ECBM E4040 Neural Networks and Deep Learning

Lecture: Machine Learning Basics, Part 1

Instructor Zoran Kostic

Columbia University
Department of Electrical Engineering

Sep. 20th, 2016

Outline of Part I - Review of Previous Lecture

2. Summary of the Previous Lecture
   - Topics Covered
   - Learning Objectives

Outline of Part II - Today's Lecture

3. Machine Learning Algorithms
   - The Task $\mathcal{T}$, Performance Measure $\mathcal{P}$ and Experience $\mathcal{E}$
   - Example: Linear Regression
4. Capacity, Overfitting and Underfitting
   - Regularization
   - Hyperparameters and Validation Sets
5. Estimation, Bias and Variance
   - Point Estimation
   - Bias
   - Variance and Standard Error
   - Trading Off Bias and Variance and MSE
6. Maximum Likelihood Estimation
   - Conditional Log-Likelihood and MSE
   - Properties of Maximum Likelihood

Part I

Review of Previous Lecture

Previous Lecture - Topics Covered

- Probability and Information Theory
  - Probability Spaces and Random Variables
  - Elements of Information Theory
- Numerical Computation
  - Gradient-Based Optimization
  - Constraint Optimization
  - Linear Least Squares

Previous Lecture - Learning Objectives

- Reviewing the main concept of probability theory that are used for (i) reasoning in machine learning systems and, (ii) to analyze and predict the performance of learning systems.
- Reviewing the key elements of optimization theory. Deep learning algorithms as the solution to optimization problems.

Part II

Today's Lecture

Learning Algorithms

A machine learning algorithm is an algorithm that is able to learn from data.

A computer program is said to learn from experience $\mathcal{E}$ with respect to some class of tasks $\mathcal{T}$ and performance measure $\mathcal{P}$, if its performance at tasks in $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$.

There are very wide variety of experiences $\mathcal{E}$, tasks $\mathcal{T}$, and performance measures $\mathcal{P}$ - no formal definition.

The Task $\mathcal{T}$

Learning is the means of attaining the ability to perform a task.
Common machine learning tasks:

- Classification
- Regression
- Density or Probability Function Estimation

Common Machine Learning Task: Classification

The computer program is asked to specify which of k categories some input belongs to.

- The learning algorithm is usually asked to produce a function $f : \mathbb{R}^n \to \{1, ..., k\}$ which may then be applied to any input. Here the output of $f(\mathbf{x})$ can be interpreted as an estimate of the category that $\mathbf{x}$ belongs to. There are other variants of the classification task, for example, where f outputs a probability distribution over classes.
- Example: object recognition, where the input is an image (usually described as a set of pixel brightness values), and the output is a numeric code identifying the object in the image. Object recognition allows computers to recognize faces and it is used to automatically tag people in photo collections.

Common Machine Learning Task: Regression

The computer program is asked to predict a numerical value given input data.

- The learning algorithm is asked to output a function $f : \mathbb{R}^n \to \mathbb{R}$. This type of task is similar to classification, except that the format of output is different.
- Example: prediction of the expected claim amount that an insured person will make (used to set insurance premia), or the prediction of future prices of securities used for algorithmic trading.

Common Machine Learning Task: Density Estimation

The learning algorithm has to capture the structure of the probability distribution on the space of examples. Density estimation allows us to explicitly capture that distribution.

- The machine learning algorithm is asked to learn a function $p_{model} : \mathbb{R}^n \rightarrow \mathbb{R}$, where $p_{model}(\mathbf{x})$ represents the probability density function (if $\mathbf{x}$ is continuous) or a probability function (if $\mathbf{x}$ is discrete) on the space that the examples were drawn from.
- Example: density estimation can be used to solve the missing value imputation task. If a value x_i is missing and all of the other values, denoted $\mathbf{x}_{-i}$ are given, then the distribution is given by $p(x_i|\mathbf{x}_{-i})$.

The Performance Measure $\mathcal{P}$

The performance measure $\mathcal{P}$ is specific to the task being carried $\mathcal{T}$ out by the system. In the real world we typically evaluate these performance measures using a test set of data that is separate from the data used for training the machine learning system.

- Accuracy of the model: the proportion of examples for which the model produces the correct output.
- Error rate: the proportion of examples for which the model produces an incorrect output.

The Experience $\mathcal{E}$

Machine learning algorithms are broadly categorized as unsupervised or supervised by what kind of experience they are allowed to have during the learning process.

Most learning algorithms experience an entire dataset. A dataset is a collection of many objects called examples. Each example contains many features that have been objectively measured. Examples are also called data points.

Unsupervised learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset.

Supervised learning algorithms experience a dataset containing features, but each example is also associated with a label or target

The Experience $\mathcal{E}$ (cont'd)

Roughly speaking

- unsupervised learning involves observing several examples of a random vector $\mathbf{x}$, and attempting to implicitly or explicitly learn the probability distribution $p(\mathbf{x})$, or some interesting properties of that distribution. In unsupervised learning, there is no instructor or teacher, and the algorithm must learn to make sense of the data without this guide.
- supervised learning involves observing several examples of a random vector $\mathbf{x}$ and an associated value or vector $\mathbf{y}$, and learning to predict $\mathbf{y}$ from $\mathbf{x}$, e.g., estimating $p(\mathbf{y}|\mathbf{x})$. The term supervised learning originates from the view of the target $\mathbf{y}$ being provided by an instructor or teacher that shows the machine learning system what to do.

Example: Linear Regression

A Simpe Machine Learning Algorithm

Build a linear system that can take a vector $\mathbf{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}$ as its output:

$$\hat{y} = \mathbf{w}^T \mathbf{x},$$

where $\mathbf{w} \in \mathbb{R}^n$ is a vector of parameters called weights.

Task $\mathcal{T}$: predict y from $\mathbf{x}$ by computing $\hat{y} = \mathbf{w}^T \mathbf{x}$.

