Machine learning fundamentals: Lecture 1

Introduction
Feasibility of learning: concepts
Bias - Variance tradeoff

Mirko Mazzoleni - Identificazione dei Modelli e Analisi dei Dati (IMAD)

11 October 2017

University of Bergamo
Department of Management, Information and Production Engineering
mirko.mazzoleni@unibg.it

Resources

These lectures give only a glimpse of the vast machine learning field. Additional material (not required for the exam) can be found using the following *free* resources

MOOCs

- Learning from data (*Yaser S. Abu-Mostafa - EDX*)
- Machine learning (*Andrew Ng - Coursera*)
- The analytics edge *(Dimitris Bertsimas - EDX)*
- Statistical learning *(Trevor Hastie and Robert Tibshirani - Standford Lagunita)*

Books

- An Introduction to Statistical Learning, with application in R *(Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani)*
- Neural Networks and Deep Learning *(Michael Nielsen)*

Outline

- Introduction
- Components of learning
- Puzzle
- Feasibility of learning
- Bias - variance tradeoff

Outline

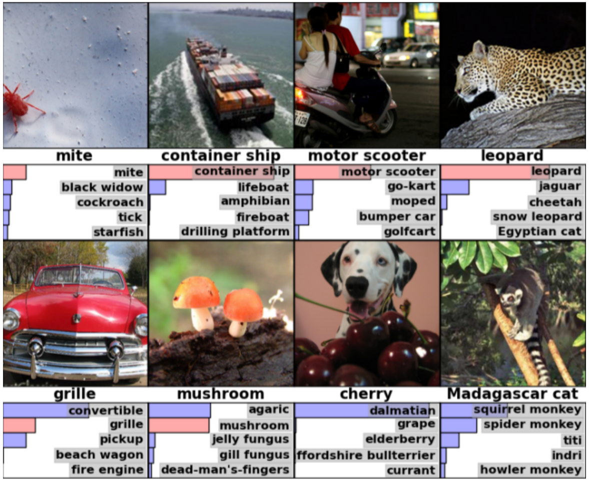
Why

Machine learning and data science have been deemed as the sexiest jobs of the 21th century

- Virtually every aspect of business is now open to data collection
- Collected information need to be analyzed properly in order to get actionable results
- A huge amount of data requires specific infrastructures to be handled
- A huge amount of data requires computational power to be analyzed
- We can let computers to perform decisions given previous examples
- Rising of specific job titles
- . . . Fun ☺

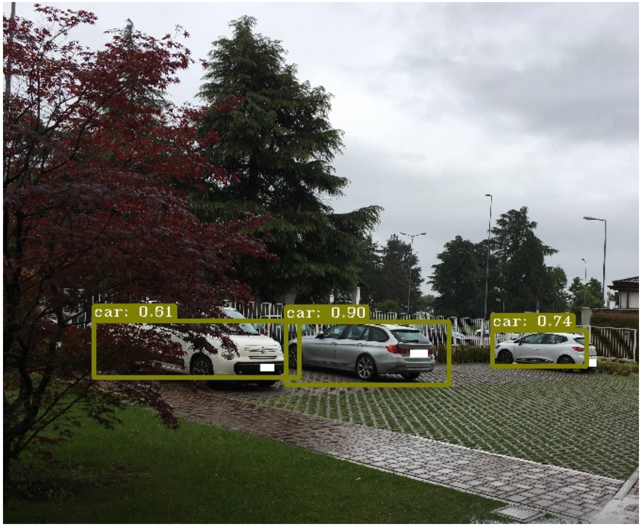
Learning examples

Recent years: stunning breakthroughs in computer vision applications

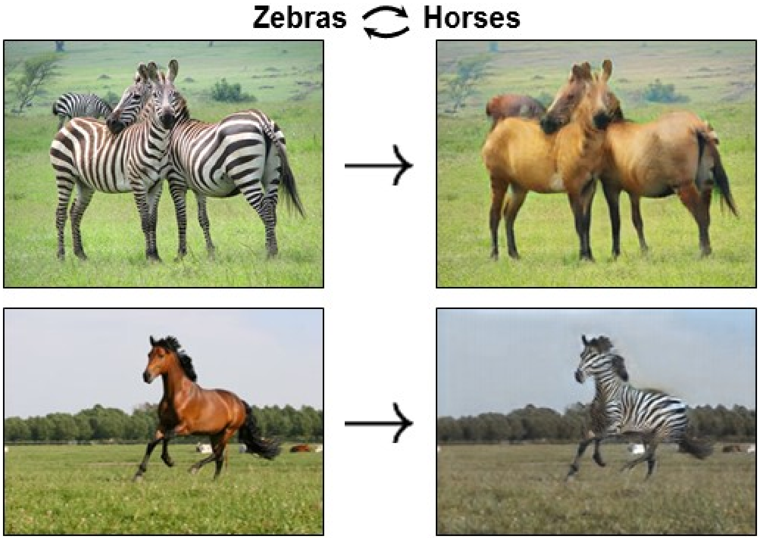


Learning examples

Learning examples



Learning examples



Learning examples

- Spam e-mail detection system
- Credit approval
- Recognize objects in images
- Find the relation between house prices and house sizes
- Predict the stock market

- Market segmentation
- Market basket analysis
- Language models (word2vec)
- Social network analysis
- Movies recommendation
- Low-order data representations

What learning is about

Machine learning is meaningful to be applied if:

1. A pattern exists
2. We cannot pin it down mathematically
3. We have data on it

Assumption 1. and 2. are not mandatory:

- If a pattern does not exist, I do not learn anything
- If I can describe the mathematical relation, I will not presumably learn the best function
- The real constraint is assumption 3

Outline

Components of learning

Formalization:

- Input: φ (*e-mail textual content*) $\rightarrow$ each dimension is some e-mail attribute
- Ouptut: y (*spam/not spam?*) $\rightarrow$ the decision that we have to take in the end
- Target function: $f : \mathcal{X} \rightarrow \mathcal{Y}$ (*Ideal spam filter formula*) $\rightarrow$ unknown, we have to learn it
- Data: $\mathcal{D} = \{\varphi(1), y(1)\}, \ldots, \{\varphi(N), y(N)\}$ (*historical records of e-mail examples*)

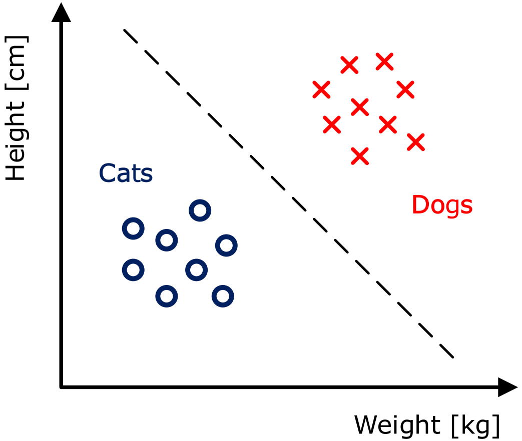
$\downarrow \quad \downarrow \quad \downarrow$

- Hypothesis: $g : \mathcal{X} \rightarrow \mathcal{Y},\ g \in \mathcal{H}$ (*formula to be used*) $\rightarrow$ g is an approximation of f

$\mathcal{H}$ is called the Hypothesis space. This, toghether with the Learning algorithm, form the *learning model*

Supervised learning

- The “correct answer” y is given
- Predict y from a set of inputs $\boldsymbol{\varphi} \in \mathbb{R}^{(d-1)\times 1}$
- Regression: predict continous output $y \in \mathbb{R}$ (real value)
- Classification: predict discrete categorical output $y \in \{1, \dots, C\}$ (class)



