Machine Learning

Introduction, Supervised Learning, and Overfitting

Aleksandr Petiushko

ML Research

July 22nd, 2023

Content

Content

1. Introduction
2. Course logistics and syllabus

Content

Content

1. Introduction
2. Course logistics and syllabus
3. Historical reference
4. Setting of basic machine learning tasks

Content

Content

Intro

About the lecturer[1]

- Aleksandr Petiushko, PhD in theoretical CS (2016)
- Lecturer in Lomonosov MSU / MIPT for Machine Learning, Computer Vision, Deep Learning Theory, Python for an ML Researcher since 2019
- Former Huawei Chief Scientist (Scientific Expert), AIRI Director of Key Research Programs (Leading Scientific Researcher)
- Currently at Nuro, leading the ML Research

[1] Homepage: `https://petiushko.info/`

Intro

Time to introduce yourselves: what are your hobbies, motivation in ML, etc

Sofia Plagiarism Policy[2]

- It covers parts "*sourced from AI*"
  - **First offense**: students need to rewrite assignment
  - **Second offense**: students fail the course
  - **Third offense**: students re to be withdrawn from their program

[2] Link

Sofia Plagiarism Policy[2]

- It covers parts "*sourced from AI*"
  - **First offense**: students need to rewrite assignment
  - **Second offense**: students fail the course
  - **Third offense**: students re to be withdrawn from their program
- Current situation: one full assignment + one discussion were written by using xGPTx (plus some specific questions inside assignments)
  - And *probably* some others, but not so obvious
  - Grades for them: 0

[2] Link

Sofia Plagiarism Policy[2]

- It covers parts "*sourced from AI*"
  - **First offense**: students need to rewrite assignment
  - **Second offense**: students fail the course
  - **Third offense**: students re to be withdrawn from their program
- Current situation: one full assignment + one discussion were written by using xGPTx (plus some specific questions inside assignments)
  - And *probably* some others, but not so obvious
  - Grades for them: 0
- Current approach: to check your text via OpenAI tool: `https://platform.openai.com/ai-text-classifier` and get the answer "**very unlikely**", "**unlikely**", or "**unclear if it is**"
  - Answers "**possibly**", or "**likely**" will be treated as plagiarism

[2] Link

Note about xGPTx

- It can produce very plausible answers in 90% of cases

Note about xGPTx

- It can produce very plausible answers in 90% of cases
- The caveats are the following:

Note about xGPTx

- It can produce very plausible answers in 90% of cases
- The caveats are the following:
  - It can really hallucinate some things which are just untrue

Note about xGPTx

- It can produce very plausible answers in 90% of cases
- The caveats are the following:
  - It can really hallucinate some things which are just untrue
  - It can produce very different information in comparison to the source used to ask question (e.g., book chapter)

Note about discussions

- Since **Module 4** your discussion answer like "I agree because of bla-bla-bla" won't be graded — they do not provide any value

Note about discussions

- Since **Module 4** your discussion answer like “I agree because of bla-bla-bla” won’t be graded — they do not provide any value
- Only the answers with some non-trivial arguments that contradict the initial post will be considered as graded ones

Course logistics

- Course grading will be done based on attendance, assignments, discussions, (probably some small programming project) and the final exam.

Course logistics

- Course grading will be done based on attendance, assignments, discussions, (probably some small programming project) and the final exam.
- Contribution:
  - 50%: attendance, assignments, discussions
  - 50%: exam

Course logistics

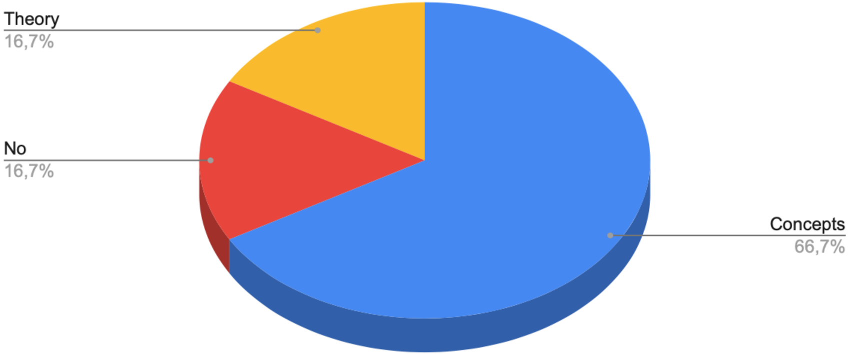
- Course grading will be done based on attendance, assignments, discussions, (probably some small programming project) and the final exam.
- Contribution:
  - 50%: attendance, assignments, discussions
  - 50%: exam
- Exam: during the final on ground class

Course logistics

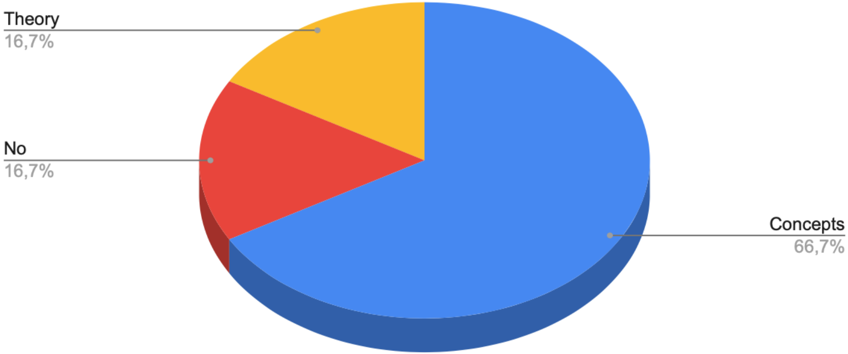
- Course grading will be done based on attendance, assignments, discussions, (probably some small programming project) and the final exam.
- Contribution:
  - 50%: attendance, assignments, discussions
  - 50%: exam
- Exam: during the final on ground class
- Preliminary grading scale:

Grade	Percent accumulated
A	90-100 %
B	75-89 %
C	60-74 %

Results of Survey: Level of Math

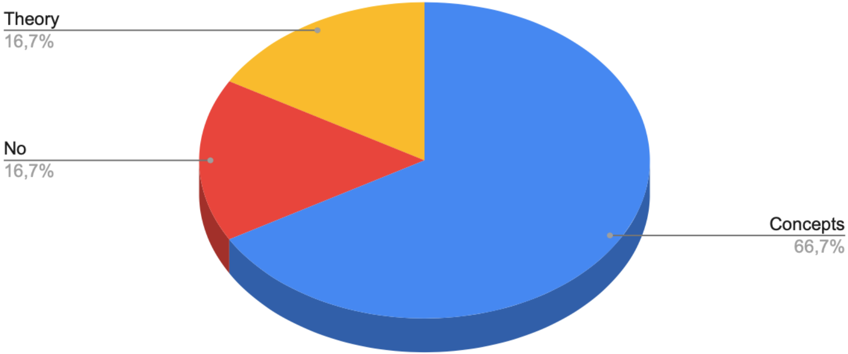


Results of Survey: Level of Math



<u>Conclusion</u>:

