Quick links: Schedule [https://intranet.fel.cvut.cz/cz/education/rozvrhy-ng.B232/public/html/predmety/61/70/p6170206.html] | Forum [https://cw.felk.cvut.cz/forum/forum-1874.html] | BRUTE [https://cw.felk.cvut.cz/brute/teacher/course/1595] | Lectures [https://cw.fel.cvut.cz/b232/courses/bev033dle/lectures] | Labs [https://cw.fel.cvut.cz/b232/courses/bev033dle/labs/start]

Lab 1: Preparations, Double Descent

This lab has an introduction part, explaining how the study process is organized and what tools you need to work with; and a light homework part.

Preparations:

- Labs and seminars format, grading, deadlines
- Upload system, forum
- How To: [/wiki/courses/bev033dle/labs/0_howto/start] – software, python IDEs

Lab: toy 2D problem with random lifting and linear regression

- useful inductive bias of GD for linear regression
- observe double descend phenomenon

Double Descent (5p)

In this and the next lab we will work with a simulated data for binary classification of 2D features drawn from a known Gaussian mixture model. Such toy data allows to visualize all the classifiers and is already sufficient to observe several interesting phenomena.

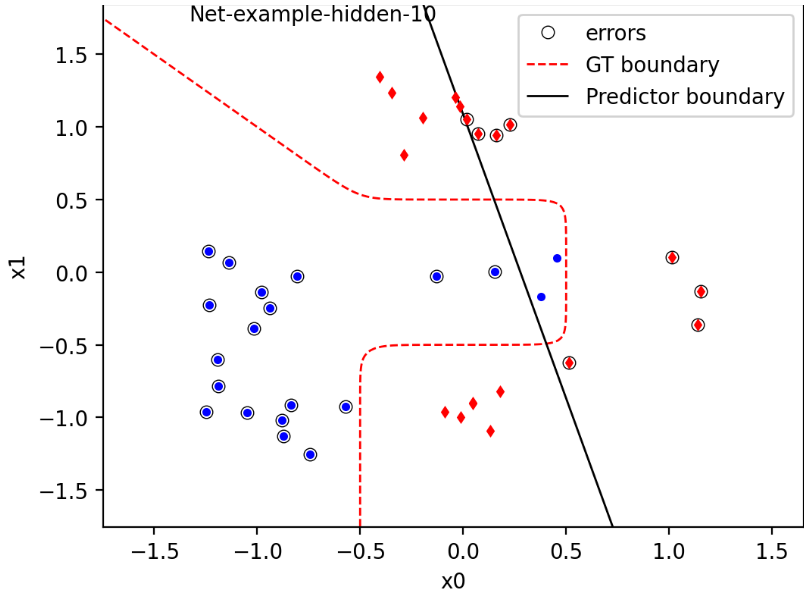
Templates

Use the provided template [/wiki/_media/courses/bev033dle/labs/lab0_ddescent/toy_model.py] with the toy generative model and lifting. It can be run as a script (example usage in "main") or by cells in the interactive mode.

Classification Problem

Let x denote observations in $\mathbb{R}^2$ and $y \in \{-1, 1\}$ denote the class label. In this lab we have chosen an artificial ground truth model $p(x, y)$, accessible to you through functions of the class `G2Model`. You can generate a training set and a test set yourself using the function `generate_sample`, which samples from this ground truth model. You can also access the optimal Bayesian classification according to the true model (function `classify`) and compare it to the classification by you neural network.

The next plot show an example of 40 data points, the GT decision boundary and a linear classifier boundary, drawn with `plot_boundary(train_data, net)` :



[/wiki/_detail/courses/bev033dle/labs/lab0_ddescent/net-example-hidden-10.png?id=courses%3Abev033dle%3Alabs%3Alab0_ddescent%3Astart]

Task

The goal of this lab is to observe a phenomenon which is somewhat counterintuitive and contradicting to the classical learning theory. The phenomenon is called double descent and is described in the paper: Belkin et al. 2018. Reconciling modern machine-learning practice and the classical bias–variance trade-off [https://www.pnas.org/content/116/32/15849]. We will replicate the experiment in a simple scenario described in Appendix C.2. of the paper. The classical learning theory (in particular structural risk minimization) suggests that when increasing the model capacity, we will be eventually overfitting to the data and generalize poorly. Therefore there should be a tradeoff between the training error and the model complexity. This experiment gives one example when it is not exactly so…

We will consider a neural network with one hidden layer:

$$\phi = \tanh(W_0 x + b_0) \qquad \text{(lifting)}$$
$$s = w^{\mathsf{T}} \phi + b, \qquad \text{(linear)}$$

The first layer, computing ϕ will will initialize randomly and will not be trained. So it will represent a fixed randomized lifting of the initial 2D features into a higher-dimensional space. Let us denote input dimension as $d = 2$ and hidden dimension as D. This lifting is provided in the \`Lifting\` class. Basically it initializes randomly but in a range relevant for the range of the dataset (a lifted feature which is constant for all data points is not useful). Due to use of $\tanh$, each feature ϕ_k can be imagined as a prediction of a random linear classifier.