Example: House prices regression

Suppose we want to find a linear function which relates the measured regressors $x_1, x_2, \dots, x_{d-1}$ with the observed output y

Size $[\text{feet}^2]$	Number of bedrooms	Number of floors	Age of home $[\text{year}]$	Price $[\$]$
2104	5	1	45	$4.60 \cdot 10^5$
1416	3	2	40	$2.32 \cdot 10^5$
1534	2	1	30	$3.15 \cdot 10^5$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
x_1	x_2	x_3	x_4	y

- The number of rows is the number of data points N
- The ith observation is the vector $\varphi(i) = [x_1(i) \;\; x_2(i) \;\; x_3(i) \;\; x_4(i)]^{\mathrm{T}} \in \mathbb{R}^{4\times 1}$
- Each feature vector φ has associated a response $y \in \mathbb{R}$ that we want to predict for new observations of φ

Example: House prices classification

The components of the features vector are the same. The difference lies in the response variable, which now is a class (categorical data type) and not a real value

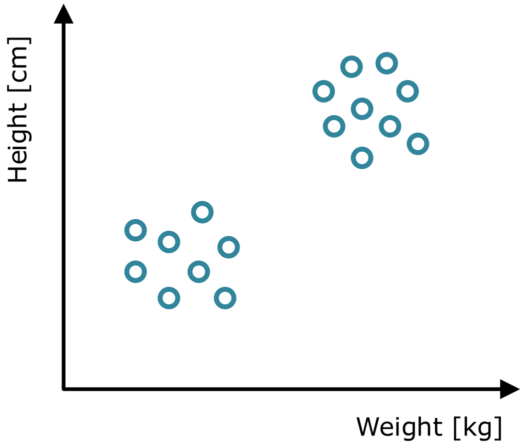
Suppose that instead of the price value in dollars, we want to classify houses as *expensive* (class $y = 1$) or *cheap* (class $y = 0$)

Size $[\text{feet}^2]$	Number of bedrooms	Number of floors	Age of home $[\text{year}]$	Price $[\text{class}]$
2104	5	1	45	1
1416	3	2	40	0
1534	2	1	30	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
x_1	x_2	x_3	x_4	y

The point $\boldsymbol{\varphi}$ can then be classified to class $y = 1$ if $s\left(\begin{bmatrix}1 & \boldsymbol{\varphi}\end{bmatrix}\right) \geq 0.5$, where $s(\cdot)$ is the logistic function. This happens when $\begin{bmatrix}1 & \boldsymbol{\varphi}\end{bmatrix}^{\mathrm{T}} \cdot \boldsymbol{\vartheta} \geq 0$ (linear classifier)

Unsupervised learning

- Instead of (**input, output**) we get (**input, ?**)
- Find properties of the inputs $\varphi \in \mathbb{R}^{(d-1)\times 1}$
- High-level representation of the input
- Elements into the same cluster have similar properties



Reinforcement learning

- Instead of (**input, output**) we get (**input, output, reward**)
- The algorithm tries to learn what action to take, in order to maximise the reward
- This is called a policy
- Applications in control, robotics, A/B testing

Learning examples revisited

Supervised Learning (Classification)

- Spam e-mail detection system
- Credit approval
- Recognize objects in images
- Find the relation between house prices and house sizes
- Predict the stock market

Unsupervised Learning

- Market segmentation
- Market basket analysis
- Language models (word2vec)
- Social network analysis
- Movies recommendation*
- Low-order data representations

In this course we will focus on the **supervised** learning case

Learning examples revisited

Supervised Learning (Regression)

- Spam e-mail detection system
- Credit approval
- Recognize objects in images
- Find the relation between house prices and house sizes
- Predict the stock market

Unsupervised Learning

- Market segmentation
- Market basket analysis
- Language models (word2vec)
- Social network analysis
- Movies recommendation*
- Low-order data representations

In this course we will focus on the **supervised** learning case

Supervised learning: problem statement

The aim is to learn an unknown function f, given a dataset $\mathcal{D}$

- The function is searched in the hypothesis space $\mathcal{H}$, where $h \in \mathcal{H}$ is a specific function
- We want to find a function h that approximates f well, on the whole domain $\mathcal{X}$

What does $h \approx f$ mean?

- We need to define an error measure or a cost function
- Almost always *pointwise definition*: $e\big[f(\boldsymbol{\varphi}), h(\boldsymbol{\varphi})\big]$

Cost functions

Pointwise error examples

- *Squared error*: $e\big(f(\varphi), h(\varphi)\big) = \big(f(\varphi) - h(\varphi)\big)^2$ → used for regression
- *Binary error*: $e\big(f(\varphi), h(\varphi)\big) = \mathbb{I}\big[f(\varphi) \neq h(\varphi)\big]$ → used for classification

It is interesting to look at the *overall error*, which considers all N examples:

Overall error examples

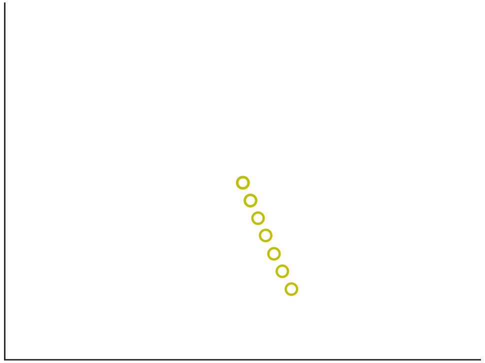
- *In sample error*: $E_{\text{in}} = J(\cdot) = \frac{1}{N}\sum_{i=1}^{N} e\big[f\big(\varphi(i)\big), h\big(\varphi(i)\big)\big]$
- *Out of sample error*: $E_{\text{out}} = \mathbb{E}_{\varphi}\big[e\big(f(\varphi), h(\varphi)\big)\big]$

Outline

- Introduction
- Components of learning
- **Puzzle**
- Feasibility of learning
- Bias - variance tradeoff

Puzzle

Focus on supervised learning: which are the plausible response values of the unknown function, on positions of the input space that we have not seen?



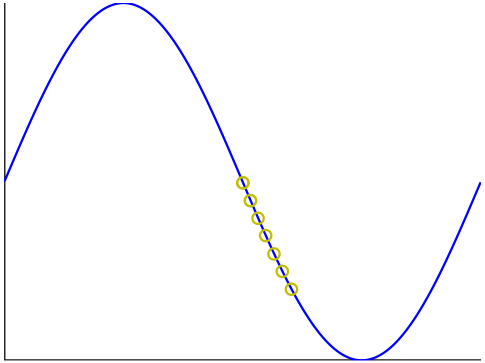
● ○ ●   ● ● ○   ● ● ● $f = 1$

○ ● ○   ○ ○ ●   ○ ○ ● $f = 0$

● ○ ○ $f = ?$

Puzzle

It is not possible to know how the function behaves outside the observed points (*Hume's induction problem*)



● ○ ●   ● ● ○   ● ● ● $f = 1$

○ ● ○   ○ ○ ●   ○ ○ ● $f = 0$

● ○ ○ $f = 1$

If first dot is black $\rightarrow f = 1$

Outline

- Introduction
- Components of learning
- Puzzle
- **Feasibility of learning**
- Bias - variance tradeoff

Feasibility of learning

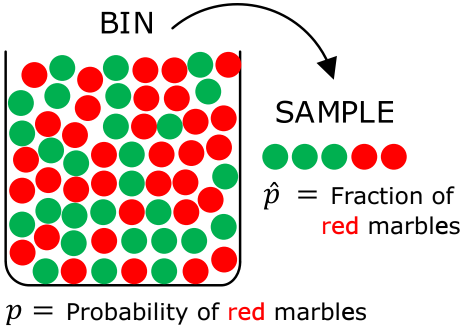
Focus on supervised learning, dichotomic classification case

Problem: Learning an unknown function

Solution: Impossible ☹. The function can assume any value outside the data we have

Experiment

- Consider a 'bin' with **red** and **green** marbles
- $\mathbb{P}[$ picking a **red** marble $] = p$
- The value of p is unknown to us
- Pick N marbles independently
- Fraction of red marbles in the sample $= \hat{p}$


BIN
SAMPLE
$\hat{p}$ = Fraction of red marbles
p = Probability of red marbles

Does $\hat{p}$ say something about p?

