ECBM E4040
Neural Networks and Deep Learning

Machine Learning Basics

Zoran Kostić
Columbia University
Electrical Engineering Department &
Data Sciences Institute

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

References and Acknowledgments

- Deep Learning by Ian Goodfellow, Yoshua Bengio and Aaron Courville, http://www.deeplearningbook.org/ , chapter 5
- John Paisley, COMS W4721 Machine Learning for Data Science, http://www.columbia.edu/~jwp2128/Teaching/W4721/Spring2017/W4721Spring2017.html
- A Course in Machine Learning, by Hal Daumé III, http://ciml.info/
- Nvidia DLI Teaching Kit

Outline - Machine Learning Basics

- Concepts
- Tasks
- Performance
- Experience
- Generalization
- Statistical Underpinnings: Estimators
- Data sets

Learning - Google Search

learn·ing

/ˈlərniNG/

noun

noun: **learning**

the acquisition of knowledge or skills through experience, study, or by being taught.
"these children experienced difficulties in learning"
synonyms: study, studying, education, schooling, tuition, teaching, academic work; research
"a center of learning"

- knowledge acquired through experience, study, or being taught.
  "I liked to parade my learning in front of my sisters"
  synonyms: scholarship, knowledge, education, erudition, intellect, enlightenment, illumination, edification, book learning, information, understanding, wisdom
  "the astonishing range of his learning"
  antonyms: ignorance

SOFTWARE WHICH IMPLEMENTS AN ALGORITHM
traditional software engineering

vs.

MODEL WHICH CAN LEARN AN ALGORITHM
implemented as (or in) software or hardware

Machine Learning

- statistically estimate complicated functions

Machine Learning

Machine learning is

- a form of applied statistics with emphasis on the use of computers to statistically estimate complicated functions
- and a decreased emphasis on proving confidence intervals around these functions

Machine Learning - Related Subjects

Statistics

Detection and Estimation

Pattern recognition

- **Applied statistics**

Adaptive signal processing

- **Closed loop control: examples in aircraft control**
- **Applications in communications - radio channel equalizers in smartphones**

Machine Learning - Key Concepts

- supervised learning vs. unsupervised learning
- generalize to new data

Machine Learning - Key Concepts

- learning algorithm
  - a cost function, a model, and a dataset
- optimization algorithm
- fitting the training data
- finding patterns that generalize to new data

- hyperparameters external to the learning algorithm
- supervised learning vs. unsupervised learning
- example: linear regression

Machine Learning Algorithm

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

Machine Learning Algorithm

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, *T*

Learning is the means of attaining the ability to perform a task.

Key machine learning **tasks:**

- Classification
- Regression
- Density or Probability Function Estimation

Key Machine Learning Tasks

Classification

Regression

Density or Probability Function Estimation

Task: Classification

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

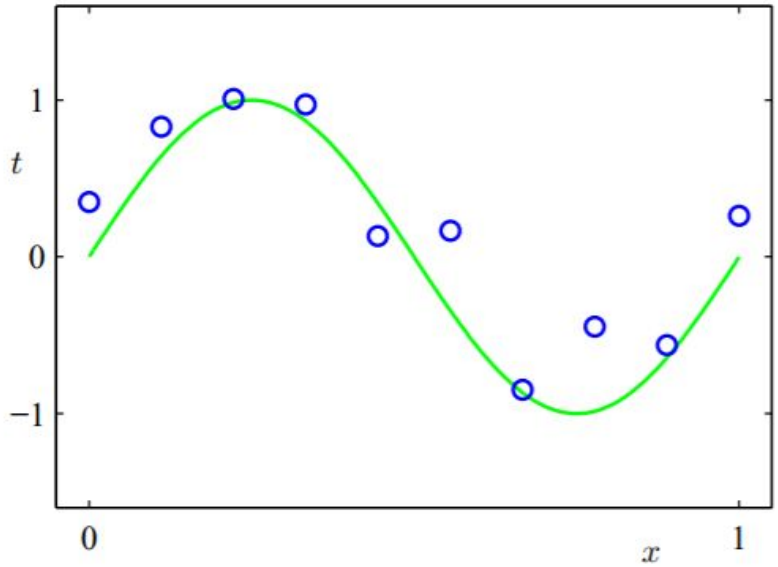
- The learning algorithm is usually asked to produce a function $f : \mathbb{R}^n \to \{1, ..., k\}$ which may then be applied to any input. Here the output of $f(\mathbf{x})$ can be interpreted as an estimate of the category that $\mathbf{x}$ belongs to. There are other variants of the classification 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.

Task: Regression

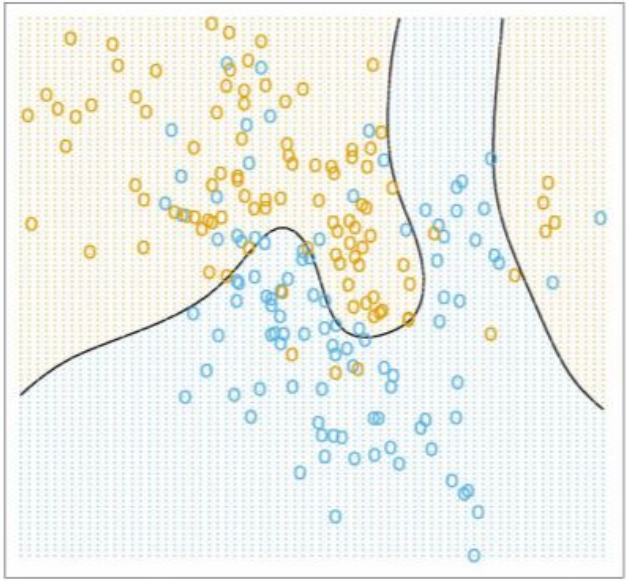
The computer program is asked to predict a numerical value given input data.

- 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.

Regression vs. Classification



(a) Regression



(b) Classification

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 $p(x_i|\mathbf{x}_{-i})$.

Applied Machine Learning Tasks

Transcription

Translation

Structured Output

Synthesis, Anomaly Detection, ...

Task: Transcription

Observe a relatively unstructured representation of some kind of data and transcribe it into discrete, textual form.

- optical character recognition (Google Street View)
- speech recognition

Task: Machine Translation

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.

Task: Structured Output

Output is a vector (multi-data structure) with important relationships between the different elements. Broad category!

- pixel-wise segmentation of images: assign every pixel in an image to a specific category
- image captioning: output a natural language sentence describing the image

Task: Anomaly Detection

Sift through a set of events or objects, and flag some of them as being unusual or atypical.

- credit card fraud detection

Task: Synthesis and Sampling

Generate new examples that are similar to those in the training data

- media applications: paint movies, create artistic images in a particular style
- speech synthesis: emit an audio waveform containing a spoken version of a sentence (variability is good)

Machine Learning Concepts

Performance Measure

Experience: Supervised vs. Unsupervised

Generalization: Capacity, Overfitting and Underfitting

Data Sets: Training, Validation and Test

Regularization, No Free Lunch Theorem

Machine Learning
Performance Measure

The performance measure $\mathcal{P}$ is specific to the task $\mathcal{T}$ being carried 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.

Machine Learning
The Experience ε

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

Machine Learning - Unsupervised Learning

Unsupervised learning involves observing several examples of a random vector **x**,

and attempting to implicitly or explicitly learn the probability distribution p(**x**),

or some interesting properties of that distribution.