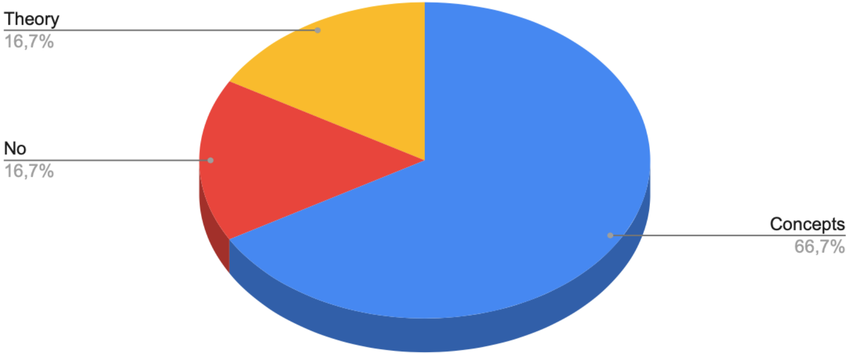
Results of Survey: Level of Math



Conclusion:

- Will be some math inside covering the basics

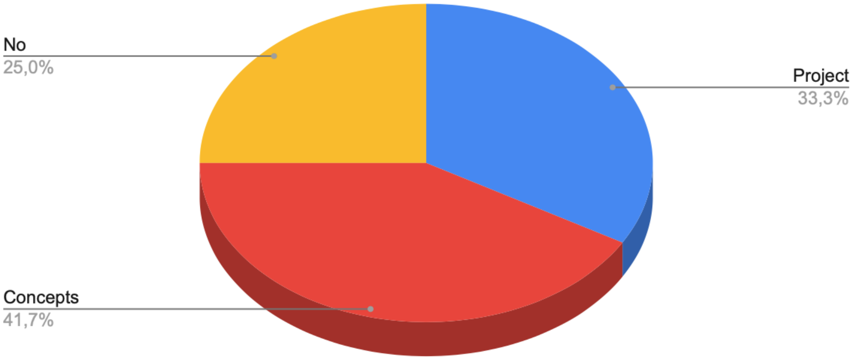
Results of Survey: Level of Math



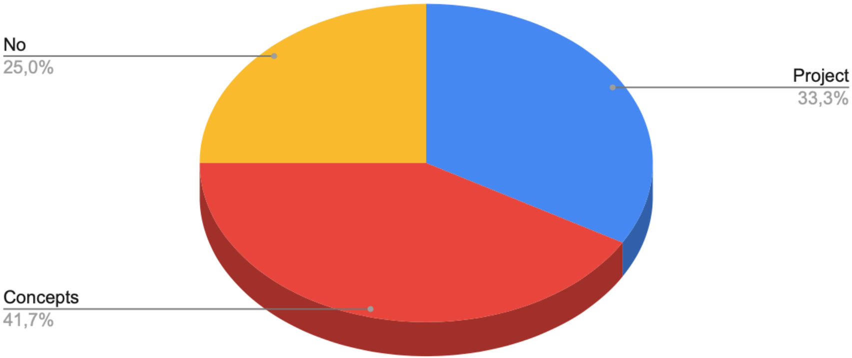
<u>Conclusion</u>:

- Will be some math inside covering the basics
- Will be links to more advanced stuff

Results of Survey: Level of Programming

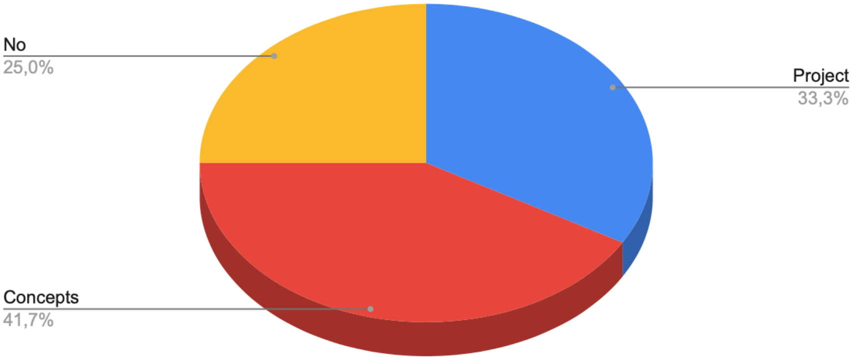


Results of Survey: Level of Programming



Conclusion:

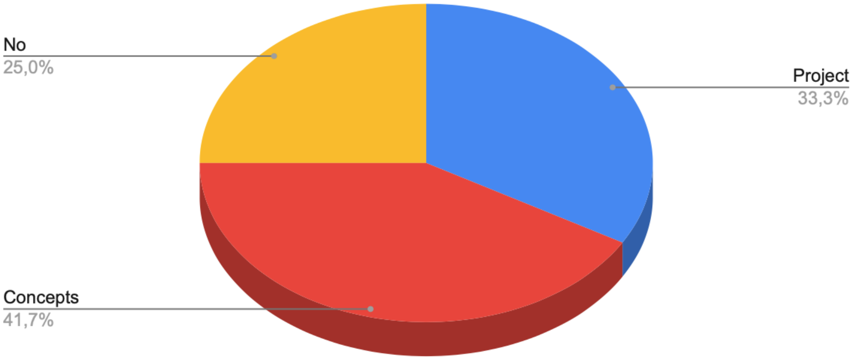
Results of Survey: Level of Programming



Conclusion:

- Will devote some time to covering the basic ml prog (using the ml lib)

Results of Survey: Level of Programming



Conclusion:

- Will devote some time to covering the basic ml prog (using the ml lib)
- Project will be discussed on the personal basis / wish

Reason behind the Survey

- Current ML is: half Math, half Programming

Reason behind the Survey

- Current ML is: half Math, half Programming
  - **Math**: for research and design of ML algorithms

Reason behind the Survey

- Current ML is: half Math, half Programming
  - **Math**: for research and design of ML algorithms
  - **Programming**: usage and tuning of ML algorithms

Reason behind the Survey

- Current ML is: half Math, half Programming
  - **Math**: for research and design of ML algorithms
  - **Programming**: usage and tuning of ML algorithms
- Hope we could touch a little both

Github

- Course page: `https://github.com/fatheral/sofia-ml-2023-1`
- The on ground lectures will be uploaded there

What is Artificial Intelligence?

Natural Intelligence (human)

- Able to perceive the information, analyze it, make decisions based on this analysis

What is Artificial Intelligence?

Natural Intelligence (human)

- Able to perceive the information, analyze it, make decisions based on this analysis

Artificial Intelligence

- (Strong) The same as natural intelligence, but computer is instead of human

What is Artificial Intelligence?