No!

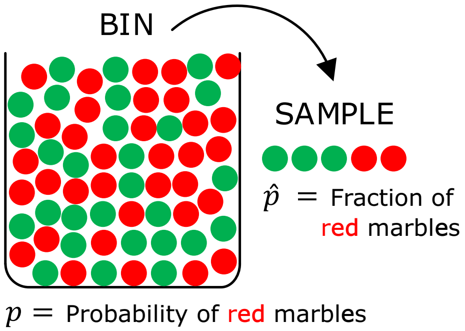
Sample can be mostly **green** while bin is mostly **red**

Possible

Yes!

Sample frequency $\hat{p}$ is likely close to bin frequency p (*if the sample is sufficiently large*)

Probable


BIN
SAMPLE
$\hat{p}$ = Fraction of red marbles
p = Probability of red marbles

Connection to learning

Bin: The unknown is a number p

Learning: The unknown is a function $f : \mathcal{X} \to \mathcal{Y}$

Each marble ● is a input point $\varphi \in \mathcal{X} \subset \mathbb{R}^{(d-1)\times 1}$

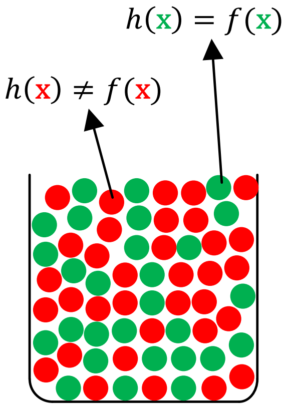
For a specific hypothesis $h \in \mathcal{H}$:

- Hypothesis got it right $\to h(\varphi) = f(\varphi)$
- Hypothesis got it wrong $\to h(\varphi) \neq f(\varphi)$

Both p and $\hat{p}$ depend on the particular hypothesis h

$\hat{p} \to$ In sample error $E_{\text{in}}(h)$

$p \to$ Out of sample error $E_{\text{out}}(h)$


$h(\mathbf{x}) = f(\mathbf{x})$
$h(\mathbf{x}) \neq f(\mathbf{x})$

The **Out of sample error** $\boldsymbol{E_{\text{out}}(h)}$ is the quantity that really matters

Connection to real learning

In a learning scenario, the function h is not fixed a priori

- The *learning algorithm* is used to fathom the hypothesis space $\mathcal{H}$, to find the best hypothesis $h \in \mathcal{H}$ that matches the sampled data $\rightarrow$ call this hypothesis g
- With many hypotheses, there is more chance to find a good hypothesis g only by chance $\rightarrow$ the function can be perfect on sampled data but bad on unseen ones

There is therefore an approximation - generalization tradeoff between:

- Perform well on the given dataset
- Perform well on unseen data

The quantity $E_{\text{out}}(g) - E_{\text{in}}(g)$ is called the generalization error

Generalization theory

There is a generalization theory, based on the concept of VC-dimension, which studies the cases in which is possible to generalize

- The takeaway concept is that learning is feasible in a probabilistic way
- If we are able to deal with the approximation-generalization tradeoff, we can say with high probability that the generalization error is small

One way to study the tradeoff is to study the concepts of **bias** and **variance** of a learning model

Outline

Approximation vs. generalization

The ultimate goal is to have a small E_{out}: good approximation of f out of sample

- More complex $\mathcal{H} \implies$ better chances of **approximating** f in sample $\rightarrow$if $\mathcal{H}$ is too simple, we fail to approximate f and we end up with large E_{in}

- Less complex $\mathcal{H} \implies$ better chance of **generalizing** out of sample $\rightarrow$if $\mathcal{H}$ is too complex, we we fail to generalize well

Approximation vs. generalization

The example shows:

- perfect fit on training data

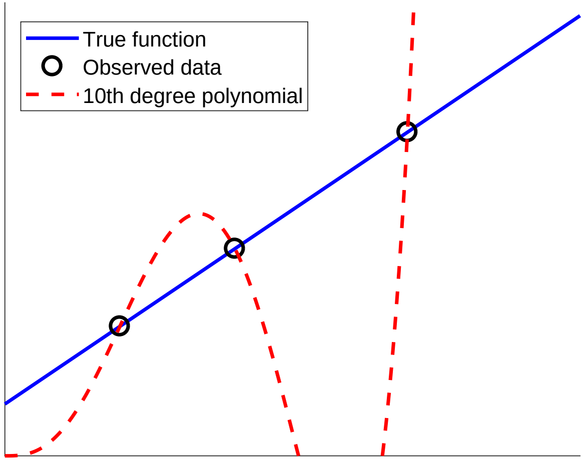
$$\downarrow$$

$$E_{\text{in}} = 0$$

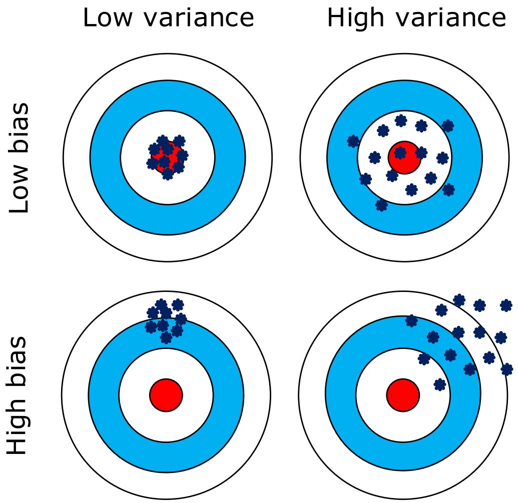
- low fit on test data

$$\downarrow$$

$$E_{\text{out}} \text{ high}$$



Bias and variance



Bias and variance

Bias-variance analysis decomposes E_{out} into two terms:

1. How well $\mathcal{H}$ can approximate $f \rightarrow$ **Bias**
2. How well we are able to find a good $h \in \mathcal{H} \rightarrow$ **Variance**

The out of sample error is (making explicit the dependence of g on $\mathcal{D}$):

$$E_{\text{out}}(g^{(\mathcal{D})}) = \mathbb{E}_{\boldsymbol{\varphi}}\left[\left(g^{(\mathcal{D})}(\boldsymbol{\varphi}) - f(\boldsymbol{\varphi})\right)^2\right]$$

The expected out of sample error of the learning model is independent of the particular realization of data set used to find $g^{(\mathcal{D})}$:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(g^{(\mathcal{D})})\right] &= \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\boldsymbol{\varphi}}\left[\left(g^{(\mathcal{D})}(\boldsymbol{\varphi}) - f(\boldsymbol{\varphi})\right)^2\right]\right] \\ &= \mathbb{E}_{\boldsymbol{\varphi}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\boldsymbol{\varphi}) - f(\boldsymbol{\varphi})\right)^2\right]\right]\end{aligned}$$