There is no instructor/teacher, the algorithm must learn to make sense of data without it.

Machine Learning - Supervised Learning

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**, e.g., estimating $p(\mathbf{y}|\mathbf{x})$.

The target **y** is provided by an instructor or teacher that shows the machine learning system what to do.

Machine Learning - Generalization

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 with.

Machine Learning - Generalization

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.

Machine Learning - Generalization

The goal of machine learning is to reduce the generalization error, also called the test error.

The generalization error is the expected value of the error on a new input.

The expectation is taken across different possible inputs, drawn from the distribution of inputs we expect the system to encounter in practice.

Generalization/Test Error

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:

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

Generalization/Test Error

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?

Generalization/Test Error vs. Training Error

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.

Generalization/Test Error under Practical Constraints

we

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

(ii) then sample the test set.

Generalization/Test Error under Practical Constraints

In practice, we

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

(ii) then sample the test set.

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

Generalization/Test Error vs. Training Error
Minimizing the Errors

The Goals

- minimize the training error;
- minimize the gap between training error and test error

Generalization/Test Error vs. Training Error
Minimizing the Errors - Underfitting

The Goals

- minimize the training error;
- minimize the gap between training error and test error

Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set.

Generalization/Test Error vs. Training Error
Minimizing the Errors - Overfitting

The Goals

- minimize the training error;
- minimize the gap between training error and test error

Overfitting occurs when the gap between the training error and test error is too large.

Generalization/Test Error vs. Training Error
Minimizing the Errors

The Modified Goals

- minimize the training error “just enough”;
- minimize the gap between training and test error

There is such a thing as too much minimization of the training error.

Generalization/Test Error vs. Training Error
Overfitting, Underfitting and Model Capacity

Model capacity represents 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

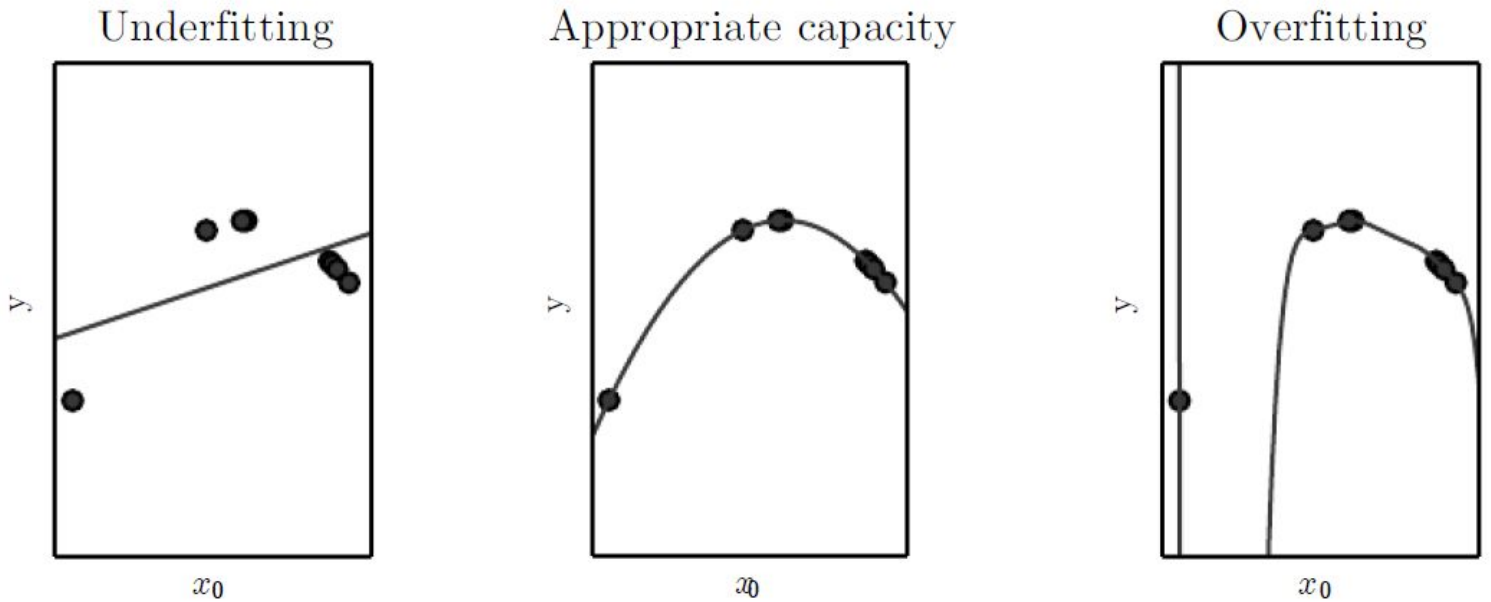
Generalization/Test Error vs. Training Error
Overfitting, Underfitting and Model Capacity

One can change a model to make it likely to overfit/underfit by having altered its capacity.

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.

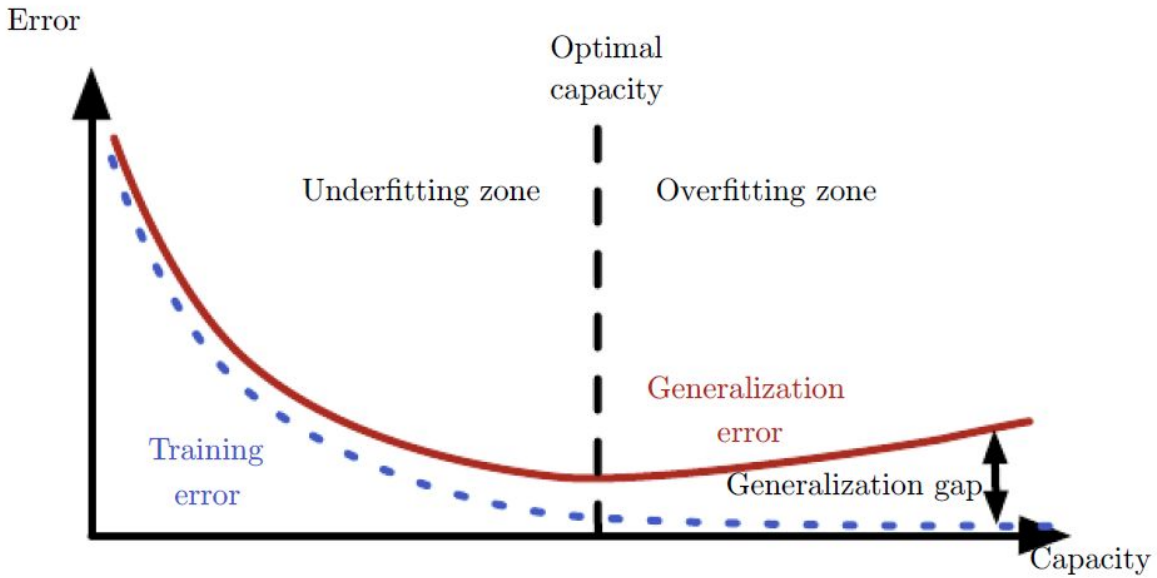
For example, the linear regression algorithm has the set of all linear functions of its input as its hypothesis space. We can generalize linear regression to include arbitrary polynomials, in its hypothesis space.

