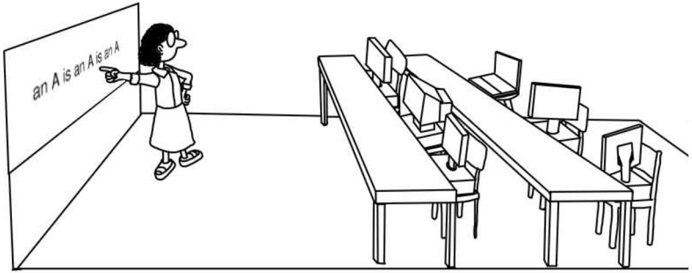
EE3-23: Machine Learning

Krystian Mikolajczyk & Deniz Gunduz

Department of Electrical and Electronic Engineering
Imperial College London

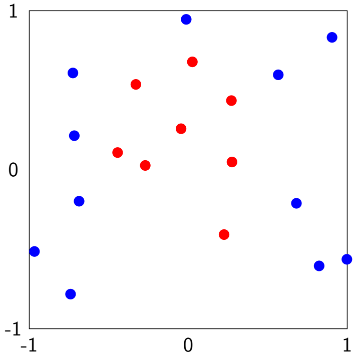


Part 3 summary

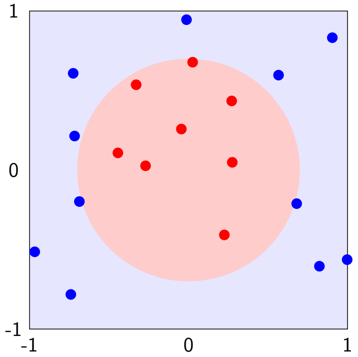
- Non-linear feature transform: polynomial, Legendre
- Overfitting/underfitting: match the hypothesis class to data
- Structural risk minimization
- Regularisation: L_2, L_1 ...

Limits of Linear Representations

Data:

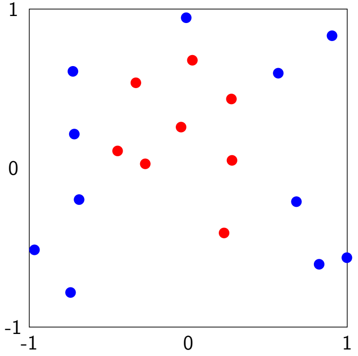


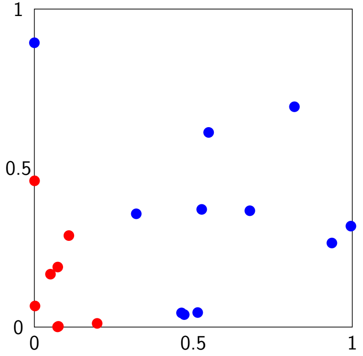
Hypothesis:



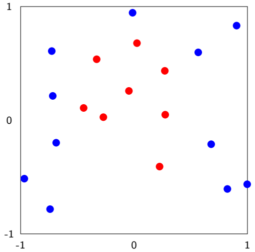
Non-linear feature transformation

$$(x_1, x_2) \xrightarrow{\Phi} (x_1^2, x_2^2)$$



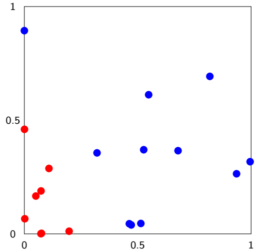


Non-linear features with linear classifier



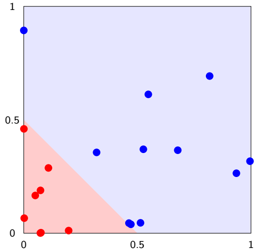
$x_i \in \mathcal{X}$

$\xrightarrow{\Phi}$



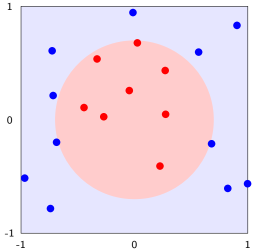
$z_i = \Phi(x_i) \in \mathcal{Z}$

$\downarrow$



$\bar{g}(z) = \mathrm{sign}(w^\top z)$

$\xleftarrow{\Phi^{-1}}$



$g(x) = \bar{g}(\Phi(x)) = \mathrm{sign}(w^\top \Phi(x))$

Feature transformation and linear classifier

$$\begin{aligned}
\mathbf{x} = (x_0, x_1, \ldots, x_d) &\quad \xrightarrow{\Phi} \quad \mathbf{z} = (z_0, z_1, \ldots, z_{\tilde{d}}) \\
\mathbf{x}_1, \ldots, \mathbf{x}_n &\quad \xrightarrow{\Phi} \quad (\mathbf{z}_1, \ldots, \mathbf{z}_n) \\
y_1, \ldots, y_n &\quad \xrightarrow{\Phi} \quad y_1, \ldots, y_n \\
\text{No weights in } \mathcal{X} &\quad \xrightarrow{\Phi} \quad \mathbf{w} = (w_0, w_1, \ldots, w_{\tilde{d}}) \\
g(\mathbf{x}) &\quad = \quad \operatorname{sign}(\mathbf{w}^\top \Phi(\mathbf{x})) \\
\text{Linear in } \mathbf{z} \text{ space } g(\mathbf{x}) &\quad = \quad \operatorname{sign}(\mathbf{w}^\top \mathbf{z})
\end{aligned}$$