Bias and variance

Focus on $\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\varphi) - f(\varphi)\right)^2\right]$

Define the 'average' hypothesis $\bar{g}(\varphi) = \mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\varphi)\right]$

This average hypothesis can be derived by imagining many datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K$, and building it by $\bar{g}(\varphi) \approx \frac{1}{N}\sum_{k=1}^{K} g^{(\mathcal{D}_k)}(\varphi)$ $\rightarrow$ this is a conceptual tool, and $\bar{g}$ does not need to belong to the hypothesis set

$$
\begin{aligned}
\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\varphi) - f(\varphi)\right)^2\right] &= \mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi) + \bar{g}(\varphi) - f(\varphi)\right)^2\right] \\
&= \mathbb{E}_{\mathcal{D}}\Bigg[\left(g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi)\right)^2 + \left(\bar{g}(\varphi) - f(\varphi)\right)^2 \\
&\quad + 2 \cdot \left(g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi)\right)\left(\bar{g}(\varphi) - f(\varphi)\right)\Bigg]
\end{aligned}
$$

Bias and variance

$$\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\varphi) - f(\varphi)\right)^2\right] = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi)\right)^2\right]}_{\textbf{var}(\varphi)} + \underbrace{\left(\bar{g}(\varphi) - f(\varphi)\right)^2}_{\textbf{bias}(\varphi)}$$

Therefore:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(g^{(\mathcal{D})})\right] &= \mathbb{E}_{\varphi}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\varphi) - f(\varphi)\right)^2\right]\right] \\ &= \mathbb{E}_{\varphi}\left[\textbf{bias}(\varphi) + \textbf{var}(\varphi)\right] \\ &= \quad \textbf{bias} \quad + \quad \textbf{var}\end{aligned}$$

Bias and variance

Interpretation

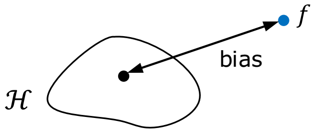
- The **bias** term $\left(\bar{g}(\varphi) - f(\varphi)\right)^2$ measures how much our learning model is biased away from the target function

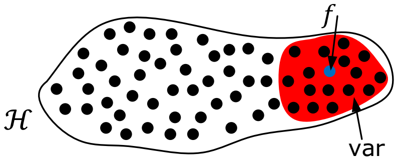
  Infact, $\bar{g}$ has the benefit of learning from an unlimited number of datasets, so it is only limited in its ability to approximate f by the limitations of the learning model itself

- The **variance** term $\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi)\right)^2\right]$ measures the variance in the final hypothesis, depending on the data set, and can be thought as how much the final chosen hypothesis differs from the 'mean' (best) hypothesis

Bias and variance

$$\textbf{bias} = \left(\bar{g}(\varphi) - f(\varphi)\right)^2 \qquad \textbf{variance} = \mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi)\right)^2\right]$$





Very small model. Since there is only one hypothesis, both the average function $\bar{g}$ and the final hypothesis $g^{(\mathcal{D})}$ will be the same, for any dataset. Thus, $\text{var} = 0$. The bias will depend solely on how well this single hypothesis approximates the target f, and unless we are extremely lucky, we expect a large bias

Very large model. The target function is in $\mathcal{H}$. Different data sets will led to different hypotheses that agree with f on the data set, and are spread around f in the red region. Thus, $\text{bias} \approx 0$ because $\bar{g}$ is likely to be close to f. The var is large (heuristically represented by the size of the red region)

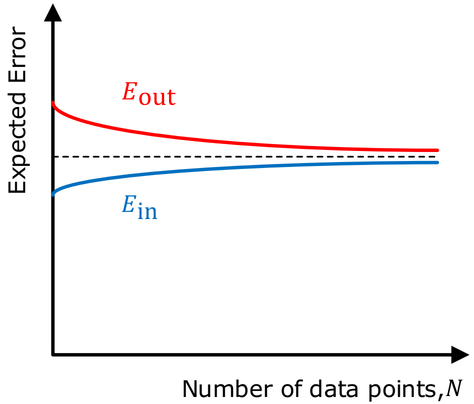
Learning curves

How it is possible to know if a model is suffering from bias or variance problems?

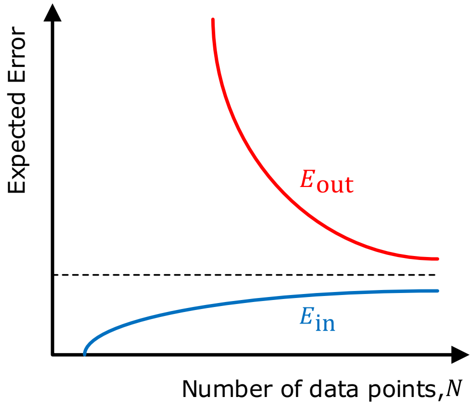
The learning curves provide a graphical representation for assessing this, by plotting the *expected out of sample error* $\mathbb{E}_{\mathcal{D}}[E_{\text{out}}(g^{\mathcal{D}})]$ and the *expected in sample error* $\mathbb{E}_{\mathcal{D}}[E_{\text{in}}(g^{\mathcal{D}})]$ vs. the number of data N

In the practice, the curves are computed from one dataset, or by dividing it into more parts and taking the mean curve resulting from various datasets

Learning curves



Simple model



Complex model

Learning curves

Interpretation

- Bias can be present when the error is quite high and E_{in} is similar to E_{out}
- When bias is present, getting more data is not likely to help
- Variance can be present when there is a gap between E_{in} and E_{out}
- When variance is present, getting more data is likely to help

Fixing bias

- Try adding more features
- Try polynomial features
- Try a more complex model
- Boosting

Fixing variance

- Try a smaller set of features
- Get more training examples
- Regularization
- Bagging

Take home lessons

Rule of thumb
How many data points N are required to ensure good generalization?

$$\boxed{N \geq 10 \cdot (\text{no. of model's parameters})}$$

General principle
Match the **'model complexity'** to the **data resources**, not to the **target complexity**

Machine learning fundamentals: Lecture 2

Overfitting

Regularization

Validation

Mirko Mazzoleni - Identificazione dei Modelli e Analisi dei Dati (IMAD)

11 October 2017

University of Bergamo
Department of Management, Information and Production Engineering
mirko.mazzoleni@unibg.it

Outline

- Overfitting
- Regularization
- Validation
- Model selection
- Cross-validation

Outline

- **Overfitting**
- Regularization
- Validation
- Model selection
- Cross-validation

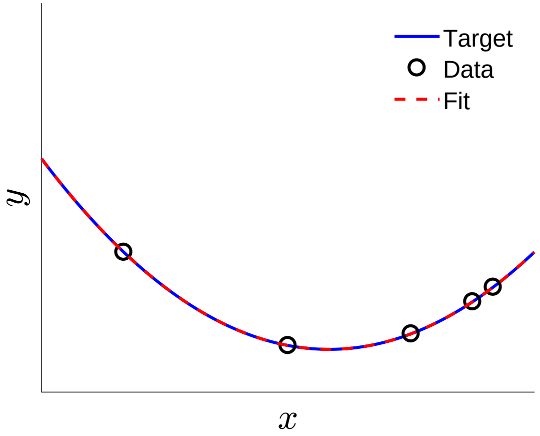
Overfitting

THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Overfitting

- Simple target function
- $N = 5$ points
- Fit with 4th order polynomial

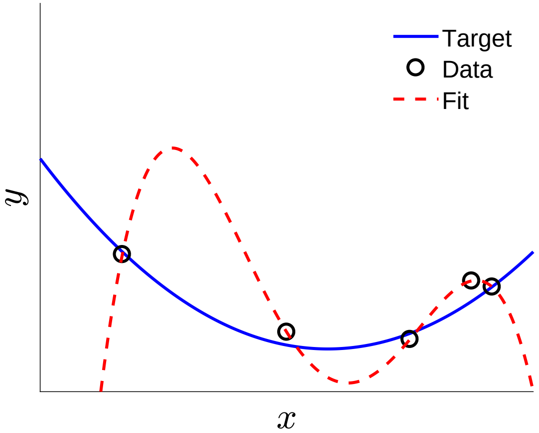
$$E_{\text{in}} = 0, \;\; E_{\text{out}} = 0$$