Examples of
Overfitting, Underfitting and Model Capacity



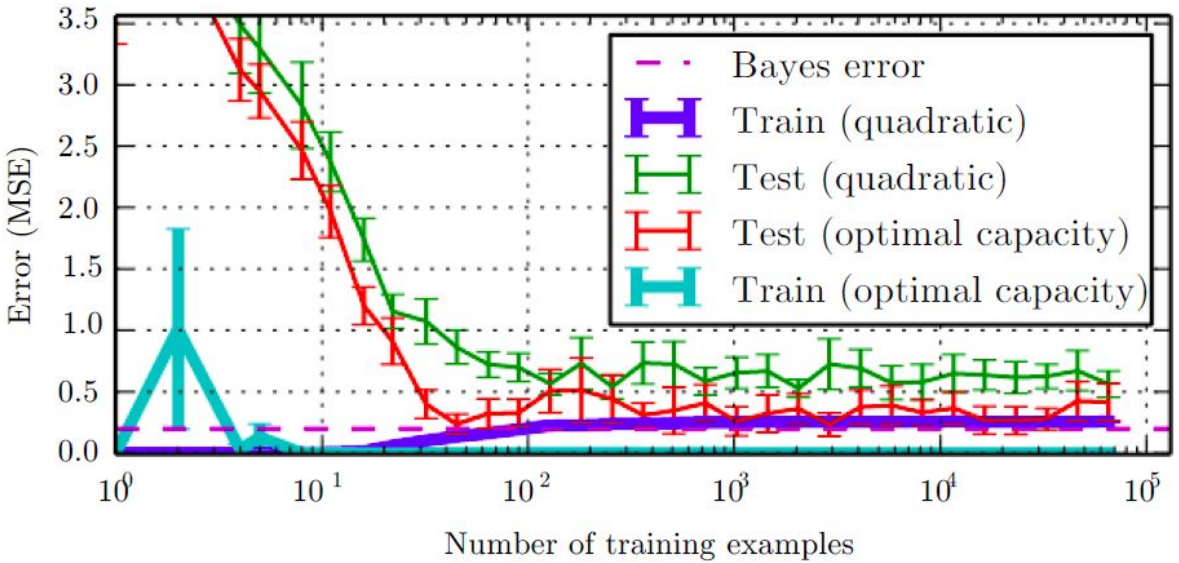
A linear, quadratic and degree-9 predictor attempting to fit a problem where the true underlying function is quadratic.

Behavior of
Overfitting, Underfitting and Model Capacity



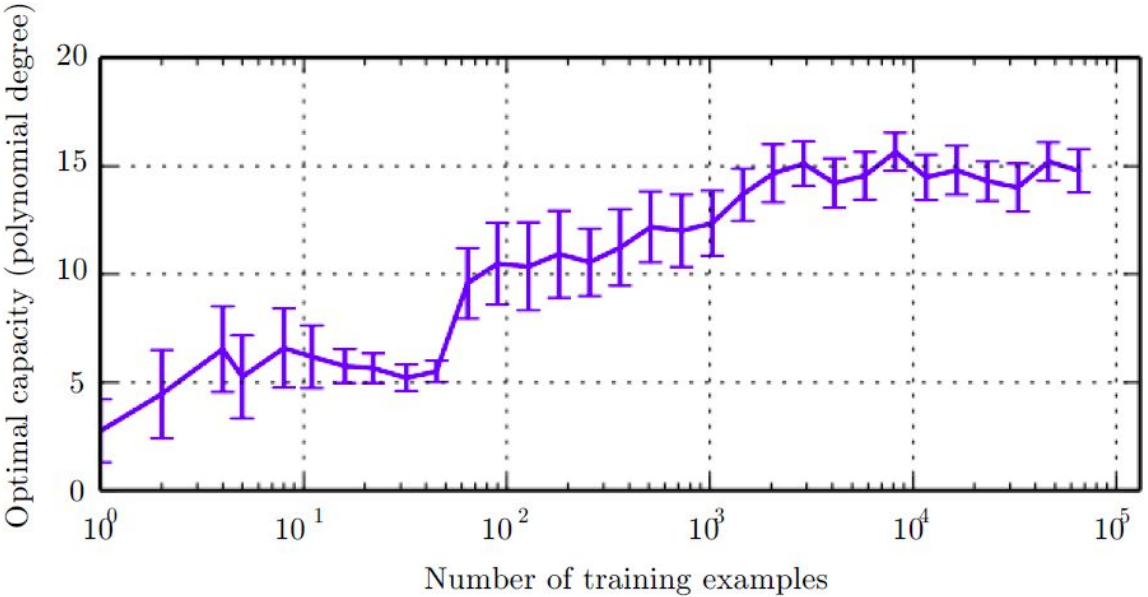
Typical relationship between capacity (horizontal axis) and both training (bottom curve, dotted) and generalization (or test) error (top curve, bold).

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



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



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.

Training, Validation and Test Data
Hyperparameters

Many machine learning algorithms have settings that one can use to control the behavior of the learning algorithm.

These settings are called hyperparameters, choices about the algorithm that we set rather than learn.

Examples:

- **Polynomial regression: the degree of the polynomial acts as a capacity hyperparameter**
- **The λ value used to control the strength of weight decay is another hyperparameter**

Training, Validation and Test Data
Hyperparameters

If learned on the training set, 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 0.**

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

Hyperparameters are very problem-dependent: Must try them all out and see what works best.

Validation Data Set

The validation set is always constructed from the training data.

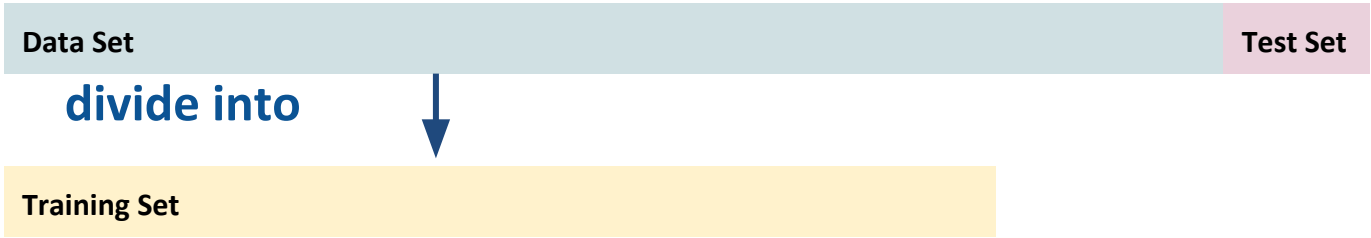
Specifically, the training data is split into two disjoint subsets.

- **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.**

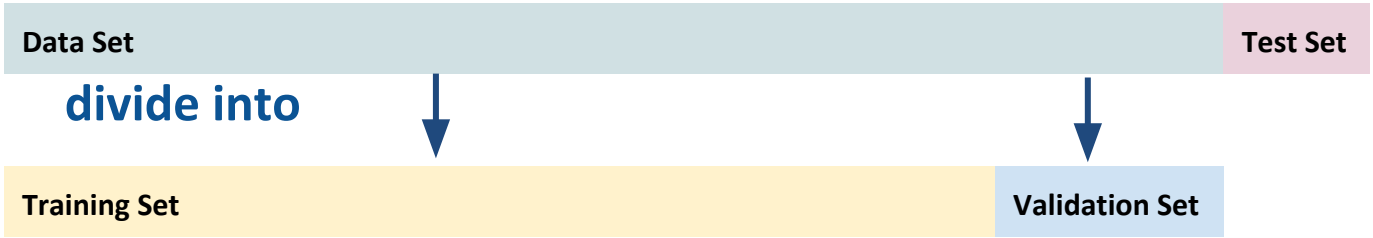
In conclusion, the subset of data used to guide the selection of hyperparameters is called the validation set. The validation set is used to train the hyperparameters.

