Introduction to Machine Learning (for AI)

Statistical Learning Theory

Dr. Matias Valdenegro

November 28, 2024

Today's Lecture

- In this lecture we are covering the statistical basis of Machine Learning.
- In particular we will study relationships between different parameters of the learning process (number of training samples, parameters, dimensionality of features).
- And we will study definitions of generalization, overfitting, and how to improve performance of your model.

Introduction

When we train a model, the most important question is how it performs in real-world settings, outside of the dataset where it was trained.

It is very common that a model *fails to generalize*, meaning it only works on the training set, and data that is close to the training set.

A large part of the ML literature focuses into methods to improve generalization and methodologies on how to evaluate models. We will cover the most common techniques in this part of the lecture.

Important Questions

There are a set of common questions that are often asked by practitioners.

- What influences generalization performance?
- How many samples are required to train a good model?
- Which model would perform the best for my data?
- How can we reduce overfitting in a model?

These questions might seem simple, but they are very difficult to answer.

Outline

Training Parameters

We first define formally the variables/parameters that influence the training process, in general.

Number of Training Samples

This is the size of the training set, in number of samples, and is denoted by N.

Feature Dimensionality

It is the length of number of dimensions of training samples (should be constant across samples), it is denoted by D.

Training Parameters

Model Parameters

It is the number of learnable parameters in the model, basically the total number of elements of the vector/matrix W (including bias), and is usually denoted by P.

Training Parameters

- Learning a function mapping from a set of data points is not trivial due to the number of degrees of freedom (N, D, P).
- Given N samples of D-dimensional features, if $N < D$ then the problem is called "ill-posed" or "ill-conditioned", as not all possible feature values are defined.
- Usually you need $N >> D$ in order to learn a concept properly, but it can still be done if additional information is provided.

Training Parameters

- Also if a model has P learnable parameters, then if $P > N$ not all model parameters have unique values. This is called an underspecified problem.
- Even if $N < P$ or $N < D$ a concept can still be learned, if additional information to the learning process is provided in order to obtain unique solutions.
- But while unique solutions are desirable in theory, in practice they are not necessarily "better". Deep Neural Networks have multiple solutions and theory has proven that these are quite similar to each other.

Training Parameters

- The relationship between all these variables is not completely understood, specially the relation between P and N, D, as the amount of information in each training sample might offset having a small sample size (for example, large images).
- If P is much larger than N, then this typically means that the network is overparametrized. This means that it has more parameters than necessary.
- But experimental results[1] show that the number of parameters is not a good indicator if overfitting will happen. Generalization can still happen if there is valuable information in the labels.

[1] Paper: Understanding Deep Learning Requires Rethinking Generalization

Training Parameters

In general for learning to succeed, the following conditions should be met.

$$N > P \quad \text{More training samples than parameters (1)}$$
$$N > D \quad \text{More training samples than dimensions (2)}$$

But these are very general rules. You usually need much more than this to generalize well.

Rules of Thumb

Number of Parameters

A good number of tranining samples is 10 times the number of parameters.

$$N \sim 10P \tag{3}$$

Feature Dimensions

Some people use this rule to compare D with N. At least have three times the number of dimensions as training samples.

$$N \sim 3D \tag{4}$$

Note that these are just approximated rules, with little theoretical backing.

Imbalanced Class Data

An important issue to consider in classification problems is the number of samples for each class.

If one class dominates with respect to others (like 90% to 50% of all samples), then this class *could* bias the classifier.

For example, we have a binary classification problem, and the positive class is 95% of all training data points, then the model will learn to predict always the positive class, and will receive a 95% accuracy score.

This effect is more pronounced with less number of classes. Also note that imbalance can happen in the training, validation, or test sets.

Imbalanced Class Data

Class Weights

One simple way to address the class imbalance problem is to use class weights. These weights are added the the loss function, to weight the terms associated to each class, in order to reverse the imbalance produced by the training data.

For the categorical cross-entropy loss:

$$L(y, \hat{y}) = -\sum_i \sum_c w_c \, y_i^c \, \log(\hat{y}_i^c) \tag{5}$$

Here the weights w_c are set for each class, and control how much each class influences the overall loss.

Imbalanced Class Data

Class Weights

Setting the class weights is not difficult. It can be done with the multiplicative inverse of the per-class frequency:

$$w_c = \frac{N}{C_c} \tag{6}$$

Where C_c is the number of samples for class c. If the dataset is very unbalanced, the weights can also be smoothed using the logarithm function:

$$w_c = \log\left(\frac{N\mu}{C_c}\right) \tag{7}$$

Where μ is a factor that controls smoothing, $\mu = 0.15$ is a common value.

Imbalanced Class Data

Sample Weights

For regression and non-classification problems, sometimes it is preferable to set weight associated to samples instead of classes, and they work in the same way as class weights (they weight the loss).

Weights are set in the same way, as the inverse of the frequency f_i of a data point.

$$w_i = \frac{1}{f_i} \tag{8}$$

The frequency can be estimated from using a kernel density estimate for the distribution of the values of y.

Imbalanced Class Data

Oversampling

Another way to deal with imbalanced datasets is to transform them into a balanced dataset, by oversampling the minority class(es). There are many methods to do this:

SMOTE This method creates new synthetic samples, by selecting a sample from the minority class, take the k nearest neighbors in feature space, and linearly interpolate with a random factor (in $[0, 1]$)) between the data point and one of the neighbors. Repeat until the dataset is close to be balanced.

Imbalanced Class Data

Oversampling

Data Augmentation Augmentation can also be applied to the minority class, increasing the number of samples in order for the dataset to be better balanced.

These methods can only be applied to the training set (due to leakage).

How many training samples?

For number of training samples, we only have basic rules that determine some hard minimums, but they do not tell us how many samples per class or total number of training samples we need to achieve a certain performance target.

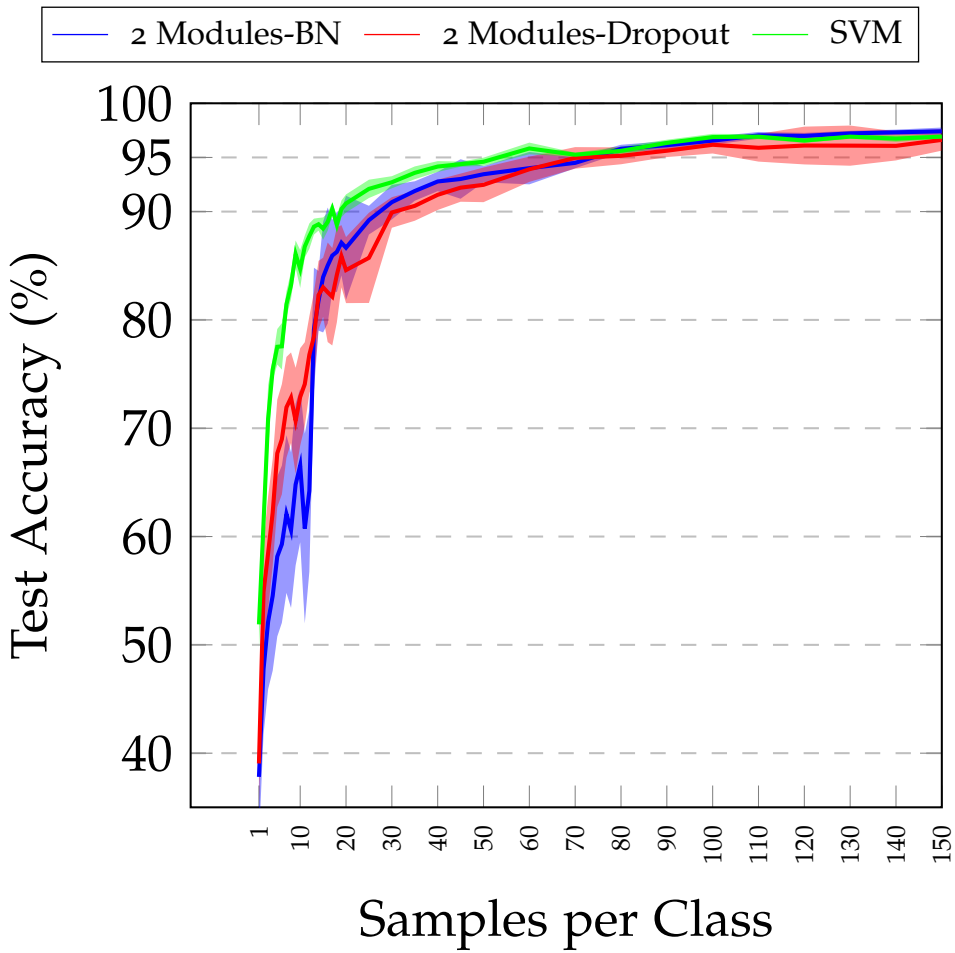
A general and very important rule is:

> *More training data increases generalization and overall model performance.*

So we should always train our models with the largest datasets that we have. We will now see some empirical results that show this effect.

