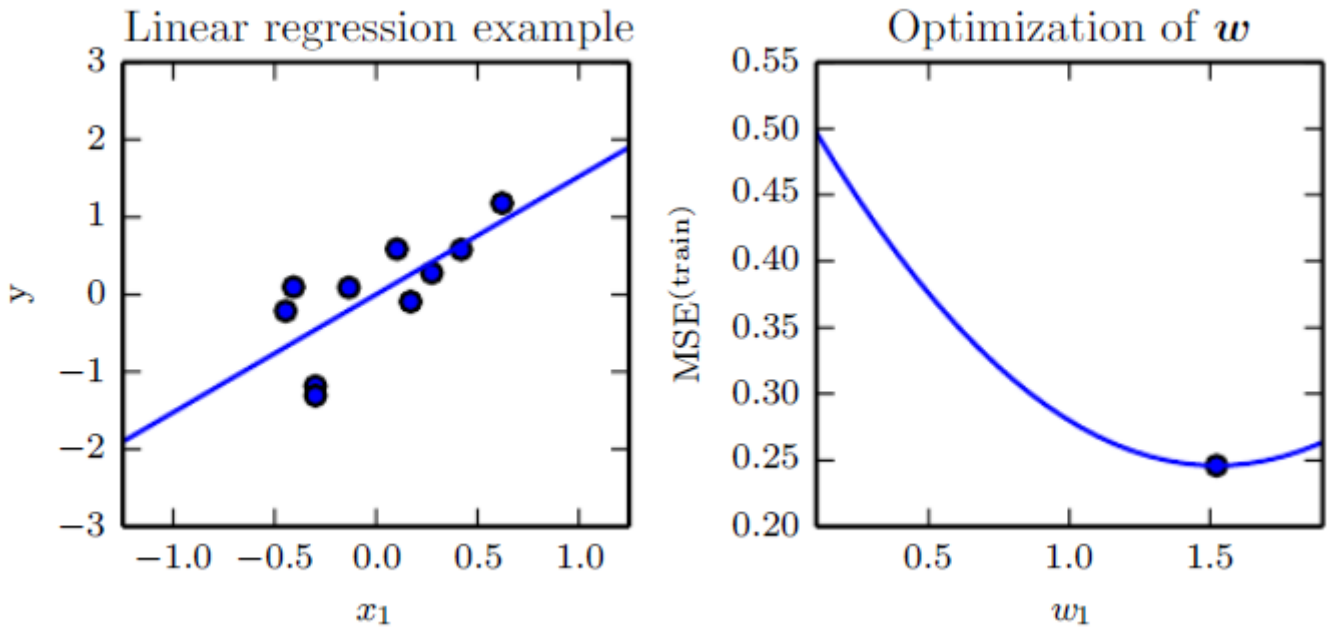


Fig. 5.1: A linear regression problem with a training set consisting of ten data points with one feature each. Therefore, the weight vector $\boldsymbol{w}$ contains only a single parameter to learn, w_1. *(Left)* Linear regression learns to set w_1 such that the line $y = w_1 x$ comes as close as possible to passing through all the training points. *(Right)* The plotted point indicates the value of w_1 found by the normal equations, which minimizes the mean squared error on the training set.

Example: Linear Regression

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- The term linear regression is often used to refer to a model with one additional parameter - an intercept term b. In this model

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b$$

  so the mapping from parameters to predictions is still a linear function, but the mapping from features to predictions is now an affine function. Now the plot of the model's predictions still looks like a line, but it need not pass through the origin.
- Instead of adding the bias parameter b, one can continue to use the model with only weights but augment $\boldsymbol{x}$ with an extra entry x_0 that is always set to 1. Its weight w_0 then plays the role of the bias parameter.

5.2 Capacity, Overfitting, Underfitting

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- The central challenge in machine learning is that we must perform well on *new, previously unseen* inputs - not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization.
- Problem: We can measure and minimize the training error, but actually want the test error to be low.
- We typically estimate the generalization error of a ML model by measuring its performance on a test set of examples collected separately from the training set.

Capacity, Overfitting, Underfitting

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- In the linear regression example, we trained the model by minimizing the training error,

$$\frac{1}{m^{(train)}} \left\| \boldsymbol{X}^{(train)} \boldsymbol{w} - y^{(train)} \right\|_2^2$$

but we actually care about the test error

$$\frac{1}{m^{(test)}} \left\| \boldsymbol{X}^{(test)} \boldsymbol{w} - y^{(test)} \right\|_2^2$$

- The field of statistical learning theory provides answers, how training and test error are related. For this we need to make some assumptions.
- We typically make the i.i.d. assumptions (see next page).

Data Generating Process

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- I.i.d. assumptions: The examples in each dataset are independent from each other, and the training set and test set are identically distributed, drawn from the same probability distribution, p_{data}.
- The expected training error of a randomly selected model is equal to the expected test error of that model.
- But in an ML algorithm, we sample the training set, then use it to optimize the parameters w to reduce the training set error, then sample the test set. Under this process, $MSE_{test} \geq MSE_{train}$.
- A good ML algorithm needs to:
  1. Make the training error small.
  2. Make the gap between training and test error small.

Capacity, Overfitting, Underfitting

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- These two factors correspond to the two central challenges in machine learning: underfitting and overfitting .
- Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set.
- Overfitting occurs when the gap between the training error and test error is too large.
- We can control overfitting or underfitting by altering a model's capacity. (approx: # of adaptable weights, layers or what functions e.g. non-linear it can model)
  - Models with low capacity may not fit the training set.
  - Models with high capacity can overfit by memorizing properties of the training set that are unsuitable on the test set.

Choosing the Hypothesis Space

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- One way to control the capacity of a learning algorithm is by choosing its hypothesis space, the set of functions that the learning algorithm is allowed to select as being the solution.
- In linear regression, we use a linear polynomial, with prediction

$$\hat{y} = wx + b$$

- By introducing a quadratic term, we allow this model:

$$\hat{y} = w_2 x^2 + w_1 x + b$$

- This model implements a quadratic function of its *input*, but the output is still a linear function of the *parameters.*
- We could use an n-order polynomial $\hat{y} = b + \sum_{i=1}^{n} w_i x^i$