Transformation of Features

$\mathbf{x} = (x_0, x_1, \ldots, x_d) \xrightarrow{\Phi} \mathbf{z} = (z_0, z_1, \ldots, z_{\bar{d}})$ that is, $\mathbf{z} = \Phi(\mathbf{x})$

Polynomial of degree q (largest power) $\Phi(x) = \sum_{k=0}^{q} a_k x^k$

- Example: $\mathbf{x} = (1, x_1, x_2) \xrightarrow{\Phi} \mathbf{z} = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$

Final hypothesis with ERM

$$g(\mathbf{x}) = \operatorname*{argmin}_{h \in \mathcal{H}} \widehat{R}_n(h) \quad g(\mathbf{x}) = \sum_{j=0}^{\bar{d}} w_j z_j$$

- classification $g(\Phi(\mathbf{x}), \mathbf{w}) = \text{sign}(\mathbf{w}^\top \Phi(\mathbf{x}))$
- regression $g(\Phi(\mathbf{x}), \mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x})$

Increased complexity of $\mathcal{H}$:

- Original VC dimension: $d_{VC} = d + 1$.
- New VC dimension: $d_{VC} \leqslant \bar{d} + 1$.

Learning with noisy data

- Data points ($\mathbf{x} \sim P(\mathbf{x})$):

$$(\mathbf{x}, y) \sim P(\mathbf{x}, y), \text{ that is } P(\mathbf{x}, y) = P(y|\mathbf{x})P(\mathbf{x})$$

- Target with noise (e.g. $\exists \mathbf{x}_a = \mathbf{x}_b : f(\mathbf{x}_a) \neq f(\mathbf{x}_b)$):

$$y = f(\mathbf{x}) + noise \text{ where } f(\mathbf{x}) = \mathbb{E}\left[y|\mathbf{x}\right] \text{ and } \mathbb{E}\left[noise|\mathbf{x}\right] = 0.$$

  Example: $y = w^\top x + N$ where $N \sim \mathcal{N}(0, \Sigma)$ is independent of $\mathbf{x}$.

- Deterministic target is a special case: $noise = 0$ and $P(y|\mathbf{x})$ is concentrated on the single point $f(\mathbf{x})$.

Learning problem

Choosing hypothesis class $\mathcal{H}$

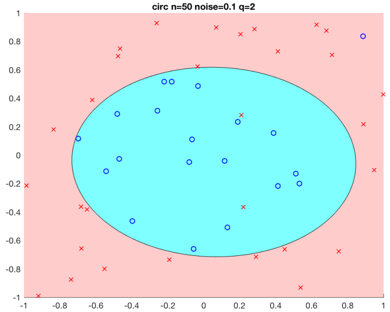
Learning $h(\mathbf{x}) = P(y|\mathbf{x})$.

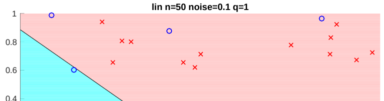
Learning with noisy data: generalisation, complexity, data size

Target : polynomial of degree q with noise $= 0.1$ and $n = 50$ samples

Target : polynomial of degree q with noise $= 0.1$ and $n = 1000$ samples

- Target A: $q = 1$ (linear)
- Target B: $q = 2$ (quadratic)

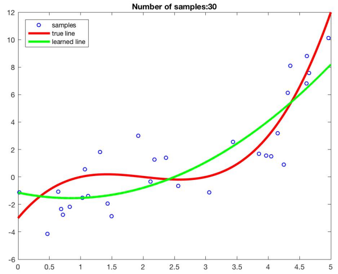


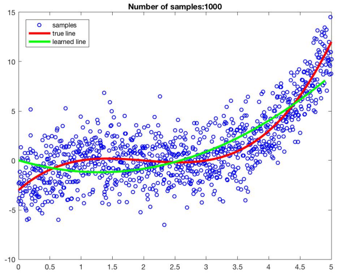


Learning with noisy data: generalisation, complexity, data size

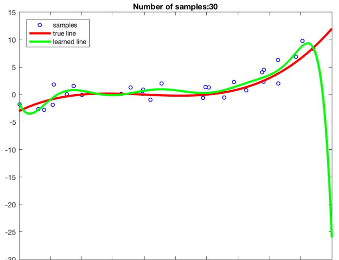
Fitting $f(x) = 0.5(x-1) * (x-2) * (x-3) + \mathcal{N}(0,4)$