How many training samples?

Classification Example

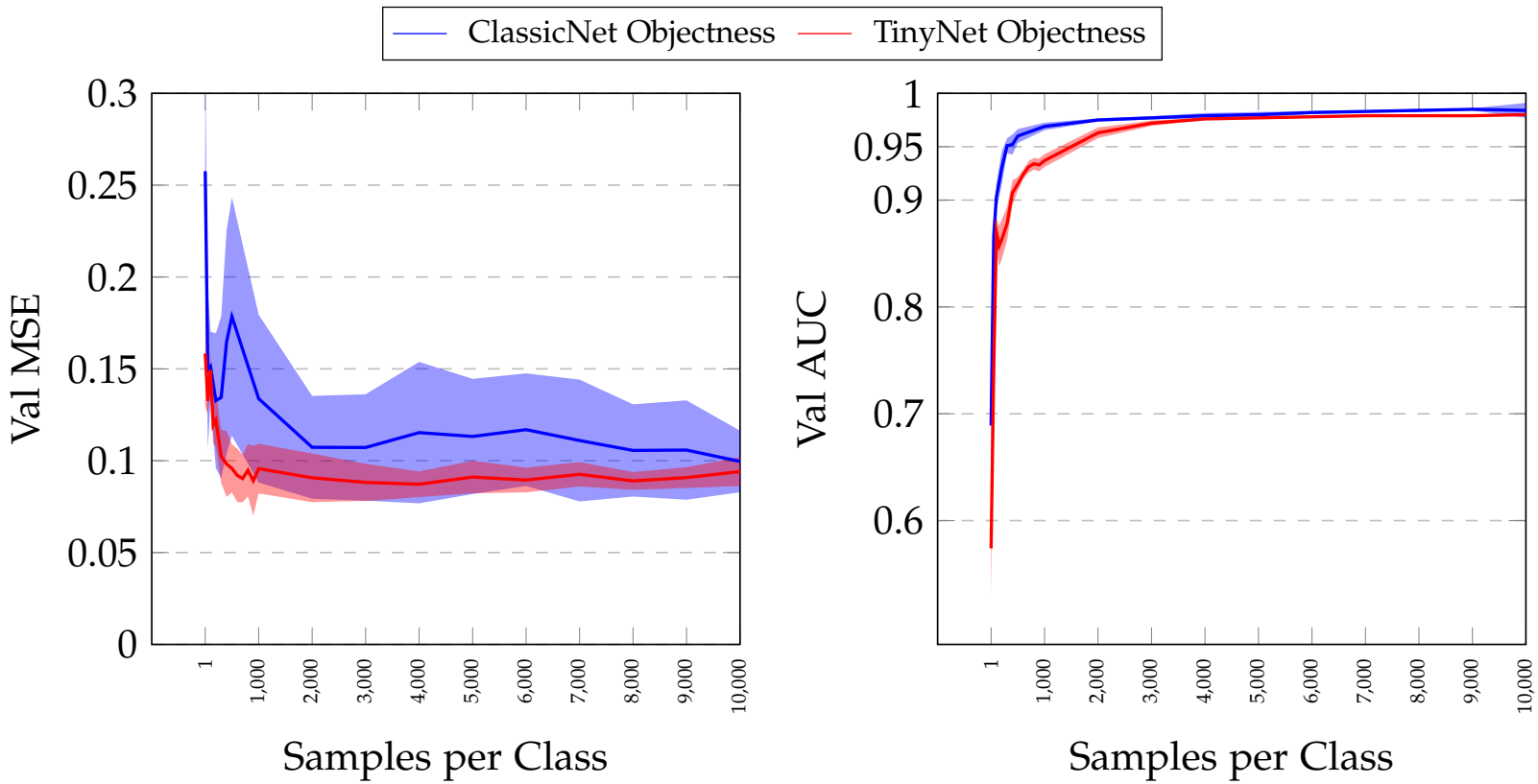


Example of sub-sampling the training set while the test set is fixed, showing the effect of training a model with more data.

More data $\rightarrow$ better model.

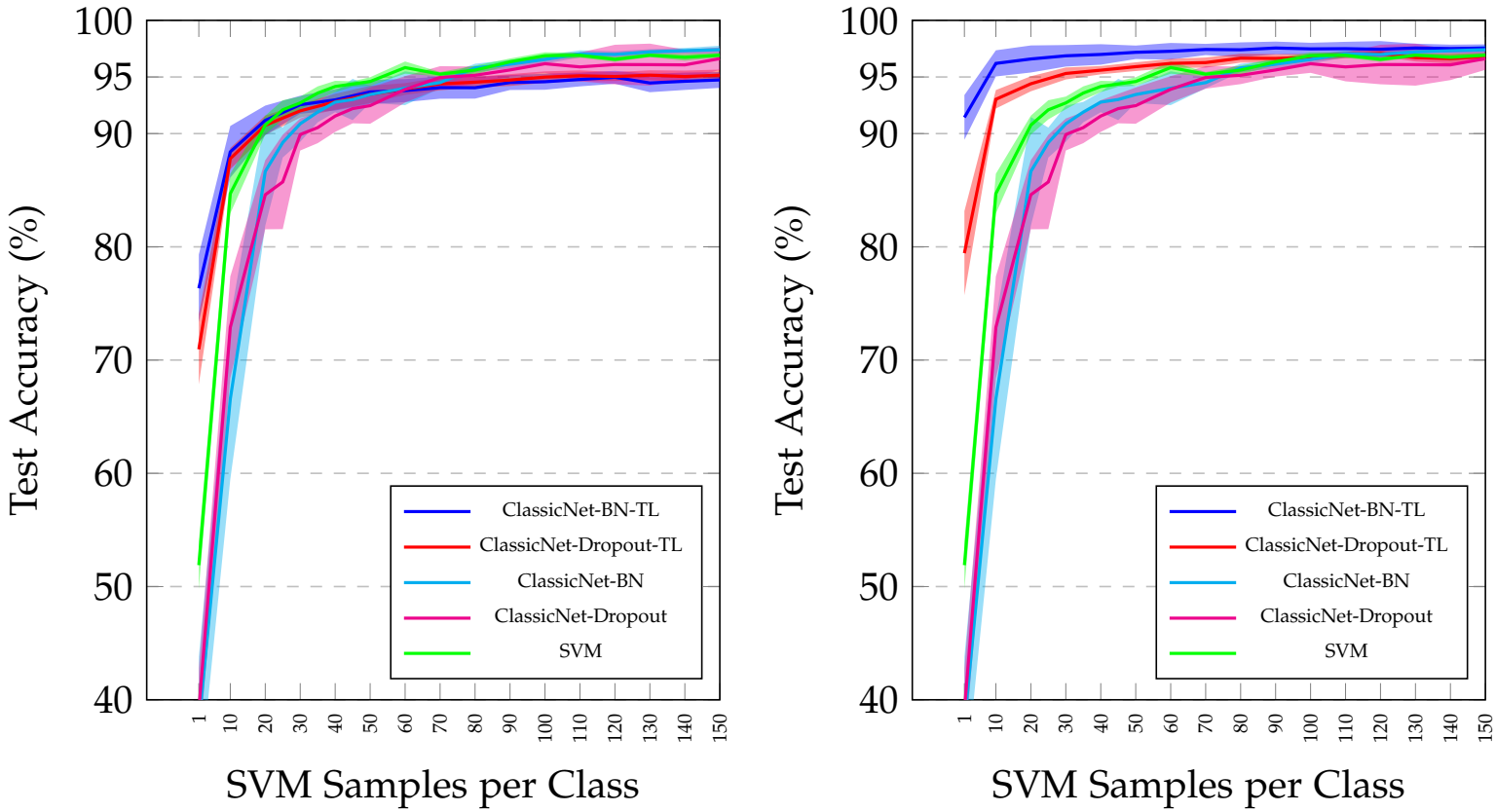
How many training samples?