Setting Hyperparameters

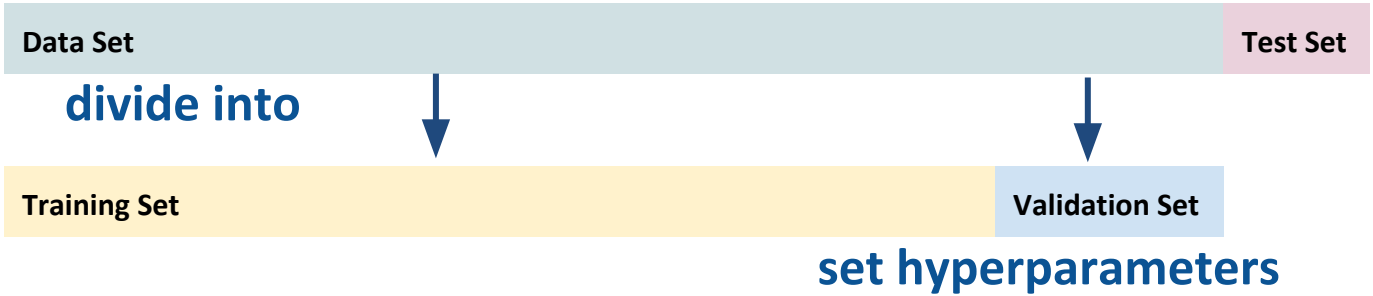
Setting Hyperparameters



Setting Hyperparameters



Setting Hyperparameters

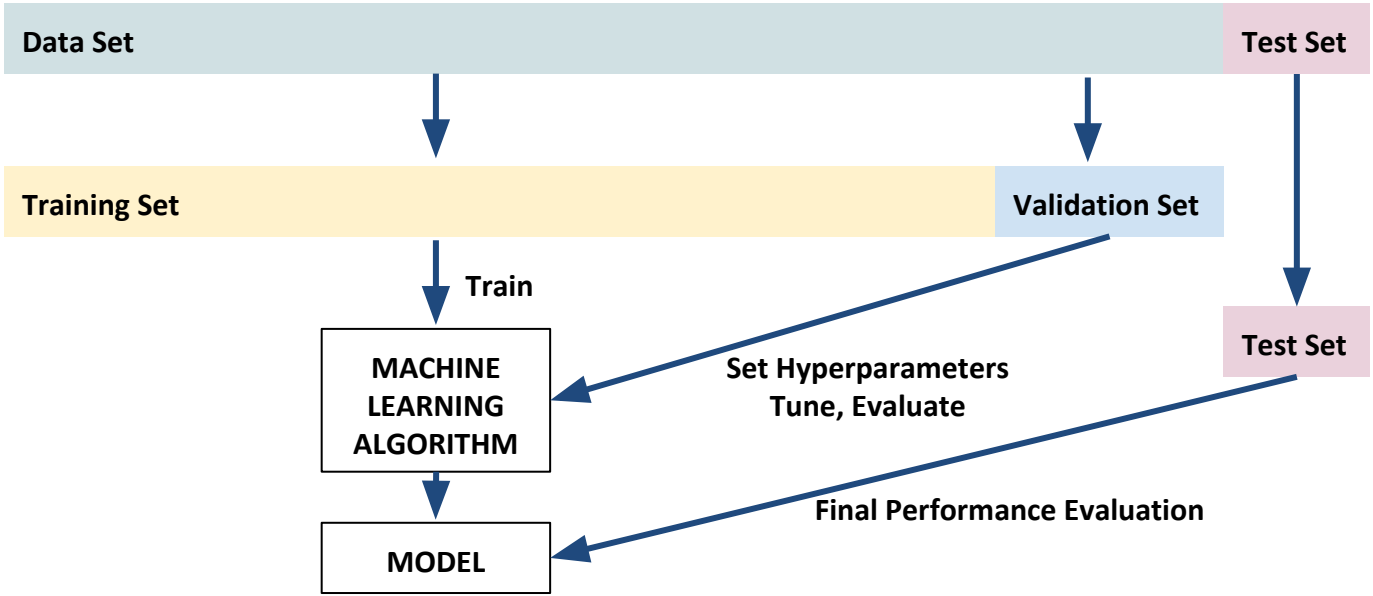


Setting Hyperparameters, Testing

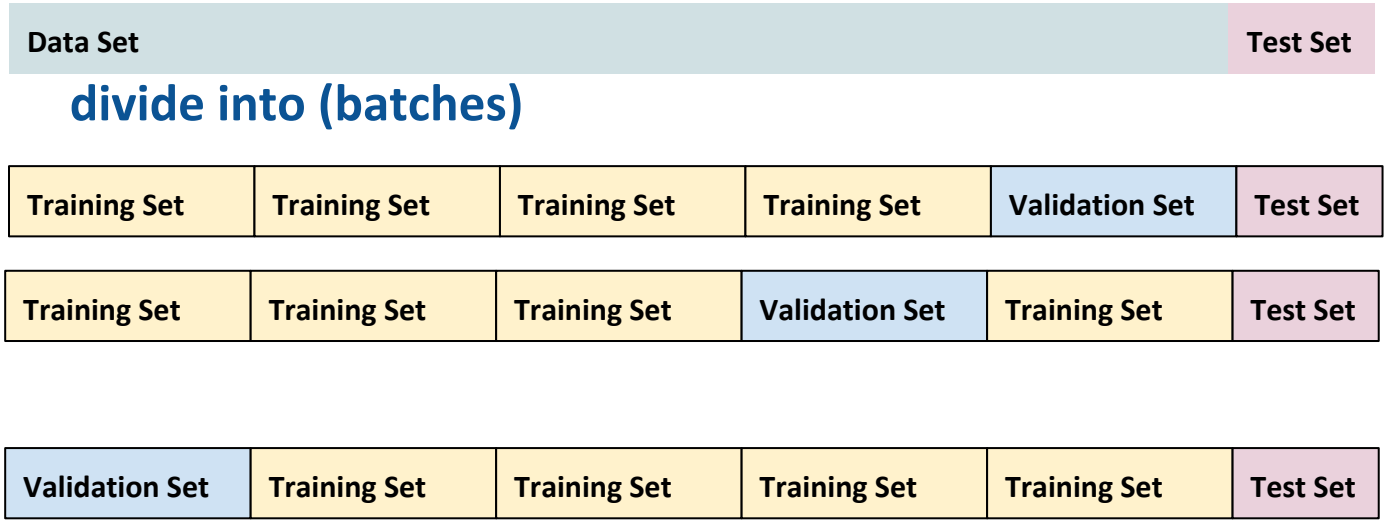
Setting Hyperparameters, Application

when training, validation and test are completed, the model can be applied to New Data which was not seen before

Setting Hyperparameters, Application



Setting Hyperparameters, Using Cross-Validation



Cross-Validation

Primary method for estimating a tuning parameter λ (such as subset size).

A model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.

The procedure for K-fold cross-validation is very simple:

1. **Randomly split the data into K roughly equal groups.**
2. **Learn the model on K - 1 groups and predict the held-out Kth group.**
3. **Do this K times, holding out each group once.**
4. **Evaluate performance using the cumulative set of predictions.**

Example: http://scikit-learn.org/stable/modules/cross_validation.html

No Free Lunch Theorem

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.

