AMMI – Introduction to Deep Learning

2.2. Over and under fitting

François Fleuret
https://fleuret.org/ammi-2018/
Wed Oct 24 09:37:08 CAT 2018
You want to hire someone, and you evaluate candidates by asking them ten technical yes/no questions.
You want to hire someone, and you evaluate candidates by asking them ten technical yes/no questions.

Would you feel confident if you interviewed one candidate and he makes a perfect score?
You want to hire someone, and you evaluate candidates by asking them ten technical yes/no questions.

Would you feel confident if you interviewed one candidate and he makes a perfect score?

What about interviewing ten candidates and picking the best?
You want to hire someone, and you evaluate candidates by asking them ten technical yes/no questions.

Would you feel confident if you interviewed one candidate and he makes a perfect score?

What about interviewing ten candidates and picking the best? What about interviewing one thousand?
With

\[ Q_n^k \sim \mathcal{B}(0.5), \quad n = 1, \ldots, 1000, \quad k = 1, \ldots, 10, \]

independent standing for “candidate \( n \) answered question \( k \) correctly”, we have

\[ \forall n, \quad P(\forall k, Q_n^k = 1) = \frac{1}{1024} \]

and

\[ P(\exists n, \forall k, Q_n^k = 1) \approx 0.62. \]
With\[Q^n_k \sim \mathcal{B}(0.5), \ n = 1, \ldots, 1000, \ k = 1, \ldots, 10,\]\independent standing for “candidate \(n\) answered question \(k\) correctly”, we have\[
\forall n, \ P(\forall k, Q^n_k = 1) = \frac{1}{1024}\]
and\[
P(\exists n, \forall k, Q^n_k = 1) \approx 0.62.
\]
So there is 62% chance that among 1,000 candidates answering completely at random, one will score perfectly.
With

\[ Q^n_k \sim \mathcal{B}(0.5), \ n = 1, \ldots, 1000, \ k = 1, \ldots, 10, \]

independent standing for “candidate \( n \) answere question \( k \) correctly”, we have

\[ \forall n, \ P(\forall k, Q^n_k = 1) = \frac{1}{1024} \]

and

\[ P(\exists n, \forall k, Q^n_k = 1) \approx 0.62. \]

So there is 62\% chance that among 1,000 candidates answering completely at random, one will score perfectly.

Selecting a candidate based on a statistical estimator biases the said estimator for that candidate. And you need a greater number of “competence checks” if you have a larger pool of candidates.
Over and under-fitting, capacity. $K$-nearest-neighbors
A simple classification procedure is the “$K$-nearest neighbors.”

Given

$$(x_n, y_n) \in \mathbb{R}^D \times \{1, \ldots, C\}, \ n = 1, \ldots, N$$

to predict the $y$ associated to a new $x$, take the $y_n$ of the closest $x_n$:

$$n^*(x) = \arg\min_n \|x_n - x\|$$

$$f^*(x) = y_{n^*(x)}.$$

This recipe corresponds to $K = 1$, and makes the empirical training error zero.
$K = 1$
Under mild assumptions of regularities of $\mu_{X,Y}$, for $N \to \infty$ the asymptotic error rate of the 1-NN is less than twice the (optimal!) Bayes' Error rate.
Under mild assumptions of regularities of $\mu_X, \gamma$, for $N \to \infty$ the asymptotic error rate of the 1-NN is less than twice the (optimal!) Bayes’ Error rate.

It can be made more stable by looking at the $K > 1$ closest training points, and taking the majority vote.
Under mild assumptions of regularities of $\mu_X, \gamma$, for $N \to \infty$ the asymptotic error rate of the 1-NN is less than twice the (optimal!) Bayes’ Error rate.

It can be made more stable by looking at the $K > 1$ closest training points, and taking the majority vote.

If we let also $K \to \infty$ “not too fast”, the error rate is the (optimal!) Bayes’ Error rate.
Training set
Prediction (K=1)
Training set
Prediction (K=1)
Training set
Votes (K=51)
Prediction (K=51)
Training set
Votes (K=51)
Prediction \((K=51)\)
Francois Fleuret AMMI – Introduction to Deep Learning / 2.2. Over and under fitting