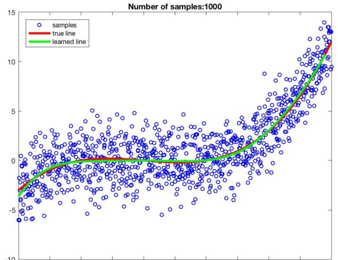
degree 2





degree 10

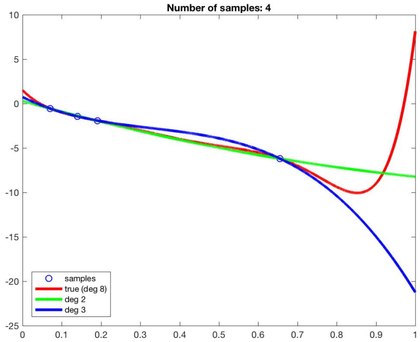




Learning with noisy data: generalisation, complexity, data size

- Target function: $f(x) = \sum_{k=0}^{q} a_k x^k + N$
  a polynomial of degree $q = 8$ on $[0, 1]$ (can fit any 9 data points)
- No noise: $N = 0$
- Data: $n = 4$ samples
- Hypothesis class: $\mathcal{H}_i$ of degree q_i ($q \sim$ complexity)

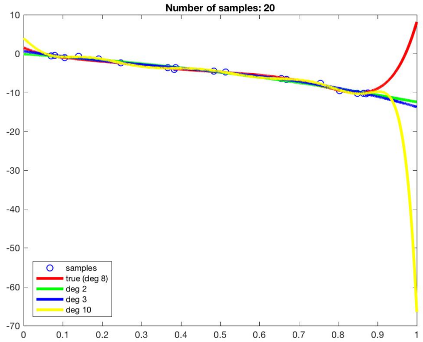
	$\mathcal{H}_1$	$\mathcal{H}_2$
	$q_1 = 2$	$q_2 = 3$
$\widehat{R}_4$	0.0013	0
R	5.97	28.47



Learning with noisy data: generalisation, complexity, data size

- Target function: $f(x) = \sum_{k=0}^{q} a_k x^k + N$
  a polynomial of degree $q = 8$ on $[0, 1]$ (can fit any 9 data points)
- Noise: $N \sim \mathcal{N}(0, \sigma^2 = 0.25)$
- Data: $n = 20$ samples
- Hypothesis class: $\mathcal{H}_i$ of degree q_i ($q \sim$ complexity)

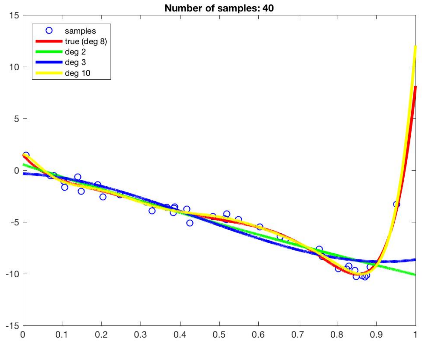
	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{H}_3$
	$q_1 = 2$	$q_2 = 3$	$q_3 = 10$
$\widehat{R}_{20}$	0.12	0.084	0.022
R	10.66	12.28	79.72



Learning with noisy data: generalisation, complexity, data size

- Target function: $f(x) = \sum_{k=0}^{q} a_k x^k + N$
  a polynomial of degree $q = 8$ on $[0,1]$ (can fit any 9 data points)
- Noise: $N \sim \mathcal{N}(0, \sigma^2 = 0.25)$
- Data: $n = 40$ samples
- Hypothesis class: $\mathcal{H}_i$ of degree q_i ($q \sim$ complexity)

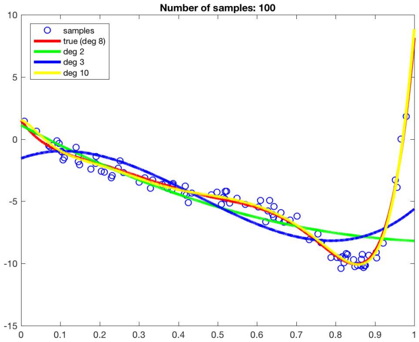
	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{H}_3$
	$q_1 = 2$	$q_2 = 3$	$q_3 = 10$
$\widehat{R}_{40}$	1.56	1.47	0.12
R	7.54	6.30	0.25



