Predictive modelling

Prediction function h : predictor $\underline{x} \longrightarrow$ target y

Parametrised: $h_{\underline{\lambda}}(\underline{x}; \underline{\theta})$

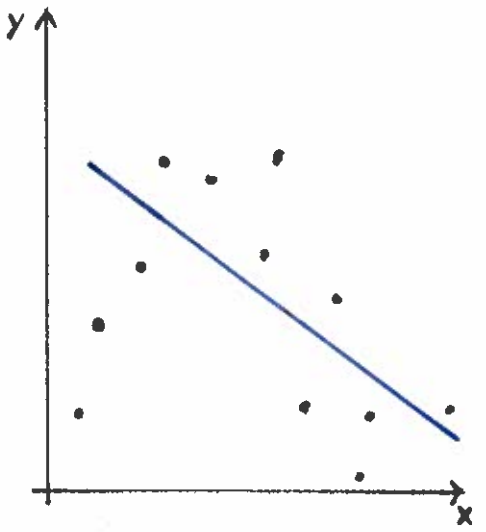
hyperparameters — tuning parameters

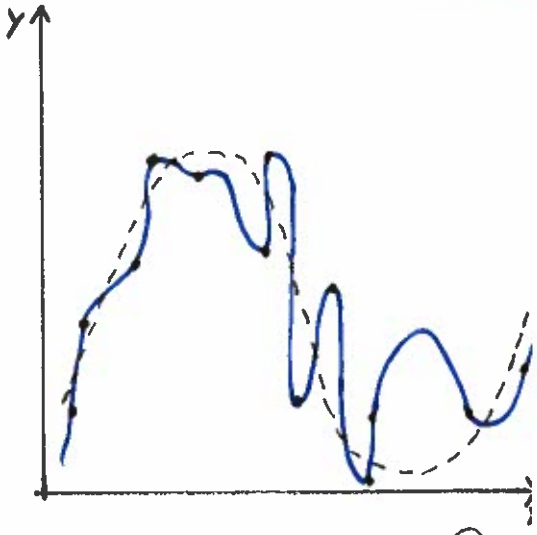
Training loss : $J_{\underline{\lambda}}(\underline{\theta}) = \frac{1}{n} \sum_{i=1}^{n} L(h_{\underline{\lambda}}(\underline{x}_i; \underline{\theta}), y_i)$

Often: $\underline{\lambda}$ optimised in outer loop

$\underline{\theta}$ optimised in inner loop (gradient descen

Typically, variance and mean of training loss decrease for increasing model complexity





Generalisation performance

How well does the model perform on unseen data?

Estimate prediction loss for a given prediction function $\hat{h}$

$$J(\hat{h}) = \mathbb{E}_{\underline{x},y}\left[\mathcal{L}(\hat{h}(\underline{x}), y)\right]$$

"generalisation loss" or "test loss"

depends on our training data through $\hat{h}$:

$\hat{h} = \mathcal{A}(D^{train})$

Algorithm $\mathcal{A}$ turns training data into a prediction function (can have hyperparam.)

$\rightsquigarrow$ expected prediction loss for algorithm $\mathcal{A}$:

$$\bar{J}(\mathcal{A}) = \mathbb{E}_{D^{train}}[J(\hat{h})]$$

$$= \mathbb{E}_{D^{train}}[J(\mathcal{A}(D^{train}))]$$

Overfitting and underfitting

- Model overfitting training data:
  reducing model complexity / flexibility
  reduces the prediction loss

Extreme example:

n free parameters for n training data points

$$h_{\text{flexible}}(\underline{x}; \underline{\theta}) = \begin{cases} \theta_i & \text{if } \underline{x} = \underline{x}_i \\ 0_i & \text{otherwise} \end{cases}$$

Set $\hat{\theta}_i = y_i \rightsquigarrow$ training loss 0

- Model underfitting training data:
  increasing model complexity / flexibility
  reduces the prediction loss

Extreme example:

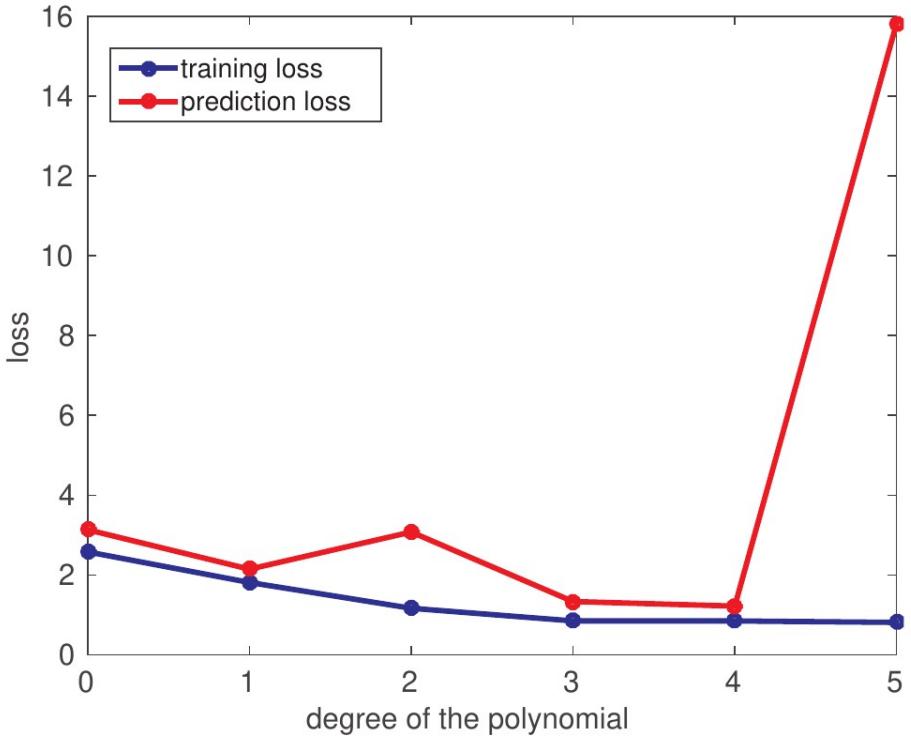
$h_{\text{rigid}}(\underline{x}; \theta) = \theta$ "constant"