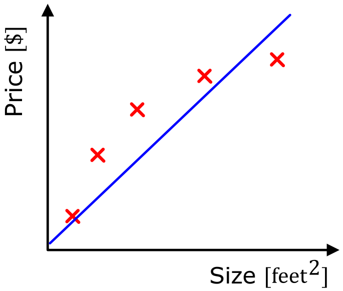
Overfitting

- Simple target function
- $N = 5$ **noisy** points
- Fit with 4th order polynomial

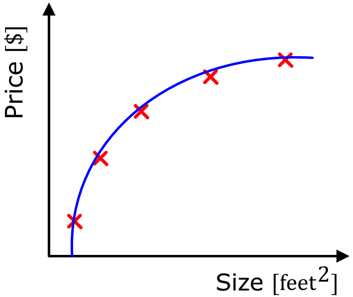
$E_{\text{in}} = 0, \;\; E_{\text{out}}$ is huge



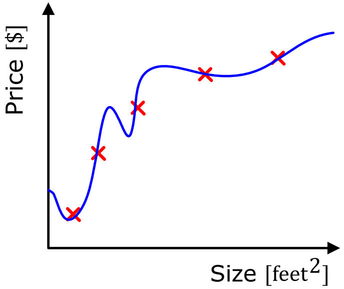
Overfitting



Underfit



Just right

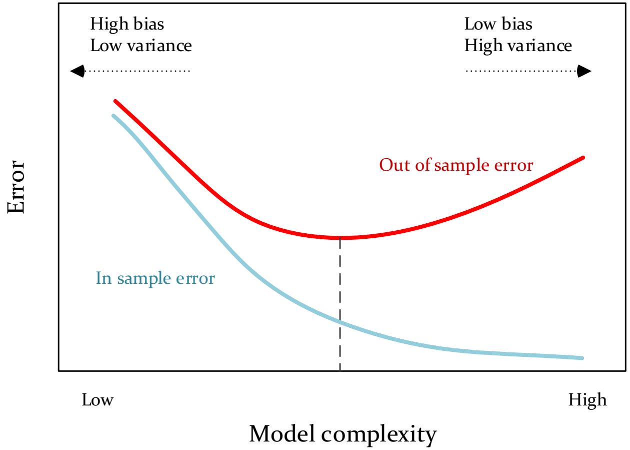


Overfit

Overfitting

We talk of overfitting when decreasing E_{in} leads to increasing E_{out}

- Major source of failure for machine learning systems
- Overfitting leads to bad generalization
- A model can exibit bad generalization even if it does not overfit



What causes overfitting

Overfitting occurs because the learning model spends its resources trying to fit the noise, on the finite dataset of size N

- The learning model is not able to discern the **signal** from the **noise**
- The effect of the noise is to show to the learning model "hallucinating patterns" that do not exist

Actually, two noise terms:

1. **Stochastic noise** $\rightarrow$ random noise on the observations
2. **Deterministic noise** $\rightarrow$ fixed quantity that depends on $\mathcal{H}$

What is noise

Noise refers to things that I **can't capture**. It is the part of the data y that I **cannot model**

1. **Stochastic noise**: I can't capture it because there is nothing to capture. There are fluctuations/measurement errors that we cannot model

2. **Deterministic noise**: I can't capture it because I am too simple, I have limited capacity

The two quantities behave therefore the same for the learning model: both are things that it can't capture. The problem is that, **on a finite data sample** N, the learning algorithm tries to fit the noise

Deterministic noise

The part of f that $\mathcal{H}$ cannot capture: $f(\varphi) - h^*(\varphi)$

- $h^*(\varphi)$ is the best hypothesis in $\mathcal{H}$ for approximating f
- Because h^* is simpler than f, any part of f that h^* cannot explain is like a noise for h^*, since that points deviate from h^* for an unknown reason

Overfitting can happen **even if the target is noiseless** (no stochastic noise)

- On a finite dataset N, the deterministic noise can led the learning model to interpret the data in a wrong way
- This happens because the model is not able to "understand" the target, and so it "misinterprets" the data, constructing its own hypothesis, which is wrong because it did not understand the true function f, given its limited hypothesis space

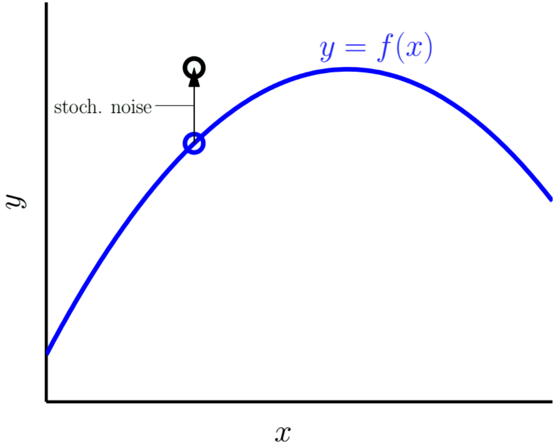
Stochastic noise - measurements error

We would like to learn from ● :

$$y(i) = f\big(\varphi(i)\big)$$

Unfortunately, we only observe ● :

$$y(i) = f\big(\varphi(i)\big) + \text{stochastic noise}$$



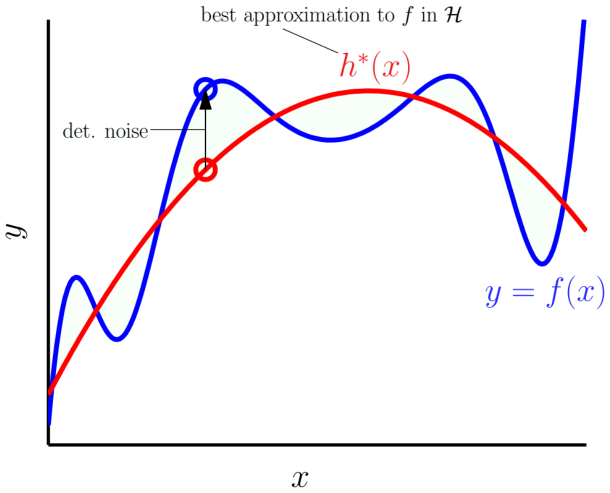
Deterministic noise - model error

We would like to learn from ● :

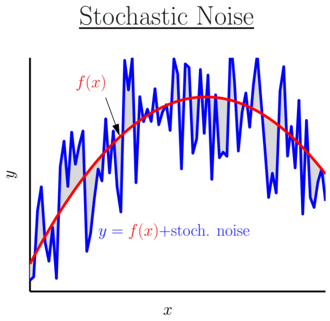
$$y(i) = h^*\big(\varphi(i)\big)$$

Unfortunately, we only observe ● :