Regression Example



How many training samples?

Transfer Learning



Example of transfer learning with different/same objects, which is an example of additional information into the learning process.

How many training samples?

Some factors that affect sample complexity:

- Inter and intra class variability.
- Dimensionality of the features (D).
- Difficulty of the task.
- Quality of the data and its ability to represent the task. Duplicated and very similar samples will not contribute much to learning.
- Noise and/or missing values on the features.
- The model and its ability to learn from the data, different models might require less samples.

How many training samples?

Inter-class Variability

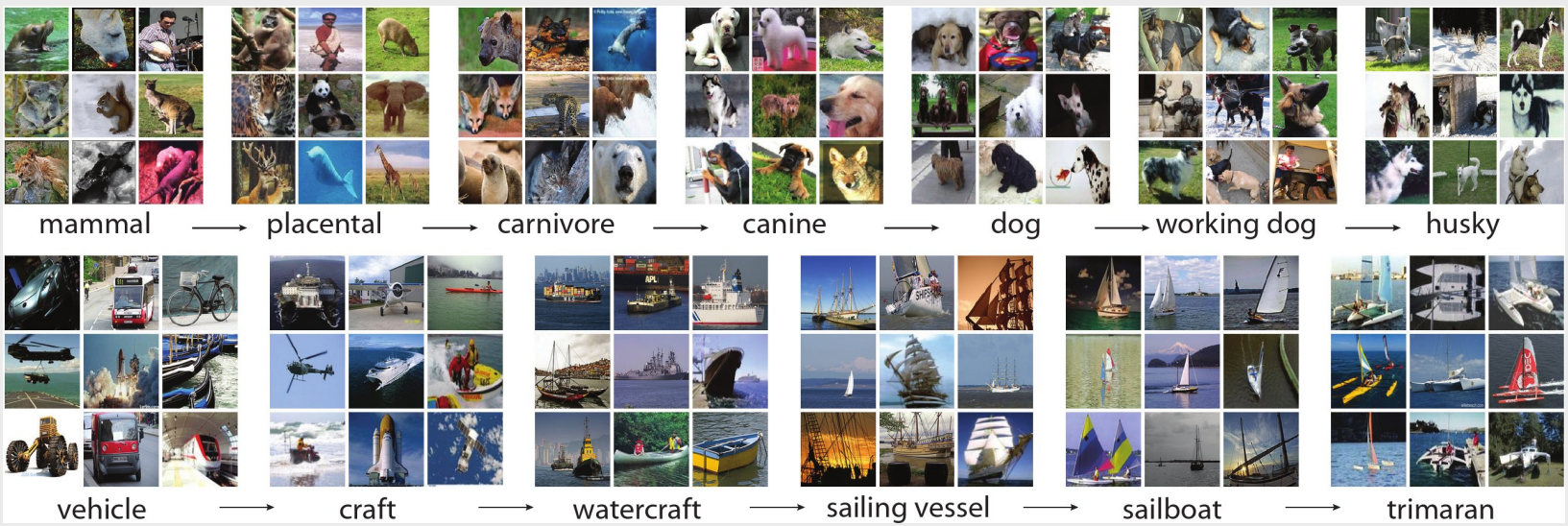
Variation between different classes, for example, Dog, Cat, Owl, Bunny, etc.

Intra-class Variability

Variation between instances of the same class, for example for Dogs: Terrier, Beagle, Chihuahua, etc. Owls: Snow Owl, European Owl, etc.

How many training samples?

Inter-class Variability



Example classes and samples for each class from the ImageNet dataset.

Similar classes are difficult to classify, having many visually different classes requires less training samples, while similar class require more training samples.

How many training samples?

Intra-class Variability

Low intra-class variability allows to use less samples, while a high intra-class variability will need more samples to represent the complexity in the data.

Effect of Feature Dimensionality

Cover's Theorem

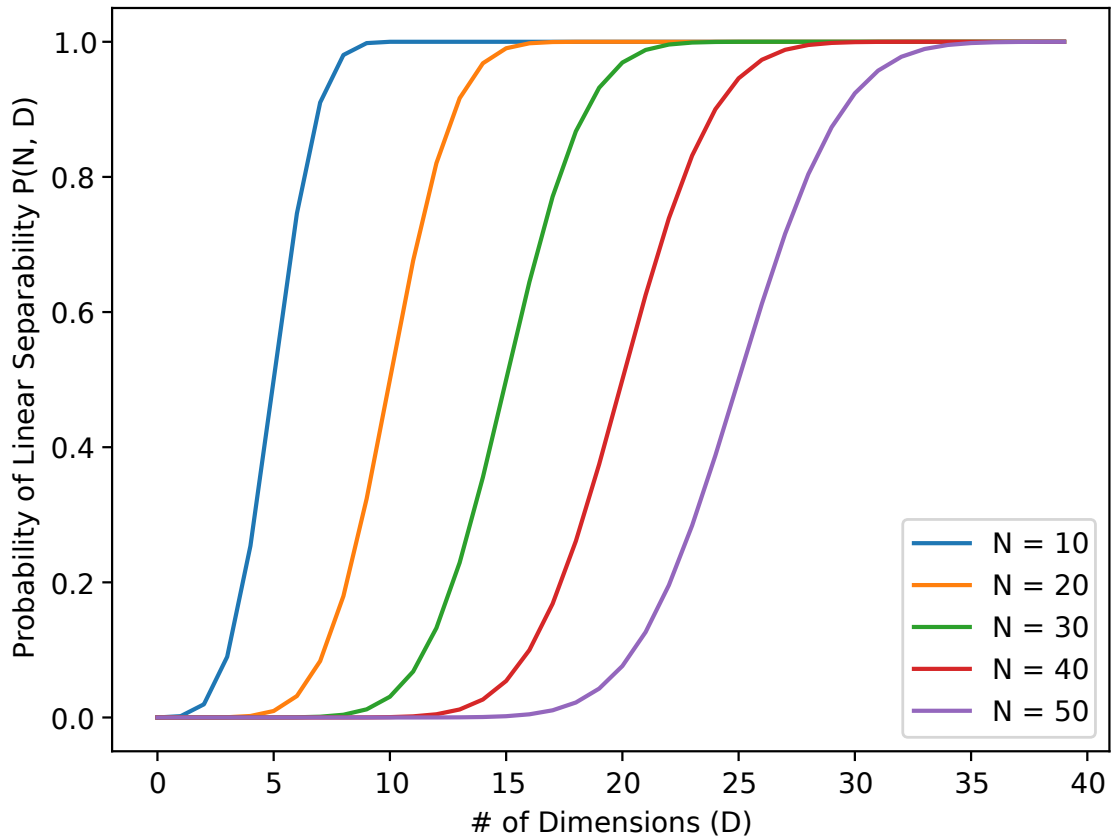
An important theoretical result that can tell us about the relationship between D and N is Cover's theorem (from 1964).

$$P(N, D) = 0.5^{N-1} \sum_{m=0}^{D} \binom{N-1}{m} \tag{9}$$

This computes the probability that N data points of D dimensions are linearly separable if randomly labeled with binary labels.