Capacity, Overfitting, Underfitting

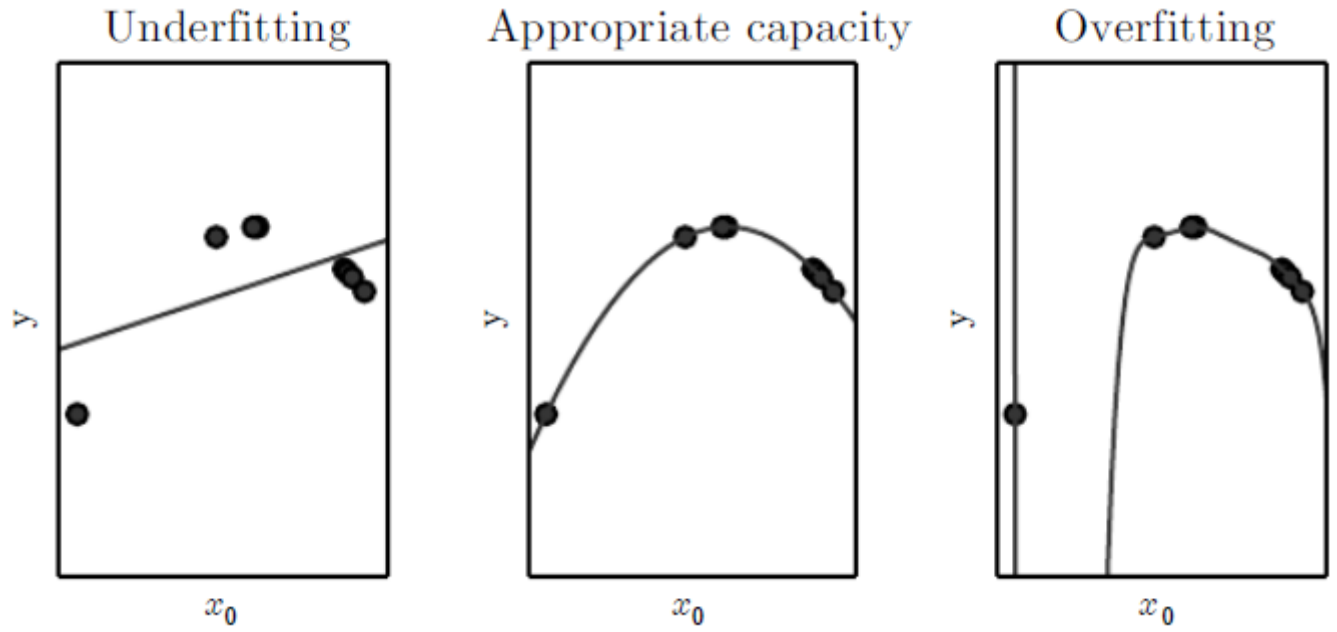


Fig. 5.2: Three models fit to this training set. The training data was generated by a quadratic function. *(Left)* A linear function fit to the data suffers from underfitting *(Center)* A quadratic function fit to the data generalizes well. *(Right)* A polynomial of degree 9 fit to the data suffers from overfitting. We used the pseudoinverse to solve the underdetermined normal equations. The solution passes through all of the training points exactly, overshoots strongly.

Representational vs Effective Capacity

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- The representational capacity of the model is determined by its function space. In many cases a learning algorithm does not find the optimal function, but one function with a local minimum of the training error. → how complex problems can be modeled
- This means that a learning algorithm's effective capacity may be less than the representational capacity of the model family. what it can really learn
- A principle of parsimony that is now widely known is Occam's razor (c. 1287-1347): among competing hypotheses that explain known observations equally well, one should choose the "simplest" one.

Typical Relation of Capacity and Error

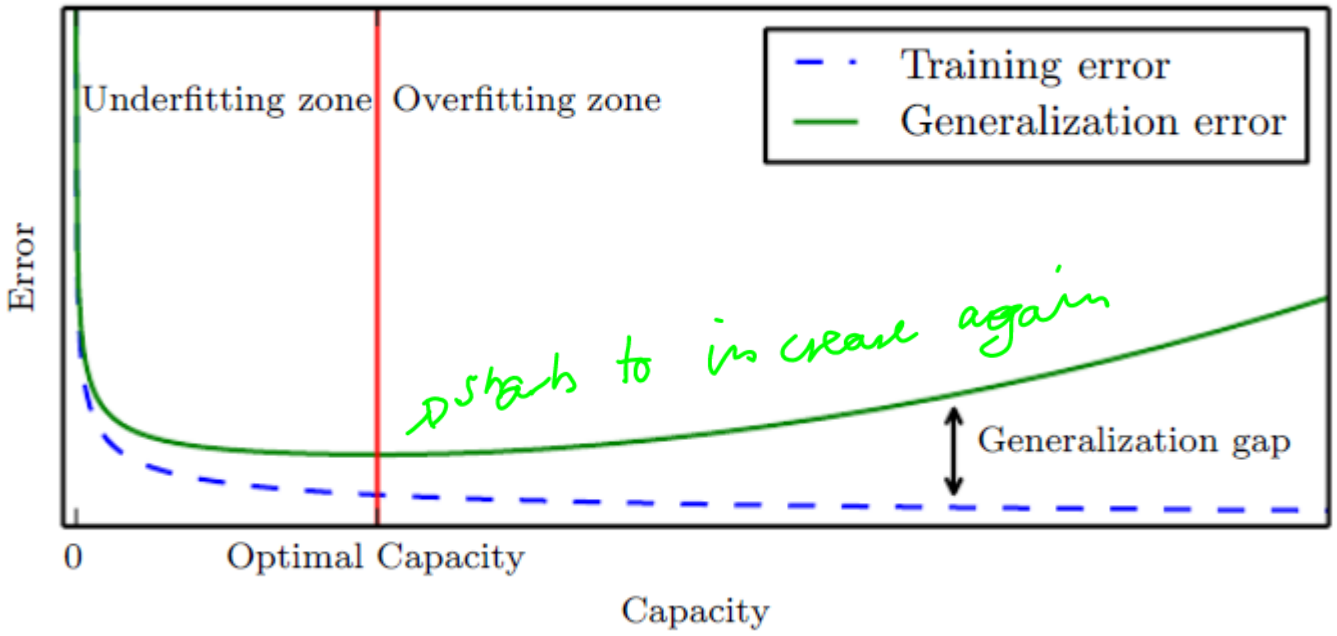


Fig. 5.3: Typical relationship between capacity and error. At the left end, training error and generalization error are both high. This is the underfitting regime. As we increase capacity, training error decreases, but the gap between training and generalization error increases. Eventually, this outweighs the decrease in training error, and we enter the overfitting regime, where the capacity is too large.

early stopping: introduce validation set to stop before overfitting

Non-parametric Models

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- To reach the most extreme case of arbitrarily high capacity, we introduce the concept of non-parametric models. These models do not have a parameter vector w whose size is fixed before training.
- One example of a non-parametric model is nearest neighbor regression. It simply stores the X and y from the training set. When asked to classify a test point x, the model looks up the nearest entry in the training set and returns the associated regression target. In other words, $\hat{y} = y_i$ where $i = \arg\min \left\| X_{i,:} - x \right\|_2^2$.