Example: Linear Regression (cont'd)

We assume that the design matrix of m example inputs will not be used for training, only for evaluating how well the model performs. The design matrix of inputs is denoted by $\mathbf{X}^{test}$ and the vector of regression targets by $\mathbf{y}^{test}$.

The mean square error is given by

$$MSE_{test} = \frac{1}{m} \sum_i [\hat{\mathbf{y}}^{test} - \mathbf{y}^{test}]_i^2.$$

Note that

$$MSE_{test} = \frac{1}{m} ||\hat{\mathbf{y}}^{test} - \mathbf{y}^{test}||_2^2.$$

Example: Linear Regression (cont'd)

To minimize MSE_{train} we simply solve

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

that is

$$\nabla_{\mathbf{w}} \frac{1}{m} ||\hat{\mathbf{y}}^{train} - \mathbf{y}^{train}||_2^2 = 0$$

or

$$\frac{1}{m} \nabla_{\mathbf{w}} ||\mathbf{X}^{train}\mathbf{w} - \mathbf{y}^{train}||_2^2 = 0$$

or in inner product for

$$\nabla_{\mathbf{w}} (\mathbf{X}^{train}\mathbf{w} - \mathbf{y}^{train})^T (\mathbf{X}^{train}\mathbf{w} - \mathbf{y}^{train}) = 0,$$

$$\nabla_{\mathbf{w}} (\mathbf{w}^T \mathbf{X}^{(train)T} \mathbf{X}^{train}\mathbf{w} - 2\mathbf{w}^T \mathbf{X}^{(train)T} \mathbf{y}^{train} + \mathbf{y}^{(train)T} \mathbf{y}^{train}) = 0.$$

Example: Linear Regression (cont'd)

Normal Equations

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

or finally

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

This system of equations is known as the normal equations.

The Concept of Generalization in Machine Learning

The central challenge in machine learning is called generalization, i.e., the ability to perform well on previously unobserved inputs and not just on the data the model was trained.

A machine learning algorithm/model is trained/optimized using a training set. The resulting error measure is called the training error. The overall goal is to reduce the training error. This is a classical optimization problem.

The goal of machine learning is to reduce the generalization error, also called the test error. The generalization error is defined as the expected value of the error on a new input. Here the expectation is taken across different possible inputs, drawn from the distribution of inputs we expect the system to encounter in practice.

Generalization in Machine Learning (cont'd)

We typically estimate the generalization error of a machine learning model by measuring its performance on a test set of examples that were collected separate from the training set.

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

$$\frac{1}{m^{train}}||\mathbf{X}^{train}\mathbf{w} - \mathbf{y}^{train}||_2^2,$$

However, we are actually interested in the test error:

$$\frac{1}{m^{test}}||\mathbf{X}^{test}\mathbf{w} - \mathbf{y}^{test}||_2^2.$$

How can we affect the performance on the test set when we only get to observe the training set?

Assumptions on the Collection of Training and the Test Set

The i.i.d. assumption:

- the examples in each dataset are independent from each other;
- the train set and test set are identically distributed, drawn from the same shared probability distribution called the data generating distribution, or data generating process.

This probabilistic framework allows us to mathematically study the relationship between training error and test error. For example, in the linear regression algorithm, for a fixed value of the weights $\mathbf{w}$ and repeated sampling, the expected training set error is exactly the same as the expected test set error.

Challenges in Machine Learning

Underfitting and Overfitting

In practice, we

(i) first sample the training set and then choose parameters that reduce the training set error,

(ii) sample the test set.

Under this process, the expected test error is greater than or equal to the expected value of training error.

The factors determining how well a machine learning algorithm will perform are its ability to:

- make the training error small;
- make the gap between training and test error small.

These two factors correspond to 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.

Challenges in Machine Learning (cont'd)