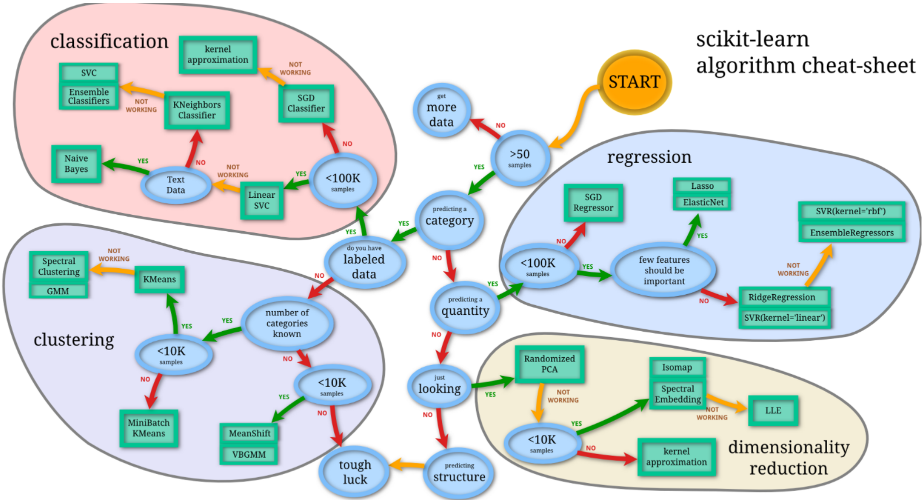
Natural Intelligence (human)

- Able to perceive the information, analyze it, make decisions based on this analysis

Artificial Intelligence

- (Strong) The same as natural intelligence, but computer is instead of human
- (**Weak**) Algorithm which is able to train using the input data in order to do tasks afterward — instead of human

Scikit-Learn[3] Roadmap



[3] https://scikit-learn.org/stable/tutorial/machine_learning_map/

(Tentative) future content

Theoretic part

- Quality metrics
  - Precision / Recall, TPR / FPR, ROC, AUC, Cross-Validation, . . .

(Tentative) future content

Theoretic part

- Quality metrics
  - Precision / Recall, TPR / FPR, ROC, AUC, Cross-Validation, ...
- Classification task and optimization
  - kNN, Linear Classifiers, Stochastic Gradient Descent, PCA, ...

(Tentative) future content

Theoretic part

- Quality metrics
  - Precision / Recall, TPR / FPR, ROC, AUC, Cross-Validation, ...
- Classification task and optimization
  - kNN, Linear Classifiers, Stochastic Gradient Descent, PCA, ...
- Regression task
  - Linear Regression, Elastic Net, Ridge Regression, LASSO, ...

(Tentative) future content

Theoretic part

- Quality metrics
  - Precision / Recall, TPR / FPR, ROC, AUC, Cross-Validation, ...
- Classification task and optimization
  - kNN, Linear Classifiers, Stochastic Gradient Descent, PCA, ...
- Regression task
  - Linear Regression, Elastic Net, Ridge Regression, LASSO, ...
- Ensembles
  - Bootstrapping, Bagging, Boosting, ...

(Tentative) future content

Theoretic part

- Quality metrics
  - Precision / Recall, TPR / FPR, ROC, AUC, Cross-Validation, ...
- Classification task and optimization
  - kNN, Linear Classifiers, Stochastic Gradient Descent, PCA, ...
- Regression task
  - Linear Regression, Elastic Net, Ridge Regression, LASSO, ...
- Ensembles
  - Bootstrapping, Bagging, Boosting, ...

Practice part

- Data processing and analysis by Python
  - Scikit-Learn, Numpy, ...

Time for questions

What is Machine Learning

In 1959 Arthur Samuel introduced the term “machine learning” into scientific use.

General definition

Machine Learning — the process leading computers to gain ability to show the behavior that wasn’t explicitly programmed.

What is Machine Learning

In 1959 Arthur Samuel introduced the term "machine learning" into scientific use.

General definition

Machine Learning — the process leading computers to gain ability to show the behavior that wasn't explicitly programmed.

In 1997 Tom M. Mitchell introduced more formal definition of a machine learning algorithm.

Formal definition

A **computer program** is said **to learn** from examples E for some set of problems T and a quality metric P if its performance on problems from T, as measured by P, is improved by using examples E.

Forerunner of Machine Learning

- People have been trying to predict the future based on their experience since time immemorial.

Forerunner of Machine Learning

- People have been trying to predict the future based on their experience since time immemorial.
- However, the scientific basis was laid by probability theory (statistics in particular) and linear algebra (as a tool).

Forerunner of Machine Learning

- People have been trying to predict the future based on their experience since time immemorial.
- However, the scientific basis was laid by probability theory (statistics in particular) and linear algebra (as a tool).

- **1795**: Gauss first uses the least squares method (LSQ) to analyze astronomical observations. In **1805** Legendre first published this method for analyzing the shape of the Earth. At present, LSQ is the simplest way to solve an overdetermined system of linear equations.

Forerunner of Machine Learning

- People have been trying to predict the future based on their experience since time immemorial.
- However, the scientific basis was laid by probability theory (statistics in particular) and linear algebra (as a tool).

- **1795**: Gauss first uses the least squares method (LSQ) to analyze astronomical observations. In **1805** Legendre first published this method for analyzing the shape of the Earth. At present, LSQ is the simplest way to solve an overdetermined system of linear equations.
- **1901**: Karl Pearson invented the Principal Component Analysis (PCA) — a master method for data dimensionality reduction.

Forerunner of Machine Learning

- People have been trying to predict the future based on their experience since time immemorial.
- However, the scientific basis was laid by probability theory (statistics in particular) and linear algebra (as a tool).

- **1795**: Gauss first uses the least squares method (LSQ) to analyze astronomical observations. In **1805** Legendre first published this method for analyzing the shape of the Earth. At present, LSQ is the simplest way to solve an overdetermined system of linear equations.
- **1901**: Karl Pearson invented the Principal Component Analysis (PCA) — a master method for data dimensionality reduction.
- **1906**: Andrey Andreyevich Markov develops the apparatus of Markov chains, which in **1913** he uses to study the text “Eugene Onegin”. Markov chains are used to generate and recognize signals.

Historical reference

- **1950**: Alan Turing creates the Turing test to evaluate the intelligence of a computer.

Historical reference

- **1950**: Alan Turing creates the Turing test to evaluate the intelligence of a computer.
- **1951**: Marvin Minsky created the first SNARC learning machine with a randomly connected neural network. In **1959**, he co-founded the Artificial Intelligence Laboratory at MIT.

Historical reference

- **1950**: Alan Turing creates the Turing test to evaluate the intelligence of a computer.
- **1951**: Marvin Minsky created the first SNARC learning machine with a randomly connected neural network. In **1959**, he co-founded the Artificial Intelligence Laboratory at MIT.
- **1952**: Arthur Samuel creates the first checkers program for the IBM 701. In **1955** Samuel adds self-learning capability to the program.