Learning with noisy data: generalisation, complexity, data size

- Target function: $f(x) = \sum_{k=0}^{q} a_k x^k + N$
  a polynomial of degree $q = 8$ on $[0,1]$ (can fit any 9 data points)
- Noise: $N \sim \mathcal{N}(0, \sigma^2 = 0.25)$
- Data: $n = 100$ samples
- Hypothesis class: $\mathcal{H}_i$ of degree q_i ($q \sim$ complexity)

	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{H}_3$
	$q_1 = 2$	$q_2 = 3$	$q_3 = 10$
$\widehat{R}_{100}$	3.29	2.76	0.18
R	5.88	4.33	0.03

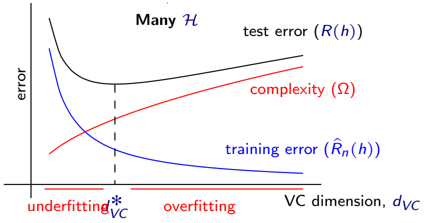


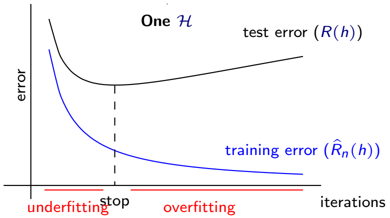
Learning with noisy data: overfitting

Overfitting

Occurs when $\widehat{R}_n(h) \downarrow \quad R(h) \uparrow$, moving away from the target.

Fitting to noise instead of the underlying target function/distribution.





Remember n matters too!

Noise types: stochastic $(N \sim \mathcal{N}(0, \sigma^2))$, deterministic (complexity)

deterministic noise	$\uparrow$	overfitting	$\uparrow$
stochastic noise	$\uparrow$	overfitting	$\uparrow$
number of data points	$\uparrow$	overfitting	$\downarrow$

Learning with noisy data: deterministic and stochastic noise

Target function: $y = f(\mathbf{x}) + N$ with $N \sim \mathcal{N}(0, \Sigma)$

Deterministic noise–pointwise bias: $f(\mathbf{x}) - h^*(\mathbf{x})$

- h^* is the best approximation to f in $\mathcal{H}$

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathbb{E}_x \left[\ell(h(\mathbf{x}), f(\mathbf{x})) \right]$$

- deterministic noise depends on $\mathcal{H}$ and f
- fixed for a given $\mathbf{x}$

Bias-variance decomposition :

$$\mathbb{E}_{\mathcal{D},N}\left[(g^{(\mathcal{D})}(\mathbf{x}) - y)^2 \right] = \mathbb{E}_{\mathcal{D},N}\left[(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x}) - N(\mathbf{x}))^2 \right]$$

$$= \underbrace{\mathbb{E}_{\mathcal{D}}\left[(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x}))^2) \right]}_{\text{var}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[(\bar{g}(\mathbf{x}) - f(\mathbf{x}))^2) \right]}_{\substack{\text{bias} \\ \text{deterministic noise}}} + \underbrace{\mathbb{E}_N\left[(N(\mathbf{x}))^2 \right]}_{\substack{\sigma^2 \\ \text{stochastic noise}}}$$

Learning with noisy data: Improving features

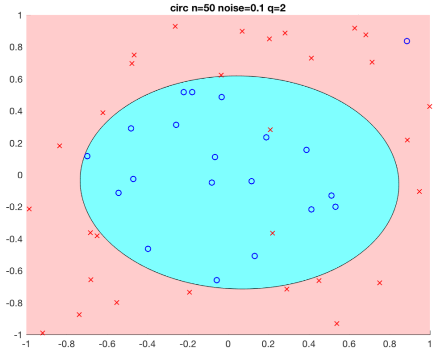
$$\mathbf{x} = (1, x_1, x_2) \rightarrow \mathcal{H}_1$$

$$\mathbf{z} = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2) \rightarrow \mathcal{H}_2$$

But why not $\mathbf{z} = (1, x_1^2, x_2^2) \rightarrow \mathcal{H}_3$

or even better $\mathbf{z} = (1, x_1^2 + x_2^2) \rightarrow \mathcal{H}_4$

or simply $\mathbf{z} = (x_1^2 + x_2^2 - 0.49) \rightarrow \mathcal{H}_5$



Data snooping

Incorporating prior knowledge is good, but

Looking at the data before choosing the model may be risky!

Learning with noisy data: Which $\mathcal{H} \sim \Phi(\mathbf{x})$ should we choose?

Given many hypotheses classes $\mathcal{H}_1, \mathcal{H}_2, \ldots$

Which $\mathcal{H}_i$ should we choose g from?

Use data to select g!