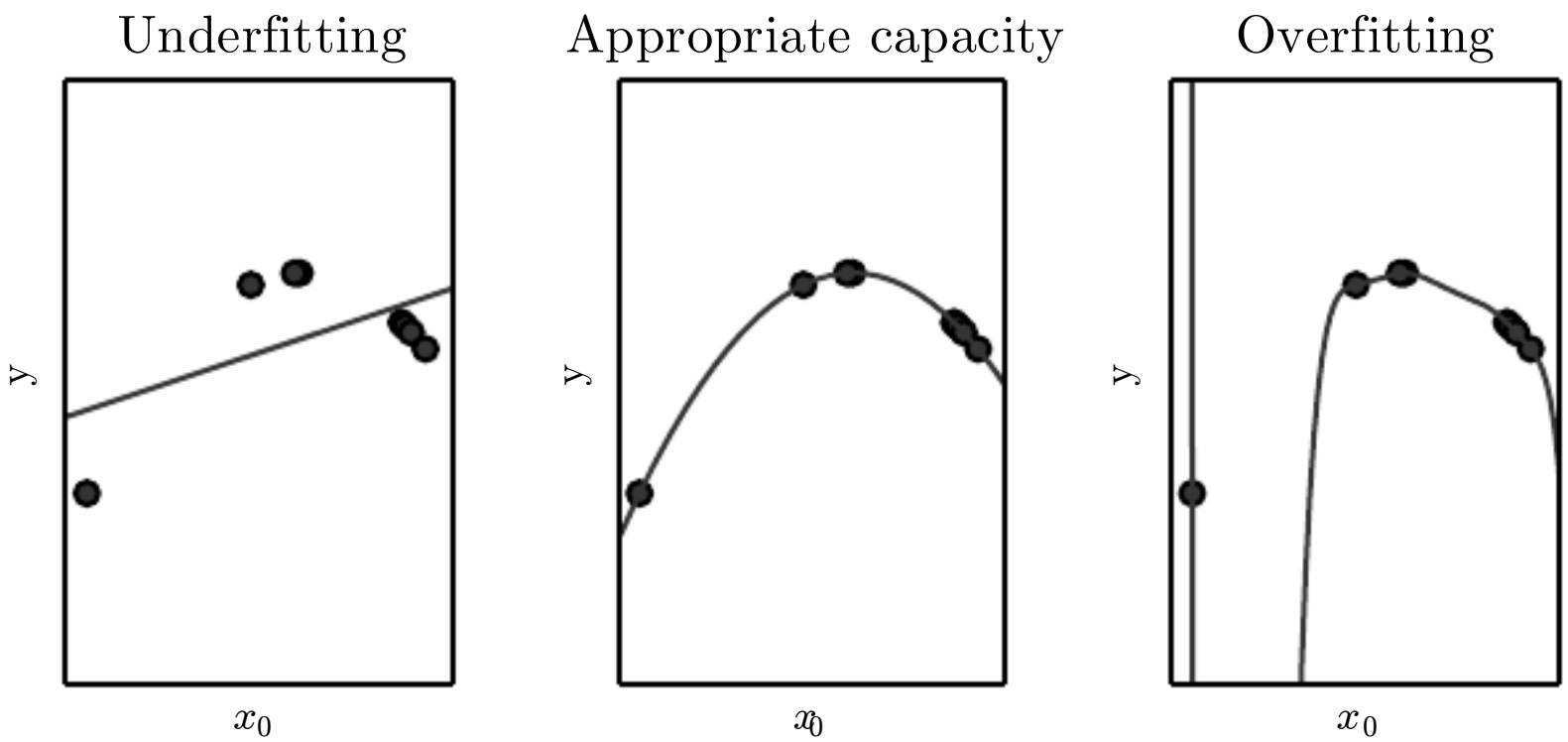
Capacity

We can control whether a model is more likely to overfit or underfit by altering its capacity.
Informally, a models capacity is its ability to fit a wide variety of functions. Models with low capacity may struggle to fit the training set. Models with high capacity can overfit, i.e., memorize properties of the training set that do not serve them well on the test set.

The capacity of a learning algorithm depends on its input or hypothesis space. The hypothesis space is the set of functions from which the learning algorithm chooses a solution. For example, the linear regression algorithm has the set of all linear functions of its input as its hypothesis space. We can generalize linear regression to include arbitrary polynomials, in its hypothesis space.

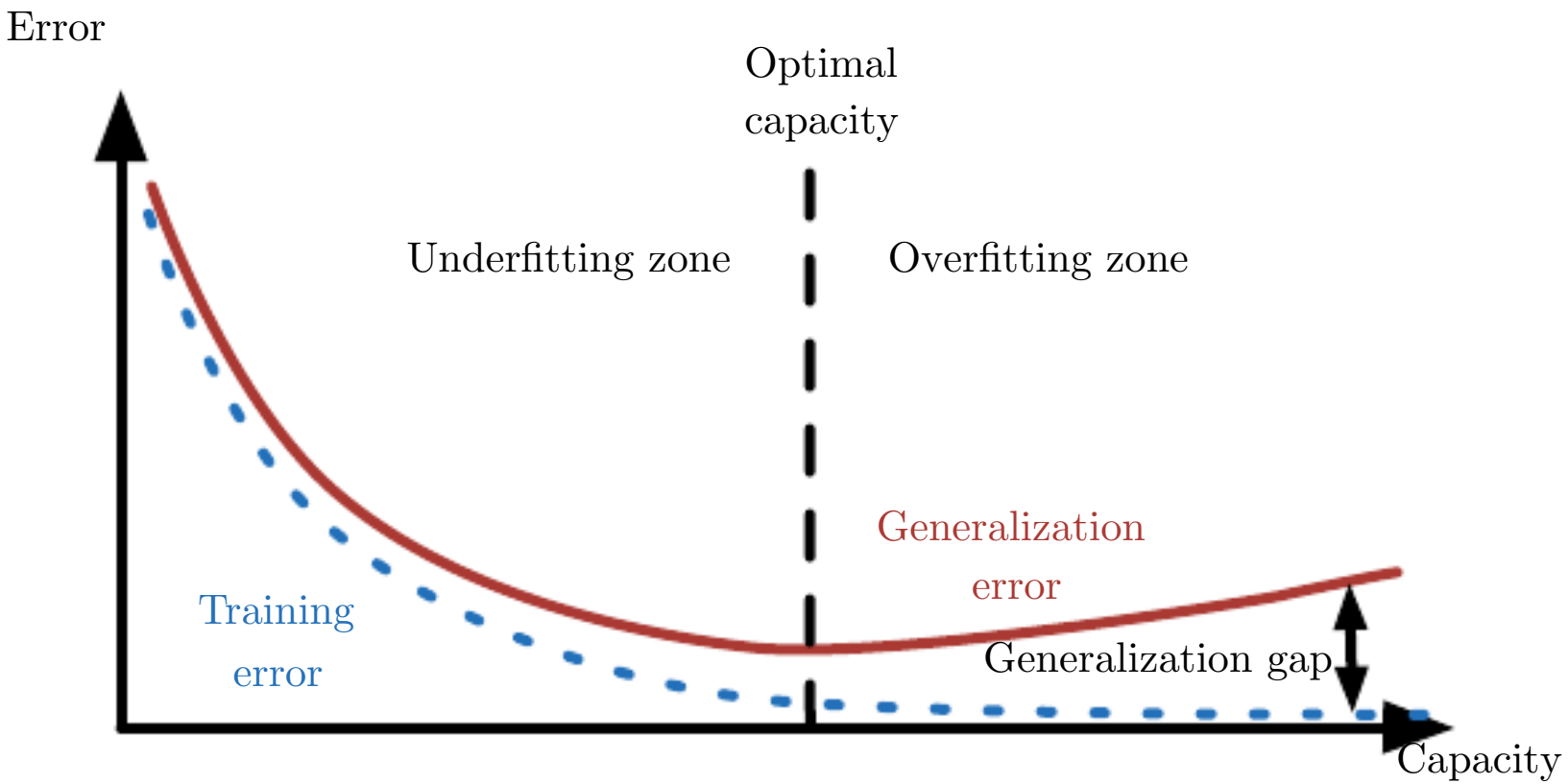
Challenges in Machine Learning (cont'd)

An Example of Underfitting and Overfitting



A linear, quadratic and degree-9 predictor attempting to fit a problem where the true underlying function is quadratic.