The graph illustrates the trade-off between underfitting and overfitting in the context of model training and testing error as a function of the parameter $K$. The x-axis represents $K$ values ranging from 1000 to 1, while the y-axis shows the error in training and testing sets. The graph shows that as $K$ decreases, the error in the training set decreases, indicating a better fit to the training data. However, this also increases the error in the test set, highlighting the risk of overfitting. Conversely, as $K$ increases, the test error decreases, but the training error also increases, indicating underfitting. The optimal value of $K$ is where the generalization error (test error) is minimized while the training error is not excessively high, indicating the best balance between fitting the training data and generalizing to unseen data.
Over and under-fitting, capacity, polynomials
Given a polynomial model

$$\forall x, \alpha_0, \ldots, \alpha_D \in \mathbb{R}, \ f(x; \alpha) = \sum_{d=0}^{D} \alpha_d x^d.$$
Given a polynomial model

$$\forall x, \alpha_0, \ldots, \alpha_D \in \mathbb{R}, \quad f(x; \alpha) = \sum_{d=0}^{D} \alpha_d x^d.$$ 

and training points \((x_n, y_n) \in \mathbb{R}^2, n = 1, \ldots, N\), the quadratic loss is

$$\mathcal{L}(\alpha) = \sum_n (f(x_n; \alpha) - y_n)^2$$
Given a polynomial model

\[ \forall x, \alpha_0, \ldots, \alpha_D \in \mathbb{R}, \quad f(x; \alpha) = \sum_{d=0}^{D} \alpha_d x^d. \]

and training points \((x_n, y_n) \in \mathbb{R}^2, n = 1, \ldots, N\), the quadratic loss is

\[ \mathcal{L}(\alpha) = \sum_n (f(x_n; \alpha) - y_n)^2 \]

\[ = \sum_n \left( \sum_{d=0}^{D} \alpha_d x_n^d - y_n \right)^2 \]
Given a polynomial model

$$\forall x, \alpha_0, \ldots, \alpha_D \in \mathbb{R}, \quad f(x; \alpha) = \sum_{d=0}^{D} \alpha_d x^d.$$ 

and training points \((x_n, y_n) \in \mathbb{R}^2, n = 1, \ldots, N\), the quadratic loss is

$$\mathcal{L}(\alpha) = \sum_n (f(x_n; \alpha) - y_n)^2$$

$$= \sum_n \left( \sum_{d=0}^{D} \alpha_d x_n^d - y_n \right)^2$$

$$= \left\| \begin{pmatrix} x_0^0 & \cdots & x_1^D \\ \vdots & \ddots & \vdots \\ x_0^0 & \cdots & x_N^D \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_D \end{pmatrix} - \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \right\|^2.$$
Given a polynomial model

\[
\forall x, \alpha_0, \ldots, \alpha_D \in \mathbb{R}, \quad f(x; \alpha) = \sum_{d=0}^{D} \alpha_d x^d.
\]

and training points \((x_n, y_n) \in \mathbb{R}^2, n = 1, \ldots, N\), the quadratic loss is

\[
\mathcal{L}(\alpha) = \sum_{n} (f(x_n; \alpha) - y_n)^2
\]

\[
= \sum_{n} \left( \sum_{d=0}^{D} \alpha_d x_n^d - y_n \right)^2
\]

\[
= \begin{bmatrix} x_0 & \cdots & x_D \\ x_1 & \cdots & x_1 \\ \vdots & \ddots & \vdots \\ x_N & \cdots & x_N \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_D \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}
\]

Hence, minimizing this loss is a standard quadratic problem, for which we have efficient algorithms.
\[
\arg\min_\alpha \left\| \begin{pmatrix} x_0^1 & \cdots & x_D^1 \\ \vdots & \vdots & \vdots \\ x_0^N & \cdots & x_D^N \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_D \end{pmatrix} - \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \right\|^2
\]

```python
def fit_polynomial(D, x, y):
    X = torch.empty(x.size(0), D + 1)
    for d in range(D + 1):
        X[:, d] = x.pow(d)

    # gels expects a matrix for target
    Y = y.view(-1, 1)

    # LAPACK's GEneralized Least-Square
    alpha, _ = torch.gels(Y, X)

    return alpha[:D+1, 0]
```
D, N = 4, 100
x = torch.linspace(-math.pi, math.pi, N)
y = x.sin()
alpha = fit_polynomial(D, x, y)

X = torch.empty(N, D + 1)
for d in range(D + 1):
    X[:, d] = x.pow(d)

yhat = X.mv(alpha)

for k in range(N):
    print(x[k].item(), y[k].item(), yhat[k].item())
We can use that model to illustrate how the prediction changes when we increase the degree or the regularization.
Degree D=0