Bayes Error

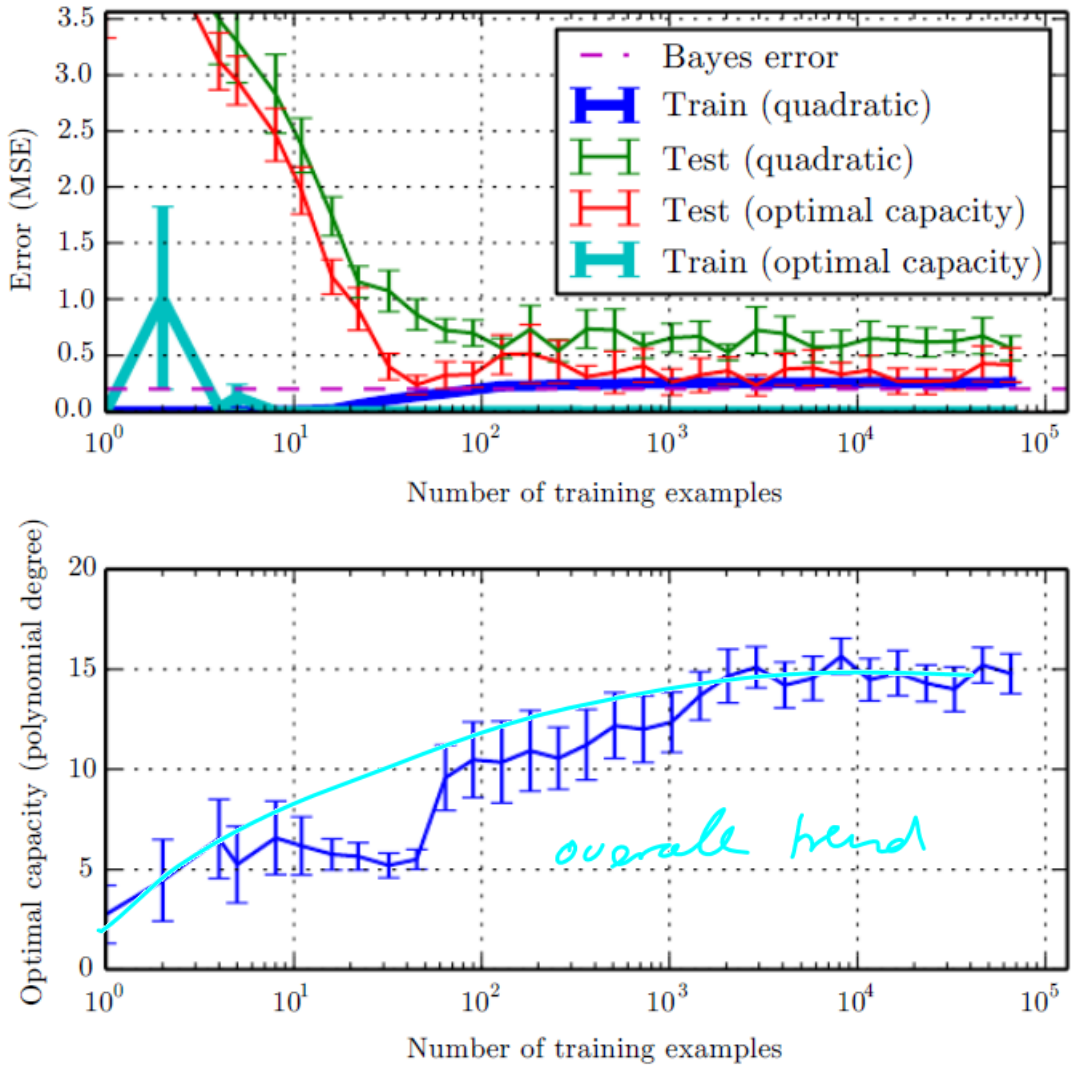
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- The ideal model is an oracle that simply knows the true probability distribution that generates the data.
- Even such a model will still incur some error on many problems, because there may still be some noise in the distribution.
- In the case of supervised learning, the mapping from x to y may be stochastic, or y may be a deterministic function that involves other variables besides those included in x.
- The error incurred by an oracle making predictions from the true distribution $p(x, y)$ is called the Bayes error.

Effect of Training Dataset size

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Fig. 5.4: Effect of the training dataset size on training and test error, as well as on the optimal model capacity, on a synthetic regression problem of a 5-deg. polynomial with noise.
There is a single test set, and several training set sizes.
(Top) The MSE on the training and test set for a quadratic model, and a model with optimal capacity. The test error at optimal capacity asymptotes to the Bayes error.
(Bottom) the optimal capacity (shown here as the degree of the optimal polynomial regressor) increases.



test error at optimal capacity approaches Bayes error (because data has error)

5.2.1 The No Free Lunch Theorem

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- The no free lunch theorem for machine learning (Wolpert, 1996) states that, averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.
- In other words, in some sense, no machine learning algorithm is universally any better than any other.
- Fortunately, these results hold only when we average over *all* possible data generating distributions. If we make assumptions about the kinds of probability distributions we encounter in real-world applications, then we can design learning algorithms that perform well on these distributions.

5.2.2 Regularization

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- We can also give a learning algorithm a preference for one solution in its hypothesis space to another. This means that both functions are eligible, but one is preferred. The other solution will be chosen only if it fits the training data significantly better than the preferred solution.
- Weight decay is one such regularization method.
- To perform linear regression with weight decay, we minimize

$$J(\boldsymbol{w}) = MSE_{train} + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

where λ is a value that controls the strength of our preference for smaller weights. $J(w) = \lambda \boldsymbol{w}^T \boldsymbol{w}$ expresses a preference for weights with smaller L^2 norm.

Regularization

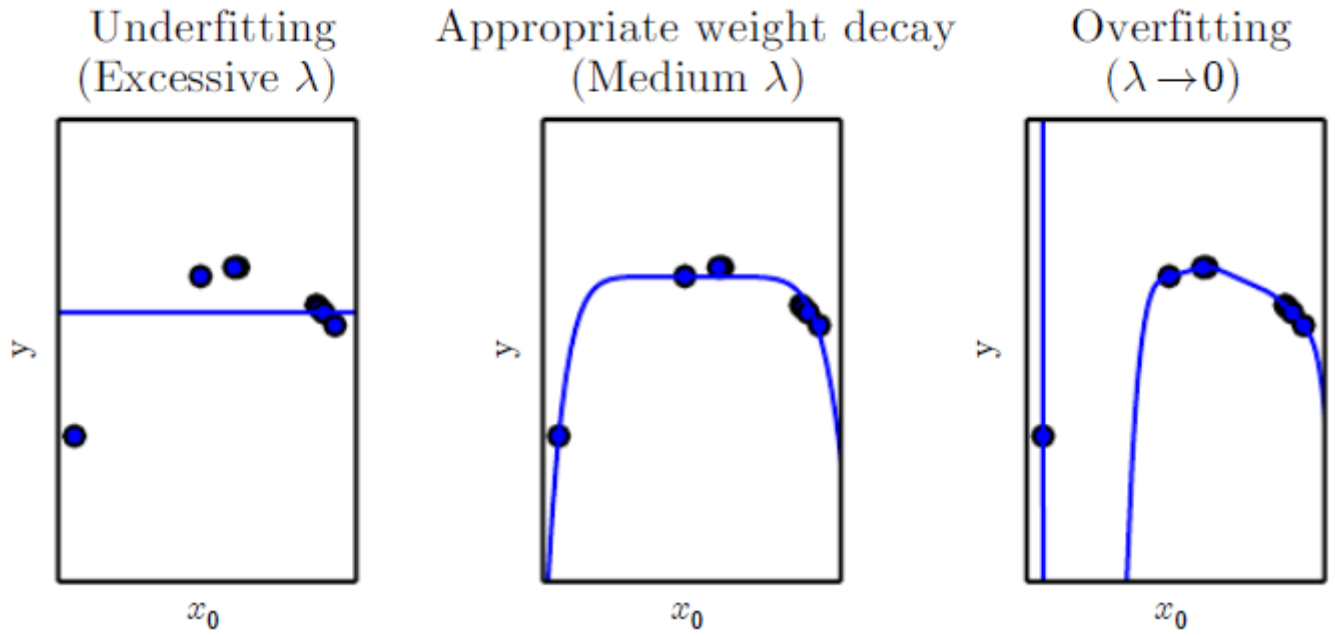


Fig. 5.5: A degree-9 polynomial regression model is fit to the training set from fig. 5.2. The true function is quadratic. *(Left)* With very large λ, we force the model to learn a function with no slope at all. This underfits. *(Center)* With a medium value of λ, the learning algorithm recovers a curve with the right general shape. *(Right)*With weight decay approaching zero (using the pseudo-inverse to solve the problem), the degree-9 polynomial overfits significantly.

Capacity, Overfitting, Underfitting

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- More generally, we can regularize a model that learns a function $f(x;\boldsymbol{\theta})$ by adding a penalty called a regularizer to the cost function.
- In the case of weight decay, the regularizer is $\Omega(\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{w}$
- There are many other ways of expressing preferences for different solutions, both implicitly and explicitly.
- These different approaches are known as regularization.
- *Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.*
- Regularization is one of the central concerns in machine learning, rivaled in importance only by optimization.