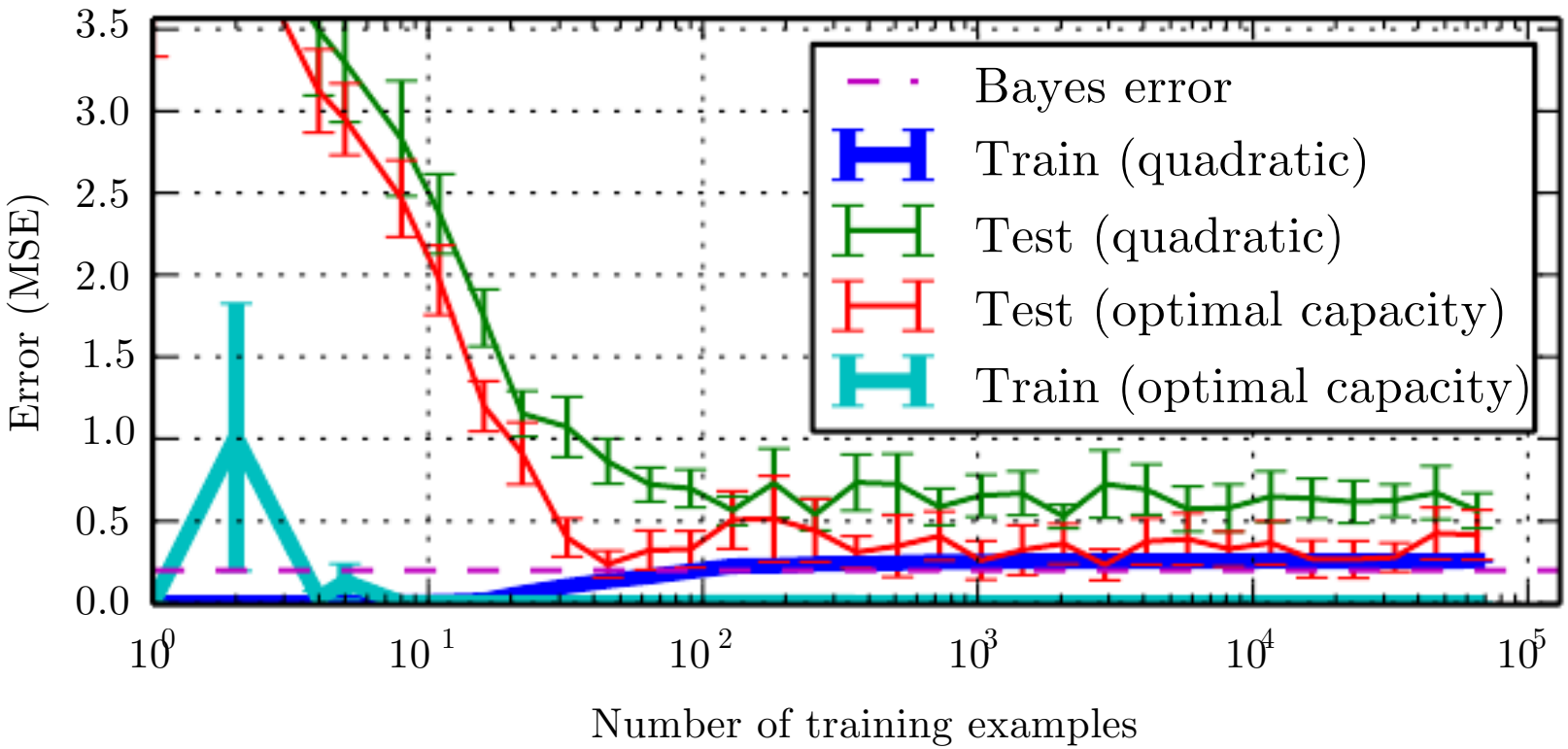
Generalization Error as a Function of Model Capacity



Typical relationship between capacity (horizontal axis) and both training (bottom curve, dotted) and generalization (or test) error (top curve, bold).