No Free Lunch Theorem

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

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

Regularization for Linear Regression

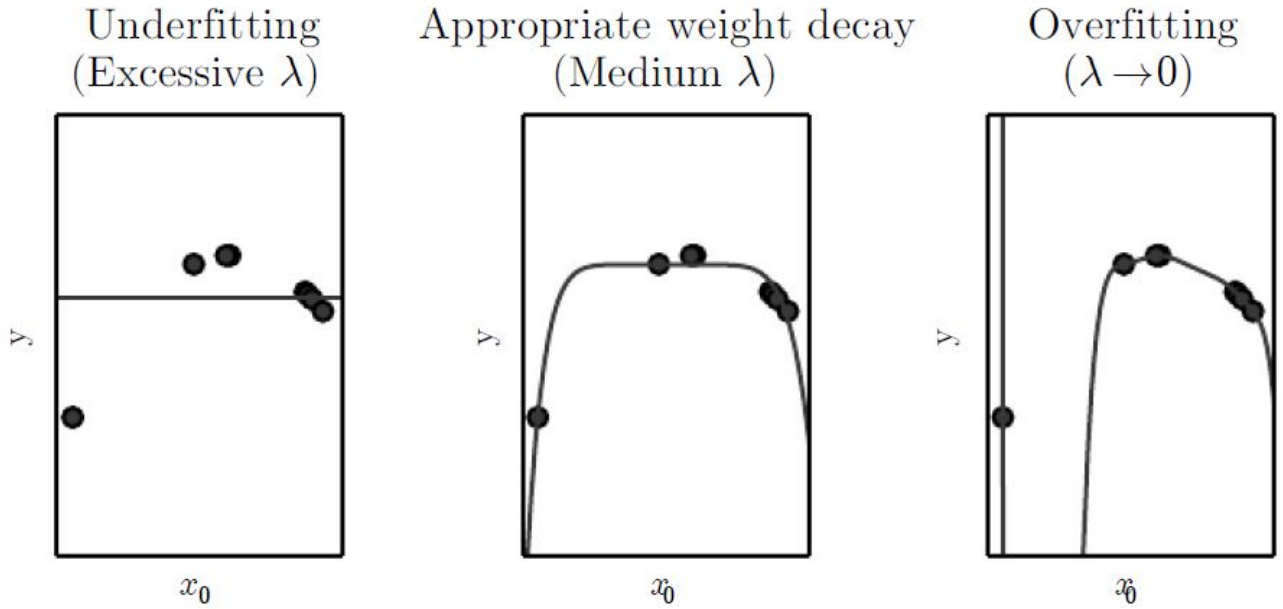
Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

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 for Linear Regression - Weight Decay



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.

Machine Learning

Linear Regression

Mean Square Error

Linear Regression

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

Linear Regression

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

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

Linear Regression - Find min Using Derivative

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

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

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

Linear Regression - Normal Equation

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

or finally

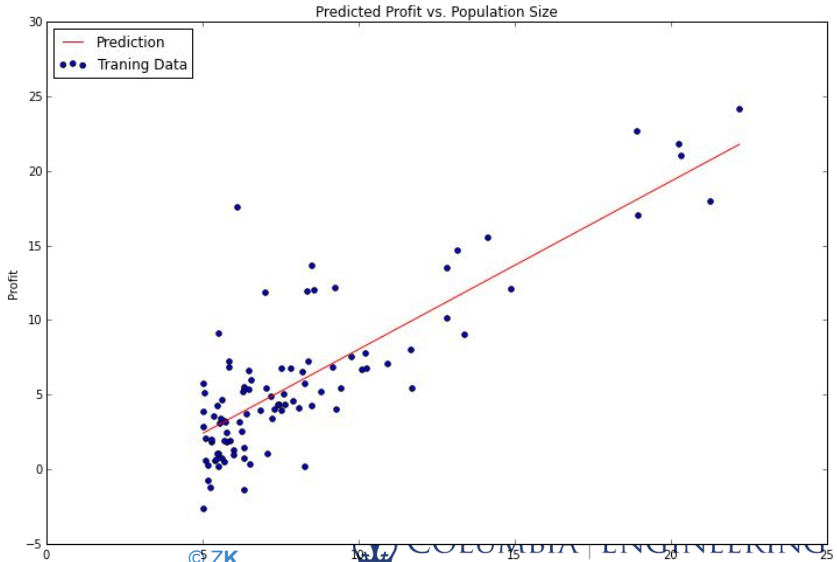
$$\mathbf{w} = (\mathbf{X}^{(train)T}\mathbf{X}^{train})^{-1}\mathbf{X}^{(train)T}\mathbf{y}^{train}.$$

This system of equations is known as the normal equations.

Linear Regression
Practical Implementation in Python

https://github.com/jdwittenauer/ipython-notebooks/blob/master/notebooks/ml/ML-Exercise1.ipynb

- implementing linear regression using gradient descent to minimize the cost function
- Predicting profit for a food truck



Machine Learning - Tools for Characterizing the Generalization

Estimation

Bias

Variance and Standard Deviation

Trading off Bias and Variance and MSE

Formalizing the Generalization
Statistical Tools

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 Estimator

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 $\lim_{m\to\infty} \mathbb{E}(\hat{\theta}_m) = \theta$.

Example: Bernoulli Distribution
Estimator of the Mean

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.,

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

Example: Gaussian Distribution
Estimator of the Mean

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

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

Example: Gaussian Distribution
(Sample) Estimator of the Variance

We consider here first the sample variance estimator:

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

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

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

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

Example: Gaussian Distribution
(Sample) Estimator of the Variance

Therefore, the sample variance is a biased estimator and

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

Example: Gaussian Distribution
(Modified Sample) Estimator of the Variance

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, $i \in \{1, ..., m\}$.

Example: Bernoulli Distribution
Variance and Standard Error

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.

Example: Gaussian Distribution
Variance and Standard Error

$\{x^i, ..., x^m\}$ 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
Example: 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

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

Trading off Bias and Variance
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 $\mathbb{P}(\lim_{n\to\infty} \mathbf{x}^n = \boldsymbol{x}) = 1$.

Trading off Bias and Variance

Consistency

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 $n \to \infty$.