5.3 Hyperparameters, Validation Sets

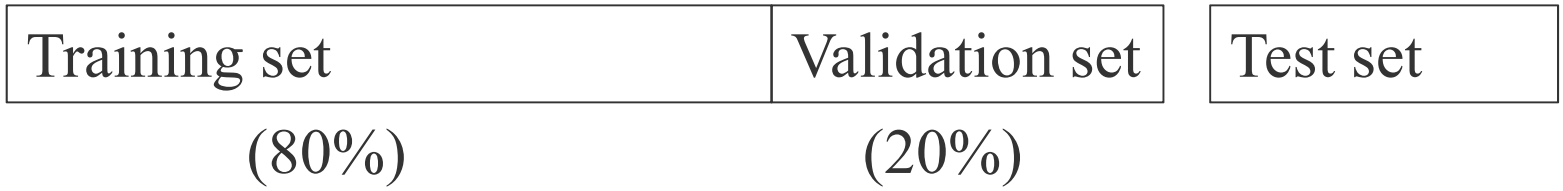
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- ML algorithms have settings used to control the behavior of the learning algorithm. These settings are called hyperparameters. Their values are not adapted by the learning algorithm itself.
- In the polynomial regression example there is a single hyperparameter: the degree of the polynomial, which acts as a capacity hyperparameter.
- The λ value used to control the strength of weight decay is another example of a hyperparameter.
- Hyperparameters that control model capacity cannot be learned on the training set. Otherwise they would always choose the maximum model capacity, resulting in overfitting.

Hyperparameters and Validation Sets

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Therefore, we need a validation set of examples that the training algorithm does not observe.
- The validation set is always split off from the training set, never from the (independent) test set.



- Training set: used to train the model, minimize training error
- Validation set: used to optimize hyperparameters, by monitoring validation set error
- Test set: used to determine generalization error.

5.3.1 Cross-Validation

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Dividing the dataset into a fixed training set and a fixed test set can be problematic if the test set gets too small.
- Cross-validation enables one to use all of the examples in the estimation of the mean test error, at the price of increased computational cost.
- *k*-fold cross-validation partitions the dataset into k non-overlapping subsets. The test error is then estimated by averaging the test error across k trials. On trial i, the i-*th* subset of the data is used as the test set and the rest of the data is used as training set.
- Leave-one-out cross validation uses n-1 examples as training set and one example as test set. This is repeated n times, using each example once as test set.

Algorithm 5.1 *k*-fold Cross-Validation

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Define `KFoldXV`$(\mathbb{D}, A, L, k)$:
Require: $\mathbb{D}$, the given dataset, with elements $\boldsymbol{z}^{(i)}$
Require: A, the learning algorithm, seen as a function that takes a dataset as input and outputs a learned function
Require: L, the loss function, seen as a function from a learned function f and an example $\boldsymbol{z}^{(i)} \in \mathbb{D}$ to a scalar $\in \mathbb{R}$
Require: k, the number of folds
  Split $\mathbb{D}$ into k mutually exclusive subsets $\mathbb{D}_i$, whose union is $\mathbb{D}$.
  for i from 1 to k **do**
    $f_i = A(\mathbb{D} \backslash \mathbb{D}_i)$
    for $\boldsymbol{z}^{(j)}$ in $\mathbb{D}_i$ **do**
      $e_j = L(f_i, \boldsymbol{z}^{(j)})$
    end for
  end for
  Return $\boldsymbol{e}$

The algorithm returns the vector of errors $\boldsymbol{e}$ for each example in $\mathbb{D}$, whose mean is the estimated generalization error.

Cross Validation

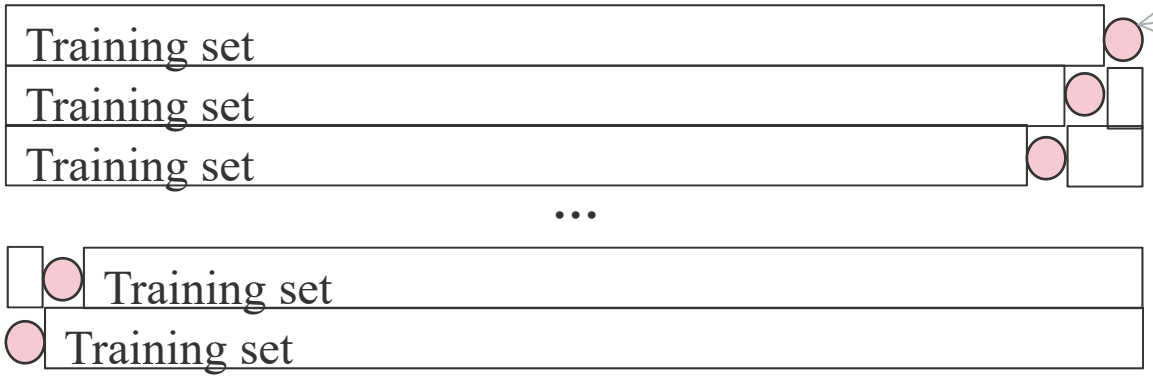
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- *5*-fold cross-validation

Train set	Train set	Train set	Train set	Test set
Train set	Train set	Train set	Test set	Train set
Train set	Train set	Test set	Train set	Train set
Train set	Test set	Train set	Train set	Train set
Test set	Train set	Train set	Train set	Train set

5 different training/ test runs

- Leave-one-out cross validation



n different training/ test runs

5.4.1 Point Estimation

- Point estimation is the attempt to provide the single "best" prediction of some quantity of interest.
- We denote the estimate of a parameter $\boldsymbol{\theta}$ with $\hat{\boldsymbol{\theta}}$.
- Let $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ be a set of m independent and identically distributed (i.i.d.) data points. A point estimator or statistic is any function of the data:

$$\hat{\boldsymbol{\theta}}_m = g(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)})$$

- A good estimator is a function whose output is close to the true underlying θ that generated the training data.
- Since the data is drawn from a random process, $\hat{\boldsymbol{\theta}}$ is a random variable.

Function Estimation

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Sometimes we are interested in performing function estimation (or function approximation). Here we are trying to predict a variable y given an input vector x.
- We assume that there is a function $f(x)$ that describes the approximate relationship between y and x.
- For example, we may assume that $y = f(x) + \epsilon$, where ϵ stands for the part of y that is not predictable from x.
- In function estimation, we want to approximate f with a model or estimate $\hat{f}$.
- Function estimation is basically the same as estimating a parameter $\boldsymbol{\theta}$; the function estimator $\hat{f}$ is simply a point estimator in function space.

5.4.2 Bias

- The bias of an estimator $\hat{\boldsymbol{\theta}}_m$ is defined as

$$\text{bias}(\hat{\boldsymbol{\theta}}_m) = \mathbb{E}(\hat{\boldsymbol{\theta}}_m) - \boldsymbol{\theta}$$

  where the expectation is over the m data and $\boldsymbol{\theta}$ is the true underlying value.
- An estimator $\hat{\boldsymbol{\theta}}_m$ is said to be unbiased if $\text{bias}(\hat{\boldsymbol{\theta}}_m) = 0$, which implies that $\mathbb{E}(\hat{\boldsymbol{\theta}}_m) = \boldsymbol{\theta}$.
- An estimator $\hat{\boldsymbol{\theta}}_m$ is said to be asymptotically unbiased if $\lim_{m \to \infty} \text{bias}(\hat{\boldsymbol{\theta}}_m) = 0$, which implies that $\lim_{m \to \infty} \mathbb{E}(\hat{\boldsymbol{\theta}}_m) = \boldsymbol{\theta}$.