The MSE Error as a Function of the Training Data Set Size

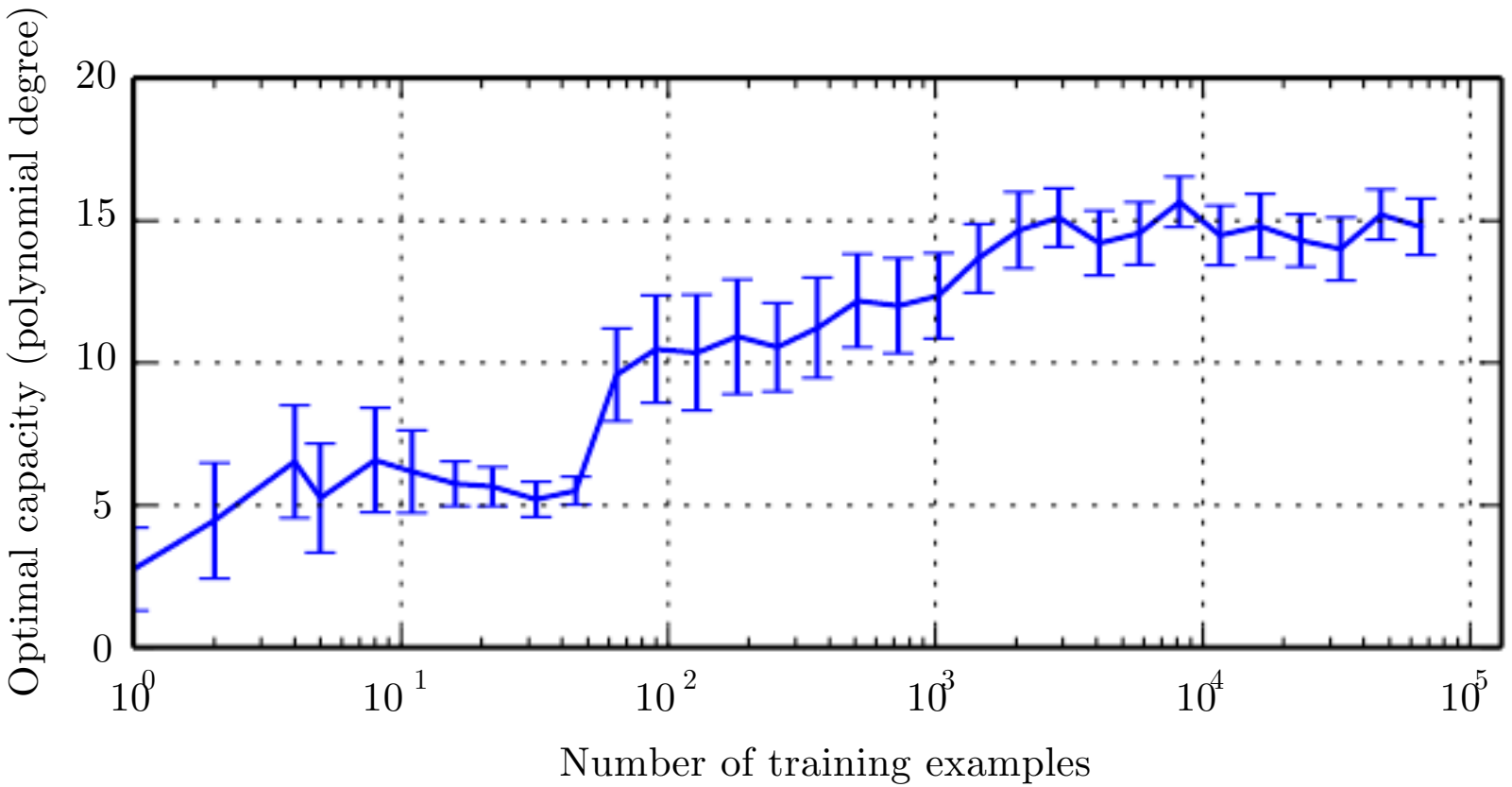
An Example



A synthetic regression problem generated a single test set, and then generated several different sizes of training set. The MSE on the train and test set for two different models: a quadratic model, while the other has its degree chosen to minimize the test error.

The Capacity as a Function of the Training Data Set Size

An Example



As the training set size increases, the optimal capacity increases. The optimal capacity (the degree of the optimal polynomial regressor) plateaus after reaching a level of sufficient complexity.

Regularization

The no free lunch theorem for machine learning 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. The no free lunch theorem implies that we must design our machine learning algorithms to perform well on a specific task.

The only method of modifying a learning algorithm we have discussed is to increase or decrease the models capacity by adding or removing functions from the hypothesis space of solutions the learning algorithm is able to choose. We gave the specific example of increasing or decreasing the degree of a polynomial for a regression problem.

Regularization (cont'd)

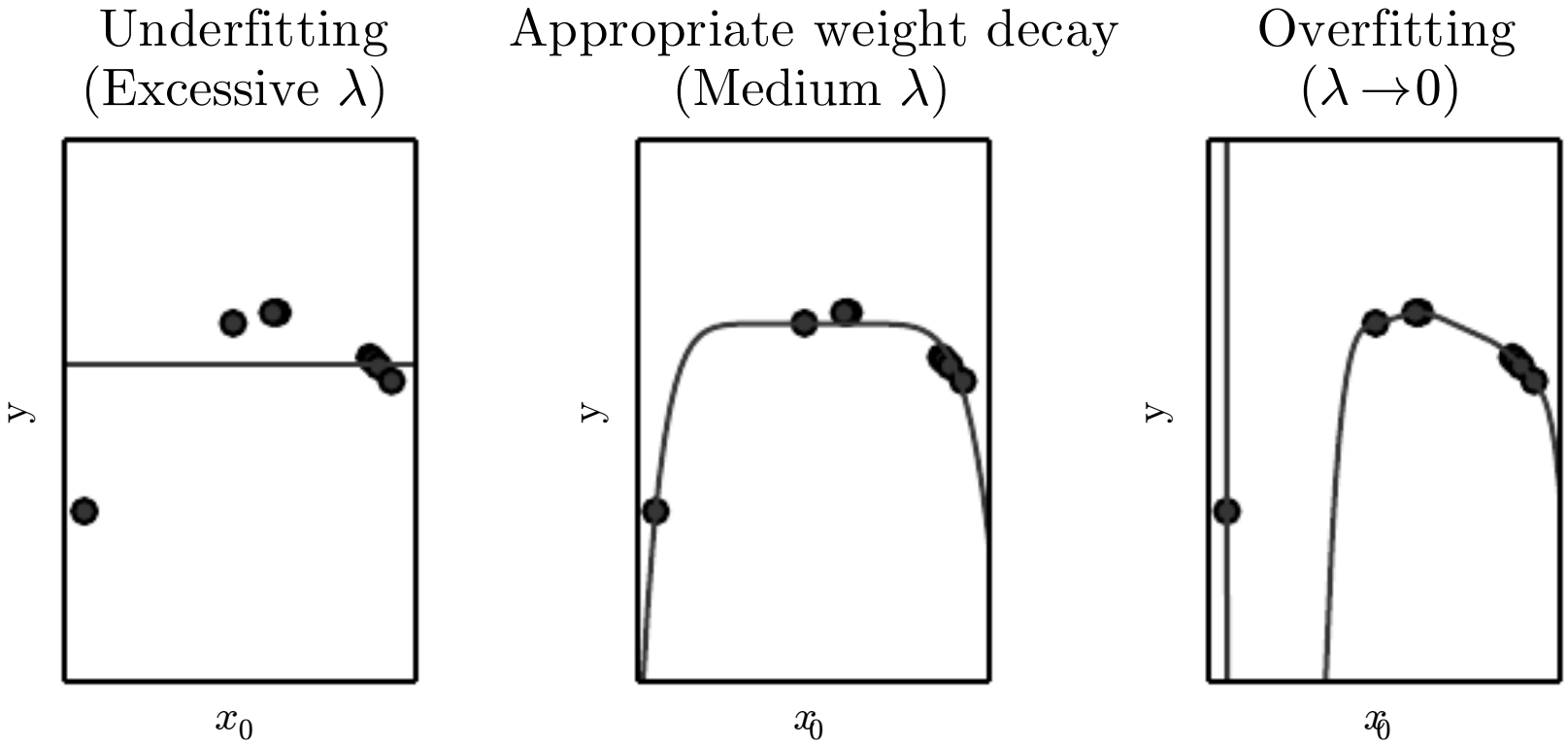
Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