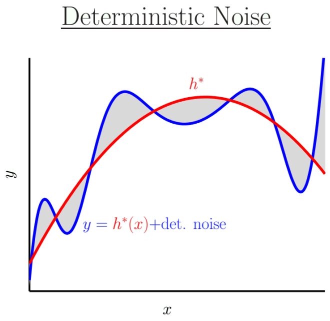
$$\begin{aligned} y(i) &= f\big(\varphi(i)\big) \\ &= h^*\big(\varphi(i)\big) + \text{deterministic noise} \end{aligned}$$



Stochastic vs. deterministic noise



source: random measurement errors



source: learner's $\mathcal{H}$ cannot model f

The problem with both noises is that the algorithm tries to fit data that it cannot explain and understand. So it comes up with its own "idea" of the function, but it does not understand, and so makes bad decision (bad learning)

Overfitting behaviour

Summarising we have that:

- Overfitting $\uparrow$ if stochastic noise $\uparrow$ $\rightarrow$ the model is deceived by erroneous data
- Overfitting $\uparrow$ if deterministic noise $\uparrow$ $\rightarrow$ deterministic noise $\uparrow$ if target complexity $\uparrow$
- Overfitting $\downarrow$ if N $\uparrow$ $\rightarrow$ with more data, it is more difficult that the model will follow the noise of all of them

Overfitting behaviour with deterministic noise

Case study 1
Suppose $\mathcal{H}$ fixed, and increase the complexity of f

- Deterministic noise $\uparrow$, since there are more parts of f that h^* cannot explain, and then they act as noise for h^*
- Deterministic noise $\uparrow \implies$ overfitting $\uparrow$

Case study 2
Suppose f fixed, and decrease the complexity of $\mathcal{H}$

- Deterministic noise $\uparrow \implies$ overfitting $\uparrow$
- Simpler model $\implies$ overfitting $\downarrow$
- Most of the times the gain in reducing the model complexity exceeds the increase in deterministic noise

Bias - variance tradeoff revisited

Let the stochastic noise $\varepsilon(\varphi)$ be a random variable with mean zero and variance σ^2

$$\mathbb{E}_{\mathcal{D},\varphi,\varepsilon}\left[\left(g^{(\mathcal{D})}(\varphi) - (f(\varphi) + \varepsilon(\varphi))\right)^2\right] =$$

$$\underbrace{\mathbb{E}_{\mathcal{D},\varphi}\left[\left(g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi)\right)^2\right]}_{\textbf{var}} + \underbrace{\mathbb{E}_{\varphi}\left(\bar{g}(\varphi) - f(\varphi)\right)^2}_{\textbf{bias}} + \underbrace{\mathbb{E}_{\varepsilon,\varphi}\left[\left(\varepsilon(\varphi)\right)^2\right]}_{\boldsymbol{\sigma^2}}$$

- Stochastic noise is related to power of the random noise $\rightarrow$ irreducible error
- Deterministic noise is related to **bias** $\rightarrow$ part of f that $\mathcal{H}$ cannot capture
- Overfitting is caused by **variance**. Variance is, in turn, affected by the noise terms, capturing a model's **susceptibility** to being led astray by the noise
- The variance term depends also on the complexity of $\mathcal{H}$

Outline

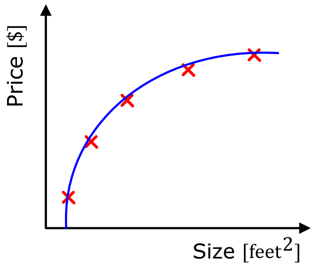
A cure for overfitting

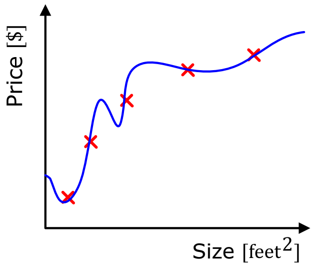
Regularization is the first line of defense against overfitting

- We have seen that complex model are more prone to overfitting
- This is because they are more powerful, and thus they can fit the noise
- Simple models exibits less variance because of their limited expressivity. This gain in variance often is greater than their greater bias
- However, if we stick only to simple models, we may not end up with a satisfying approximation of the target function f

How can we retain the benefits of both worlds?

A cure for overfitting





$$\mathcal{H}_2 : \vartheta_0 + \vartheta_1 x + \vartheta_2 x^2$$

$$\mathcal{H}_4 : \vartheta_0 + \vartheta_1 x + \vartheta_2 x^2 + \vartheta_3 x^3 + \vartheta_4 x^4$$

- We can recover the model $\mathcal{H}_2$ from the model $\mathcal{H}_4$ by imposing $\vartheta_3 = \vartheta_4 = 0$

$$\arg\min_{\boldsymbol{\vartheta}} \quad \frac{1}{N}\sum_{i=1}^{N}\Big(h\big(\boldsymbol{\varphi}(i);\boldsymbol{\vartheta}\big) - f\big(\boldsymbol{\varphi}(i)\big)\Big)^2 + 1000\cdot(\vartheta_3)^2 + 1000\cdot(\vartheta_4)^2$$

A cure for overfitting

$$\arg\min_{\boldsymbol{\vartheta}} \ \frac{1}{N}\sum_{i=1}^{N}\Big(h\big(\boldsymbol{\varphi}(i);\boldsymbol{\vartheta}\big) - f\big(\boldsymbol{\varphi}(i)\big)\Big)^2 + \underbrace{1000\cdot(\vartheta_3)^2 + 1000\cdot(\vartheta_4)^2}_{\Omega}$$

- The cost function has been augmented with a penalization term $\Omega(\vartheta_3, \vartheta_4)$
- The minimization algorithm now has to minimize both E_{in} and $\Omega(\vartheta_3, \vartheta_4)$
- Due to the minimization process the value of ϑ_3 and ϑ_4 will be shrinked toward a small value $\rightarrow$ not exactly zero: soft order constraint
- If this value is very small, then the contribution of x_3 and x_4 is negligible $\rightarrow$ ϑ_3 and ϑ_4 (very small) multiply x_3 and x_4
- In this way, we ended up with a model that it is like the model $\mathcal{H}_2$ in terms of complexity $\rightarrow$ we can think as like the features x_3 and x_4 were not present
- It is like we reduced the number of parameters of the $\mathcal{H}_4$ model

Regularization

The concept introduced in the previous slides can be extented on the entire model's parameters

Instead of minimizing the in-sample error E_{in}, minimize the **augmented error**:

$$\boxed{E_{\text{aug}}(\boldsymbol{\vartheta}) = \frac{1}{N}\sum_{i=1}^{N}\Big(h(\varphi(i);\boldsymbol{\vartheta}) - f(\varphi(i))\Big)^2 + \lambda\sum_{j=1}^{d-1}(\vartheta_j)^2}$$

- Usually we do not want to penalize the intercept ϑ_0, so j starts from 1
- The term $\Omega(h) = \sum_{j=1}^{d-1}(\vartheta_j)^2$ is called regularizer
- The regularizer is a penalty term which depends on the hypothesis h
- The term λ weights the importance of minimizing E_{in}, with respect to minimizing $\Omega(h)$

Regularization

The minimization of E_{aug} can be viewed as a constrained minimization problem

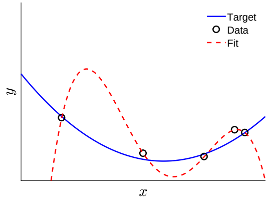
$$\text{Minimize } \; E_{\text{aug}}(\boldsymbol{\vartheta}) = \frac{1}{N}\sum_{i=1}^{N}\Big(h\big(\boldsymbol{\varphi}(i);\boldsymbol{\vartheta}\big) - f\big(\boldsymbol{\varphi}(i)\big)\Big)^2$$

$$\text{Subject to : } \; \boldsymbol{\vartheta}^{\mathrm{T}}\boldsymbol{\vartheta} \leq C$$