Historical reference

- **1950**: Alan Turing creates the Turing test to evaluate the intelligence of a computer.
- **1951**: Marvin Minsky created the first SNARC learning machine with a randomly connected neural network. In **1959**, he co-founded the Artificial Intelligence Laboratory at MIT.
- **1952**: Arthur Samuel creates the first checkers program for the IBM 701. In **1955** Samuel adds self-learning capability to the program.
- **1958**: Frank Rosenblatt invented the Perceptron — the first artificial neural network — and built the first "Mark-1" brain computer. *New York Times: The Perceptron is "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence".*

Historical reference

- **1950**: Alan Turing creates the Turing test to evaluate the intelligence of a computer.
- **1951**: Marvin Minsky created the first SNARC learning machine with a randomly connected neural network. In **1959**, he co-founded the Artificial Intelligence Laboratory at MIT.
- **1952**: Arthur Samuel creates the first checkers program for the IBM 701. In **1955** Samuel adds self-learning capability to the program.
- **1958**: Frank Rosenblatt invented the Perceptron — the first artificial neural network — and built the first "Mark-1" brain computer. *New York Times: The Perceptron is "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence".*
- **1963**: Lawrence Roberts formulated the thesis of computer vision in his dissertation at MIT.

Historical reference

- **1963**: Vladimir Vapnik and Aleksey Chervonenkis invented the SVM algorithm.

Historical reference

- **1963**: Vladimir Vapnik and Aleksey Chervonenkis invented the SVM algorithm.
- **1965**: One of the first books on machine learning (pattern classification) was published — Nilsson N. Learning Machines, McGraw Hill.

Historical reference

- **1963**: Vladimir Vapnik and Aleksey Chervonenkis invented the SVM algorithm.
- **1965**: One of the first books on machine learning (pattern classification) was published — Nilsson N. Learning Machines, McGraw Hill.
- **1966**: Joseph Weizenbaum wrote a computerized conversation simulator ELIZA, able to imitate (or rather parody) a dialogue with a psychotherapist (the program owes its name to the heroine from the play by B. Shaw).

Historical reference

- **1963**: Vladimir Vapnik and Aleksey Chervonenkis invented the SVM algorithm.
- **1965**: One of the first books on machine learning (pattern classification) was published — Nilsson N. Learning Machines, McGraw Hill.
- **1966**: Joseph Weizenbaum wrote a computerized conversation simulator ELIZA, able to imitate (or rather parody) a dialogue with a psychotherapist (the program owes its name to the heroine from the play by B. Shaw).
- **1967**: Alexey Ivakhnenko and Valentin Lapa publish the first general working learning algorithm for deep multilayer perceptrons for supervised learning problems.

Historical reference

- **1963**: Vladimir Vapnik and Aleksey Chervonenkis invented the SVM algorithm.
- **1965**: One of the first books on machine learning (pattern classification) was published — Nilsson N. Learning Machines, McGraw Hill.
- **1966**: Joseph Weizenbaum wrote a computerized conversation simulator ELIZA, able to imitate (or rather parody) a dialogue with a psychotherapist (the program owes its name to the heroine from the play by B. Shaw).
- **1967**: Alexey Ivakhnenko and Valentin Lapa publish the first general working learning algorithm for deep multilayer perceptrons for supervised learning problems.
- **1986**: Rina Dechter introduced the term “Deep Learning” to the machine learning community.

Historical reference

- **1963**: Vladimir Vapnik and Aleksey Chervonenkis invented the SVM algorithm.
- **1965**: One of the first books on machine learning (pattern classification) was published — Nilsson N. Learning Machines, McGraw Hill.
- **1966**: Joseph Weizenbaum wrote a computerized conversation simulator ELIZA, able to imitate (or rather parody) a dialogue with a psychotherapist (the program owes its name to the heroine from the play by B. Shaw).
- **1967**: Alexey Ivakhnenko and Valentin Lapa publish the first general working learning algorithm for deep multilayer perceptrons for supervised learning problems.
- **1986**: Rina Dechter introduced the term “Deep Learning” to the machine learning community.
- **1997**: The Deep Blue computer beat world chess champion Garry Kasparov.

Historical reference

- **2010**: Founding of DeepMind.

[4] https://en.wikipedia.org/wiki/Perfect_information

Historical reference

- **2010**: Founding of DeepMind.
- **2011**: Andrew Ng, Greg Corrado and Jeff Dean founded Google Brain.

[4] `https://en.wikipedia.org/wiki/Perfect_information`

Historical reference

- **2010**: Founding of DeepMind.
- **2011**: Andrew Ng, Greg Corrado and Jeff Dean founded Google Brain.
- **2011**: IBM Watson AI supercomputer wins TV quiz show *Jeopardy!*

[4] `https://en.wikipedia.org/wiki/Perfect_information`

Historical reference

- **2010**: Founding of DeepMind.
- **2011**: Andrew Ng, Greg Corrado and Jeff Dean founded Google Brain.
- **2011**: IBM Watson AI supercomputer wins TV quiz show *Jeopardy!*
- **2014**: Facebook invented the DeepFace software algorithm for face recognition. The accuracy of the algorithm was 97%.

[4] https://en.wikipedia.org/wiki/Perfect_information

Historical reference

- **2010**: Founding of DeepMind.
- **2011**: Andrew Ng, Greg Corrado and Jeff Dean founded Google Brain.
- **2011**: IBM Watson AI supercomputer wins TV quiz show *Jeopardy!*
- **2014**: Facebook invented the DeepFace software algorithm for face recognition. The accuracy of the algorithm was 97%.
- **2016**: AlphaGo, developed by (now) Google's DeepMind, won four out of five games against Korea's world Go champion Lee Se-dol. The computer won the last game with perfect information [4] against a human (an example of a game with incomplete information is poker, although robots are already beginning to perform successfully there).

[4] https://en.wikipedia.org/wiki/Perfect_information

Historical reference

- **2010**: Founding of DeepMind.
- **2011**: Andrew Ng, Greg Corrado and Jeff Dean founded Google Brain.
- **2011**: IBM Watson AI supercomputer wins TV quiz show *Jeopardy!*
- **2014**: Facebook invented the DeepFace software algorithm for face recognition. The accuracy of the algorithm was 97%.
- **2016**: AlphaGo, developed by (now) Google's DeepMind, won four out of five games against Korea's world Go champion Lee Se-dol. The computer won the last game with perfect information [4] against a human (an example of a game with incomplete information is poker, although robots are already beginning to perform successfully there).
- **2016**: OpenAI, a non-profit research company, is launched with the support of Elon Musk.

[4] https://en.wikipedia.org/wiki/Perfect_information

Historical reference

- **2010**: Founding of DeepMind.
- **2011**: Andrew Ng, Greg Corrado and Jeff Dean founded Google Brain.
- **2011**: IBM Watson AI supercomputer wins TV quiz show *Jeopardy!*
- **2014**: Facebook invented the DeepFace software algorithm for face recognition. The accuracy of the algorithm was 97%.
- **2016**: AlphaGo, developed by (now) Google's DeepMind, won four out of five games against Korea's world Go champion Lee Se-dol. The computer won the last game with perfect information [4] against a human (an example of a game with incomplete information is poker, although robots are already beginning to perform successfully there).
- **2016**: OpenAI, a non-profit research company, is launched with the support of Elon Musk.
- **2022**: OpenAI, a (not so) non-profit research company, provided the breakthrough in LLMs: ChatGPT.