Formally, we can modify the training criterion for linear regression to include weight decay. To perform linear regression with weight decay, we minimize not only the MSE on the training set, but instead a criterion $J(\mathbf{w})$ that expresses a preference for the weights to have smaller squared L^2 norm. Specifically,

$$J(\mathbf{w}) = \text{MSE}_{train} + \lambda \mathbf{w}^T \mathbf{w},$$

where λ is a value chosen ahead of time that specifies our preference for smaller weights.

Regularization (cont'd)



When $\lambda = 0$, we impose no preference, and larger λ forces the weights to become smaller. Minimizing $J(\mathbf{w})$ results in a choice of weights that make a tradeoff between fitting the training data and using small weights.

Hyperparameters

Most machine learning algorithms have several settings that we can use to control the behavior of the learning algorithm. These settings are called hyperparameters.

In polynomial regression 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.

If learned on the training set, such hyperparameters would always choose the maximum possible model capacity, resulting in overfitting. For example, we can always fit the training set better with a higher degree polynomial and a weight decay setting of $\lambda = 0$.

Hyperparameters (cont'd)

To solve this problem, we need a validation set of examples that the training algorithm does not observe.

The validation set is always constructed from the training data. Specifically, the training data is split into two disjoint subsets. One of these subsets is used to learn the parameters. The other subset is the validation set, used to estimate the generalization error during or after training, allowing for the hyperparameters to be updated accordingly.

In conclusion, the subset of data used to guide the selection of hyperparameters is called the validation set. The validation set is used to train the hyperparameters.

Basic Goals

The field of statistics gives us many tools that can be used to achieve the machine learning goal of solving a task not only on the training set but also to generalize.

Foundational concepts such as parameter estimation, bias and variance are useful to formally characterize notions of generalization, underfitting and overfitting.

Point Estimation

Let $\{\mathbf{x}^1, ..., \mathbf{x}^m\}$ be a set of m i.i.d. data points. A point estimator is a function of the data

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

The true parameter value θ is fixed but unknown, while the point estimate $\hat{\theta}_m$ is a function of the data. Since the data is drawn from a random process, any function of the data is random.

Therefore $\hat{\theta}_m$ is a random variable whose probability distribution is called the sampling distribution.

Bias

The bias of an estimator is defined as

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

where the expectation is over the data (seen as samples from a random variable) and θ is the true underlying value of θ according to the data generating distribution.

An estimator $\hat{\theta}_m$ is said to be unbiased if $\text{bias}\,(\hat{\theta}_m) = 0$, i.e., if $\mathbb{E}(\hat{\theta}_m) = \theta$.

An estimator $\hat{\theta}_m$ is said to be asymptotically unbiased if $\lim_{m\to\infty} \text{bias}(\hat{\theta}_m) = 0$, i.e., if $\lim_{m\to\infty} \mathbb{E}(\hat{\theta}_m) = \theta$.

Bernoulli Distribution Estimator of the Mean

An Example

The set of samples $\{x^1, ..., x^m\}$ with $x^i \in \{0, 1\}$, for all $i \in \{1, m\}$ are i.i.d. with a Bernoulli distribution. The Bernoulli probability function is given by $\mathbb{P}(x^i; \theta) = \theta^{x^i}(1-\theta)^{1-x^i}$.

Is the estimator $\hat{\theta}_m = \frac{1}{m}\sum_{i=1}^{m} x^i$ biased?

$$\text{bias}\,(\hat{\theta}_m) = \mathbb{E}\,[\hat{\theta}_m] - \theta = \mathbb{E}\,[\frac{1}{m}\sum_{i=1}^{m} x^i] - \theta = \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\,[x^i] - \theta$$

i.e.,

$$\text{bias}\,(\hat{\theta}_m) = \frac{1}{m}\sum_{i=1}^{m}\sum_{x^i=0}^{1}[x^i\theta^{x^i}(1-\theta)^{1-x^i}] - \theta = \frac{1}{m}\sum_{i=1}^{m}\theta - \theta = 0.$$

Gaussian Distribution Estimator of the Mean

An Example

The set of samples $\{x^1, ..., x^m\}$ are i.i.d. with a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ for all $i \in \{1, m\}$. The Gaussian probability density function is given by

$$p(x^i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2}\frac{(x^i - \mu)^2}{\sigma^2}).$$