Effect of Feature Dimensionality

Cover's Theorem



Effect of Feature Dimensionality

Cover's theorem points that increasing the feature dimensionality (D) improves the probability that the data points are linearly separable (for any assignment of binary labels).

This is the reason why Kernel methods are used, as projecting features to a highly dimensional space, makes the more likely to be linearly separable.

It also motivates the use of as many features as possible, but less than the number of data points that are available.

Outline

Overfitting

- Overfitting is when the trained model memorizes undesirable patterns from training data.
- Like models learning noise in the data or irrelevant features.
- It reduces generalization performance. Models perform badly on the test set or on different data.
- Happens when model has too much learning capacity or it is trained for too long.

Cross Validation

Cross validation is the concept of evaluating a model on an independent dataset in order to evaluate its performance and assess the level of generalization. This is usually done by using two datasets:

Training Set

The dataset where the model is trained and the training optimization process runs. Metrics computed in this dataset are called training metrics.

Test Set

An *independent* evaluation dataset, ideally containing samples that are not present in the training set. Metrics computed in this dataset are called testing metrics.

Generalization

General Concept

Generalization in Machine Learning means that the trained model is widely applicable and produces correct predictions for inputs that are far from the training set.

This is generally measured by the gap between training and validation/testing loss.

But usually a model can also fail to generalize if the inputs are very different from the training set, and this is a very difficult problem.

For a given model, there is always inputs that will make non-sensical or highly incorrect predictions.

Overfitting

General Concept

Overfitting is when a model fails to generalize (make correct predictions).

To detect it, look at the gap between train and validation/test losses.

$$L_{\text{gap}} = L_{\text{val}} - L_{\text{train}} \tag{10}$$

Overfitting is a continuous concept, not binary (yes or no overfitting). There are multiple "degrees" of overfitting.

An important concept is that all models overfit to some degree, but some are useful. Any model can also overfit, there is no model that completely prevents overfitting.

Overfitting

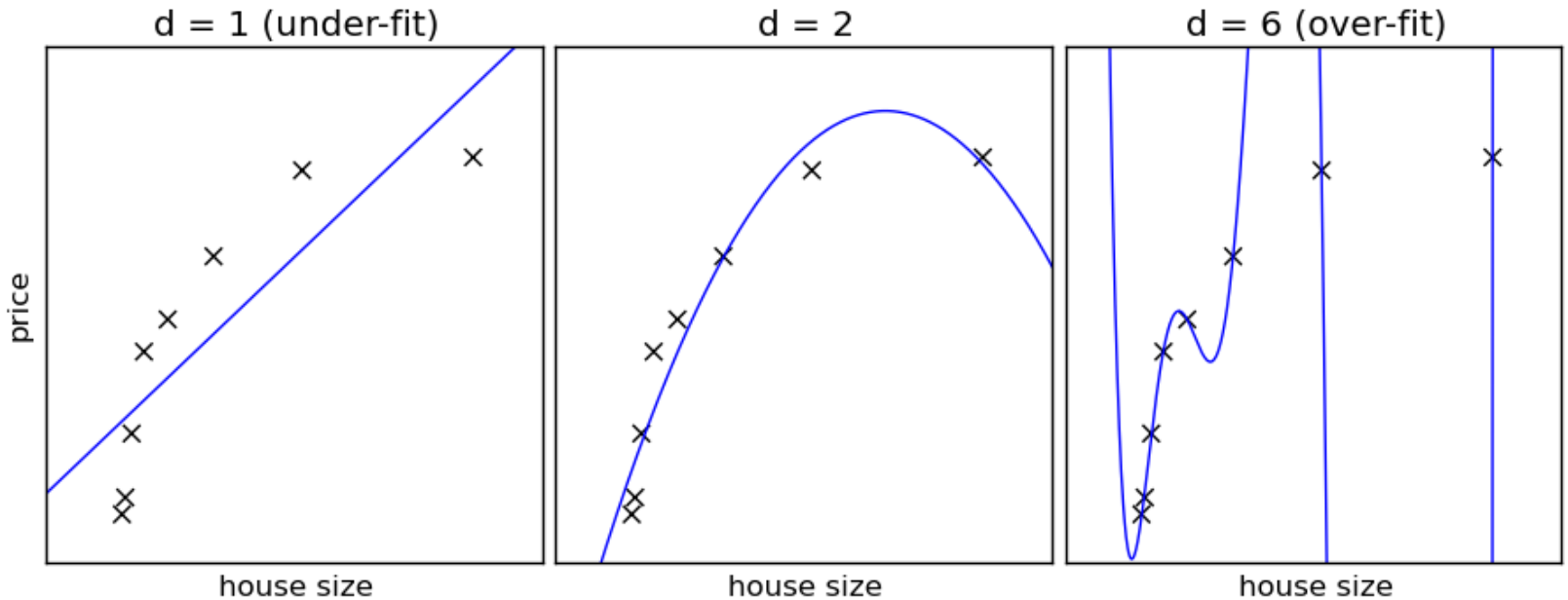
Causes

1. Little training data ($N << D$ and/or $N << P$).
2. Model misspecification (incorrect model).
3. Model learning from the incorrect features (that do not generalize).
4. Model too large for the data available.
5. No guarantees over validation/test set performance.

Solution

Regularization, will be covered in next week's lectures. But does not always work, and fighting overfitting is a complex problem in Machine Learning. No guarantees.

Overfitting



Overfitting

Golden Rule

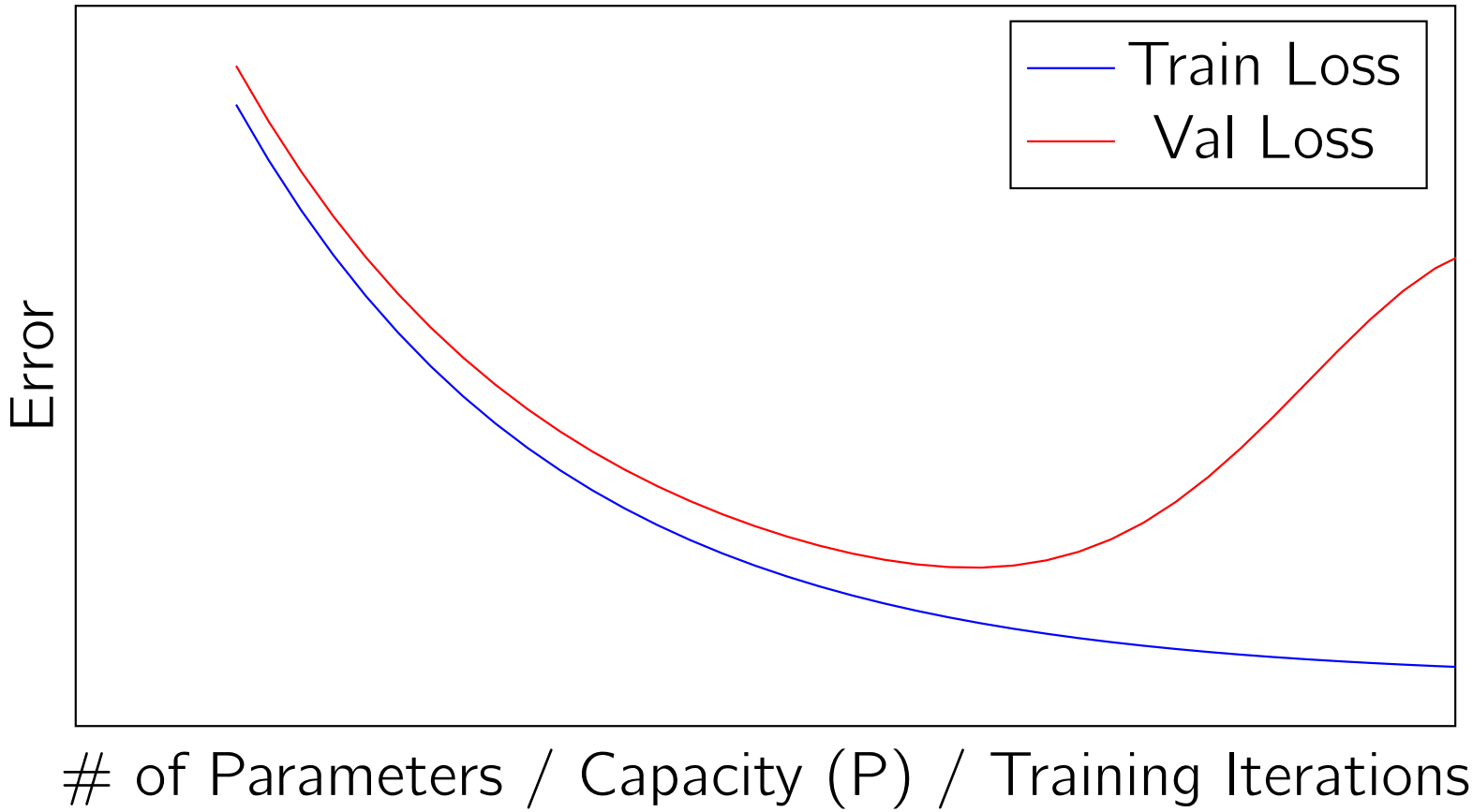
When training a model, compare the loss value on the training set and on the testing set.