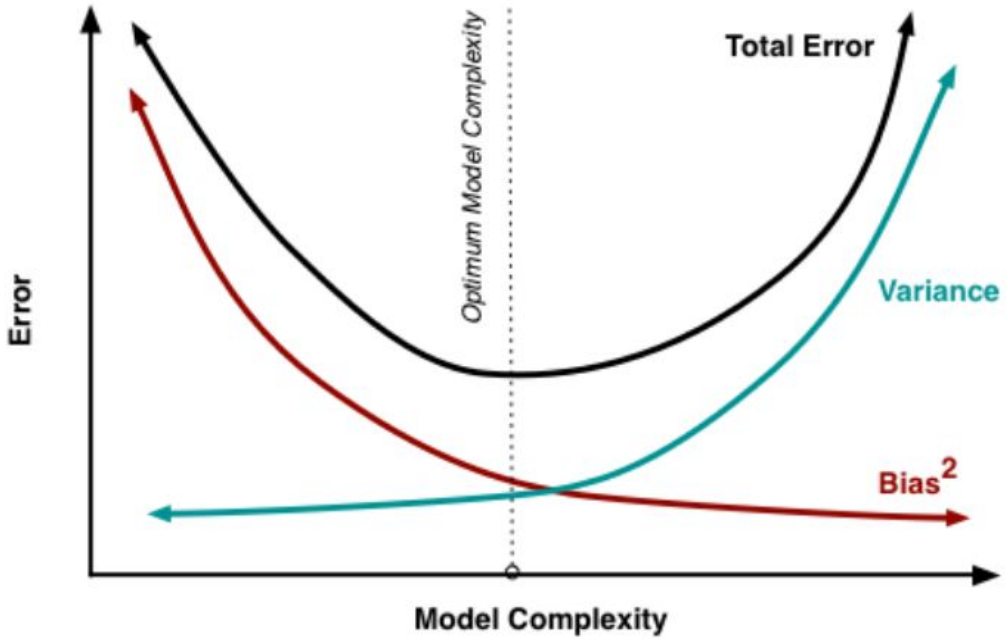
Bias / Variance Tradeoff

As we add more and more parameters to our model, its complexity increases, which results in increasing variance and decreasing bias, i.e., overfitting.

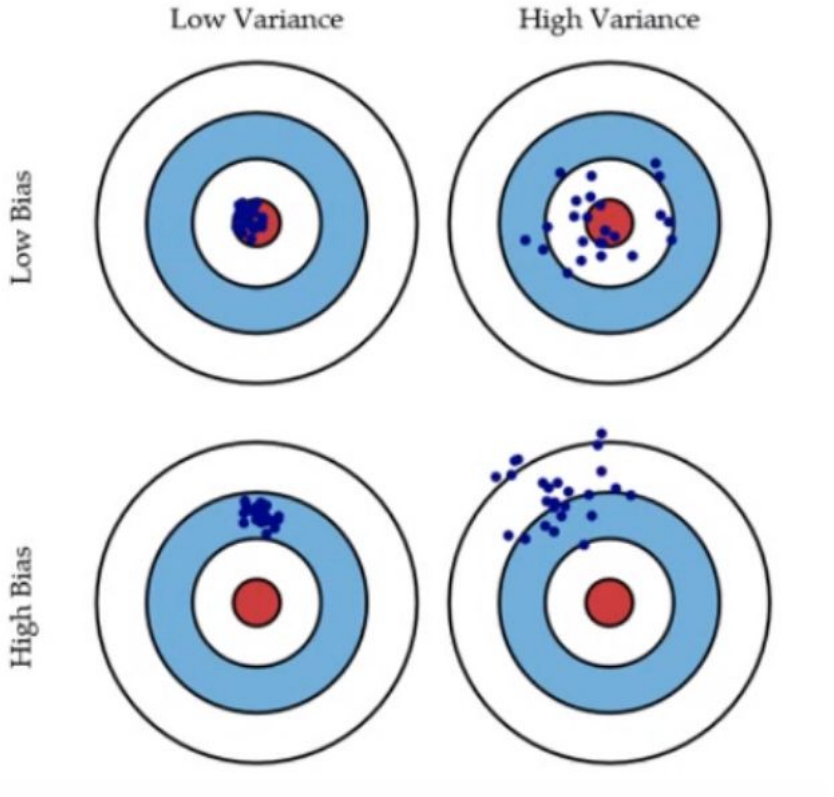
So we need to find out one optimum point in our model where the decrease in bias is equal to increase in variance.

In practice, there is no analytical way to find this point. So how to deal with high variance or high bias?

To overcome underfitting or high bias, we can basically add new parameters to our model so that the model complexity increases, and thus reducing high bias.



Bias / Variance Tradeoff



Machine Learning - Estimators

Maximum Likelihood Estimate (MLE)
Bayesian Estimate

Maximum A-posteriori Estimation (MAP)

Estimators

With large systems/number of data points

- “the best” is in statistical sense.

And therefore, single numbers are not the goal

- but distributions are,
- where in practice the numbers which are obtained in calculations are drawn from those distributions.

Estimators
Based on 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

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

Estimators
Based on Maximum Likelihood Principle

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

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

Conditional Log-Likelihood and MSE
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.

Conditional Log-Likelihood and MSE
Example: Linear Regression

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 infinitely 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 fit the distribution p(y|x) to all of those different y values that are all compatible with x.

Conditional Log-Likelihood and MSE
Example: Linear Regression

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.

Conditional Log-Likelihood and MSE
Example: Linear Regression

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.

Conditional Log-Likelihood and MSE
Example: Linear Regression

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.

Conditional Log-Likelihood and MSE
Example: Linear Regression

This justifies the use of MSE as a maximum likelihood estimation procedure!

Conditional Log-Likelihood and MSE

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.

Basics of Bayesian Statistics

There are two main modeling approaches/perpectives in statistics:

(i) Frequentist statistics. In the frequentist perspective on statistics the true parameter value $\boldsymbol{\theta}$ is fixed but unknown, while the point estimate $\hat{\boldsymbol{\theta}}$ is a random variable on account of it being a function of the data. The data is assumed to be random.

(ii) Bayesian statistics. The Bayesian perspective on statistics uses probability to reflect degrees of certainty of states of knowledge. The data is directly observed and so it is not assumed to be random. On the other hand, the true parameter $\boldsymbol{\theta}$ is unknown or uncertain and thus it is represented as a random variable.

Bayes Rule

Knowledge about $\boldsymbol{\theta}$ is represented using the prior probability distribution (or prior), $p(\boldsymbol{\theta})$.
Let $\{x^1, ..., x^m\}$ be a set of data samples.
Our belief about $\boldsymbol{\theta}$ is given by the posterior distribution via Bayes rule:

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

where $p(x^1, ..., x^m|\boldsymbol{\theta})$ is the likelihood of observing the data samples $\{x^1, ..., x^m\}$ given $\boldsymbol{\theta}$.

Comparison of MLE and Bayesian Estimates

Unlike the maximum likelihood point estimate of $\boldsymbol{\theta}$, the Bayesian makes decision with respect to a full distribution over $\boldsymbol{\theta}$. For example, after observing m examples, the predicted distribution over the next data sample, x^{m+1}, is given by

$$p(x^{m+1}|x^1,...,x^m) = \int_{\mathbb{D}} p(x^{m+1},\boldsymbol{\theta}|x^1,...,x^m)d\boldsymbol{\theta}$$
$$= \int_{\mathbb{D}} p(x^{m+1}|x^1,...,x^m,\boldsymbol{\theta})p(\boldsymbol{\theta}|x^1,...,x^m)d\boldsymbol{\theta}.$$

Here each value of $\boldsymbol{\theta}$ with positive probability density contributes to the prediction of the next example, with the contribution weighted by the posterior density itself. After having observed $\{x^1,...,x^m\}$, if we are still quite uncertain about the value of $\boldsymbol{\theta}$, then this uncertainty is incorporated directly into any predictions we might make.

Comparison of MLE and Bayesian Estimates

The frequentist statistics addresses the uncertainty in a given point estimator of θ by evaluating its variance. The variance of the estimator is an assessment of how the estimate might change with alternative samplings of the observed (or training) data.

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.

The Bayesian prior shifts the 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 more smooth. One important effect of the prior is to actually reduce the uncertainty (or entropy) in the posterior density over θ.