Estimator Bias of Bernoulli Distribution

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Example: Bernoulli Distribution:

- Consider a set of samples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ that are i.i.d. according to a Bernoulli distribution with mean θ.

$$P(x(i); \theta) = \theta^{x^{(i)}} (1-\theta)^{(1-x^{(i)})}$$

- A common estimator for the parameter θ of this distribution is the mean of the training samples

$$\hat{\theta}_m = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}^{(i)}$$

- To determine whether this estimator is biased, we compute the estimator bias according to the formula from the previous page:

Estimator Bias of Bernoulli Distribution

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

$$
\begin{aligned}
\text{bias}(\hat{\theta}_m) &= \mathbb{E}\left[\hat{\theta}_m\right] - \theta \\
&= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}^{(i)}\right] - \theta \\
&= \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\left[\boldsymbol{x}^{(i)}\right] - \theta \\
&= \frac{1}{m}\sum_{i=1}^{m} \sum_{\boldsymbol{x}^{(i)}=0}^{1} \boldsymbol{x}^{(i)} \theta^{\boldsymbol{x}^{(i)}} (1-\theta)^{(1-\boldsymbol{x}^{(i)})} - \theta \\
&= \frac{1}{m}\sum_{i=1}^{m} (\theta) - \theta \\
&= \theta - \theta = 0
\end{aligned}
$$

So we see that our estimator is unbiased.

Gaussian Distrib., Estimator of Mean

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Example: Gaussian Distribution, Estimator of the Mean
- Now, consider a set of samples $\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(m)}\}$ that are i.i.d. according to a Gaussian distribution $p(\boldsymbol{x}^{(i)}) = \quad (\boldsymbol{x}^{(i)};\mu,\sigma^2)$, where $i \in \{1,\ldots,m\}$.
- Recall that the Gaussian probability density function is given by $$p(\boldsymbol{x}^{(i)};\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(\boldsymbol{x}^{(i)}-\mu)}{\sigma^2}\right)$$
- A common estimator of the Gaussian mean is known as the sample mean: $$\hat{\mu}_m = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}^{(i)}$$
- To determine the bias of the sample mean, we again calculate $$\text{bias}(\hat{\mu}_m) = \mathbb{E}\left[\hat{\mu}_m\right] - \mu$$

Gaussian Distrib., Estimator of Mean

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

$$
\begin{aligned}
\text{bias}(\hat{\mu}_m) &= \mathbb{E}\left[\hat{\mu}_m\right] - \mu \\
&= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}^{(i)}\right] - \mu \\
&= \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\left[\boldsymbol{x}^{(i)}\right] - \mu \\
&= \left(\frac{1}{m}\sum_{i=1}^{m} \mu\right) - \mu \\
&= \mu - \mu = 0
\end{aligned}
$$

- Thus we find that the sample mean is an unbiased estimator of Gaussian mean parameter.

- Example: Gaussian Distrib., Estimator of the Variance
- The first estimator of σ^2 we consider is known as the sample variance:

$$\hat{\sigma}_m^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \hat{\mu}_m)^2$$

where $\hat{\mu}_m$ is the sample mean from above.

- Its bias is: $\text{bias}(\hat{\sigma}_m^2) = \mathbb{E}\left[\hat{\sigma}_m^2\right] - \sigma^2$

$$= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \hat{\mu}_m)^2\right] - \sigma^2$$

$$= \frac{m-1}{m}\sigma^2 - \sigma^2 = \frac{-\sigma^2}{m}$$

therefore the sample variance is a biased estimator.

Gaussian Distrib., Estimator of Variance

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- The unbiased sample variance estimator is

$$\tilde{\sigma}_m^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x^{(i)} - \hat{\mu}_m)^2$$

- We find that

$$\mathbb{E}\left[\tilde{\sigma}_m^2\right] = \mathbb{E}\left[\frac{1}{m-1}\sum_{i=1}^{m}(x^{(i)} - \hat{\mu}_m)^2\right]$$

$$= \frac{m}{m-1}\mathbb{E}\left[\hat{\sigma}_m^2\right]$$

$$= \frac{m}{m-1}\left(\frac{m-1}{m}\sigma^2\right)$$

$$= \sigma^2$$

so this estimator is unbiased.

5.4.3 Variance and Standard Error

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- The variance of an estimator is simply the variance

$$\mathrm{Var}(\hat{\theta})$$

  where the random variable is the training set.
- Alternately, the square root of the variance is called the standard error, denoted $\mathrm{SE}(\hat{\theta})$.
- The expected degree of variation in any estimator is a source of error that we want to quantify.
- The standard error of the mean is given by

$$\mathrm{SE}(\hat{\mu}_m) = \sqrt{\mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}^{(i)}\right]} = \sqrt{\frac{\sigma^2}{m}} = \frac{\sigma}{\sqrt{m}}$$

  where σ^2 is the true variance of the samples x^i.

$$SE(\mu) = \sqrt{Var(\mu)} = \sqrt{Var\left(\frac{1}{n}\sum_i^n x_i\right)}$$

$$= \sqrt{\frac{1}{n^2} Var\left(\sum_i x_i\right)}$$

$$= \sqrt{\frac{1}{n^2} \sum_i Var(x_i)}$$

$$= \sqrt{\frac{1}{n^2} \sum_i \sigma^2}$$

$$= \sqrt{\frac{1}{n^2} n \cdot \sigma^2} = \sqrt{\frac{\sigma^2}{n}} = \frac{\sigma}{\sqrt{n}}$$

Standard Error

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Unfortunately, neither the square root of the sample variance nor the square root of the unbiased estimator of the variance provide an unbiased estimate of the standard deviation.
- Both approaches tend to underestimate the true standard deviation, but are still used in practice.
- The square root of the unbiased estimator of the variance is less of an underestimate.
- For large m, the approximation is quite reasonable.
- $\mathrm{SE}(\hat{\mu}_m)$ is very useful in ML experiments.
- We often estimate the generalization error by computing the sample mean of the error on the test set.

Standard Error

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- The number of examples in the test set determines the accuracy of this estimate. With the central limit theorem, which tells us that the mean will be approximately distributed with a normal distribution, we can use the standard error to compute the probability that the true expectation falls in any chosen interval.
- For example, the 95% confidence interval around the mean $\hat{\mu}_m$ is $(\hat{\mu}_m - 1.96 \cdot \mathrm{SE}(\hat{\mu}_m), \hat{\mu}_m + 1.96 \cdot \mathrm{SE}(\hat{\mu}_m))$ under the normal distrib. with mean $\hat{\mu}_m$ and variance $\mathrm{SE}(\hat{\mu}_m)^2$.
- In ML experiments, algorithm A is “better” than algorithm B, if the upper bound of the 95% confidence interval for the error of algorithm A is less than the lower bound of the 95% confidence interval for the error of algorithm B.

Variance of the Bernoulli Distribution

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Now, consider a set of samples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ drawn i.i.d. from a Bernoulli distribution $P(x^{(i)}; \theta) = \theta^{x^{(i)}} (1-\theta)^{(1-x^{(i)})}$.
- We are interested in the variance of the estimator $\hat{\theta}_m = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}^{(i)}$.

$$\mathrm{Var}(\hat{\theta}_m) = \mathrm{Var}\left(\frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}^{(i)} \right)$$

$$= \frac{1}{m^2} \sum_{i=1}^{m} \mathrm{Var}\left(\boldsymbol{x}^{(i)}\right) = \frac{1}{m^2} \sum_{i=1}^{m} \theta(1-\theta)$$

$$= \frac{1}{m^2} m\theta(1-\theta) = \frac{1}{m} \theta(1-\theta)$$

- The variance of the estimator decreases as a function of m, the number of examples in the dataset.