Data

\[ f^* \]
Degree D=1

Data

\[ f^* \]
Degree D=2

Data

$\hat{f}$

François Fleuret

AMMI – Introduction to Deep Learning / 2.2. Over and under fitting
Degree $D=3$

Data

\[ f^* \]
Degree D=6

Diagram showing data points and a curve fit for degree D=6.
Degree D=8

Data

$\mathbf{f}^*$
Degree D=9

Data

\[ f^* \]

François Fleuret

AMMI – Introduction to Deep Learning / 2.2. Over and under fitting
Over and under fitting

Error (MSE) vs Degree

Train
Test

Degree

Error (MSE)

Train
Test

François Fleuret
AMMI – Introduction to Deep Learning / 2.2. Over and under fitting
Degree D=1

\[ f \]

\[ f^* \]}
Degree $D=2$

![Degree $D=2$ graph](image)
Degree D=4

![Graph showing function curves](image-url)
Degree D=9

$\deg D=9$
We can reformulate this control of the degree with a penalty

\[
\mathcal{L}(\alpha) = \sum_n (f(x_n; \alpha) - y_n)^2 + \sum_d l_d(\alpha_d)
\]

where

\[
l_d(\alpha) = \begin{cases} 
0 & \text{if } d \leq D \text{ or } \alpha = 0 \\
+\infty & \text{otherwise.}
\end{cases}
\]

Such a penalty kills any term of degree \(> D\).
We can reformulate this control of the degree with a penalty

$$\mathcal{L}(\alpha) = \sum_n (f(x_n; \alpha) - y_n)^2 + \sum_d l_d(\alpha_d)$$

where

$$l_d(\alpha) = \begin{cases} 
0 & \text{if } d \leq D \text{ or } \alpha = 0 \\
\infty & \text{otherwise.} 
\end{cases}$$

Such a penalty kills any term of degree $> D$.

This motivates the use of more subtle variants. For instance, to keep all this quadratic

$$\mathcal{L}(\alpha) = \sum_n (f(x_n; \alpha) - y_n)^2 + \rho \sum_d \alpha_d^2.$$
$D=9, \rho=1e1$

Graph showing $f^*$ and $f$.
D = 9, \rho = 1 \times 10^0

f^*
f
D=9, $\rho=1e^{-2}$

The graph illustrates the function $f^*$ and $f$ as $x$ increases from 0 to 1. The function $f^*$ is represented by blue lines, while $f$ is represented by red lines. The graph shows a clear separation between the two functions, indicating that $f^*$ is a better fit for the data compared to $f$. The parameter $\rho=1e^{-2}$ significantly affects the performance of the model, with a lower $\rho$ value leading to a tighter fit around $f^*$. This suggests that optimizing the hyperparameters, such as $\rho$, is crucial for achieving a good fit in deep learning models.
$D=9, \rho=1e^{-3}$
D=9, ρ=1e-4

- Plot showing the functions $f^*$ and $f$.

- The graph illustrates the relationship between the variables $x$ and $f(x)$ for different values of $x$ ranging from 0 to 1.

- The plot demonstrates the impact of overfitting and underfitting in the context of deep learning.
$D=9, \rho=1e^{-5}$

The graph shows the function $f^*$ (blue) and $f$ (red) as a function of $x$ ranging from 0 to 1. The function $f^*$ is a smooth curve that approaches the $x$-axis as $x$ increases, while $f$ is a series of curves that converge to $f^*$ for large values of $x$. The scale on the y-axis ranges from -0.5 to 1.5.
D=9, ρ=1e-7

\[ f^* \]

\[ f \]
D=9, \( \rho=1e-9 \)
D=9, $\rho=1\times10^{-13}$
We define the **capacity** of a set of predictors as its ability to model an arbitrary functional. This is a vague definition, difficult to make formal.
We define the **capacity** of a set of predictors as its ability to model an arbitrary functional. This is a vague definition, difficult to make formal.

A mathematically precise notion is the Vapnik–Chervonenkis dimension of a set of functions, which, in the Binary classification case, is the cardinality of the largest set that can be labeled arbitrarily (Vapnik, 1995).

It is a very powerful concept, but is poorly adapted to neural networks. We will not say more about it in this course.
Although the capacity is hard to define precisely, it is quite clear in practice how to modulate it for a given class of models.

In particular one can control over-fitting either by

- Impoverishing the space $\mathcal{F}$ (less functionals, constrained or degraded optimization).

- Make the choice of $f^*$ less dependent on data (penalty on coefficients, margin maximization, ensemble methods).
The end
References