Machine Learning

Empirical and Structural Risk. Error Decomposition. Model Selection. Underfitting and overfitting

Aleksandr Petiushko

ML Research

January 22nd, 2024

SOFIA UNIVERSITY

AP

Content

Content

1. Structural Risk and its Minimization
2. Overfittning and underfitting

Content

Content

1. Structural Risk and its Minimization
2. Overfittning and underfitting
3. Model Selection overview
4. Bias-variance tradeoff

Content

Instance-based learning

- X – set of objects descriptions, Y – set of objects labels
- Unknown target dependency: mapping $y : X \to Y$
- Finite training set: $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, so as $y_i = y(x_i)$

Instance-based learning

- X – set of objects descriptions, Y – set of objects labels
- Unknown target dependency: mapping $y : X \to Y$
- Finite training set: $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, so as $y_i = y(x_i)$
- **Task of inductive (or instance-based) learning**: construct the algorithm $a : X \to Y$, to approximate the target dependency y not only on training set X^m, but also on the whole set X

Instance-based learning

- X – set of objects descriptions, Y – set of objects labels
- Unknown target dependency: mapping $y : X \to Y$
- Finite training set: $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$, so as $y_i = y(x_i)$
- **Task of inductive (or instance-based) learning**: construct the algorithm $a : X \to Y$, to approximate the target dependency y not only on training set X^m, but also on the whole set X
- **Empirical Risk** – average error of a on X^m

Instance-based learning

- X – set of objects descriptions, Y – set of objects labels
- Unknown target dependency: mapping $y : X \to Y$
- Finite training set: $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, so as $y_i = y(x_i)$
- **Task of inductive (or instance-based) learning**: construct the algorithm $a : X \to Y$, to approximate the target dependency y not only on training set X^m, but also on the whole set X
- **Empirical Risk** – average error of a on X^m
- **Empirical Risk Minimization** (ERM) – the common approach to solve the broad range of tasks of inductive learning (e.g., classification / regression tasks)

Empirical risk: definitions

Loss function $L(\hat{y}, y)$

Characteristics of difference between the prediction $\hat{y} = a(x)$ and the *ground truth* label $y = y(x)$ for object $x \in X$

Empirical risk: definitions

Loss function $L(\hat{y}, y)$

Characteristics of difference between the prediction $\hat{y} = a(x)$ and the *ground truth* label $y = y(x)$ for object $x \in X$

Set of algorithms $A = \{a : X \to Y\}$

We will conduct the search of mapping approximating the unknown target dependency inside this set

Empirical risk: definitions

Loss function $L(\hat{y}, y)$

Characteristics of difference between the prediction $\hat{y} = a(x)$ and the *ground truth* label $y = y(x)$ for object $x \in X$

Set of algorithms $A = \{a : X \to Y\}$

We will conduct the search of mapping approximating the unknown target dependency inside this set

Empirical Risk (ER)

Performance metric reflecting the average error made by an algorithm a upon the set X^m: $R(a, X^m) = \frac{1}{m}\sum_{i=1}^{m} L(a(x_i), y(x_i))$

Empirical Risk Minimization

Empirical Risk Minimization (ERM)

Given a set of algorithms A need to find the algorithm minimizing the empirical risk:

$$a = \arg\min_{a \in A} R(a, X^m)$$

Empirical Risk Minimization

Empirical Risk Minimization (ERM)

Given a set of algorithms A need to find the algorithm minimizing the empirical risk:

$$a = \arg\min_{a \in A} R(a, X^m)$$

ERM pros

Universal and constructive approach allowing to reduce the learning task to the task of numerical optimization

Empirical Risk Minimization

Empirical Risk Minimization (ERM)

Given a set of algorithms A need to find the algorithm minimizing the empirical risk:

$$a = \arg\min_{a \in A} R(a, X^m)$$

ERM pros

Universal and constructive approach allowing to reduce the learning task to the task of numerical optimization

ERM cons

Overfitting on the training set X^m. Happens almost always when using ERM, because the performance criteria is the error **on the very same set** (solution: to measure the performance it makes sense to change the set)

Loss functions examples

Classification task

- Classification error: $L(a, x) = L(\hat{y}, y) = [\hat{y} \neq y] = 1 - \delta_y(\hat{y})$
- The function is discontinuous $\Rightarrow$ ERM is a task of combinatorial optimization $\Rightarrow$ in many practical applications can be reduced to the search of maximal consistent subsystem of inequality system (number of inequalities is equal to the number of training examples m) $\Rightarrow$ NP-hard

Loss functions examples

Classification task

- Classification error: $L(a,x) = L(\hat{y}, y) = [\hat{y} \neq y] = 1 - \delta_y(\hat{y})$
- The function is discontinuous $\Rightarrow$ ERM is a task of combinatorial optimization $\Rightarrow$ in many practical applications can be reduced to the search of maximal consistent subsystem of inequality system (number of inequalities is equal to the number of training examples m) $\Rightarrow$ NP-hard

Regression task

- Squared error: $L(\hat{y}, y) = (\hat{y} - y)^2$

Structural Risk

- In order to deal with overfitting tied to ERM, the Structural improvement is commonly used