Structural Risk Minimization

Regularisation

Validation

Learning with noisy data: Structural Risk Minimization (SRM)

Structural Risk Minimization

- Hypothesis class: $\mathcal{H} = \cup_i \mathcal{H}_i$
- Generalization bound: for all i w.p. at least $1 - \delta$,

$$R(h) \leqslant \widehat{R}_n(h) + \Omega(n, \mathcal{H}_i, w_i \delta).$$

- SRM solution:

$$g = \operatorname{argmin}_{h \in \mathcal{H}} \widehat{R}_n(h) + \Omega(n, \mathcal{H}(h), w_i \delta)$$

  - $\mathcal{H}(h)$ is the simplest class h belongs to, and w_i is the class weight.
  - Training error $\widehat{R}_n(h)$ plus complexity term Ω.
  - Identical to ERM if there is only one class $\mathcal{H}$.

Example

$$\mathcal{H}_1 : \Phi_1(\mathbf{x}) = (1, x_1, x_2)$$

$$\mathcal{H}_2 : \Phi_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$$

$$\mathcal{H}_3 : \Phi_3(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, x_1^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, x_2^4)$$

From VC inequality:

$$\Omega(n, \mathcal{H}_i, w_i\delta) = \sqrt{\frac{8d_{VC}(\mathcal{H}_i)}{n} \log \frac{2ne}{d_{VC}(\mathcal{H}_i)} + \frac{8}{n} \log \frac{4}{w_i\delta}}$$

$$= \begin{cases} 1.298 & i = 1, n = 100 \\ 1.612 & i = 2, n = 100 \\ 2.178 & i = 3, n = 100 \end{cases}$$

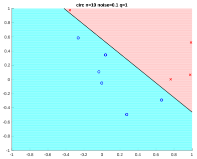
$\Omega(n, \mathcal{H}_i, \delta)$ is too large: $\Omega > 1$

Use $c\Omega(n, \mathcal{H}_i, \delta)$ instead, e.g., for $c = 0.1$

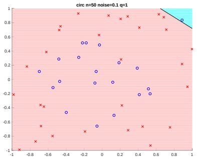
Use $\widehat{R}_n(h) + c\Omega(n, \mathcal{H}_i, \delta)$ to choose g.

Learning with noisy data: Structural Risk Minimization

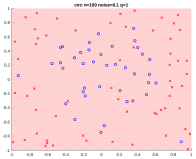
Columns $n \in \{8, 50, 100, 1000\}$, rows $q \in \{1, 2, 4\}$, % of trials selected by SRM, with train and test errors



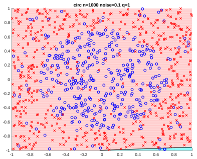
9% tr=0.189 te=0.481



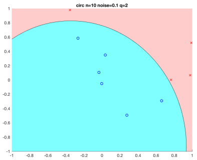
0% tr=0.332 te=0.462



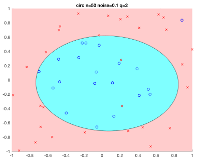
0% tr=0.364 te=0.44



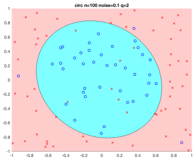
0% tr=0.402 te=0.417



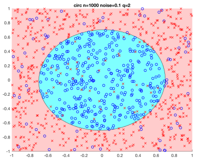
87% tr=0.004 te=0.334



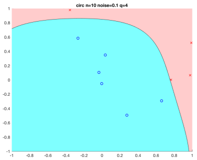
70% tr=0.085 te=0.185



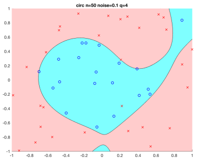
99% tr=0.097 te=0.150



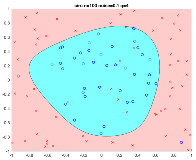
100% tr=0.106 te=0.110



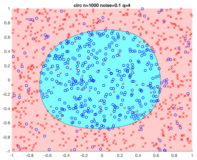
4% tr=0 te=0.328



30% tr=0.031 te=0.243



1% tr=0.084 te=0.186

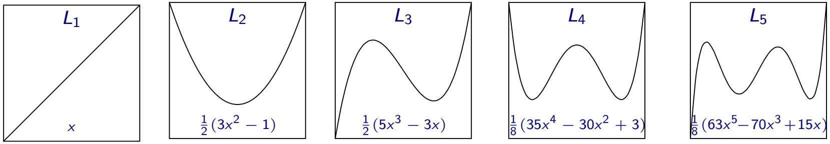


0% tr=0.111 te=0.119

Better features: Legendre polynomials

$$y = f(x) + N = \sum_{q=1}^{Q_f} a_q L_q(x) + N$$

- $L_q(x)$: Legendre polynomials, form an **orthogonal basis** for piecewise smooth functions on $[-1,1]$.
$$L_q(x) = \frac{1}{2^q q!}\frac{d^q}{dx^q}(x^2-1)^q \qquad \text{with} \qquad \mathbb{E}_x\left[L_q^2(x)\right] = \frac{2}{2q+1}$$
- $\frac{d^q}{dx^q}$: q-order derivative.
- a_q standard normal, normalized such that $\mathbb{E}_{a,x}\left[f(x)^2\right] = 1$.
- N zero-mean Gaussian noise, with variance σ^2
- Hypothesis class of degree Q, $\mathcal{H}_Q = \left\{\sum_{q=0}^{Q} w_q L_q(x)\right\}$



Constraining the hypothesis class $\longrightarrow$ regularisation!