We want to train the second layer, i.e. the linear (affine) transform defined by w, b.

Let $(x_i, y_i)_{i=1}^N$ be our training data, where the class label is encoded as $\{-1, 1\}$. Let $\theta = (w, b)$ be the vector of trainable parameters, of the size $D+1$. We will use the MSE loss as the training loss function, i.e.

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_i \|s_i - y_i\|^2,$$

The choice of fixed random features and MSE loss allows us to solve for the optimal parameters in a closed form instead of performing gradient descent.

Let Φ be the data matrix of size $N \times (D+1)$ with columns $(\phi(x_i), 1)$, i.e. our features augmented with constant 1. Let y be the vector of labels. With this notation we can write the loss as

$$\mathcal{L}(\theta) = \frac{1}{n}\|\Phi\theta - y\|^2 = \frac{1}{n}(\Phi\theta - y)^{\mathsf{T}}(\Phi\theta - y) = \frac{1}{n}\Big(\theta^{\mathsf{T}}\Phi^{\mathsf{T}}\Phi\theta - 2\theta^{\mathsf{T}}\Phi^{\mathsf{T}}y + y^{\mathsf{T}}y\Big).$$

Note that $\Phi^{\mathsf{T}}\Phi$ is a $(D+1)\times(D+1)$ matrix. The critical point conditions for this loss function are

$$\Phi^{\mathsf{T}}\Phi\theta = \Phi^{\mathsf{T}}y.$$

If we have at least D training data points in general positions, the matrix $\Phi^{\mathsf{T}}\Phi$ is invertible and we can find the unique minimum as

$$\theta = (\Phi^{\mathsf{T}}\Phi)^{-1}\Phi^{\mathsf{T}}y.$$

If we have less data points, there is a linear subspace of optimal solutions. The solution with the smallest norm is given by

$$\theta = (\Phi^{\mathsf{T}}\Phi)^{\dagger}\Phi^{\mathsf{T}}y,$$

where $A^{\dagger}$ is the pseudo-inverse of A. Important for us is that gradient descent started at $\theta = 0$ converges exactly to this pseudo-inverse solution (see the analysis in the recommended reading). This is an implicit property of GD dynamics, which turns out to be a good inductive bias.

If we add a regularization to the linear regression problem in the form of additional penalty $\lambda\|\theta\|^2$, the optimal solution will be always unique and given by

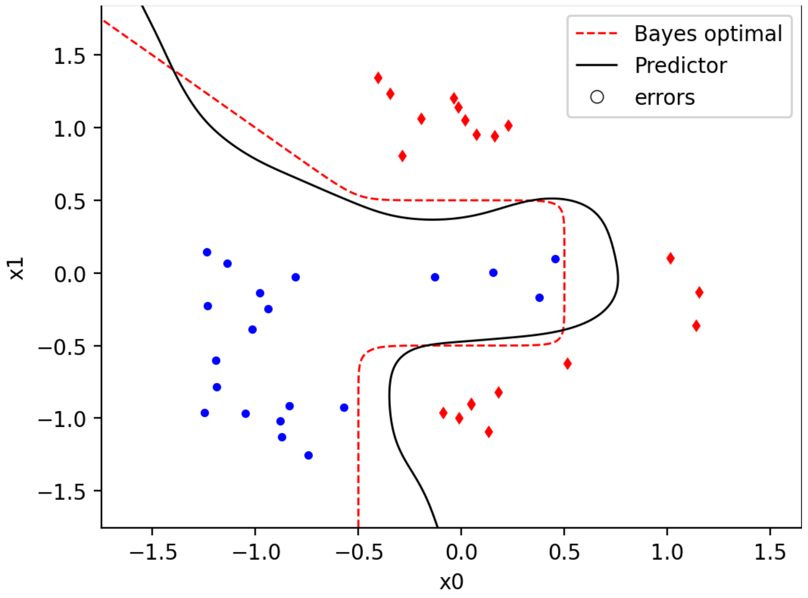
$$\theta = (\Phi^{\mathsf{T}}\Phi + \lambda I)^{-1}\Phi^{\mathsf{T}}y.$$

This is explicit regularization towards a small norm solution. If λ is chosen small, e.g., 10^{-6}, we expect the explicitly and implicitly regularized solutions to be similar.

Task 1 (2p)

Use the training set size of 200 points and hidden size $D = 10$. Find the optimal setting of parameters w, b by linear regression. Plot the classifier decision boundary. Report its training and test error. Try several values of D observing how the boundary complexity increases towards 40 and then gets smoother and smoother in the highly overparameterized mode (try $D = 1000$).