5.4.4. Trading off Bias and Variance

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Bias and variance measure two different sources of error in an estimator.
- Bias measures the expected deviation from the true value of the function or parameter.
- Variance measures the deviation from the expected estimator value that any particular sampling of the data is likely to cause.
- How do we choose between two estimators, one with more bias, the other with more variance?
- The most common way to negotiate this trade-off is to use cross-validation. Empirically, cross-validation is highly successful on many real-world tasks.

Trading off Bias and Variance

- Alternatively, we can also compare the mean squared error (MSE) of the estimates

$$\mathrm{MSE} = \mathbb{E}\left[(\hat{\theta}_m - \theta)^2\right] = \mathbb{E}\left[\hat{\theta}_m^{\,2} - 2\hat{\theta}_m\theta + \theta^2\right]$$

because doesn't depend on sample

$$= \mathbb{E}\left[\hat{\theta}_m^{\,2}\right] - 2\theta\mathbb{E}\left[\hat{\theta}_m\right] + \theta^2$$

$$= \mathbb{E}\left[\hat{\theta}_m^{\,2}\right] - 2\theta\mathbb{E}\left[\hat{\theta}_m\right] + \theta^2 + \mathbb{E}\left[\hat{\theta}_m\right]^2 - \mathbb{E}\left[\hat{\theta}_m\right]^2$$

$$= \mathbb{E}\left[\hat{\theta}_m\right]^2 - 2\theta\mathbb{E}\left[\hat{\theta}_m\right] + \theta^2 + \mathbb{E}\left[\hat{\theta}_m^{\,2}\right] - \mathbb{E}\left[\hat{\theta}_m\right]^2$$

$$= \left(\mathbb{E}\left[\hat{\theta}_m\right] - \theta\right)^2 + \left(\mathbb{E}\left[\hat{\theta}_m^{\,2}\right] - \mathbb{E}\left[\hat{\theta}_m\right]^2\right)$$

$$= \mathrm{Bias}(\hat{\theta}_m)^2 + \mathrm{Var}(\hat{\theta}_m)$$

This is the well-known bias-variance tradeoff

Trading off Bias and Variance

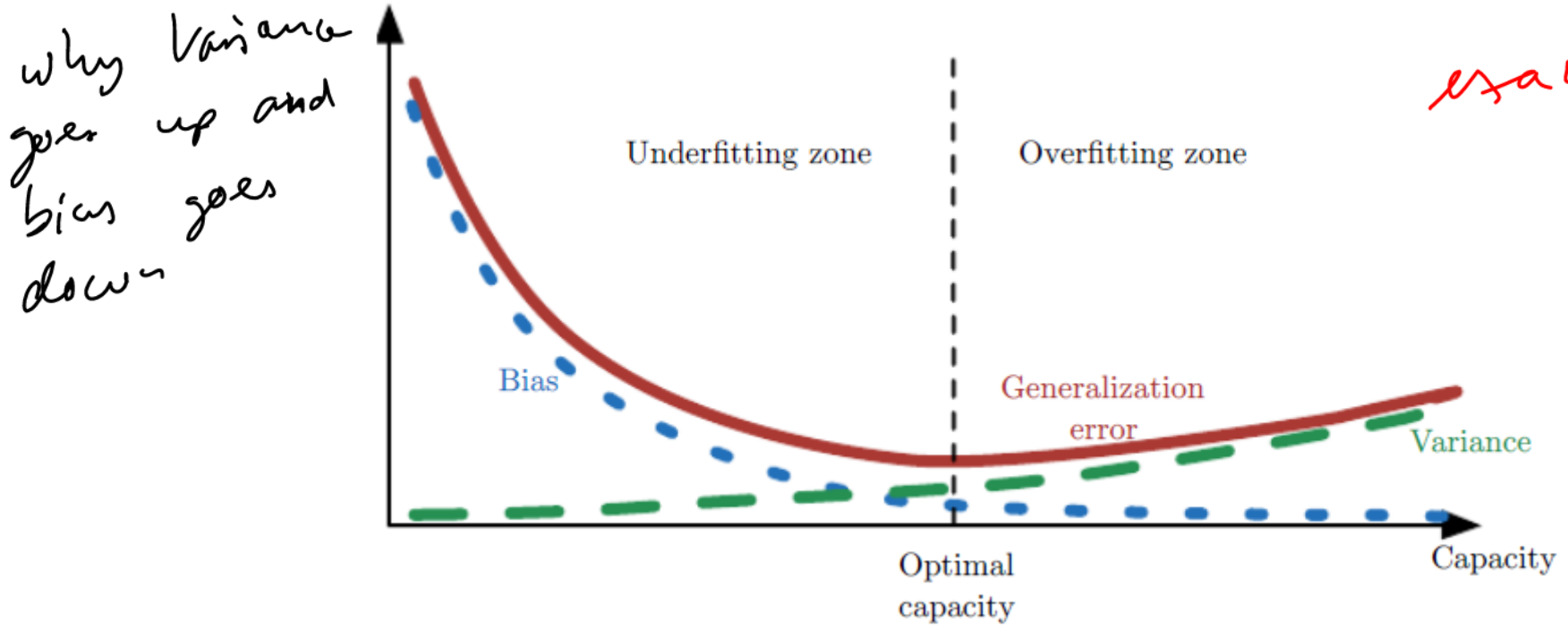


Fig. 5.6: As capacity increases (x-axis), bias (dotted) tends to decrease and variance (dashed) tends to increase, yielding another U-shaped curve for generalization error (red curve). If we vary capacity along one axis, there is an optimal capacity, with underfitting when the capacity is below this optimum and overfitting when it is above. This is similar to the relationship between capacity, underfitting, and overfitting,

5.5 Maximum Likelihood Estimation

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- The most common principle to devise an estimator is the maximum likelihood principle.
- Consider a set of m examples $\mathbb{X} = \{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(m)}\}$ drawn independently from the true but unknown data distribution $p_{data}(\mathrm{x})$.
- Let $p_{model}(\boldsymbol{x}; \boldsymbol{\theta})$ be a family of probability distributions over the same space indexed by θ. $p_{model}(\boldsymbol{x}; \boldsymbol{\theta})$ maps any configuration $\boldsymbol{x}$ to a real number estimating the true probability $p_{data}(\boldsymbol{x})$. The maximum likelihood estimator for θ is then defined as

$$\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} p_{model}(\mathbb{X}; \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{model}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

because independently

Maximum Likelihood Estimation

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- This product over many probabilities is inconvenient and prone to underflow.
- Taking the logarithm of the likelihood does not change the $\arg\max$, but transforms the product into a sum:

$$\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{model}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

- We can rescale the cost function by dividing it by m and express the criterion as an expectation with respect to the empirical distribution $\hat{p}_{data}(\boldsymbol{x})$ defined by the training data:

$$\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \sim \tilde{p}_{data}} \log p_{model}(\boldsymbol{x}; \boldsymbol{\theta})$$

Maximum Likelihood Estimation

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Maximum likelihood estimation can viewed as minimizing the dissimilarity between the empirical distribution $\hat{p}_{data}(\boldsymbol{x})$ on the training set and the model distribution, with the degree of dissimilarity measured by the KL divergence. The KL divergence is given by

$$D_{KL}(\hat{p}_{data} \,||\, p_{model}) = \mathbb{E}_{x \sim \hat{p}_{data}} \left[\log \hat{p}_{data}(\boldsymbol{x}) - \log p_{model}(\boldsymbol{x}; \boldsymbol{\theta}) \right]$$

- The term on the left is a function only of the data, not the model. This means that when we train the model to minimize the KL divergence, we need only minimize

$$\arg\min_{\boldsymbol{\theta}} - \mathbb{E}_{x \sim \hat{p}_{data}} \left[\log p_{model}(\boldsymbol{x}; \boldsymbol{\theta}) \right]$$

  which is the same as the maximization on the previous page.