Example

In an ML regression task you decided to use a non-linear transformation of the input data with polynomials. You observe that the training error is zero but the test error is much larger

- Give an example of a target function $f(x) = ?$, polynomial feature transformation $\mathbf{z} = \Phi(x) = \{\ldots\}?$ and error function $\widehat{R} = ?$ for the given scenario. In what specific training setting this may occur and how to improve the test error?
- Assume you fix the polynomial degree to $k = 3$. What is the approximate number of data points that are needed if you want to guarantee that the test error of the predictor is within 0.1 from the training error with probability 0.99? Show your calculations.

Example

Suppose that n data points are generated according to a polynomial of degree q $f(x) = y = ax^q + bx^{q-1} + \dots$, where x is a real feature $x \in \mathbb{R}$ value uniformly distributed on $[0, 1]$

y is the target variable

consider the squared error $\widehat{R} = \frac{1}{n} \sum_n (x - y)^2$.

Given $n > q$ data points, if we chose the hypothesis class to be polynomials degree $n - 1$:

$\Phi(x) = \{1, x, x^2, \dots, x^{n-1}\}$

the training error will be 0 (as any n points can be fitted properly), but the test error of the learned polynomial will be large as the complexity of the hypothesis class is too high.

To improve the test error reduce the polynomials degree, use Legendre polynomials, and apply regularisation.

Example

We need to choose n large enough so that $|R - \widehat{R}_n| \leqslant 0.1$ with probability at least 99%. The probability $P_{\cdot}() = 0.01$ of error larger than $\varepsilon = 0.1$ for the VC dimension $d_{VC} = k + 1 = 4$, since $\mathbf{z} = \Phi(x) = \{1, x, x^2, x^3\}$.

By VC's inequality we get

$$P\left(|R - \widehat{R}_n| > 0.1\right) \leqslant 4n^{d_{VC}} e^{-\varepsilon^2 n/8} = 4n^4 e^{-n \cdot 0.1^2/8} = 0.01.$$

trying e.g. $n \in \{1000, 10000, 50000\}$ one can quickly converge to 40000.

Regularisation: Linear regression

Minimize

$$\widehat{R}_n = \frac{1}{n}\sum_{i=1}^{n}(w^\top \mathbf{z}_i - y_i)^2$$
$$= \frac{1}{n}(Z\mathbf{w} - y)^\top (Z\mathbf{w} - y)$$

$$\textit{unconstrained solution} \Rightarrow \mathbf{w}_{lin} = (Z^\top Z)^{-1} Z^\top y.$$

Will overfit if complexity of $\mathbf{z} = \Phi(\mathbf{x})$ is too high (e.g. high degree q)!

Constrain the function class!

- Hard constraint: $\mathcal{H}_Q$: $w_q = 0$ for all $q > Q$! (how to choose Q?)

  E.g., $\mathcal{H}_2$ is a constrained version of $\mathcal{H}_{10}$.

- Soft constraint: $\sum_{i=0}^{Q} w_q^2 \leqslant C$.

  E.g. Small polyn. coeffs $\rightarrow$ function value varies little with varying input.

Regularisation: Solving constrained optimization, L_2 penalty

Minimize

$$\widehat{R}_n = \frac{1}{n}(Z\mathbf{w} - y)^\top (Z\mathbf{w} - y)$$

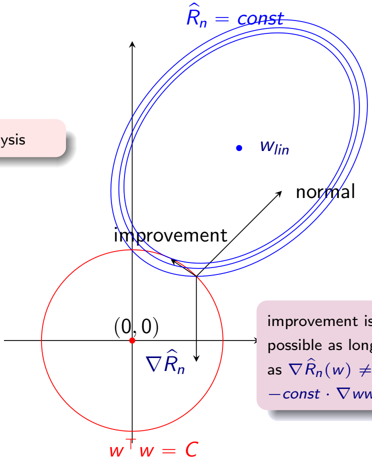
$$\text{subject to} \quad w^\top w \leqslant C$$

$\leftarrow$ Restricting $\mathcal{H} \leftarrow$ VC analysis

$\|\mathbf{w}\|_2^2 = \sum_i w_i^2 = w^\top w$

If $\mathbf{w}_{lin}^\top \mathbf{w}_{lin} \leqslant C$ then $g(\mathbf{z}, \mathbf{w}) = \mathbf{w}_{lin}^\top \mathbf{z}$