[4] https://en.wikipedia.org/wiki/Perfect_information

Time for questions

Machine Learning Paradigms

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the <u>unknown</u> dependency

Machine Learning Paradigms

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the unknown dependency

Machine learning paradigms

- **Supervised** (now)
  - Sufficient amount of training material, i.e. pairs (x_i, y_i)

Machine Learning Paradigms

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the unknown dependency

Machine learning paradigms

- **Supervised** (now)
  - Sufficient amount of training material, i.e. pairs (x_i, y_i)
- Semi-supervised
  - Few labeled data (x_i, y_i) and many unlabeled examples x_j

Machine Learning Paradigms

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the unknown dependency

Machine learning paradigms

- **Supervised** (now)
  - Sufficient amount of training material, i.e. pairs (x_i, y_i)
- Semi-supervised
  - Few labeled data (x_i, y_i) and many unlabeled examples x_j
- *Unsupervised* (in future lectures?)
  - No labeled pairs, only x_i examples

Machine Learning Paradigms

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the <u>unknown</u> dependency

Machine learning paradigms

- **Supervised** (now)
  - Sufficient amount of training material, i.e. pairs (x_i, y_i)
- Semi-supervised
  - Few labeled data (x_i, y_i) and many unlabeled examples x_j
- *Unsupervised* (in future lectures?)
  - No labeled pairs, only x_i examples
- Reinforced
  - Action generation based on interaction with the environment

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - A decision function $a : X \to Y$ that approximates the target dependency y.

Problem setting for supervised learning

- Given:
  - — $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - — A decision function $a : X \to Y$ that approximates the target dependency y.
- Need to clarify:

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - A decision function $a : X \to Y$ that approximates the target dependency y.
- Need to clarify:
  - How objects are defined

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - A decision function $a : X \to Y$ that approximates the target dependency y.
- Need to clarify:
  - How objects are defined
  - How answers are given

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - A decision function $a : X \to Y$ that approximates the target dependency y.
- Need to clarify:
  - How objects are defined
  - How answers are given
  - What does it mean that one dependency approximates another

How objects are defined

Definition

Object = set of features

How objects are defined

Definition

Object = set of features

Feature types

- Categorical feature

How objects are defined

Definition

Object = set of features

Feature types

- Categorical feature
- Binary attribute
  - A special case of categorical, when category = “does this property exist or not”

How objects are defined

Definition

Object = set of features

Feature types

- Categorical feature
- Binary attribute
  - A special case of categorical, when category = "does this property exist or not"
- Ordinal attribute
  - Full (or partial) order within categories

How objects are defined

Definition

Object = set of features

Feature types

- Categorical feature
- Binary attribute
  - A special case of categorical, when category = “does this property exist or not”
- Ordinal attribute
  - Full (or partial) order within categories
- Quantitative attribute

How answers are given

Classification tasks

- Binary classification $Y = \{-1, 1\}$ or $Y = \{0, 1\}$

How answers are given

Classification tasks

- Binary classification $Y = \{-1, 1\}$ or $Y = \{0, 1\}$
- Multiclass classification $Y = \{0, 1, ..., M - 1\}$

How answers are given

Classification tasks

- Binary classification $Y = \{-1, 1\}$ or $Y = \{0, 1\}$
- Multiclass classification $Y = \{0, 1, ..., M-1\}$
- Multivalued binary classification $Y = \{0, 1\}^M$

How answers are given

Classification tasks

- Binary classification $Y = \{-1, 1\}$ or $Y = \{0, 1\}$
- Multiclass classification $Y = \{0, 1, ..., M - 1\}$
- Multivalued binary classification $Y = \{0, 1\}^M$

Regression Tasks

$Y = \mathbb{R}$ or $Y = \mathbb{R}^n$

Loss Function

Definition

Loss function $L(a, x)$ — error value of algorithm a on object x

Loss Function

Definition

Loss function $L(a, x)$ — error value of algorithm a on object x

Loss functions for classification problems

$L(a, x) = [a(x) \neq y]$ — error indicator function (either 0 or 1)

Loss Function

Definition

Loss function $L(a, x)$ — error value of algorithm a on object x

Loss functions for classification problems

$L(a, x) = [a(x) \neq y]$ — error indicator function (either 0 or 1)

Loss functions for regression problems

$L(a, x) = (a(x) - y)^2$ — squared error

Time for questions

Comparison of machine learning models

How do you know that one model is better than another?

To do this, we use a set which is independent of **training** set, which is called **test** set

Comparison of machine learning models

How do you know that one model is better than another?

To do this, we use a set which is independent of **training** set, which is called **test** set

Why even bother with this?

- There are many machine learning algorithms and it is important to understand which one is more applicable to a particular task

Comparison of machine learning models

How do you know that one model is better than another?

To do this, we use a set which is independent of **training** set, which is called **test** set

Why even bother with this?

- There are many machine learning algorithms and it is important to understand which one is more applicable to a particular task
- Even within the same model, there can be many (hyper)parameters to choose from

How to choose the best model

Naive approach

Train models with different parameters and choose the best one on the test

How to choose the best model

Naive approach

Train models with different parameters and choose the best one on the test

Disadvantages of the naive approach

- Since the test usually consists of a random subset of the original sample, the result on the test is also some approximation of a random variable

How to choose the best model

Naive approach

Train models with different parameters and choose the best one on the test

Disadvantages of the naive approach

- Since the test usually consists of a random subset of the original sample, the result on the test is also some approximation of a random variable
- If all models are tested on a test dataset and thus choose the best one, then implicit training will occur on the test, and surprises are possible on another independent test

How to choose the best model

Naive approach

Train models with different parameters and choose the best one on the test

Disadvantages of the naive approach

- Since the test usually consists of a random subset of the original sample, the result on the test is also some approximation of a random variable
- If all models are tested on a test dataset and thus choose the best one, then implicit training will occur on the test, and surprises are possible on another independent test

So... what to do?

In order not to implicitly learn from test data — you need to use **cross-validation**

Cross Validation

General idea

The main idea of cross-validation is to split the training set into two non-overlapping sets (possibly multiple times):

$$X^{learn} = X^{train} \sqcup X^{val}$$

On one of them, training takes place, and on the other, the model is validated.

Cross Validation

General idea

The main idea of cross-validation is to split the training set into two non-overlapping sets (possibly multiple times):

$$X^{learn} = X^{train} \sqcup X^{val}$$

On one of them, training takes place, and on the other, the model is validated.

Why validate?

Usually, any machine learning algorithm contains a whole set of so-called "**hyperparameters**" (i.e. parameters that are not learned, but set initially): dimension, various weighting factors, etc.
And in order to select these parameters "fairly", without using any test data at all, a validation procedure is carried out.