If the validation/test loss is **much larger** than in the training set, then the model is overfitting, specially if the training loss is low.

It is normal that the test loss is **slightly** larger than training loss.

The gap size L_{gap} is a matter of discussion/opinion and needs some human judgment to decide if the model overfits or not.

Overfitting



As your model contains more parameters or it is trained for more iterations, then the train and validation losses diverge, producing the gap L_{gap}, indicating overfitting.

Practical View Overfitting

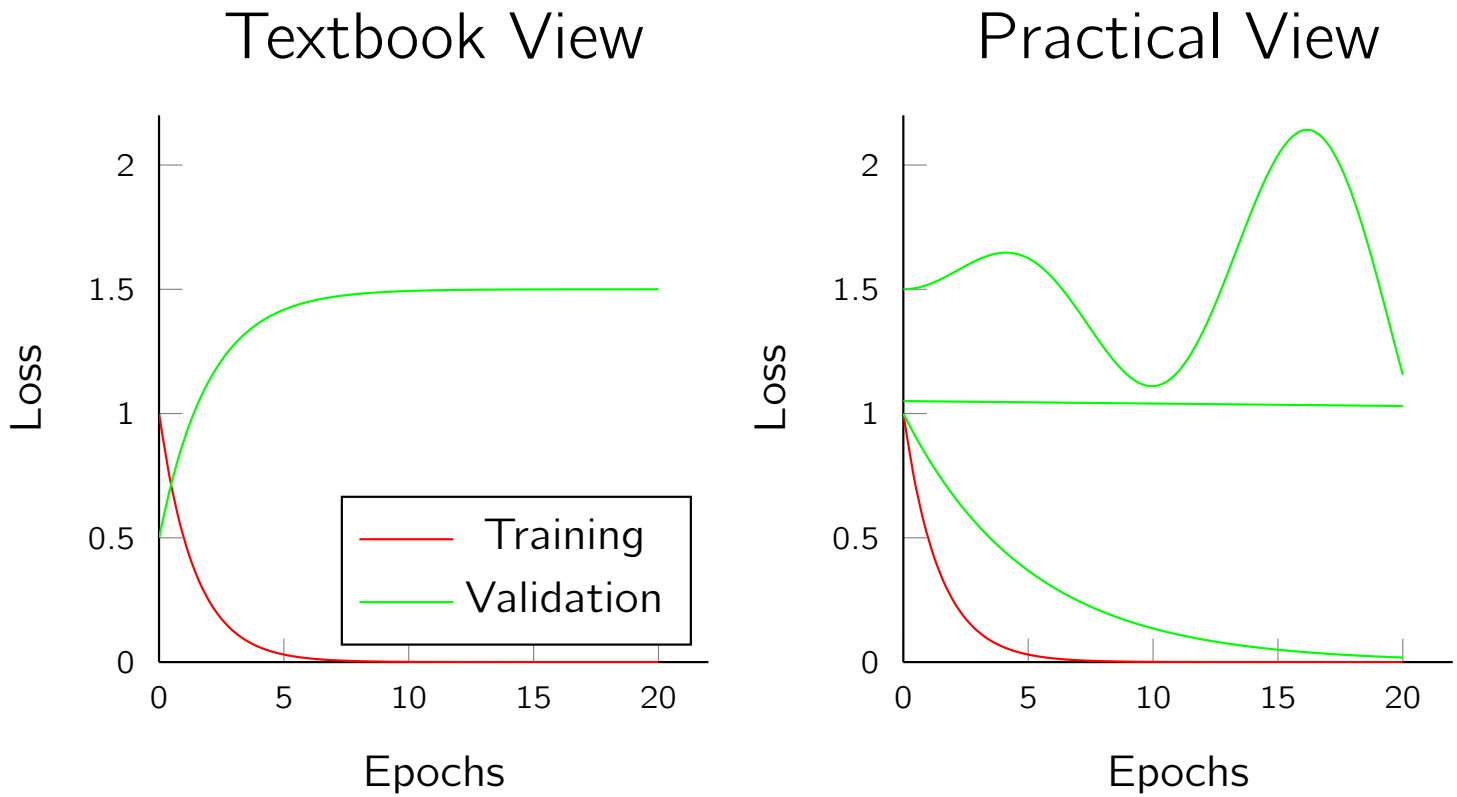


Figure: Example view of overfitting in textbooks (top) vs how students see overfitting in practice (bottom)

Underfitting

Another phenomena related to overfitting, is underfitting.

This is when the training loss is high, and does not decrease during training. It can also happen that the training loss decreases a little but not enough for convergence (close to zero).

It basically means that the model does not fit the data. Could be model misspecification or incorrect model.

What is **NOT** Overfitting

1. Training loss of zero.
2. Looking at training loss but no validation/test loss.
3. Looking at validation/test loss without a training loss.
4. Validation loss lower than the training loss.
5. Oscillating validation loss.
6. Using metrics (accuracy, non-loss values) to decide overfitting. It is only defined for losses.

For more information, you can read our short paper: *Machine Learning Students Overfit about Overfitting*, Valdenegro and Sabatelli, Teach ML Workshop @ ECML 2022. `https://arxiv.org/abs/2209.03032`

How to Prevent Overfitting?

- Reduce model learning capacity or use a different model.
- Use regularization or methods that combat overfitting.
- Train with more data, including capturing more real data, using data augmentation, or synthetic data.
- Use early stopping. Stop training if **validation** loss is not improving.
- Auxiliary tasks and Multi-Task learning can also improve generalization. We will see this later in the course.

No Free Lunch Theorem

Another important theoretical result is the NFL theorem, which basically indicates:

In the average, no learning algorithm is superior to another.

Meaning that if you average performance of all learning algorithms over all random training and testing sets, then average performance is pretty much the same.

Bias-Variance Trade-off

The mean squared error can always be decomposed as follows, for true model $f(x)$ and estimated model $\hat{f}(x)$:

$$\begin{aligned}\mathbb{E}[(y - \hat{f}(x))^2] &= \overbrace{\mathbb{E}[\hat{f}(x) - f(x)]^2}^{\text{Bias}} + \overbrace{\mathbb{E}[\hat{f}(x)^2] - \mathbb{E}[\hat{f}(x)]^2}^{\text{Variance}} + \sigma^2 \\ &= \text{Bias}[\hat{f}(x)]^2 + \text{Var}[\hat{f}(x)] + \sigma^2\end{aligned}$$

Where $\sigma^2 = \text{Var}(y)$ is an irreducible error, usually noise in the data or labels.

The bias comes from assumptions in the model, like fitting non-linear data with a linear model (**underfitting**). Variance comes from the model fitting noise in the training set (**overfitting**).