If $\mathbf{w}_{lin}^\top \mathbf{w}_{lin} > C$?



improvement is possible as long as $\nabla \widehat{R}_n(w) \neq -const \cdot \nabla ww^\top$

Loss: $\mathcal{L}_n(w) = \widehat{R}_n(w) + \frac{\lambda}{n}\|w\|_2^2 \quad (\lambda > 0)$

Lagrangian

Why $\frac{2\lambda}{n}$? $\widehat{R}_n = \frac{\mathbf{1}}{\mathbf{n}} \sum \ell(h(x), y)$

Optimum: $\nabla \widehat{R}_n(w_{reg}) = -\frac{2\lambda}{n} w_{reg}$

$$\nabla \widehat{R}_n(w_{reg}) + \frac{2\lambda}{n} w_{reg} = 0$$

Ridge regression solution

$$\lambda \downarrow \Leftrightarrow C \uparrow$$

Regularisation: Minimizing the Lagrangian

Minimize

$$\begin{aligned}\mathcal{L}_n(w) &= \widehat{R}_n(w) + \frac{\lambda}{n}\|w\|_2^2 \\ &= \frac{1}{n}(Zw - y)^\top (Zw - y) + \frac{\lambda}{n}\|w\|_2^2\end{aligned}$$

Setting $\nabla \mathcal{L}_n(w) = 0$

$$Z^\top (Zw - y) + \lambda w = 0$$

Optimal regularized weight – Ridge regression

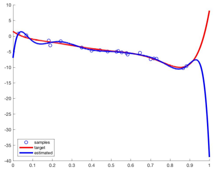
$$w_{reg} = (Z^\top Z + \lambda I)^{-1} Z^\top y$$

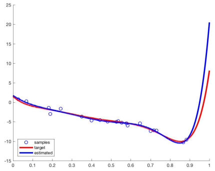
Unconstrained optimum

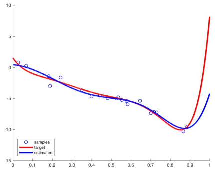
$$w_{lin} = (Z^\top Z)^{-1} Z^\top y$$

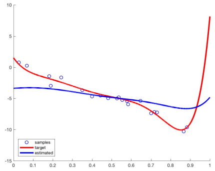
Regularisation: regression with Legendre polynomials

$\lambda = 0$ $\lambda = 0.0001$ $\lambda = 1$ $\lambda = 100$









overfitting $\longrightarrow$ $\longrightarrow$ underfitting

Regularization: Tikhonov

Minimize Loss:

$$\mathcal{L}_n(w) = \widehat{R}_n(w) + \lambda \|\mathbf{w}\|_Q^2$$

Instead of $\|\mathbf{w}\|_2^2 = \mathbf{w}^\top \mathbf{w}$ put emphasis on certain weights

$$\|\mathbf{w}\|_Q^2 = \sum_{q=0}^{Q} \gamma_q w_q^2$$

Examples:

- $\gamma_q = 2^q$ low-order fit
- $\gamma_q = 2^{-q}$ high order fit

General form: Tikhonov regularizer

Tikhonov regularizer: $\|\mathbf{w}\|_Q^2 = \mathbf{w}^\top \Gamma^\top \Gamma \mathbf{w}$, with $Q = \Gamma^\top \Gamma$,

where Γ is a matrix with weights (and cross term weights)

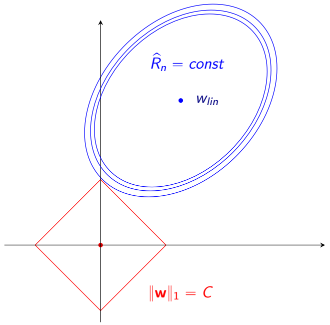
Regularisation: LASSO L_1 penalty

- Lasso: Least absolute shrinkage and selection operator
  - when $\mathbf{x} \in \mathbb{R}^d$ and $d \approx n$
  - need to reduce d to avoid overfitting
- Select useful dimensions d

  $\Rightarrow$ constrain $\|\mathbf{w}\|_0 = \sum_{j=1}^{d} \mathbb{I}\left(\mathbf{w}_i \neq 0\right)$

  Computationally very hard (non-convex)
- Instead, constrain $\|\mathbf{w}\|_1 = \sum_{j=1}^{d} |\mathbf{w}_j|$
  - Solve $\widehat{R}_n(\mathbf{w}) + \lambda\|\mathbf{w}\|_1$

The ellipse $\widehat{R}_n$ is likely to touch the corner of $\|\mathbf{w}\|_1 = C$ where a subset of dimensions is 0



Sparse solution!