Cross Validation

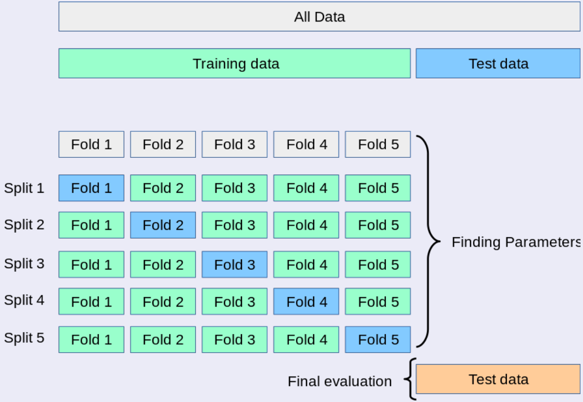
Special cases

1. The simplest cross-validation is **hold-out** control, in which the set is split once:

Cross Validation

Special cases

2. k block control (**k-fold** validation)[5]:



[5] Image source: `https://scikit-learn.org/`

Cross Validation

Special cases

3. Control by individual objects (**leave-one-out**, or LOO validation) — a special case of k-fold validation, if k is equal to the cardinality of the training set

Cross Validation

Special cases

3. Control by individual objects (**leave-one-out**, or LOO validation) — a special case of k-fold validation, if k is equal to the cardinality of the training set
4. **Multiple k-fold** validation — repeat k-fold validation several times with different splits.

Time for questions

Overfitting

Definition

Overfitting is an undesirable phenomenon that occurs when solving problems of learning by precedents, when the probability of the error of the trained algorithm on the objects of the test sample is significantly higher than the average error on the training sample.
Overfitting occurs when using an overly complex model

Overfitting

Definition

Overfitting is an undesirable phenomenon that occurs when solving problems of learning by precedents, when the probability of the error of the trained algorithm on the objects of the test sample is significantly higher than the average error on the training sample.
Overfitting occurs when using an overly complex model

One of the main causes

Excessive dimension of the model parameter space, "extra" degrees of freedom are used to "memorize" the training set

Overfitting

Definition

Overfitting is an undesirable phenomenon that occurs when solving problems of learning by precedents, when the probability of the error of the trained algorithm on the objects of the test sample is significantly higher than the average error on the training sample.
Overfitting occurs when using an overly complex model

One of the main causes

Excessive dimension of the model parameter space, "extra" degrees of freedom are used to "memorize" the training set

One of the main detection methods

Using Cross Validation

Underfitting

Definition

Underfitting is an undesirable phenomenon that occurs when solving problems of learning by precedents, when the learning algorithm does not provide a sufficiently small value of the average error on the training set.
Underfitting occurs when using insufficiently complex models

Underfitting

Definition

Underfitting is an undesirable phenomenon that occurs when solving problems of learning by precedents, when the learning algorithm does not provide a sufficiently small value of the average error on the training set.
Underfitting occurs when using insufficiently complex models

One of the main causes

Insufficient dimension of the model parameter space, model just learns very simple patterns

Underfitting

Definition

Underfitting is an undesirable phenomenon that occurs when solving problems of learning by precedents, when the learning algorithm does not provide a sufficiently small value of the average error on the training set.
Underfitting occurs when using insufficiently complex models

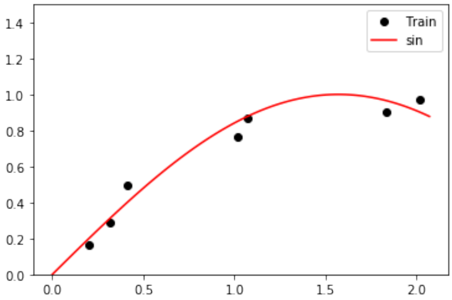
One of the main causes

Insufficient dimension of the model parameter space, model just learns very simple patterns

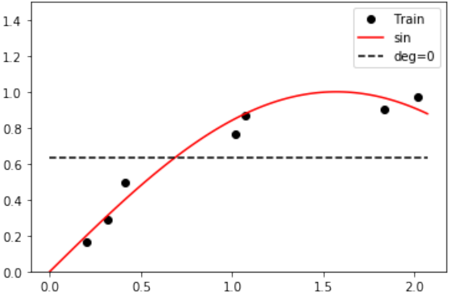
One of the main detection methods

Train error observation

Overfitting and Underfitting: examples

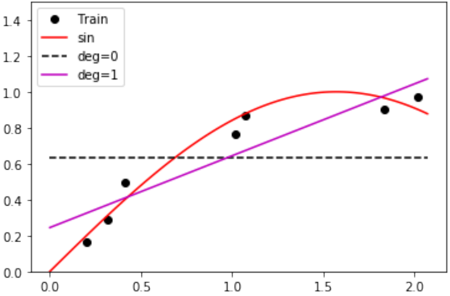


Overfitting and Underfitting: examples



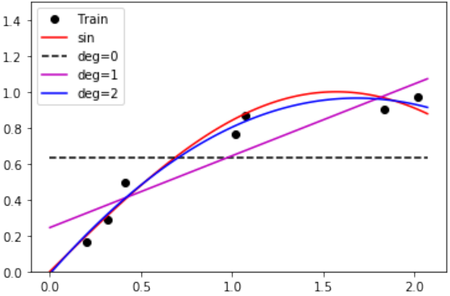
- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model

Overfitting and Underfitting: examples



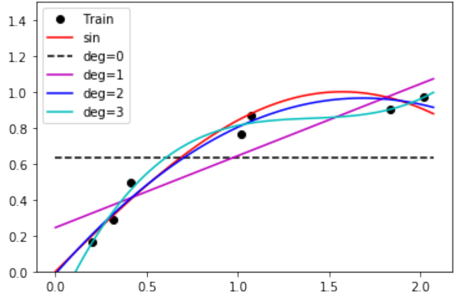
- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model
- Linear and quadratic models adequately describe the pattern

Overfitting and Underfitting: examples



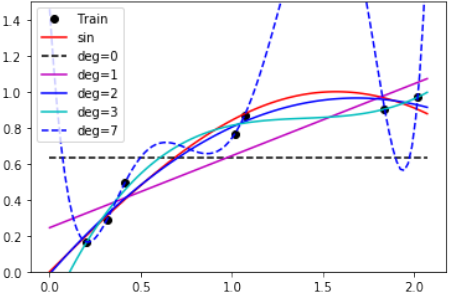
- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model
- Linear and quadratic models adequately describe the pattern
- High-degree polynomials can exactly pass through the points of the training sample

Overfitting and Underfitting: examples



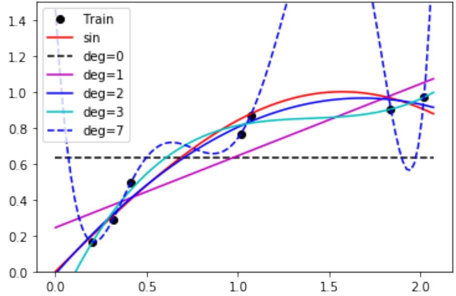
- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model
- Linear and quadratic models adequately describe the pattern
- High-degree polynomials can exactly pass through the points of the training sample