Training Loss and Prediction Loss: Complexity

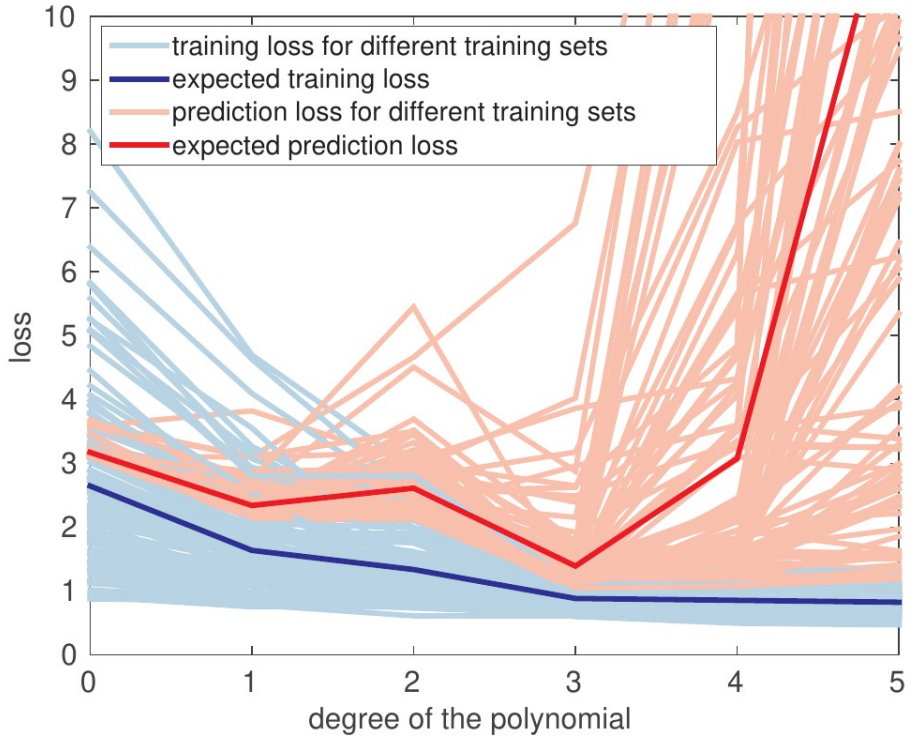
$$p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right),$$

$$p(y|x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(y - g(x))^2\right), \qquad g(x) = \frac{1}{4}x + \frac{3}{4}x^2 + x^3 \qquad \mathcal{L}(\hat{y}, y) = (\hat{y} - y)^2$$

Polynomial regression: $h_\lambda(x; \boldsymbol{\theta}) = \sum_{k=0}^{\lambda} \theta_k x^k$



(a) Single training set



(b) Distribution over training sets

Use model selection or regularisation to avoid overfitting and underfitting

Regularisation:

- Introduce penalty for model flexibility
- Augment training loss function:

$$\underset{\underline{\theta}}{\text{minimise}} \; J_{\underline{\lambda}}(\underline{\theta}) + \lambda_{reg} R(\underline{\theta})$$

where $R(\underline{\theta})$: parameter penalty term

λ_{reg}: strength of regularisation (hyperparameter)

Examples:

$R(\underline{\theta}) = \sum_i \theta_i^2$ "L_2" or "Tikhonov" regular.

$R(\underline{\theta}) = \sum_i |\theta_i|$ "L_1" regularisation

Estimating generalisation performance

- Held-out:

Idea to use a separate dataset to estimate the prediction loss

Typically: given single dataset D

→ randomly split D into D^{train} "training set" and $\tilde{D}$ "test set" or "validation set"

Different values of target variables should be balanced in D^{train} and $\tilde{D}$

Obtain $\hat{h} = \mathcal{A}(D^{train})$ from training data.

Estimate prediction loss

$$J(\hat{h}) \simeq \hat{J}(\hat{h}; \tilde{D}) = \frac{1}{n} \sum_{i=1}^{\hat{n}} \mathcal{L}(\hat{h}(\tilde{x}_i), \tilde{y}_i)$$

estimate can vary strongly

- Cross validation:

Divide dataset into K subsets

$D_1, \dots, D_K$ "folds"

$\rightsquigarrow$ K pairs of training sets and validation sets: $D_k^{train} = \bigcup_{i \neq k} D_i$, $D_k^{val} = D_k$

$\rightsquigarrow$ K prediction functions $\hat{h}_k = \mathcal{A}(D_k^{train})$

K prediction loss estimates $\hat{J}_k = \hat{J}(\hat{h}_k, D_k^{val})$

Overall score: $CV = \frac{1}{K} \sum_{k=1}^{K} \hat{J}_k$

is an estimate for $\bar{J}(\mathcal{A})$, not $\hat{J}$:

$\hat{\bar{J}}(\mathcal{A}) = CV$

Special case $K = n$: "Leave-one-out cross validation" (LOOCV)

CV depends on dataset and on particular split

$\rightsquigarrow$ CV is a random variable 2020 L13 (6)

Rough estimate of variability:

$Var[CV] \approx \frac{1}{K} Var[\hat{J}]$ where $Var[\hat{J}] \approx \frac{1}{K} \sum_{k=1}^{K} (\hat{J}_k - CV)^2$

5-Fold Cross Validation