Feature selection

Lasso

$$\begin{aligned} \text{minimize} \quad & \widehat{R}_n = \frac{1}{n}(Z\mathbf{w} - \mathbf{y})^\top(Z\mathbf{w} - \mathbf{y}) \\ \text{subject to} \quad & \|\mathbf{w}\|_1 \leqslant C \end{aligned}$$

Regularisation: Variants

Regularised Loss (Augmented Error)

$$\mathcal{L}_n(h) = \widehat{R}(h) + \lambda \underbrace{\Omega(h)}_{\textit{overfit penalty}}$$

VC inequality

$$R(h) \leqslant \widehat{R}(h) + \underbrace{\Omega(n, \mathcal{H}, \sigma)}_{\textit{complexity penalty}}$$

- $\mathcal{L}_n(h)$ is a better approximation of $R(h)$
- $\Omega(h)$ – regulariser
  - $\|\mathbf{w}\|_2^2$: ridge regression
  - $\|\mathbf{w}\|_1$: lasso
  - $\|\mathbf{w}\|_1 + \|\mathbf{w}\|_2^2$: elastic net
  - $\mathbf{w}^\top \Gamma^\top \Gamma \mathbf{w}$: Tikhonov
- Generalization bounds also hold

Occam's razor: Simpler is usually better

Regularize towards smoother, simpler functions. Why?

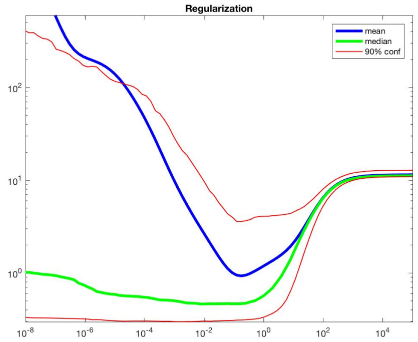
Because noise is not smooth!

How to select the regularization coefficient λ?

Noise types: stochastic ($N \sim \mathcal{N}(0, \sigma^2)$), deterministic (complexity e.g. Q)

1000 experiments of
polynomial fit with 20 points.

$R(w_\lambda)$



$\log(\lambda)$

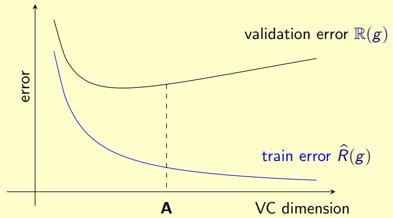
deterministic noise	$\uparrow$	overfitting	$\uparrow$
stochastic noise	$\uparrow$	overfitting	$\uparrow$
number of data points	$\uparrow$	overfitting	$\downarrow$
λ	$\uparrow$	overfitting	$\downarrow$

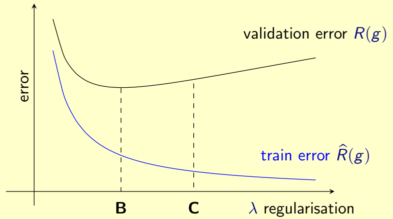
From regularisation point of view, deterministic noise behaves same as the stochastic noise

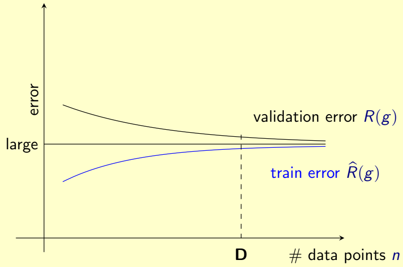
How to select λ ?Validation!

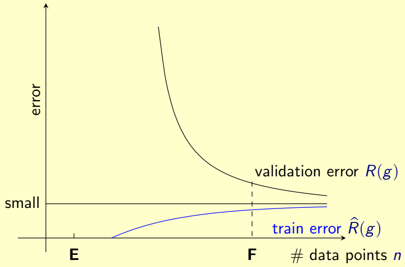
Example

Figure shows various learning curves that you may observe in your ML experiment. Discuss the main observations at each of the points A, B, C, D, E, and F in the figures and propose a change in the setup to improve learning (1-2 sentences for each point).









Example

- A Complexity of the hypothesis class is too high (overfitting), reduce the complexity or increase the number of data points.
- B λ gives optimal regularisation for the given learning setup. Nothing to do.
- C λ too high, regularization too strong (underfitting), reduce the value of λ
- D Complexity of the hypothesis class is too low (underfitting) hence large error even with large number of data points. Use different hypothesis class with higher VC dimension.
- E the number of data points is to low, hence overfitting, increase the number of data points.
- F the number of data points could be increased otherwise the learning seems effective.

Part 3 summary

- Non-linear feature transform: polynomial, Legendre
- Overfitting/underfitting: match the hypothesis class to data
- Structural risk minimization
- Regularisation: L_2, L_1 ...
- Validation