- With this view, we are explicitly constraining the weights to not have certain large values
- There is a relation between C and λ in such a way that if $C \uparrow$ the $\lambda \downarrow$
- Infact, bigger C means that the weights can be greater. This is equal to set for a lower λ, because the regularization term will be less important, and therefore the weights will not be shrunked as much

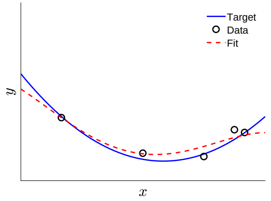
Effect of λ

λ_1



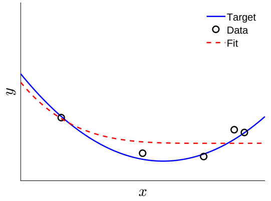
Overfit

$\lambda_2 > \lambda_1$



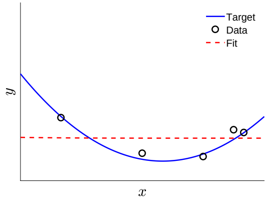
$\rightarrow$

$\lambda_3 > \lambda_2$



$\rightarrow$

$\lambda_4 > \lambda_3$



Underfit

Augmented error

General form of the augmented error

$$E_{\text{aug}}(\boldsymbol{\vartheta}) = E_{\text{in}}(\boldsymbol{\vartheta}) + \lambda\Omega(h)$$

- $\Omega(h)$ is a measure of complexity of a specific hypothesis $h \in \mathcal{H}$
- This information about the model's complexity is used in the cost function and drives the minimization process

The augmented error E_{aug} is **better** than E_{in} as a proxy for E_{out}

Augmented error

The holy Grail of machine learning would be to have a formula for E_{out} to minimize

- In this way, it would be possible to directly minimize the out of sample error instead of the in sample one
- Regularization helps by estimating the quantity $\Omega(h)$, which, added to E_{in}, gives E_{aug}, an estimation of E_{out}

Choice of the regularizer

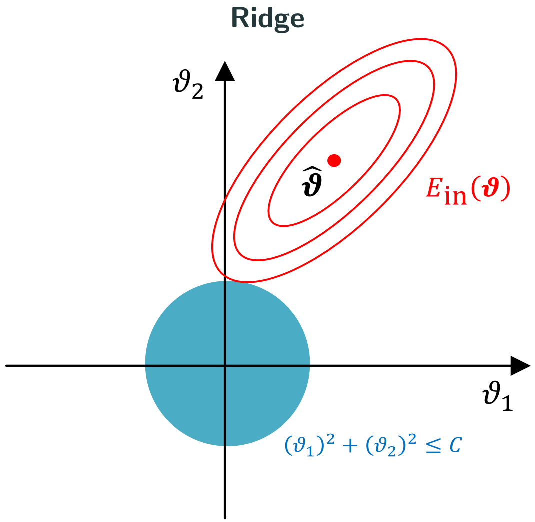
There are many choices of possible regularizes. The most used ones are:

- L_2 **regularizer:** also called Ridge regression, $\Omega(h) = \sum_{j=1}^{d-1} \vartheta_j^2$
- L_1 **regularizer:** also called Lasso regression, $\Omega(h) = \sum_{j=1}^{d-1} |\vartheta_j|$

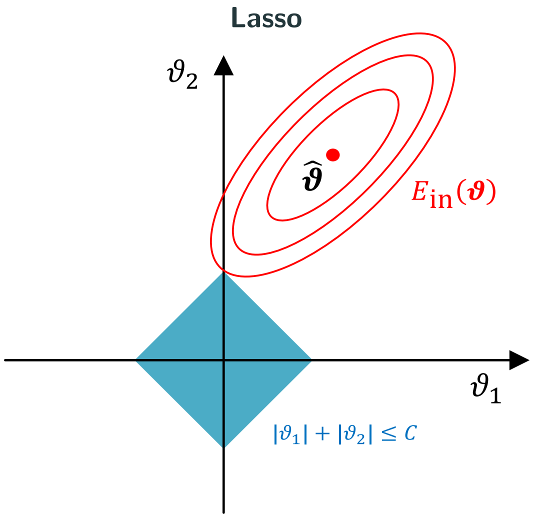
The different regularizers behaves differently:

- The ridge penalty tends to shrink all coefficients to a lower value
- The lasso penalty tends to set more coefficients exactly to zero

Geometrical interpretation



Ridge
ϑ_2
$\hat{\boldsymbol{\vartheta}}$
$E_{\text{in}}(\boldsymbol{\vartheta})$
ϑ_1
$(\vartheta_1)^2 + (\vartheta_2)^2 \leq C$



Lasso
ϑ_2
$\hat{\boldsymbol{\vartheta}}$
$E_{\text{in}}(\boldsymbol{\vartheta})$
ϑ_1
$|\vartheta_1| + |\vartheta_2| \leq C$

Regularization and bias-variance

The effects of the regularization procedure can be observed in the bias and variance terms

- Regularization trades more bias in order to considerely decrease the variance of the model
- Regularization strives for smoother hypothesis, thereby reducing the opportunities to overfit
- The amount of regularization λ has to be chosen specifically for each type of regularizer
- Usually λ is chosen by cross-validation
- When there is more stochastic noise or more deterministic noise, a higher value of λ is required to contrast their effect

Outline

- Overfitting
- Regularization
- **Validation**
- Model selection
- Cross-validation

Validation vs. regularization

The out of sample error can be seen as: $E_{\text{out}}(h) = E_{\text{in}}(h) \ + \text{overfit penalty}$

Regularization

$$E_{\text{out}}(h) = E_{\text{in}}(h) \ + \underbrace{\text{overfit penalty}}_{\text{regularization estimates this quantity}}$$

Validation

$$\underbrace{E_{\text{out}}(h)}_{\text{validation estimates this quantity}} = E_{\text{in}}(h) + \text{overfit penalty}$$

Validation set

The idea of a validation set is to estimate the model's performance out of sample

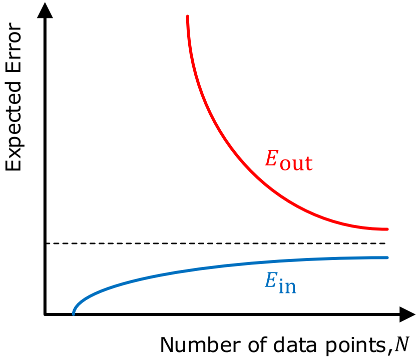
1. Remove a subset from the training data $\rightarrow$ this subset is not used in training
2. Train the model on the remaining training data $\rightarrow$ the model will be trained on *less* data
3. Evaluate the model's performance on the held-out set $\rightarrow$ this is an unbiased estimation of the out of sample error
4. Retrain the model on *all* the data

K is taken out of N

Given the dataset $\mathcal{D} = \{\varphi(1), y(1)\}, \ldots, \{\varphi(N), y(N)\}$

$\underbrace{K \text{ points}}_{\mathcal{D}_{\text{val}}} \rightarrow$ validation $\qquad \underbrace{N - K \text{ points}}_{\mathcal{D}_{\text{train}}} \rightarrow$ training