Bias-Variance Trade-off

Bias

An abstract measure of how well the model fits the data, comparing it to a true model $f(x)$ (which generally we do not know).

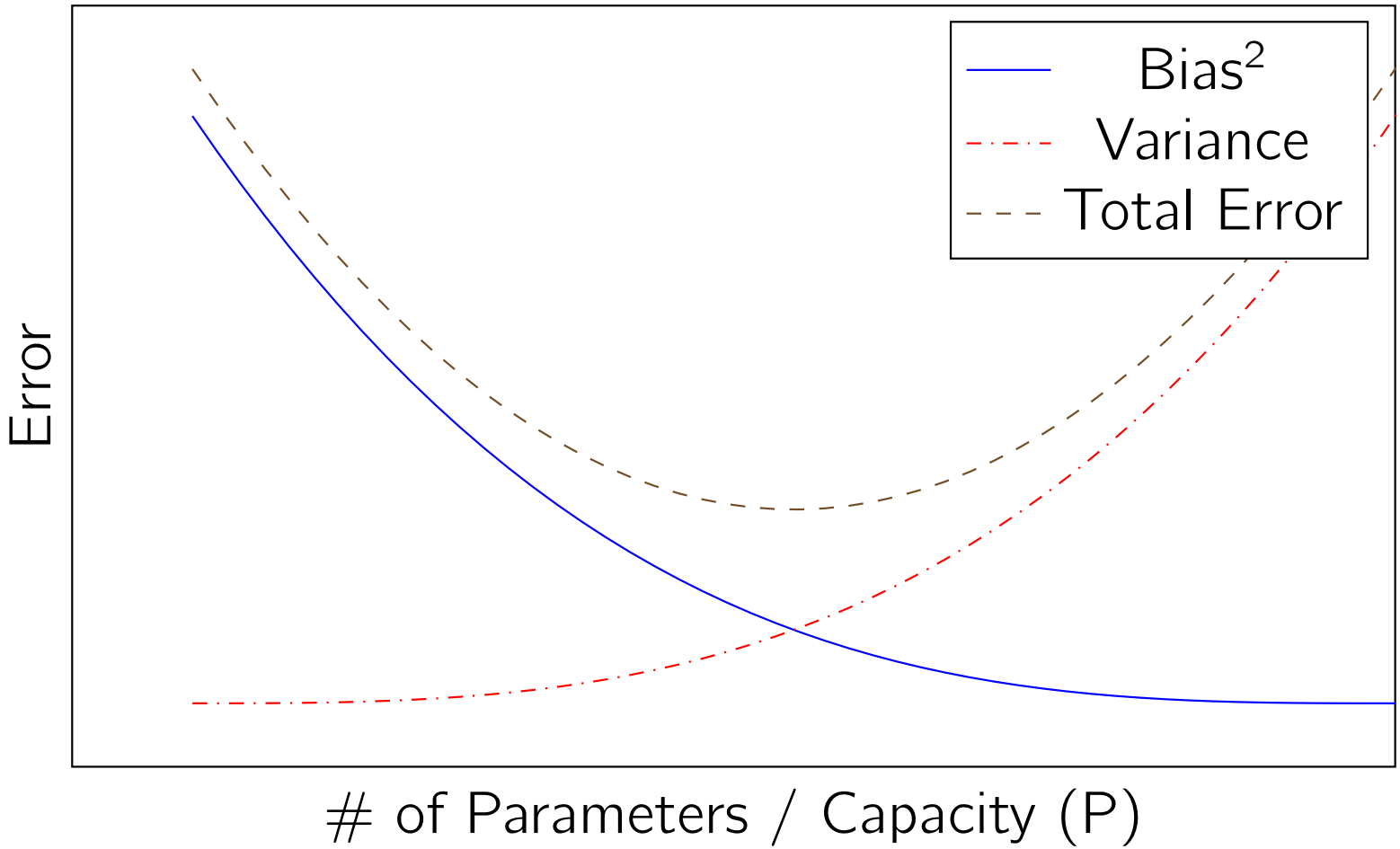
Usually related to underfitting.

Variance

Variance of the estimated model $\hat{f}(x)$ with respect to data, basically how well the model fits small variations of the training data.

Usually related to overfitting.

Bias-Variance Trade-off



Bias-Variance Trade-off

The trade-off is as follows:

- A model with small variance usually has a large bias (its underfitting).
- A model with small bias usually has a large variance due to overfitting.
- Too many parameters/capacity leads to overfitting, and not enough parameters/capacity leads to underfitting. It is not really trivial to determine when under/overfitting happens.
- The number of parameters/capacity needs to be carefully adjusted in order to "match" the tasks' complexity and number of training samples.
- This should always be done on a validation set!

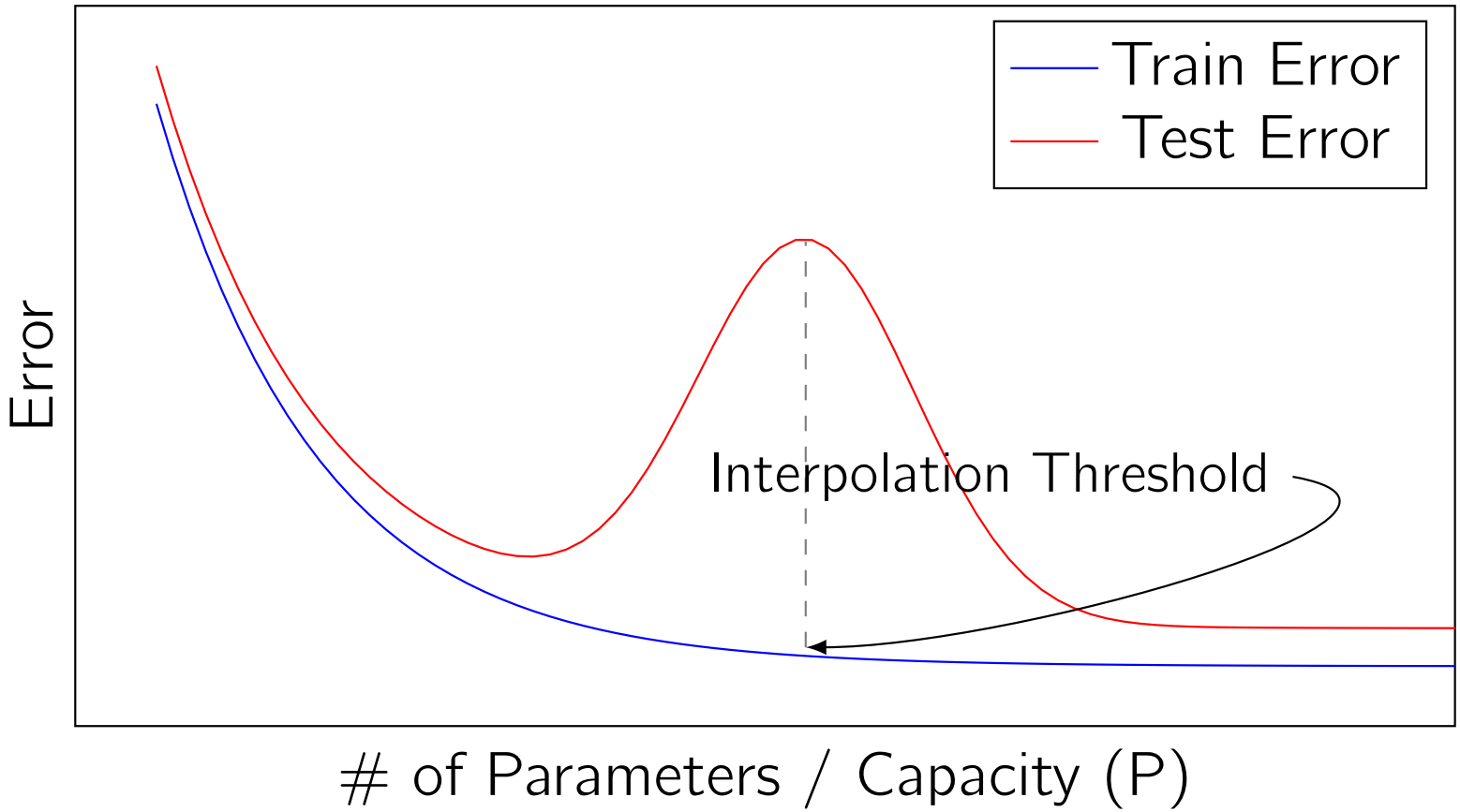
Double Descent or Grokking

A recently known phenomena is Double Descent or also known as Grokking.