A Bayesian Approach to Learning
Example: Linear Regression

We learn a linear mapping from an input vector $\mathbf{x} \in \mathbb{R}^n$ to 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.
Given m training samples $(\mathbf{X}^{train}, \mathbf{y}^{train})$, the prediction of y over the entire training set is given by

$$\hat{\mathbf{y}}^{train} = \mathbf{X}^{train}\mathbf{w}.$$

Expressed as Gaussian conditional distribution on $\mathbf{y}^{train}$, we have

$$p(\mathbf{y}^{train}|\mathbf{X}^{train}, \mathbf{w}) = \mathcal{N}(\mathbf{y}^{train}; \mathbf{X}^{train}\mathbf{w}, \mathbf{I})$$

$$\propto \exp\left(-\frac{1}{2}(\mathbf{y}^{train} - \mathbf{X}^{train}\mathbf{w})^T(\mathbf{y}^{train} - \mathbf{X}^{train}\mathbf{w})\right).$$

The assumption here is that the variance of y is 1.

A Bayesian Approach to Learning
Example: Linear Regression

For real-valued parameters it is common to use a Gaussian as a prior distribution for the model parameter vector $\mathbf{w}$:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0) \propto \exp\left(-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu}_0)^T \boldsymbol{\Lambda}_0^{-1}(\mathbf{w} - \boldsymbol{\mu}_0)\right),$$

where $\boldsymbol{\mu}_0$ and $\boldsymbol{\Lambda}_0$ are the prior distribution mean vector and the covariance matrix, respectively. We typically assume a diagonal covariance matrix $\boldsymbol{\Lambda}_0 = \text{diag}(\lambda_0)$.
In what follows, we refer to $(\mathbf{X}^{train}, \mathbf{y}^{train})$ as simply $(\mathbf{X}, \mathbf{y})$. The posterior distribution of the model parameters amounts to

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{X}, \mathbf{y}, \mathbf{w})}{p(\mathbf{X}, \mathbf{y})} = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{X}, \mathbf{w})}{p(\mathbf{X}, \mathbf{y})} \propto p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})$$

A Bayesian Approach to Learning
Example: Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) \propto$$

$$\propto \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w})\right)\exp\left(-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu}_0)^T\boldsymbol{\Lambda}_0^{-1}(\mathbf{w}-\boldsymbol{\mu}_0)\right)$$

$$\propto \exp\left(-\frac{1}{2}(-2\mathbf{y}^T\mathbf{X}\mathbf{w}+\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}+\mathbf{w}^T\boldsymbol{\Lambda}_0^{-1}\mathbf{w}-2\boldsymbol{\mu}_0^T\boldsymbol{\Lambda}_0^{-1}\mathbf{w})\right).$$

In the derivation above we dropped all terms that do not include the parameter vector $\mathbf{w}$. With the substitution

$$\boldsymbol{\Lambda}_m = (\mathbf{X}^T\mathbf{X}+\boldsymbol{\Lambda}_0^{-1})^{-1} \quad \text{and} \quad \boldsymbol{\mu}_m = \boldsymbol{\Lambda}_m(\mathbf{X}^T\mathbf{y}+\boldsymbol{\Lambda}_0^{-1}\boldsymbol{\mu}_0)$$

we obtain ($\boldsymbol{\Lambda}_m$ is symmetric)

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) \propto \exp\left(-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu}_m)^T\boldsymbol{\Lambda}_m^{-1}(\mathbf{w}-\boldsymbol{\mu}_m)+\frac{1}{2}\boldsymbol{\mu}_m^T\boldsymbol{\Lambda}_m^{-1}\boldsymbol{\mu}_m\right).$$

A Bayesian Approach to Learning
Example: Linear Regression

Finally,

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) \propto \exp\left(-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu}_m)^T\boldsymbol{\Lambda}_m^{-1}(\mathbf{w}-\boldsymbol{\mu}_m)\right).$$

Note that the posterior distribution has the form of a Gaussian distribution with mean vector $\boldsymbol{\mu}_m$ and covariance matrix $\boldsymbol{\Lambda}_m$. This justifies our dropping all terms unrelated to $\mathbf{w}$, since the posterior distribution must be normalized and, as a Gaussian, we know what that normalization constant must be (where n is the dimension of the input):

$$p(\mathbf{w}|\mathbf{X}^{train},\mathbf{y}^{train}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Lambda}_m|}}\exp\left(-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu}_m)^T\boldsymbol{\Lambda}_m^{-1}(\mathbf{w}-\boldsymbol{\mu}_m)\right).$$

A Bayesian Approach to Learning
Example: Linear Regression

Examining this posterior distribution allows us to gain some intuition for the effect of Bayesian inference. In most situations, we set $\boldsymbol{\mu}_0 = 0$. If we set $\boldsymbol{\Lambda}_m = \frac{1}{\alpha}\mathbf{I}$, then $\boldsymbol{\mu}_m$ gives the same estimate of $\mathbf{w}$ as does frequentist linear regression with a weight decay penalty of $\alpha\mathbf{w}^T\mathbf{w}$.

One difference is that the Bayesian estimate is undefined if $\alpha = 0$ - we are not allowed to begin the Bayesian learning process with an infinitely wide prior on $\mathbf{w}$. The more important difference is that the Bayesian estimate provides a covariance matrix, showing how likely all the different values of $\mathbf{w}$ are, rather than providing only the estimate $\boldsymbol{\mu}_m$.

Maximum A Posteriori Estimation (MAP) Basics

The MAP estimate chooses the point of maximal posterior probability (maximal probability density in case of continuous $\boldsymbol{\theta}$):

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

On the right hand side, $\log p(\boldsymbol{x} \mid \boldsymbol{\theta})$, is the standard log likelihood term and $\log p(\boldsymbol{\theta})$ corresponds to the prior distribution.
The prior on $\boldsymbol{\theta}$ effects the MAP estimate by leveraging information other than that contained in the training data. This additional information helps to reduce the variance in the MAP point estimate (in comparison to the ML estimate). However, it does so at the price of increased bias.

MLE vs. MAP vs. MLSE
Example: Regularized Linear Regression

Given a set of m training samples of input output pairs $(\mathbf{X}^{train}, \mathbf{y}^{train})$, and using the Bayesian approach to linear regression, the prediction of y over the entire training set is:

$$\hat{\mathbf{y}}^{train} = \mathbf{X}^{train}\mathbf{w},$$

where prediction is parametrized by the vector $\mathbf{w} \in \mathbb{R}^n$.
The MLE for the model parameters also minimizes the MSE and it is, therefore, given by

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

Assuming that the mean is $\boldsymbol{\mu}_0 = 0$ and the covariance matrix of the prior is $\boldsymbol{\Lambda}_0 = \lambda_0\mathbf{I}$, the MAP estimate of the mean of the Gaussian posterior density becomes

$$\hat{\mathbf{w}}_{MAP} = \boldsymbol{\Lambda}_m\mathbf{X}^{(train)T}\mathbf{y}^{train}.$$

MLE vs. MAP vs. MLSE
Example: Regularized Linear Regression

Since

$$\mathbf{\Lambda}_m = (\mathbf{X}^{(train)T}\mathbf{X}^{train} + \lambda_0^{-1}\mathbf{I})^{-1}.$$

$$\hat{\mathbf{w}}_{MAP} = (\mathbf{X}^{(train)T}\mathbf{X}^{train} + \lambda_0^{-1}\mathbf{I})^{-1}\mathbf{X}^{(train)T}\mathbf{y}^{train}.$$

Recall that

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

Therefore, as the variance of the prior distribution tends to infinity, the MAP estimate tends to the ML estimate. As the variance of the prior tends to zero, the MAP estimate tends to zero (i.e., the value of $\boldsymbol{\mu}_0 = 0$).