Structural Risk

- In order to deal with overfitting tied to ERM, the Structural improvement is commonly used
- **Structural Ris**k: an Empirical Risk with an additional term called *regularization penalty*

Structural Risk

- In order to deal with overfitting tied to ERM, the Structural improvement is commonly used
- **Structural Risk**: an Empirical Risk with an additional term called *regularization penalty*
- **Structural Risk Minimization** (SRM): $S(a, X^m) = R(a, X^m) + \lambda C(a) \to \min$, where $\lambda > 0$ is some weight of the regularization term, and $C(a) \geq 0$ is the regularization cost associated with the function $a : X \to Y$

Structural Risk

- In order to deal with overfitting tied to ERM, the Structural improvement is commonly used
- **Structural Risk**: an Empirical Risk with an additional term called *regularization penalty*
- **Structural Risk Minimization** (SRM): $S(a, X^m) = R(a, X^m) + \lambda C(a) \to \min$, where $\lambda > 0$ is some weight of the regularization term, and $C(a) \geq 0$ is the regularization cost associated with the function $a : X \to Y$

SRM pros

A simple add-on to ERM allowing to avoid the overfitting

Structural Risk

- In order to deal with overfitting tied to ERM, the Structural improvement is commonly used
- **Structural Risk**: an Empirical Risk with an additional term called *regularization penalty*
- **Structural Risk Minimization** (SRM): $S(a, X^m) = R(a, X^m) + \lambda C(a) \to \min$, where $\lambda > 0$ is some weight of the regularization term, and $C(a) \geq 0$ is the regularization cost associated with the function $a : X \to Y$

SRM pros

A simple add-on to ERM allowing to avoid the overfitting

SRM cons

Hard to guess in advance what is the right form of the regularization term $C(a)$ and what should be the regularization weight λ

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

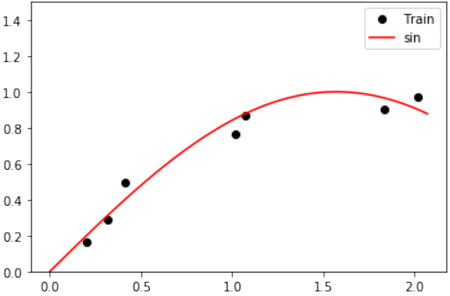
One of the main causes

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

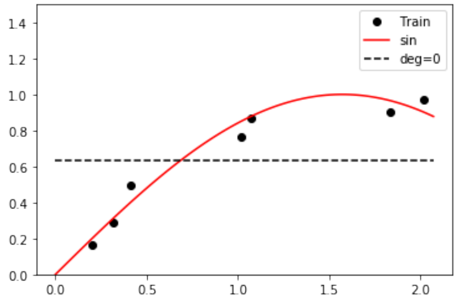
One of the main detection methods

Train error observation

Overfitting and Underfitting: examples

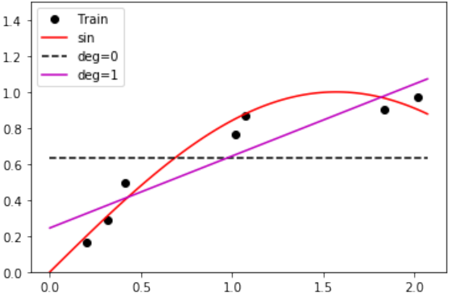


Overfitting and Underfitting: examples



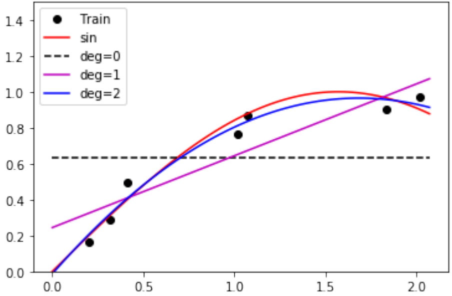
- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model

Overfitting and Underfitting: examples



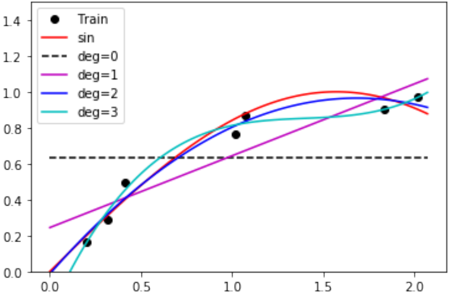
- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model
- Linear and quadratic models adequately describe the pattern

Overfitting and Underfitting: examples



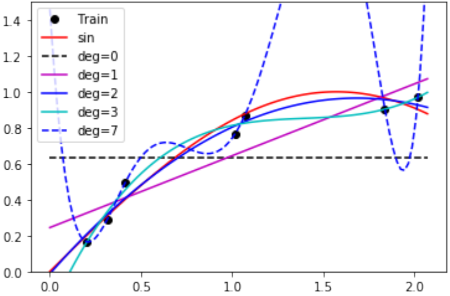
- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model
- Linear and quadratic models adequately describe the pattern
- High-degree polynomials can exactly pass through the points of the training sample