A widely used Gaussian mean estimator is given by

$$\hat{\mu}_m = \frac{1}{m}\sum_{i=1}^{m} x^i.$$

The bias of the sample mean amounts to

$$\text{bias}\,(\hat{\mu}_m) = \mathbb{E}\,[\hat{\mu}_m] - \mu = \mathbb{E}\frac{1}{m}\sum_{i=1}^{m} x^i - \mu = \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\,x^i - \mu = 0.$$

Gaussian Distribution Estimators of the Variance

An Example

We consider here first the sample variance estimator:

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

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

$$\mathbb{E}\,\hat{\sigma}^2 = \frac{1}{m}(1-\frac{1}{m})\sum_{i=1}^{m}\mathbb{E}\,(x^i)^2 - \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j\neq i}\mathbb{E}\,x^i x^j$$

$$\mathbb{E}\,\hat{\sigma}^2 = \frac{1}{m}(1-\frac{1}{m})m(\sigma^2+\mu^2) - \frac{1}{m^2}m(m-1)\mu^2 = \frac{m-1}{m}\sigma^2.$$

Gaussian Distribution Estimators of the Variance (cont'd)

An Example

Therefore, the sample variance is a biased estimator and

$$\text{bias}\,(\hat{\sigma}^2) = \mathbb{E}\,\hat{\sigma}_m^2 - \sigma^2 = -\frac{1}{m}\sigma^2.$$

The modified sample variance

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

has variance

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

Variance and Standard Error

Recall that the variance is given by

$$\text{Var}\,(\hat{\theta}) = \mathbb{E}\,\hat{\theta}^2 - [\,\mathbb{E}\hat{\theta}\,]^2.$$

and the standard error by

$$\text{SE}\,(\hat{\theta}) = \sqrt{\text{Var}\,(\hat{\theta})}.$$

Also, the standard deviation of the mean $\hat{\mu}_m = \frac{1}{m}\sum_{i=1}^{m} x^i$ is

$$\text{Var}(\hat{\mu}_m) = \sqrt{\text{Var}[\frac{1}{m}\sum_{i=1}^{m} x^i]} = \frac{\sigma}{\sqrt{m}},$$

where σ is the variance of the samples x^i, $i \in \{1, ..., m\}$.

Variance and Standard Error (cont'd)

Example: Bernoulli Distribution

We consider the estimator $\hat{\theta}_m = \frac{1}{m}\sum_{i=1}^{m} x^i$, where the samples $\{x^i, ..., x^m\}$ are iid with Bernoulli distribution with parameter θ.

$$\text{Var}(\hat{\theta}_m) = \text{Var}(\frac{1}{m}\sum_{i=1}^{m} x^i) = \frac{1}{m^2}\sum_{i=1}^{m}\text{Var}(x^i) = \frac{1}{m^2}\sum_{i=1}^{m}\theta(1-\theta) = \frac{\theta(1-\theta)}{m}.$$

Note that the above result was also computed in the more general framework of i.i.d. random data samples in the previous slide.

Variance and Standard Error (cont'd)

Example: Gaussian Distribution Estimators of the Variance

$\{x^i, ..., x^m\}$ are iid with Gaussian distribution with (μ, σ^2).

$$\text{Var}(\frac{m-1}{\sigma^2}\tilde{\sigma}^2) = 2(m-1)$$

or

$$\frac{(m-1)^2}{\sigma^4}\text{Var}\left(\tilde{\sigma}^2\right) = 2(m-1)$$

Finally, since $\hat{\sigma}^2 = \frac{m-1}{m}\tilde{\sigma}^2$ and the relationship to χ^2 distribution

$$\text{Var}(\hat{\sigma}^2) = \frac{2(m-1)\sigma^4}{m^2}.$$

Trading Off Bias and Variance

Gaussian Distribution of the Variance

$$\text{MSE} = \mathbb{E}[\hat{\theta}_n - \theta]^2 = \text{Bias}(\hat{\theta}_n^2) + \text{Var}(\hat{\theta}_n)$$

$$\text{MSE}(\hat{\sigma}_m^2) = \text{Bias}(\hat{\sigma}_m^2)^2 + \text{Var}(\hat{\sigma}_m^2)^2 = (\frac{-\sigma^2}{m})^2 + \frac{2(m-1)\sigma^4}{m^2} = \frac{2m-1}{m^2}\sigma^4.$$

The MSE of the unbiased alternative is given by

$$\text{MSE}(\tilde{\sigma}_m^2) = \text{Bias}(\tilde{\sigma}_m^2)^2 + \text{Var}(\tilde{\sigma}_m^2)^2 = 0 + \frac{2\sigma^4}{m-1} = \frac{2}{m-1}\sigma^4.$$

Consistency

At times we may choose a biased estimator in order to minimize its variance.
We might still wish that, as the number of data points in our dataset increases, our point estimates converge to the true value of the parameter. More formally, we would like that

$$\lim_{n\to\infty} \hat{\theta}_n \xrightarrow{p} \theta.$$

The symbol $\xrightarrow{p}$ means that the convergence is in probability, i.e. for any $\varepsilon > 0$,

$$\lim_{n\to\infty} \mathbb{P}(|\hat{\theta} - \theta| > \epsilon) = 0.$$

This condition is known as weak consistency. Strong consistency refers to the almost sure convergence of $\hat{\theta}$ to θ. Almost sure convergence of a sequence of random variables $\mathbf{x}^1, \mathbf{x}^2, \ldots$ to a value $\boldsymbol{x}$ occurs when $\mathbb{P}(\lim_{n\to\infty} \mathbf{x}^n = \boldsymbol{x}) = 1$.

Consistency (cont'd)

Consistency ensures that the bias induced by the estimator is assured to diminish as the number of data examples grows.

Asymptotic unbiasedness is not equivalent to consistency.

Consider estimating the mean parameter μ of a normal distribution $\mathcal{N}(\mu, \sigma^2)$, with a dataset consisting of m samples: $\{\mathbf{x}^1, ..., \mathbf{x}^m\}$.