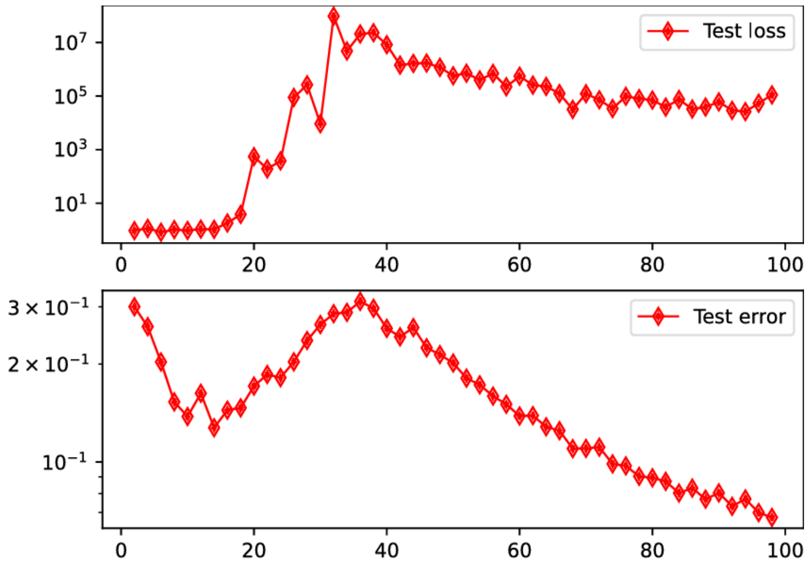
The expected result for D=1000 with either pseudo-inverse or regularized with $\lambda = 10^{-6}$ should be a smooth decision boundary around the Bayesian optimal solution:



Test error 4.25%. You may obtain a different decision boundary and different test error depending on the random initialization of the lifting. The displayed result correspond to the random draw according to the following order of generation:

```
np.random.seed(seed=1)
train_data = model.generate_sample(40)
test_data = model.generate_sample(50000)
net = MyNet(2, 1000)
```

The next task is to reproduce the following experiment. We fix the hidden dimension size $D = 40$ and vary the training data size as $N = 2, 4, \dots 100$ (we use stratified sampling, so it must be even). The observed test error (measured on independently drawn large enough test set) behaves as this:

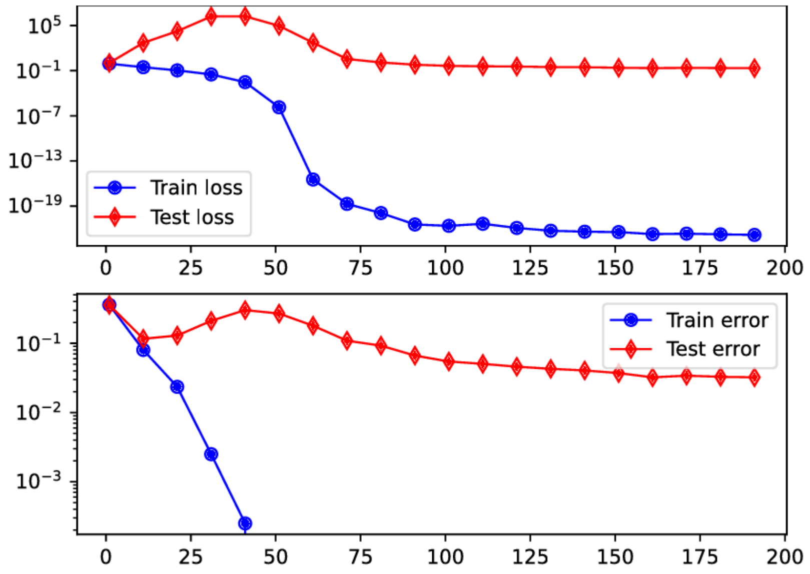


There is a pathological behaviour from 10 to 40 samples: the amount of training data increases, but it leads to worse results. Only after the threshold of 40 more data starts to help.

In this experiment, we averaged over 100 trials for each N. In each trial, the lifting of size D is constructed at random and the training data is drawn at random. The plot therefore shows an overall statistical tendency.

Task 2 (3p)

In the next experiment to reproduce, we fix the training data size $N = 40$ and vary the hidden dimension $D = 1, 10, 20, ... 200$. The observed dependence is as follows:



Again, with hidden dimensions below 40, we observe rather a classical picture of generalization: if the model capacity is too high we start to overfit and performance worthens. However, after the threshold at around 50, the test error starts decreasing again and eventually achieves even a lower value in the highly-overparameterized mode.

In this experiment, we again average over 100 trials, where in each trial the training set is sampled anew and the model of each studied hidden size is initialized anew.

Discussion

- We have investigated a rather special example and linear regression. Does it apply to deep networks? Read about Deep Double Descent [https://openai.com/blog/deep-double-descent/] on open.ai.
- Why do you think this is observed?

Submit

Submit the report to BRUTE:

- PDF (LATeX, MathPix) / Jupyter notebook **with no redundant code** and results (numbers, plots).
- all source code
- nothing extra (temporary files, datasets, *.jpeg, __MACOS)
- The report may contain some conclusions and questions to us

courses/bev033dle/labs/lab0_ddescent/start.txt · Last modified: 2024/02/27 09:15 by shekhole