If your model has too many parameters, it is expected to overfit (low train error, high test error). Then train and test error/losses will start to diverge at some point.

But **in some cases**, after the "peak of divergence", there are some models that do continue to generalize after this peak, which is called the interpolation threshold. The overall concept is called double descent.

Double Descent



This figure shows the typical bias-variance trade-off with respect to overfitting, but in some cases, a larger model can still generalize better and have decreasing test error.

Outline

Shattering

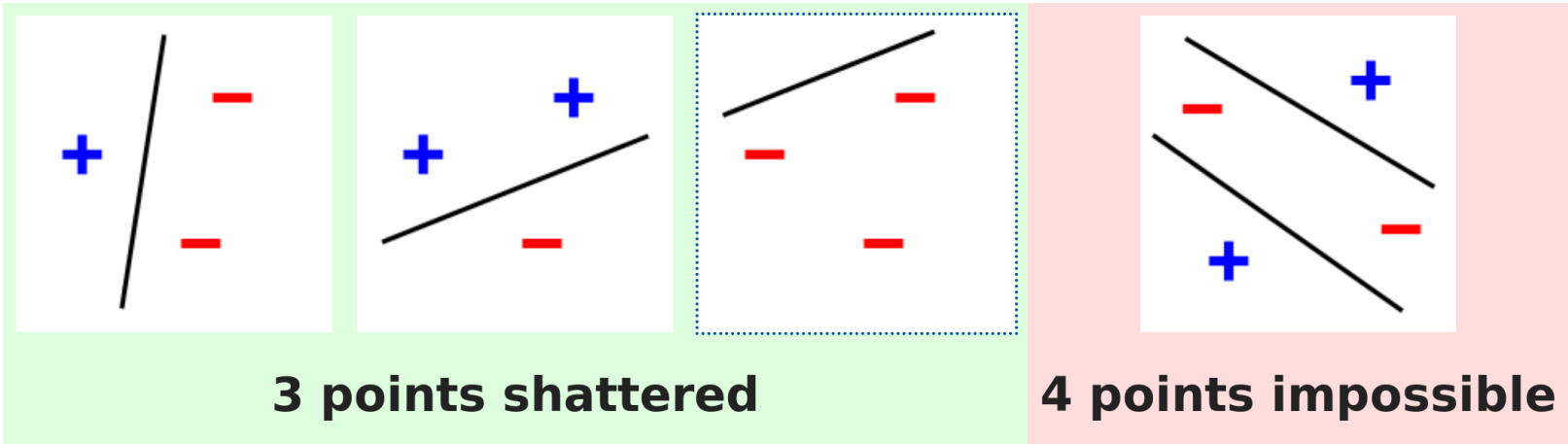
Shattering

Given a binary classifier f with parameters θ and a set of points $(x_1, x_2, ..., x_n)$. The classifier is said to *shatter* these points if for any possible (binary) labeling of these points, there is a set of parameters θ that obtains zero error.

In simple words, shattering means a model can perfectly classify a dataset, for all possible binary labeling of the dataset.

Shattering Example

This example is in 2D with a linear classifier, so its VC dimension is 3.



VC Dimension

VC-Dimension

The Vapnik-Chervonenkis dimension D of a binary classifier f is the maximum number of data points that can be shattered, irrespective of the actual label values of the points.

Every model has a limited ability to classify datasets perfectly (shattering), and the VC dimension aims to measure that capacity, in an imperfect way.

VC Dimension

Given a D-dimensional feature space:

- Simple Threshold ($\text{sgn}(X + T)$): 1
- Linear Classifier: $D + 1$.
- Multilayer Perceptron with w parameters: $O(w^2)$
- Directed Acyclic Neural Network with $s \geq 2$ internal nodes each of VC Dimension V: $2V \cdot s \log_2 e \cdot s$
- Random Forest: Believed to be ∞.
- Sine Classifier ($\sin \theta x > 0$): ∞

VC Dimension

The VC dimension basically measures how many functions can be approximated perfectly by a machine learning model.

Different models can approximate different number of functions, which defines their power and/or capabilities.

This is most relevant for non-linear models, and it is only a rough estimate of model capabilities. But the best use of VC dimension is about bounding the difference between train and test error.

VC Dimension - Theoretical Bounds

Given train and testing errors, assuming that training points (x_i, y_i) are independent and identically distributed (iid), and model parameters θ:

$$\text{TRAINERR}(\theta) = 0.5N^{-1} \sum_i |y_i - f(x_i, \theta)|$$

$$\text{TESTERR}(\theta) = 0.5\mathbb{E}[|y - f(x, \theta)|]$$

These are just the mean absolute error applied to a classification/regression output, it is a special case.

VC Dimension - Theoretical Bounds

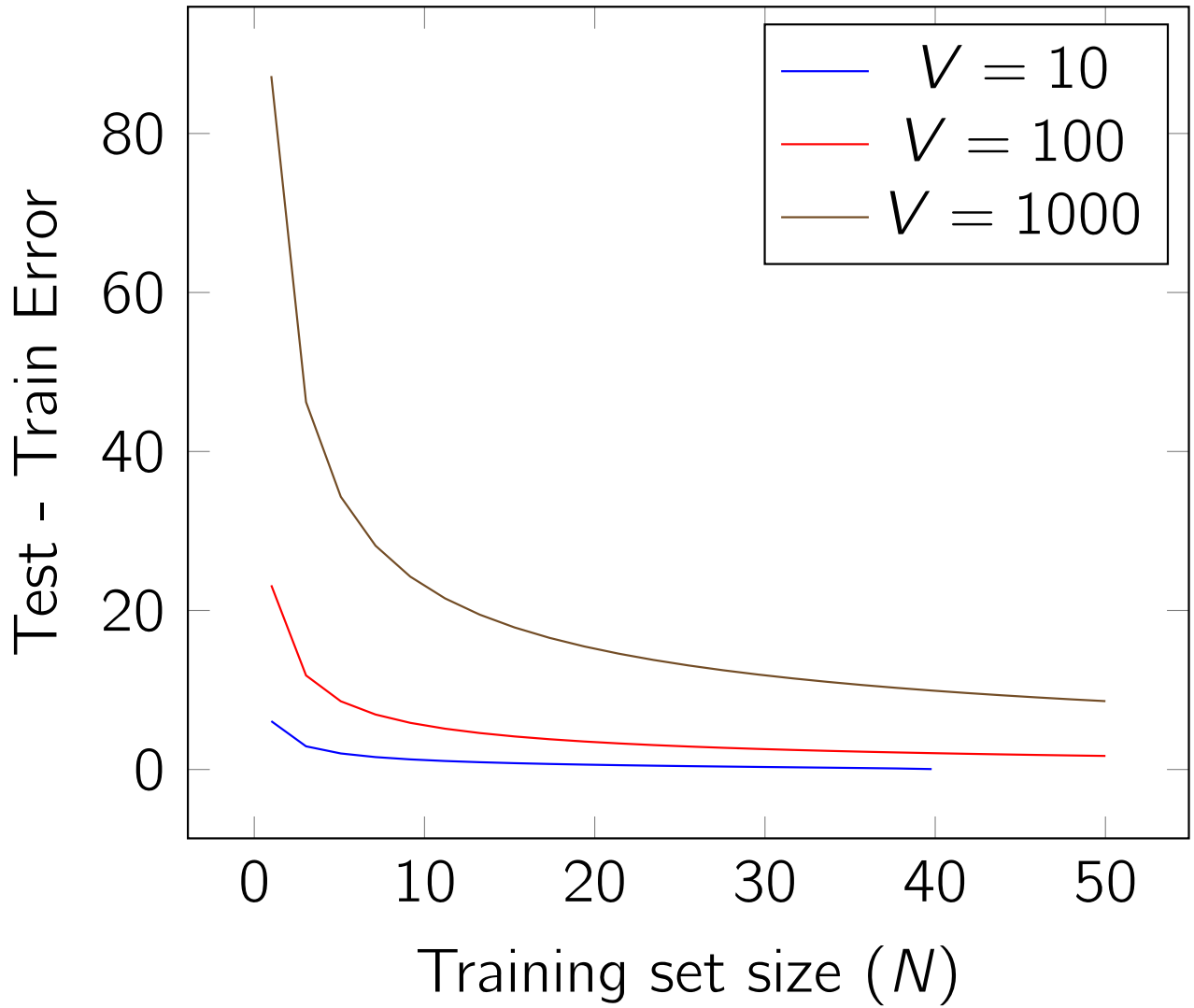
Then the following bound applies with probability $1 - \eta$:

$$\text{TESTERR}(\theta) \leq \text{TRAINERR}(\theta) + \sqrt{\frac{V(\log \frac{2N}{V} + 1) - \log \frac{\eta}{4}}{N}} \tag{11}$$

This bound loosely tells you how much data (N) and model complexity via the VC dimension (V) is needed to obtain a certain test error, as function of the training error.

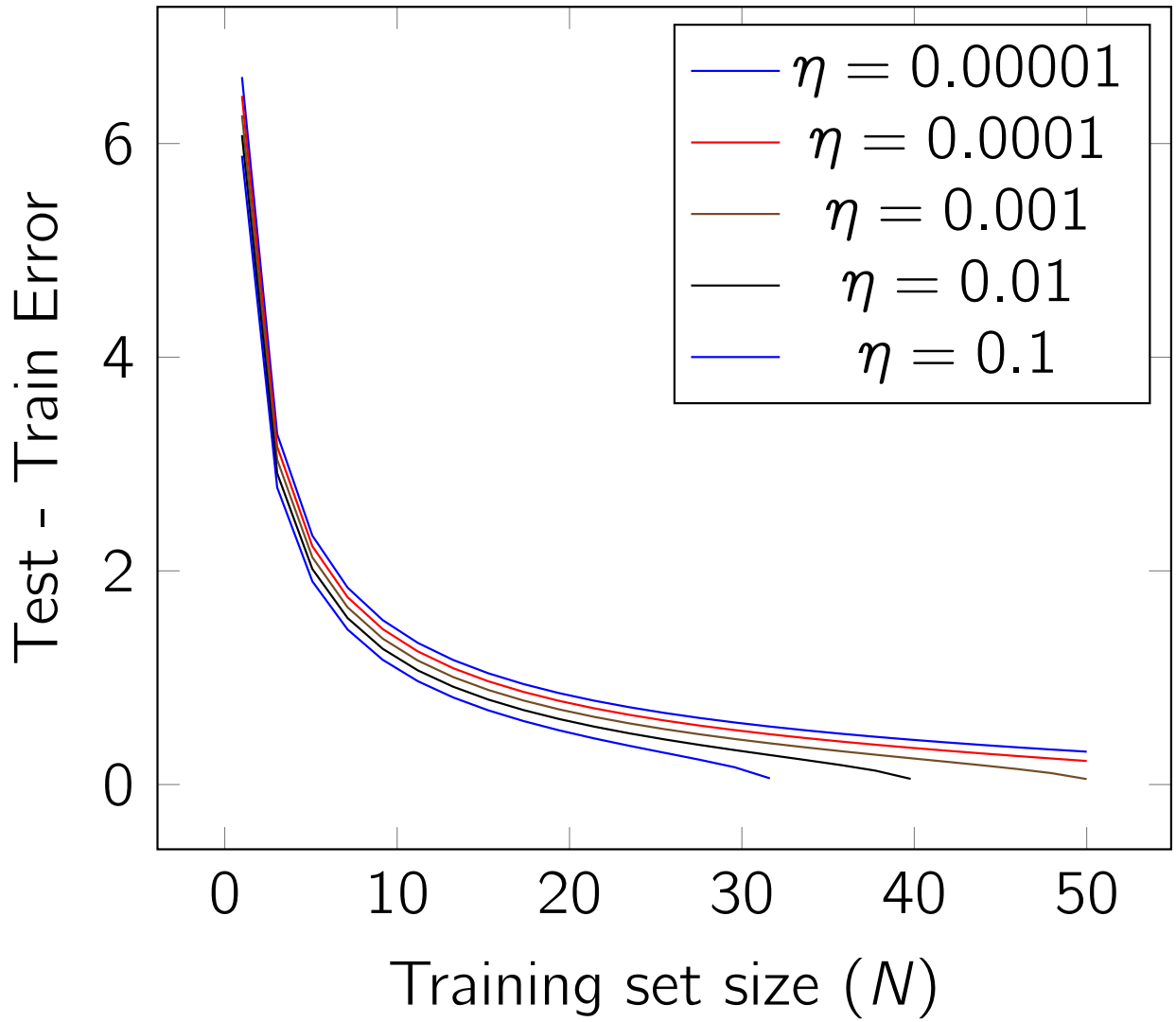
This implies that test error is usually larger than training error. Note that the VC Dimension and this bound do not consider the training data itself in its computation.

VC Dim - Theoretical Bound Visualization for N



Plot of $\sqrt{\frac{V(\log \frac{2N}{V}+1)-\log \frac{\eta}{4}}{N}}$ as N is varied with $\eta = 0.01$

VC Dim - Theoretical Bound Visualization for η



Plot of $\sqrt{\frac{V(\log \frac{2N}{V}+1)-\log \frac{\eta}{4}}{N}}$ as N is varied with $V = 10$

What is the point?

The VC dimension is one of many theoretical analysis about generalization of machine learning models. In simple words, try to predict performance based on some variables of the learning process (N, V, etc).

These theories are incomplete and not so useful in practice, but they give basic intuitions: **more data improves a model's generalization ability**, *different models have varying learning capacities*, and predicting difference between test and train error is very difficult.

Overall the lesson is that you always design a model specifically for a task and data availability, else your model overfits.

What you must learn

- The relationships between N, D, and P, and their relationship to overfitting.
- Cover's theorem and why we prefer to use large dimensional features.
- The concept of overfitting and underfitting **in detail**, how to detect it, its causes, and possible solutions. **This is by far the most important concept in practical machine learning, it will haunt you forever**...
- The bias-variance trade-off, its relationship with overfitting/underfitting, and the double descent phenomena.

Questions to Think About

1. What is overfitting in your own words?
2. How is overfitting different from underfitting?
3. What variables influence the learning problem success? And how?
4. What is the VC Dimension?
5. Your model overfits, suggest some steps to improve?
6. Conceptual question: We would show you some train/val loss curves and ask what over/underfitting is happening.

Take Home Messages

- There are complex relationships that control the learning process.
- How many samples are needed? It is hard to give a exact number, only guidelines.
- Overfitting, Underfitting, and Bias-variance trade-off are important concepts to understand why your ML model works/not works. You should dominate these concepts.
- All these concepts apply to Neural Networks and Deep Learning, even more than classical models.

Book Chapter Readings

If you want to dive **deeper** in this topic, we recommend:

- Bishop Book: Chapter 1.1 for overfitting, Chapter 3.2 for bias-variance.
- Duda Book: Chapter 7 for overfitting.
- UDL Book: Chapter 8 for overfitting and bias-variance.
- Our paper: *Machine Learning Students Overfit about Overfitting*, Valdenegro and Sabatelli, Teach ML Workshop @ ECML 2022. `https://arxiv.org/abs/2209.03032`

UDL Book is freely available at `https://udlbook.github.io/udlbook/`.

Questions?