- Small K: bad estimate of E_{out}
- Large K: possibility of learning a bad model (learning curve)



K is put back into N

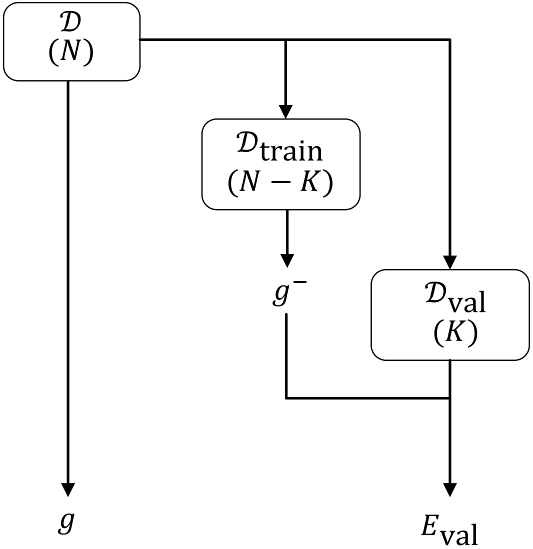
$$\begin{tikzcd}
\mathcal{D} \arrow[r] \arrow[d] & \mathcal{D}_{\text{train}} \arrow[r, phantom, "\cup"] \arrow[d] & \mathcal{D}_{\text{val}} \arrow[d] \\
N & N-K & K
\end{tikzcd}$$

$\mathcal{D} \Longrightarrow g \qquad \mathcal{D}_{\text{train}} \Longrightarrow g^-$

$E_{\text{val}} = E_{\text{val}}(g^-)$

Rule of thumb:

$$K = \frac{N}{5}$$



$\mathcal{D}$ (N)
$\mathcal{D}_{\text{train}}$ $(N-K)$
g^-
$\mathcal{D}_{\text{val}}$ (K)
g
E_{val}

Outline

- Overfitting
- Regularization
- Validation
- **Model selection**
- Cross-validation

Model selection

The most important use of a validation set is model selection

- Choose between a linear model and a nonlinear one
- Choice of the order of the polynomial
- Choice of the regularization parameter
- Any other choice that affects the learning of the model

If the validation set is used to perform choices (e.g. to select the regularization parameter λ), then it **no longer** provides an unbiased estimate of E_{out}

There is the need of a third dataset: the **test set**, onto which to measure the model's performance E_{test}

Using $\mathcal{D}_{\text{val}}$ more than once

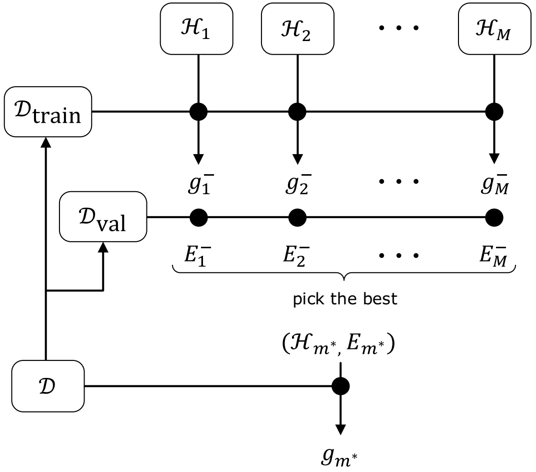
M models $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$

Use $\mathcal{D}_{\text{train}}$ to learn g_m^- for each model

Evaluate g_m^- using $\mathcal{D}_{\text{val}}$

$$E_m = E_{\text{val}}(g_m^-) \qquad m = 1, \ldots, M$$

Pick the model $m = m^*$ with the smallest E_m



How much bias

For the M models $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$, $\mathcal{D}_{\text{val}}$ is used for "training" on the **finalist model set**:

$$\mathcal{H}_{\text{val}} = \left\{g_1^-, g_2^-, \ldots, g_M^-\right\}$$

- The validation performance of the final model is $E_{\text{val}}\left(g_{m^*}^-\right)$
- This quantity is biased and not representative of $E_{\text{out}}\left(g_{m^*}^-\right)$, just as the in sample error E_{in} was not representative of E_{out}
- What happened is that $\mathcal{D}_{\text{val}}$ has become the "training set" for $\mathcal{H}_{\text{val}}$
- The risk is to overfit the validation set

Data contamination

Error estimates: $E_{\text{in}},\ E_{\text{val}},\ E_{\text{test}}$

Contamination: Optimistic bias in estimating E_{out}

- **Training set:** totally contaminated
- **Validation set:** slightly contaminated
- **Test set:** totally 'clean'

Outline

The dilemma about K

The following chain of reasoning:

$$E_{\text{out}}(g) \underset{\text{(small } K)}{\approx} E_{\text{out}}(g^-) \underset{\text{(large } K)}{\approx} E_{\text{val}}(g^-)$$

highlights the dilemma in selecting K

Can we have K both small and large?

Leave one out cross-validation

Use $N-1$ points for training and $K=1$ point for validation

$$\mathcal{D}_i = \{\boldsymbol{\varphi}(1), y(1)\}, \dots, \cancel{\{\boldsymbol{\varphi}(i), y(i)\}}, \dots, \{\boldsymbol{\varphi}(N), y(N)\},$$

where $\mathcal{D}_i$ is the training set without the point i

The final hypothesis learned from $\mathcal{D}_i$ is g_i^-

The validation error on the point $\boldsymbol{\varphi}(i)$ is $e(i) = E_{\text{val}}(g_i^-) = e\Big(g_i^-(\boldsymbol{\varphi}(i)), y(i)\Big)$

It is then possible to define the **cross-validation error**

$$\boxed{E_{\text{cv}} = \frac{1}{N}\sum_{i=1}^{N} e(i)}$$

Cross-validation for model selection

Cross-validation can be used effectively to perform model selection by selecting the right regularization parameter λ

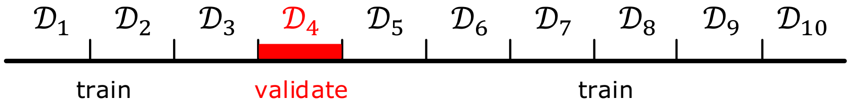
1. Define M models by choosing different values for λ: $(\mathcal{H}, \lambda_1), (\mathcal{H}, \lambda_2), \ldots, (\mathcal{H}, \lambda_M)$
2. **for** each model $m = 1, \ldots, M$ **do**
    2.1 Use cross-validation to obtain estimates of the out of sample error for each model
3. Select the model m^* with the smallest cross-validation error $E_{\text{cv}}(m^*)$
4. Use the model $(\mathcal{H}, \lambda_{m^*})$ and all the data $\mathcal{D}$ to obtain the final hypothesis g_{m^*}

Leave more than one out

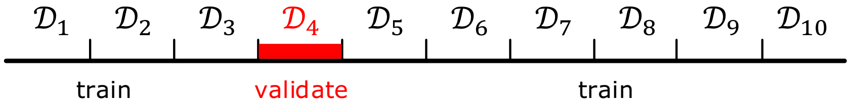
Leave-one-out cross-validation as the disadvantage that:

- It is computationally expensive, requiring a total of N training sessions for each of the M models
- The estimated cross-validation error has high variance, since it is based only on one point

It is possible to reserve more points for validation by dividing the training set in "folds"



Leave more than one out



- This produces $\frac{N}{K}$ training session on $N - K$ points each
- A good comprosime for the number of folds is 10

10-fold cross validation: $K = \frac{N}{10}$

- Pay attention to not reduce the training set to much
    - Look at the learning curves