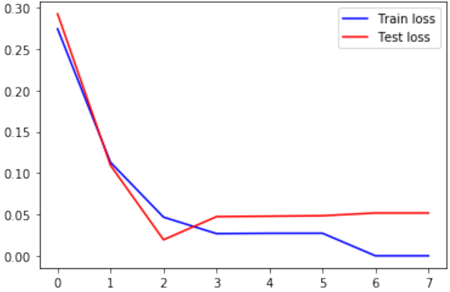
Overfitting and Underfitting: examples



- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model
- Linear and quadratic models adequately describe the pattern
- High-degree polynomials can exactly pass through the points of the training sample

Overfitting and Underfitting: examples





- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model
- Linear and quadratic models adequately describe the pattern
- High-degree polynomials can exactly pass through the points of the training sample

On parameters and hyperparameters

In the example with the approximation of the unknown dependence by the polynomial $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$:

On parameters and hyperparameters

In the example with the approximation of the unknown dependence by the polynomial $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$:

- **Parameters**: coefficients $a_n, a_{n-1}, \ldots, a_1, a_0$, and they are adjusted during model training

On parameters and hyperparameters

In the example with the approximation of the unknown dependence by the polynomial $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$:

- **Parameters**: coefficients $a_n, a_{n-1}, \ldots, a_1, a_0$, and they are adjusted during model training
- **Hyperparameters**: the degree of the polynomial n, which is chosen before training starts; then chosen from the set of hyperparameters tested on the validation set

Derivation of mean squared error expression

Definitions

Let $y = y(x) = f(x) + \varepsilon$ be the target dependence, where $f(x)$ is the deterministic function, $\varepsilon \sim N(0, \sigma^2)$ and $a(x)$ is the machine learning algorithm.

Derivation of mean squared error expression

Definitions

Let $y = y(x) = f(x) + \varepsilon$ be the target dependence, where $f(x)$ is the deterministic function, $\varepsilon \sim N(0, \sigma^2)$ and $a(x)$ is the machine learning algorithm.

We assume that ε and a are independent $(Ea\varepsilon = EaE\varepsilon)$. $Ey = Ef, Dy = D\varepsilon = \sigma^2$.

Derivation of mean squared error expression

Definitions

Let $y = y(x) = f(x) + \varepsilon$ be the target dependence, where $f(x)$ is the deterministic function, $\varepsilon \sim N(0, \sigma^2)$ and $a(x)$ is the machine learning algorithm.

We assume that ε and a are independent $(Ea\varepsilon = EaE\varepsilon)$. $Ey = Ef, Dy = D\varepsilon = \sigma^2$.

Squared Error Decomposition

$$E(y-a)^2 = E(y^2 + a^2 - 2ya) = Ey^2 + Ea^2 - 2Eya =$$

Derivation of mean squared error expression

Definitions

Let $y = y(x) = f(x) + \varepsilon$ be the target dependence, where $f(x)$ is the deterministic function, $\varepsilon \sim N(0, \sigma^2)$ and $a(x)$ is the machine learning algorithm.

We assume that ε and a are independent $(Ea\varepsilon = EaE\varepsilon)$. $Ey = Ef, Dy = D\varepsilon = \sigma^2$.

Squared Error Decomposition

$$E(y-a)^2 = E(y^2 + a^2 - 2ya) = Ey^2 + Ea^2 - 2Eya =$$
$$= Ey^2 + Ea^2 - 2E(f+\varepsilon)a = Ey^2 + Ea^2 - 2Efa - 2E\varepsilon a =$$

Derivation of mean squared error expression

Definitions

Let $y = y(x) = f(x) + \varepsilon$ be the target dependence, where $f(x)$ is the deterministic function, $\varepsilon \sim N(0, \sigma^2)$ and $a(x)$ is the machine learning algorithm.

We assume that ε and a are independent $(Ea\varepsilon = EaE\varepsilon)$. $Ey = Ef, Dy = D\varepsilon = \sigma^2$.

Squared Error Decomposition

$$
\begin{gathered}
E(y-a)^2 = E(y^2 + a^2 - 2ya) = Ey^2 + Ea^2 - 2Eya = \\
= Ey^2 + Ea^2 - 2E(f+\varepsilon)a = Ey^2 + Ea^2 - 2Efa - 2E\varepsilon a = \\
= Ey^2 - (Ey)^2 + (Ey)^2 + Ea^2 - (Ea)^2 + (Ea)^2 - 2fEa =
\end{gathered}
$$

Derivation of mean squared error expression

Definitions

Let $y = y(x) = f(x) + \varepsilon$ be the target dependence, where $f(x)$ is the deterministic function, $\varepsilon \sim N(0, \sigma^2)$ and $a(x)$ is the machine learning algorithm.

We assume that ε and a are independent $(Ea\varepsilon = EaE\varepsilon)$. $Ey = Ef, Dy = D\varepsilon = \sigma^2$.

Squared Error Decomposition

$$\begin{gathered} E(y-a)^2 = E(y^2 + a^2 - 2ya) = Ey^2 + Ea^2 - 2Eya = \\ = Ey^2 + Ea^2 - 2E(f+\varepsilon)a = Ey^2 + Ea^2 - 2Efa - 2E\varepsilon a = \\ = Ey^2 - (Ey)^2 + (Ey)^2 + Ea^2 - (Ea)^2 + (Ea)^2 - 2fEa = \\ = Dy + Da + (Ey)^2 + (Ea)^2 - 2fEa = \end{gathered}$$

Derivation of mean squared error expression

Definitions

Let $y = y(x) = f(x) + \varepsilon$ be the target dependence, where $f(x)$ is the deterministic function, $\varepsilon \sim N(0, \sigma^2)$ and $a(x)$ is the machine learning algorithm.

We assume that ε and a are independent $(Ea\varepsilon = EaE\varepsilon)$. $Ey = Ef, Dy = D\varepsilon = \sigma^2$.

Squared Error Decomposition

$$
\begin{gathered}
E(y-a)^2 = E(y^2 + a^2 - 2ya) = Ey^2 + Ea^2 - 2Eya = \\
= Ey^2 + Ea^2 - 2E(f+\varepsilon)a = Ey^2 + Ea^2 - 2Efa - 2E\varepsilon a = \\
= Ey^2 - (Ey)^2 + (Ey)^2 + Ea^2 - (Ea)^2 + (Ea)^2 - 2fEa = \\
= Dy + Da + (Ey)^2 + (Ea)^2 - 2fEa = \\
= Dy + Da + (Ef)^2 - 2fEa + (Ea)^2 =
\end{gathered}
$$

Derivation of mean squared error expression

Definitions

Let $y = y(x) = f(x) + \varepsilon$ be the target dependence, where $f(x)$ is the deterministic function, $\varepsilon \sim N(0, \sigma^2)$ and $a(x)$ is the machine learning algorithm.