Maximum Likelihood Estimation

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Minimizing this KL divergence corresponds exactly to minimizing the cross-entropy between the distributions.
- We can thus see maximum likelihood as an attempt to make the model distribution match the empirical distribution $\hat{p}_{data}(\boldsymbol{x})$.
- While the optimal $\boldsymbol{\theta}$ is the same whether we are maximizing the likelihood or minimizing the KL divergence, the objective function values are different.
- Maximum likelihood thus becomes minimization of the negative log-likelihood (NLL), or equivalently, minimization of the cross entropy.

5.5.1 Conditional Log-Likelihood and Mean Squared Error

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- The maximum likelihood estimator can be generalized to estimate a conditional probability $P(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})$ in order to predict $\boldsymbol{y}$ given $\boldsymbol{x}$.
- This is actually the most common situation because it forms the basis for most supervised learning.
- If $\boldsymbol{X}$ represents all inputs and $\boldsymbol{Y}$ all observed targets, then the conditional maximum likelihood estimator is

$$\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{Y} \mid \boldsymbol{X}; \boldsymbol{\theta})$$

- If the examples are assumed to be i.i.d., then this can be decomposed into

$$\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log P(\boldsymbol{y}^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

5.5.2 Properties of Maximum Likelihood

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- It can be shown that the maximum likelihood estimator is the best estimator asymptotically, as $m \rightarrow \infty$, in terms of its rate of convergence as m increases.
- Under the following conditions, the maximum likelihood estimator has the property of consistency, meaning that if $m \rightarrow \infty$, the maximum likelihood estimate of a parameter converges to the true value of the parameter.
  - The true distribution $p_{data}(\boldsymbol{x})$ must lie within the model family $p_{model}(\boldsymbol{x};\boldsymbol{\theta})$. Otherwise, no estimator can recover $p_{data}(\boldsymbol{x})$. data must be representable by the model
  - The true distribution $p_{data}(\boldsymbol{x})$ must correspond to exactly one value of θ.

Properties of Maximum Likelihood

not so important

- Consistent estimators can differ in their statistic efficiency, meaning that one consistent estimator may obtain lower generalization error for a fixed number of samples m, or may require fewer examples to obtain a fixed level of generalization error.
- In the parametric case, a way to measure how close we are to the true parameter is by the expected MSE, computing the squared difference between the estimated and true parameter values, over m training samples.
- That parametric MSE decreases as m increases, and for m large, the Cramér-Rao lower bound shows that no consistent estimator has a lower mean squared error than the maximum likelihood estimator.

5.6 Bayesian Statistics

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Frequentist statistics, discussed so far, is based on estimating a single value of θ, then making all predictions based on that one estimate.
- The frequentist perspective is that the true parameter value θ is fixed but unknown, while the point estimate $\hat{\theta}$ is a random variable, being a function of the dataset (seen as random).
- Bayesian statistics considers all possible values of θ when making a prediction.
- The Bayesian perspective uses probability to reflect degrees of certainty of states of knowledge. The dataset is directly observed and so is not random. On the other hand, the true parameter θ is unknown or uncertain and thus is represented as a random variable.

Bayesian Statistics

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Before observing the data, we represent our knowledge of θ using the prior probability distribution, $p(\theta)$ (often referred to as simply "the prior").
- Now consider we have a set of examples $\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(m)}\}$. We can recover the effect of data on our belief about θ by combining the data likelihood $p\left(\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(m)}\middle|\boldsymbol{\theta}\right)$ with the prior via Bayes' rule:

$$p(\boldsymbol{\theta}\,|\,x^{(1)},\ldots,x^{(m)}) = \frac{p(x^{(1)},\ldots,x^{(m)}\,|\,\boldsymbol{\theta})\cdot p(\boldsymbol{\theta})}{p(x^{(1)},\ldots,x^{(m)})}$$

- Usually the prior begins as a uniform or Gaussian distribution with high entropy, and the observation of the data usually causes the posterior to lose entropy and concentrate around a few highly likely parameter values.

Bayesian Statistics

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- The Bayesian approach is to make predictions using a full distribution over θ. For example, after observing m examples, the predicted distribution over the next data sample, $x^{(m+1)}$, is given by

$$p(x^{(m+1)} \mid x^{(1)}, \ldots, x^{(m)}) = \int p(x^{(m+1)} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid x^{(1)}, \ldots, x^{(m)}) d(\boldsymbol{\theta})$$

  Here each value of θ with positive probability density contributes to the prediction of the next example, with the contribution weighted by the posterior density itself.

- The Bayesian answer to the question of how to deal with the uncertainty in the estimator is to simply integrate over it, which tends to protect well against overfitting.

Bayesian Statistics

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- The second important difference between the Bayesian approach to estimation and the maximum likelihood approach is due to the contribution of the Bayesian prior distribution.
- The prior has an influence by shifting probability mass density towards regions of the parameter space that are preferred *a priori*. In practice, the prior often expresses a preference for models that are simpler or smoother.
- Bayesian methods typically generalize much better when limited training data is available, but suffer from high computational cost when the number of training examples is large.

5.6.1 Maximum A Posteriori Estimation

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Even in Bayesian statistics we might want a point estimate.
- Rather than simply returning to the maximum likelihood estimate, we can still gain some of the benefit of the Bayesian approach by allowing the prior to influence the choice of the point estimate.
- One rational way to do this is to choose the maximum a posteriori (MAP) point estimate. The MAP estimate chooses the point of maximal posterior probability (or maximal probability density in the more common case of continuous θ):

$$\boldsymbol{\theta}_{MAP} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$$

can ignore $p(x)$ in denom because constant

Maximum A Posteriori (MAP) Estimation

$$\boldsymbol{\theta}_{MAP} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$$

- We recognize above $\log p(\boldsymbol{x} \mid \boldsymbol{\theta})$, i.e. the standard log-likelihood term, and $\log p(\boldsymbol{\theta})$ from the prior distribution.
- As an example, consider a linear regression model with a Gaussian prior on the weights w. If this prior is given by $p(\boldsymbol{w}) = \quad (\boldsymbol{w}; 0; \frac{1}{\lambda}\boldsymbol{I}^2)$, then the log-prior term in the previous equation is proportional to the familiar $\lambda \boldsymbol{w}^T \boldsymbol{w}$ weight decay penalty, plus a term that does not depend on w and does not affect the learning process.
- MAP Bayesian inference with a Gaussian prior on the weights thus corresponds to weight decay.
- MAP Bayesian inference may use information from the prior that cannot be found in the training data.

Maximum A Posteriori (MAP) Estimation

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- This additional information helps to reduce the variance in the MAP point estimate (vs. the ML estimate). However, it does so at the price of increased bias.
- Many regularized estimation strategies, such as ML learning regularized with weight decay, can be viewed as MAP approximation to Bayesian inference. Regularization then consists of adding an extra term to the objective function that corresponds to $\log p(\boldsymbol{\theta})$.
- Not all regularization penalties correspond to MAP Bayesian inference. E.g., some regularizer terms may not be the logarithm of a probability distribution.
- Other regularization terms depend on the data, which of course a prior probability distribution is not allowed to do.

5.7 Supervised Machine Learning

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Recall that supervised learning algorithms are learning algorithms that learn to associate some input with some output, given a training set of examples of inputs x and outputs y.