MLE vs. MAP vs. MLSE
Example: Regularized Linear Regression

Here we explore the model capacity tradeoff between the ML estimate and the MAP estimate by analyzing the bias and variance of these estimates.

The ML estimate is unbiased, i.e., $\mathbb{E}\hat{\mathbf{w}}_{ML} = \mathbf{w}$ and the variance is given by

$$\text{Var}(\hat{\mathbf{w}}_{ML}) = (\mathbf{X}^{(train)T}\mathbf{X}^{train})^{-1}.$$

Work it out!

MLE vs. MAP vs. MLSE
Example: Regularized Linear Regression

We now calculate the average of the MAP estimate:

$$\begin{aligned}\mathbb{E}\hat{\mathbf{w}}_{MAP} &= \mathbb{E}\boldsymbol{\Lambda}_m \mathbf{X}^{(train)T}\mathbf{y}^{train} = \mathbb{E}\boldsymbol{\Lambda}_m[\mathbf{X}^{(train)T}(\mathbf{X}^{train}\mathbf{w}+\boldsymbol{\epsilon})] \\ &= \boldsymbol{\Lambda}_m \mathbf{X}^{(train)T}\mathbf{X}^{train}\mathbf{w} + \boldsymbol{\Lambda}_m \mathbf{X}^{(train)T}\mathbb{E}\,\boldsymbol{\epsilon} \\ &= (\mathbf{X}^{(train)T}\mathbf{X}^{train} + \lambda_0^{-1}\mathbf{I})^{-1}\mathbf{X}^{(train)T}\mathbf{X}^{train}\mathbf{w}\end{aligned}$$

Note that the expected value of the ML estimate is the true parameter value $\mathbf{w}$ (i.e. the parameters that we assume generated the data); the expected value of the MAP estimate is a weighted average of w and the prior mean $\mathbf{w}$.
The bias amounts to

$$\text{Bias}(\hat{\mathbf{w}}_{MAP}) = \mathbb{E}\hat{\mathbf{w}}_{MAP} - \mathbf{w} = -(\lambda_0 \mathbf{X}^{(train)T}\mathbf{X}^{train} + \mathbf{I})^{-1}\mathbf{w}$$

As the variance of the prior $\lambda_0 \to \infty$, the bias tends to zero. As the variance of the prior $\lambda_0 \to 0$, the bias tends to $\mathbf{w}$.

MLE vs. MAP vs. MLSE
Example: Regularized Linear Regression

In order to compute the variance, we use the identity $\text{Var}(\hat{\theta}) = \mathbb{E}\,\hat{\theta}^2 - (\mathbb{E}\,\hat{\theta})^2$. Let us denote by $\boldsymbol{\Xi}^{train} = \mathbf{X}^{(train)T}\mathbf{X}^{train}$

$$
\begin{aligned}
&\mathbb{E}\,\hat{\mathbf{w}}_{MAP}\hat{\mathbf{w}}_{MAP}^T \\
&= \mathbb{E}\boldsymbol{\Lambda}_m\mathbf{X}^{(train)T}\hat{\mathbf{y}}^{train}\hat{\mathbf{y}}^{(train)T}\mathbf{X}^{train}\boldsymbol{\Lambda}_m \\
&= \mathbb{E}\boldsymbol{\Lambda}_m\mathbf{X}^{(train)T}(\mathbf{X}^{train}\mathbf{w}+\boldsymbol{\epsilon})(\mathbf{X}^{train}\mathbf{w}+\boldsymbol{\epsilon})^T\mathbf{X}^{train}\boldsymbol{\Lambda}_m \\
&= \boldsymbol{\Lambda}_m\boldsymbol{\Xi}^{train}\mathbf{w}\mathbf{w}^T\boldsymbol{\Xi}^{train}\boldsymbol{\Lambda}_m + \boldsymbol{\Lambda}_m\mathbf{X}^{(train)T}\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]\mathbf{X}^{train}\boldsymbol{\Lambda}_m \\
&= \boldsymbol{\Lambda}_m\boldsymbol{\Xi}^{train}\mathbf{w}\mathbf{w}^T\boldsymbol{\Xi}^{train}\boldsymbol{\Lambda}_m + \boldsymbol{\Lambda}_m\boldsymbol{\Xi}^{train}\boldsymbol{\Lambda}_m \\
&= \mathbb{E}\hat{\mathbf{w}}_{MAP}\,\mathbb{E}\hat{\mathbf{w}}_{MAP}^T + \boldsymbol{\Lambda}_m\boldsymbol{\Xi}^{train}\boldsymbol{\Lambda}_m.
\end{aligned}
$$

MLE vs. MAP vs. MLSE
Example: Regularized Linear Regression

With $\mathbb{E}\hat{\mathbf{w}}_{MAP}\,\hat{\mathbf{w}}_{MAP}^T$ thus computed, the variance of the MAP estimate of our linear regression model is given by:

$$\begin{aligned}
\text{Var}(\hat{\mathbf{w}}_{MAP}) &= \mathbb{E}\hat{\mathbf{w}}_{MAP}\,\hat{\mathbf{w}}_{MAP}^T - \mathbb{E}\hat{\mathbf{w}}_{MAP}\,\mathbb{E}\hat{\mathbf{w}}_{MAP}^T \\
&= \mathbb{E}\hat{\mathbf{w}}_{MAP}\,\mathbb{E}\hat{\mathbf{w}}_{MAP}^T + \boldsymbol{\Lambda}_m\boldsymbol{\Xi}^{train}\boldsymbol{\Lambda}_m - \mathbb{E}\hat{\mathbf{w}}_{MAP}\,\hat{\mathbf{w}}_{MAP}^T \\
&= \boldsymbol{\Lambda}_m\boldsymbol{\Xi}^{train}\boldsymbol{\Lambda}_m \\
&= (\boldsymbol{\Xi}^{train} + \lambda_0^{-1}\mathbf{I})^{-1}\boldsymbol{\Xi}^{train}(\boldsymbol{\Xi}^{train} + \lambda_0^{-1}\mathbf{I})^{-1}.
\end{aligned}$$

Recall that

$$\text{Var}(\hat{\mathbf{w}}_{ML}) = (\mathbf{X}^{(train)T}\mathbf{X}^{train})^{-1} = (\boldsymbol{\Xi}^{train})^{-1}.$$

MLE vs. MAP vs. MLSE
Example: Regularized Linear Regression

To get some intuition, assume that $\mathbf{w}$ is one-dimensional (along with $\mathbf{x}$), it becomes a bit easier to see that, as long as λ_0 is bounded, then

$$\text{Var}(\hat{w}_{ML}) = \frac{1}{\sum_{i=1}^{m} x_i^2} \text{Var}(\hat{w}_{MAP}) = \frac{\lambda_0 \sum_{i=1}^{m} x_i^2}{(1 + \lambda_0 \sum_{i=1}^{m} x_i^2)^2}.$$

The role of the prior in the MAP estimate is to trade increased bias for a reduction in variance. The goal, of course, is to try to avoid overfitting. The incurred bias is a consequence of the reduction in model capacity caused by limiting the space of hypotheses to those with significant probability density under the prior.

Thanks