We assume that ε and a are independent $(Ea\varepsilon = EaE\varepsilon)$. $Ey = Ef, Dy = D\varepsilon = \sigma^2$.

Squared Error Decomposition

$$
\begin{gathered}
E(y-a)^2 = E(y^2 + a^2 - 2ya) = Ey^2 + Ea^2 - 2Eya = \\
= Ey^2 + Ea^2 - 2E(f+\varepsilon)a = Ey^2 + Ea^2 - 2Efa - 2E\varepsilon a = \\
= Ey^2 - (Ey)^2 + (Ey)^2 + Ea^2 - (Ea)^2 + (Ea)^2 - 2fEa = \\
= Dy + Da + (Ey)^2 + (Ea)^2 - 2fEa = \\
= Dy + Da + (Ef)^2 - 2fEa + (Ea)^2 = \\
= Dy + Da + (E(f-a))^2 = \sigma^2 + variance(a) + bias^2(f,a)
\end{gathered}
$$

Additional definitions

Definition

Variance — variance of responses of algorithms a(x).
Characterizes the variety of algorithms (due to the randomness of the training sample, noise, learning stochasticity, etc.)

Additional definitions

Definition

Variance — variance of responses of algorithms a(x).
Characterizes the variety of algorithms (due to the randomness of the training sample, noise, learning stochasticity, etc.)

Definition

Bias — expectation of the difference between the true answer and the one chosen by an algorithm.
In the example above — this is $E(f - a)$.
Characterizes the ability of the model to adjust to the target dependence

Additional definitions

Definition

Variance — variance of responses of algorithms a(x).
Characterizes the variety of algorithms (due to the randomness of the training sample, noise, learning stochasticity, etc.)

Definition

Bias — expectation of the difference between the true answer and the one chosen by an algorithm.
In the example above — this is $E(f - a)$.
Characterizes the ability of the model to adjust to the target dependence

Definition

The mean squared error decomposition in the example above is called the **bias-variance tradeoff**

Model of Optimal Complexity: Classic View

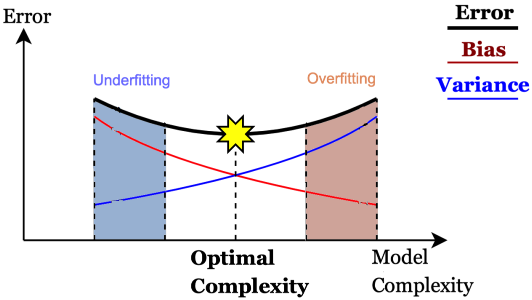
- Simple models tend to be underfit

Model of Optimal Complexity: Classic View

- Simple models tend to be underfit
- Complex models tend to overfit

Model of Optimal Complexity: Classic View

- Simple models tend to be underfit
- Complex models tend to overfit
- The optimal complexity of the model is somewhere between



Model of Optimal Complexity: Recent Empirical Evidence

- Previously, it was not technically possible to look at the quality in the case of a model of huge complexity

[6] Advani, Madhu S., Andrew M. Saxe, and Haim Sompolinsky. "High-dimensional dynamics of generalization error in neural networks." 2017

Model of Optimal Complexity: Recent Empirical Evidence

- Previously, it was not technically possible to look at the quality in the case of a model of huge complexity
- With the development of technology, it has become possible to train models with millions and even billions of parameters

[6] Advani, Madhu S., Andrew M. Saxe, and Haim Sompolinsky. "High-dimensional dynamics of generalization error in neural networks." 2017

Model of Optimal Complexity: Recent Empirical Evidence

- Previously, it was not technically possible to look at the quality in the case of a model of huge complexity
- With the development of technology, it has become possible to train models with millions and even billions of parameters
- It turned out that as complexity increases, the error first behaves as predicted by the bias-variance tradeoff, and then suddenly starts to decrease[6] and goes even to a lower error level!

[6] Advani, Madhu S., Andrew M. Saxe, and Haim Sompolinsky. "High-dimensional dynamics of generalization error in neural networks." 2017

Model of Optimal Complexity: Recent Empirical Evidence

- Previously, it was not technically possible to look at the quality in the case of a model of huge complexity
- With the development of technology, it has become possible to train models with millions and even billions of parameters
- It turned out that as complexity increases, the error first behaves as predicted by the bias-variance tradeoff, and then suddenly starts to decrease[6] and goes even to a lower error level!
- Tipping point — the point at which the complexity of the model is comparable to the cardinality of the training set (interpolation threshold)

[6] Advani, Madhu S., Andrew M. Saxe, and Haim Sompolinsky. "High-dimensional dynamics of generalization error in neural networks." 2017

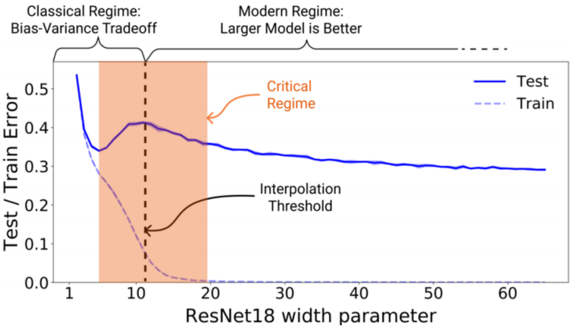
Model of Optimal Complexity: Recent Empirical Evidence

- Previously, it was not technically possible to look at the quality in the case of a model of huge complexity
- With the development of technology, it has become possible to train models with millions and even billions of parameters
- It turned out that as complexity increases, the error first behaves as predicted by the bias-variance tradeoff, and then suddenly starts to decrease[6] and goes even to a lower error level!
- Tipping point — the point at which the complexity of the model is comparable to the cardinality of the training set (interpolation threshold)
- This behavior is called **double descent**

[6] Advani, Madhu S., Andrew M. Saxe, and Haim Sompolinsky. "High-dimensional dynamics of generalization error in neural networks." 2017

Model of Optimal Complexity: Double Descent

- Example of double descent in practice[7]:



[7] Image source: `https://arxiv.org/pdf/1912.02292.pdf`

Takeaway notes

1. It is necessary to divide the available data into training, validation and test sets — from the very beginning

Takeaway notes

1. It is necessary to divide the available data into training, validation and test sets — from the very beginning
2. It is necessary to monitor the complexity of the model — too large will lead to overfitting, too small — to underfitting.

Takeaway notes

1. It is necessary to divide the available data into training, validation and test sets — from the very beginning
2. It is necessary to monitor the complexity of the model — too large will lead to overfitting, too small — to underfitting.
   - Both will increase the error on the test set

Takeaway notes

1. It is necessary to divide the available data into training, validation and test sets — from the very beginning
2. It is necessary to monitor the complexity of the model — too large will lead to overfitting, too small — to underfitting.
   - Both will increase the error on the test set
3. In the case of a huge amount of data and parameters ($\approx$billions), classical estimates stop working

Time for questions

Thank you!