Use the first sample x^1 of the dataset as an unbiased estimator: $\hat{\theta}_n = x^1$. In that case, $\mathbb{E}(\hat{\theta}_n) = \theta$ so the estimator is unbiased no matter how many data points are seen. This, of course, implies that the estimate is asymptotically unbiased. However, this is not a consistent estimator as it is not the case that $\hat{\theta}_n \to \theta$ as $n \to \infty$.

The Maximum Likelihood Principle

Consider a set of m examples $\mathbb{X} = \{\mathbf{x}^1, ..., \mathbf{x}^m\}$ drawn from the unknown distribution $p_{data}(\mathbf{x})$.

Let $p_{model}(\mathbf{x}; \theta)$ be a parametric family of probability distributions over the same space indexed by θ. In other words, $p_{model}(\mathbf{x}; \theta)$ maps any configuration $\mathbf{x}$ to a real number estimating the true probability $p_{data}(\mathbf{x})$.

The maximum likelihood estimator for θ is then defined as

$$\theta_{ML} = \arg\max_{\theta} p_{model}(\mathbb{X}; \theta) = \arg\max_{\theta} \prod_{i=1}^{m} p_{model}(\mathbf{x}^i; \theta).$$

Note that it is easier to work with the formulation

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

The Maximum Likelihood Principle (cont'd)

By rescaling the cost function (dividing by m) we obtain

$$\theta_{ML} = \arg\max_{\theta} \mathbb{E}\, \log p_{model}(\mathbf{x}^i; \theta).$$

Maximum likelihood estimation is equivalent with minimizing the dissimilarity between the empirical distribution defined by the training set and the model distribution, with the degree of dissimilarity between the two measured by the KL divergence. The KL divergence is given by

$$\mathrm{DKL}(\hat{p}_{data}||p_{model}) = \mathbb{E}[\log \hat{p}_{data}(\mathbf{x}) - \log p_{model}(\mathbf{x})].$$

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

$$-\mathbb{E}\log p_{model}(\mathbf{x}).$$

Conditional Log-Likelihood and MSE

In order to predict $\mathbf{y}$ given $\mathbf{x}$, we need to estimate the conditional probability $\mathbb{P}(\mathbf{y}|\mathbf{x};\theta)$. This is actually the most common situation because it forms the basis for most supervised learning, the setting where the examples are pairs $(\mathbf{x}, \mathbf{y})$.

If $\mathbf{X}$ represents all our inputs and $\mathbf{Y}$ all our observed targets, then the conditional maximum likelihood estimator is

$$\theta_{ML} = \arg\max_{\theta} \mathbb{P}(\mathbf{Y}|\mathbf{X};\theta).$$

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

$$\theta_{ML} = \arg\max_{\theta} \sum_{i=1}^{m} \log \mathbb{P}(\mathbf{y}^i|\mathbf{x}^i;\theta).$$

Example: Linear Regression

Previously, we motivated linear regression as an algorithm that learns to take an input x and produce an output value y. The mapping from x to y is chosen to minimize mean squared error, a criterion that we introduced more or less arbitrarily. We now revisit linear regression from the point of view of maximum likelihood estimation. Instead of producing a single prediction y, we now think of the model as producing a conditional distribution $p(y|x)$. We can imagine that with an innitely large training set, we might see several training examples with the same input value x but different values of y. The goal of the learning algorithm is now to t the distribution $p(y|x)$ to all of those different y values that are all compatible with x.

Example: Linear Regression (cont'd)

To derive the same linear regression algorithm we obtained before, we define $p(y|x) = \mathcal{N}(y; \hat{y}(\mathbf{x}; \mathbf{w}), \sigma^2)$. The function $\hat{y}(\mathbf{x}; \mathbf{w})$ gives the prediction of the mean of the Gaussian. In this example, we assume that the variance is fixed to some constant σ^2 chosen by the user. We will see that this choice of the functional form of $p(y|x)$ causes the maximum likelihood estimation procedure to yield the same learning algorithm as we developed before.

Example: Linear Regression (cont'd)

The conditional density of $\mathbf{y}$, given $\mathbf{x} = \boldsymbol{x}$, is a Gaussian with mean $\mu(\boldsymbol{x})$ that is a learned function of $\boldsymbol{x}$, with unconditional variance σ^2. Since the examples are assumed to be i.i.d., the conditional log-likelihood

$$\log \mathbb{P}(\mathbf{Y}|\mathbf{X};\theta) = \sum_{i=1}^{m} \log \mathbb{P}(\mathbf{y}^i|\mathbf{x}^i;\theta)$$

$$\log \mathbb{P}(\mathbf{Y}|\mathbf{X};\theta) = \sum_{i=1}^{m} \frac{-1}{2\sigma^2}||\hat{\mathbf{y}}^i - \mathbf{y}^i)||^2 - m \log \sigma - \frac{m}{2}\log(2\pi)$$

where $\hat{\mathbf{y}}^i = \mu(\mathbf{x}^i)$ is the output of the linear regression on the i-th input $\mathbf{x}^i$ and m is the dimension of the $\mathbf{y}$ vectors.

Example: Linear Regression (cont'd)

If σ is fixed, maximizing the above is equivalent (up to an additive and a multiplicative constant that do not change the value of the optimal parameter) to minimizing the training set mean squared error, i.e.,

$$\text{MSE}_{train} = \frac{1}{m} \sum_{i=1}^{m} ||\hat{\mathbf{y}}^i - \mathbf{y}^i)||^2.$$

Note that maximizing the log-likelihood with respect to $\mathbf{w}$ yields the same estimate of the parameters $\mathbf{w}$ as does minimizing the MSE. The two criteria have different values but the same location of the optimum. This justifies the use of the MSE as a maximum likelihood estimation procedure.

Properties of Maximum Likelihood

The maximum likelihood estimator has the property of consistency. That parametric mean squared error 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.