5.7.1 Probabilistic Supervised Learning

- Many supervised learning algorithms are based on estimating a probability distribution $p(y \mid x)$.
- We can do this by using maximum likelihood estimation to find the best parameter vector θ for a parametric family of distributions $p(y \mid x; \theta)$.
- We have already seen that linear regression corresponds to the family

$$p(y \mid \boldsymbol{x}; \boldsymbol{\theta}) = \quad (y; \boldsymbol{\theta}^T \boldsymbol{x}, \boldsymbol{I})$$

5.7.1 Probabilistic Supervised Learning

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- We can generalize linear regression to classification by defining a different family of probability distributions.
- If we have two classes, class 0 and class 1, then we need only specify the probability of one of these classes. The probability of class 1 determines the probability of class 0, because these two values must add up to 1.
- The mean of a binary variable must always be in (0, 1).
- We may use the logistic sigmoid function σ to squash the output of the linear function into the interval (0, 1) and interpret that value as a probability:

$$p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$$

- This approach is known as logistic regression.

5.9 Stochastic Gradient Descent

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Nearly all of deep learning is powered by one very important algorithm: stochastic gradient descent or SGD.
- Stochastic gradient descent is an extension of the gradient descent algorithm introduced in section 4.3.
- The cost function used by a machine learning algorithm often decomposes as a sum over training examples of some per-example loss function. For example, the negative conditional log-likelihood of the training data can be written as

$$J(\boldsymbol{\theta}) = \mathbb{E}_{x,y\sim\tilde{p}_{data}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

where L is the per-example loss $L(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\log p(y \mid \boldsymbol{x}; \boldsymbol{\theta})$

Stochastic Gradient Descent

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- For these additive cost functions, gradient descent requires computing

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

- The computational cost of this operation is $O(m)$. As the training set size grows to millions of examples, the time to take a single gradient step becomes prohibitively long.
- The insight of stochastic gradient descent is that the gradient is an expectation.
- The expectation may be approximately estimated using a small set of samples.

Stochastic Gradient Descent

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Specifically, on each step of the algorithm, we can sample a minibatch of examples $\mathbb{B} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m')}\}$ drawn uniformly from the training set.
- The minibatch size is typically fixed to be a relatively small number of examples, ranging from 1 to 500.
- The estimate of the gradient is formed as

$$\boldsymbol{g} = \frac{1}{m'} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

  using examples from the minibatch $\mathbb{B}$.
- The stochastic gradient descent algorithm then follows the estimated gradient downhill:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$

  where ϵ is the learning rate.

Stochastic Gradient Descent

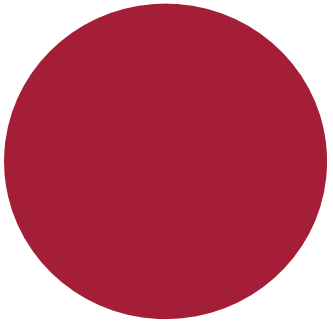
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Gradient descent may not be guaranteed to arrive at even a local minimum in a reasonable amount of time, but it often finds a very low value of the cost function quickly enough to be useful.
- Stochastic gradient descent has many important uses outside the context of deep learning. It is the main way to train large linear models on very large datasets.
- For a fixed model size, the cost per SGD update does not depend on the training set size m.

- Prior to the advent of deep learning, the main way to learn nonlinear models was to use the kernel trick in combination with a linear model.
- Many kernel learning algorithms require constructing an $m \times m$ matrix $G_{i,j} = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$.
- Constructing this matrix has computational cost $O(m^2)$, which is clearly undesirable for datasets with millions of examples.

5.10 Building a Machine Learning Algorithm

- Nearly all deep learning algorithms can be described as particular instances of a fairly simple recipe: combine
  - a specification of a dataset,
  - a cost function,
  - an optimization procedure and
  - a model.



- For example, the linear regression algorithm combines
  - a dataset consisting of $\boldsymbol{X}$ and y,
  - the cost function $J(\boldsymbol{w}, b) = -\mathbb{E}_{x,y\sim\tilde{p}_{data}} \log p_{model}(y \mid \boldsymbol{x})$
  - the model specification $p_{model}(y \mid \boldsymbol{x}) = \quad (y; \boldsymbol{x}^T \boldsymbol{w} + b, 1)$,
  - the optimization algorithm for where the gradient of the cost function is zero using the normal equations.

Building a Machine Learning Algorithm

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- By realizing that we can replace any of these components mostly independently from the others, we can obtain a very wide variety of algorithms.
- The most common cost function is the negative log-likelihood, so that minimizing the cost function causes maximum likelihood estimation.
- The cost function may also include additional terms, such as regularization terms.
- For example, we can add weight decay to the linear regression cost function to obtain

$$J(\boldsymbol{w}, b) = \lambda \|\boldsymbol{w}\|_2^2 - \mathbb{E}_{x,y \sim \tilde{p}_{data}} \log p_{model}(y \mid \boldsymbol{x})$$

- This still allows closed-form optimization.

Building a Machine Learning Algorithm

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- If we change the model to be nonlinear, then most cost functions can no longer be optimized in closed form.
- This requires us to choose an iterative numerical optimization procedure, such as gradient descent.
- The recipe for constructing a learning algorithm by combining models, costs, and optimization algorithms supports both supervised and unsupervised learning.
- The linear regression example shows how to support supervised learning.
- Unsupervised learning can be supported by defining a dataset that contains only $\boldsymbol{X}$ and providing an appropriate unsupervised cost and model.

- This section is about how the challenge of generalizing to new examples becomes exponentially more difficult when working with high-dimensional data

5.11.1 The Curse of Dimensionality

- Many machine learning problems become exceedingly difficult when the number of dimensions in the data is high. This is known as the curse of dimensionality.
- Of particular concern is that the number of possible distinct configurations of a set of variables increases exponentially as the number of variables increases.

The Curse of Dimensionality

Fig. 5.9: As the number of dimensions of the data increases, the number of configurations may grow exponentially. *(Left)* In 1D, we have one variable distinguished into 10 regions of interest. With enough examples falling within each of these regions, learning algorithms can easily generalize. *(Center)* In 2D, we have 10×10=100 regions, and we need at least that many examples to cover all those regions. *(Right)* In 3D this grows to $10^3 = 1000$ regions and at least that many examples. For d dimensions and v values along each axis, we need $O(v^d)$ regions and examples.

5.11.2 Local Constancy and Smoothness Regularization

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- In order to generalize well, machine learning algorithms need to be guided by prior beliefs about what kind of function they should learn.
- Among the most widely used of these implicit priors is the smoothness prior or local constancy prior.
- It states that the function we learn should not change very much within a small region.
- There are many different ways to express a prior belief that the learned function should be smooth.
- All of these different methods are designed to learn a function f^* that satisfies the condition

$$f^*(\boldsymbol{x}) \approx f^*(\boldsymbol{x} + \ )$$

for most configurations x and small change ϵ.

Local Constancy and Smoothness Regularization

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- The smoothness assumption and the associated non-parametric learning algorithms work extremely well so long as there are enough examples for the learning algorithm to observe high points on most peaks and low points on most valleys of the true function to be learned.
- This is generally true when the function to be learned is smooth enough and varies in few enough dimensions.
- In high dimensions, even a very smooth function can change smoothly but in a different way along each dimension. If the function additionally behaves differently in different regions, it can become extremely complicated to describe with a set of training examples.

Local Constancy and Smoothness Regularization

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- If the function is complicated (we want to distinguish a huge number of regions compared to the number of examples), is there any hope to generalize well?
- The answer is yes. The key insight is that a very large number of regions, e.g., $O(2^k)$, can be defined with $O(k)$ examples, if we introduce some dependencies between the regions via additional assumptions about the underlying data generating distribution.
- In this way, we can actually generalize non-locally.
- The core idea in deep learning is that we assume that the data was generated by the *composition of factors* or features, potentially at multiple levels in a hierarchy.