Overfitting and Underfitting: examples



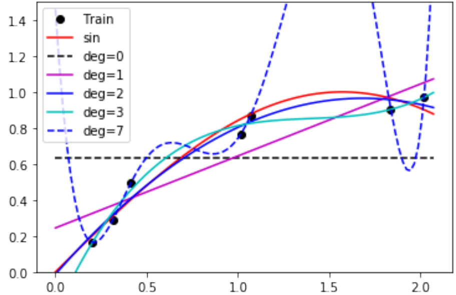
- A polynomial of degree zero cannot approximate the dependence well due to the limited parameter space of the model
- Linear and quadratic models adequately describe the pattern
- High-degree polynomials can exactly pass through the points of the training sample

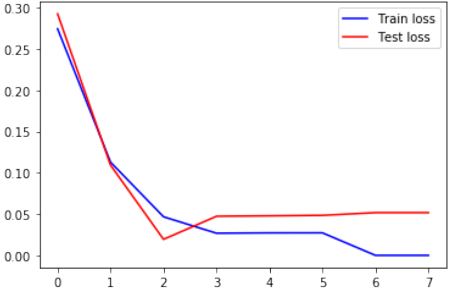
Overfitting and Underfitting: examples



- 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 $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$:

- **Parameters**: coefficients $a_n, a_{n-1}, \dots, a_1, a_0$, 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

Notes about model selection process

The following steps should be considered:

Notes about model selection process

The following steps should be considered:

- Data considerations (a huge or small amount => in advance to think about the model complexity to avoid either under- or overfitting)

Notes about model selection process

The following steps should be considered:

- Data considerations (a huge or small amount => in advance to think about the model complexity to avoid either under- or overfitting)
- Categorical vs Numerical input features (different models, e.g. different versions of gradient boosting, works differently with different feature types)

Notes about model selection process

The following steps should be considered:

- Data considerations (a huge or small amount => in advance to think about the model complexity to avoid either under- or overfitting)
- Categorical vs Numerical input features (different models, e.g. different versions of gradient boosting, works differently with different feature types)
- Training / inference speed (tradeoff between good and fast model)

Notes about model selection process

The following steps should be considered:

- Data considerations (a huge or small amount => in advance to think about the model complexity to avoid either under- or overfitting)
- Categorical vs Numerical input features (different models, e.g. different versions of gradient boosting, works differently with different feature types)
- Training / inference speed (tradeoff between good and fast model)
- Explainability (tradeoff between good and interpretable model)

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

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

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

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

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

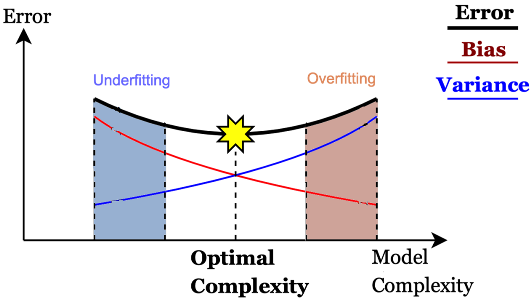
- Simple models tend to be underfit

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

[1] 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

[1] 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[1] and goes even to a lower error level!

[1] 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[1] 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)

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

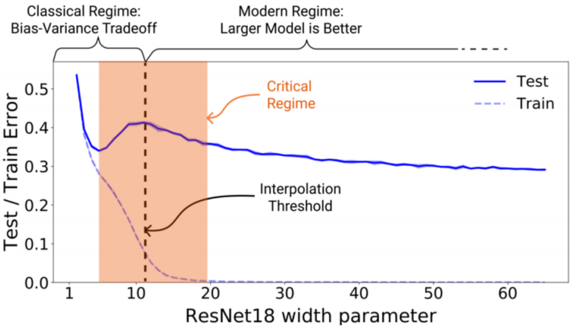
Model of Optimal Complexity: Recent Empirical Evidence

- 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[1] 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**

[1] 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[2]:



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

Takeaway notes

1. Structural Risk is needed to avoid overfitting doing Empirical Risk Minimization

Takeaway notes

1. Structural Risk is needed to avoid overfitting doing Empirical Risk Minimization
2. There is a tradeoff between bias and variance for any ML model

Takeaway notes

1. Structural Risk is needed to avoid overfitting doing Empirical Risk Minimization
2. There is a tradeoff between bias and variance for any ML model
3. It is necessary to monitor the complexity of the model — too large will lead to overfitting, too small — to underfitting.

Takeaway notes

1. Structural Risk is needed to avoid overfitting doing Empirical Risk Minimization
2. There is a tradeoff between bias and variance for any ML model
3. 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. Structural Risk is needed to avoid overfitting doing Empirical Risk Minimization
2. There is a tradeoff between bias and variance for any ML model
3. 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
4. A lot of different considerations should be taken into account while thinking of the most appropriate model choice

Takeaway notes

1. Structural Risk is needed to avoid overfitting doing Empirical Risk Minimization
2. There is a tradeoff between bias and variance for any ML model
3. 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
4. A lot of different considerations should be taken into account while thinking of the most appropriate model choice
5. In case of a huge amount of data and parameters (≈billions), classical estimates stop working

Thank you!