

Machine Learning

Section 10: Model selection & hyperparameter tuning

SLIDES BY Stefan Harmeling

(-1)st of 10 November 2021
2

Overview

- ▶ Evaluating on test error / cross-validation
- ▶ Bayesian model selection

Evaluating on test error / cross validation

Example: linear regression

Data:

- ▶ dataset \mathcal{D} of scalar-valued input/output pairs $(x_1, y_1) \dots, (x_n, y_n)$
- ▶ the outputs are noisy

Model:

- ▶ basis function $\phi_d(x) = [1, x, x^2, \dots, x^d]^T$ with hyperparameter d
- ▶ model the function as $f(x, w) = \phi_d(x)^T w$, with parameter $w \in \mathbb{R}^{d+1}$

Loss and fit:

- ▶ mean-squared error

$$\text{MSE}(\mathcal{D}, w, d) = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} (y - f(x, w))^2 = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} (y - \phi_d(x)^T w)^2$$

- ▶ minimize the MSE (to get the maximum likelihood estimator):

$$w_{ML}(\mathcal{D}, d) = \arg \min_w \text{MSE}(\mathcal{D}, w, d)$$

Model selection / hyperparameter tuning

- ▶ What is the best choice for d ?

Hyperparameter tuning: 1st attempt

Idea:

- ▶ choose d such that the MSE is minimal:

$$d_1 = \arg \min_d \text{MSE}(\mathcal{D}, w_{ML}(\mathcal{D}, d), d)$$

Problem:

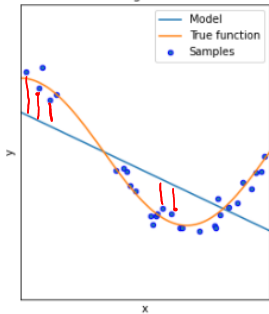
- ▶ this overfits the data
- ▶ the best fit will be a large d , such that every single noisy data point is hit
- ▶ however, actually, we would like to minimize the *test error* which is the MSE on unseen data
- ▶ idea: split data into seen (training) and unseen (evaluation) data

```
def phi(x, d):  
    return np.dstack([np.power(x, i) for i in range(d+1)]).squeeze()
```

```
def poly_regression_plot(x, y, d):  
    # perform regression analysis:  
    transformed_x = phi(x, d)  
  
    linear_regressor = LinearRegression()  
    linear_regressor.fit(transformed_x, y)  
    Y_pred = linear_regressor.predict(transformed_x)
```

Capacity of model

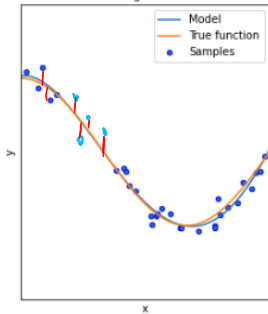
Degree 1



big MSE

training error.
large
underfitting

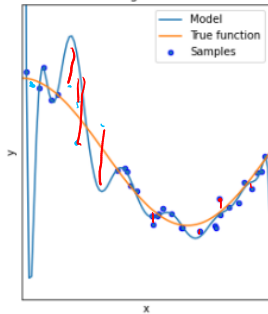
Degree 4



small MSE

medium

Degree 15



smallest MSE

small
overfitting

Aside: Polynomial regression will always lead $MSE = 0$ for high enough degree

$$\underbrace{\begin{pmatrix} x_1^d & x_1^{d-1} & \dots & x_1 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{d+1}^d & x_{d+1}^{d-1} & \dots & x_{d+1} & 1 \end{pmatrix}}_{=: A} \begin{pmatrix} w_1 \\ \vdots \\ w_{d+1} \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_{d+1} \end{pmatrix}$$

$$\det(A) = \prod_{i < j} (x_i - x_j) \neq 0 \quad \text{if all } x_i \text{ different}$$

\leadsto always solvable...

Hyperparameter tuning: 2nd attempt

Idea:

- ▶ split dataset \mathcal{D} into two disjoint sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{eval}}$
- ▶ 1. *train* on $\mathcal{D}_{\text{train}}$ (choose parameter w)
2. *evaluate* on $\mathcal{D}_{\text{eval}}$ (choose hyperparameter d)
- ▶ choose d such that the MSE on the evaluation set is minimal for the parameter w chosen on the training set:

$$d_2 = \arg \min_d \text{MSE}(\mathcal{D}_{\text{eval}}, w_{ML}(\mathcal{D}_{\text{train}}, d), d)$$

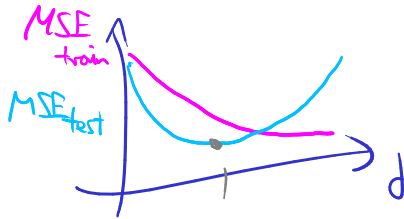
Note:

- ▶ much better! works quite well!
- ▶ no overfitting while learning the parameter

Problems/challenges:

- ▶ how good is the parameter/hyperparameter choice on unseen data? i.e. what is the test error (the MSE on unseen data)?
- ▶ the estimator $\text{MSE}(\mathcal{D}_{\text{eval}}, w_{ML}(\mathcal{D}_{\text{train}}, d_2), d_2)$ of the MSE is too optimistic (this is similar to underestimating the variance with a sample mean)

Typical evaluation diagram:



$$\arg \min_d MSE(D_{test}, d)$$

Data splitting:

```
1 from sklearn.model_selection import train_test_split
2
3 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

Grid search:

for i in range(500):

$$X_{\text{train}} = [1, X, X^2, \dots, X^i]$$

$$\text{linear_model.fit}(X_{\text{train}}, Y_{\text{train}})$$

$$\text{MSE}[i] = \frac{1}{|X_{\text{test}}|} (\text{linear_model.predict}(X_{\text{test}}) - Y_{\text{test}})^2 \cdot \text{sum}()$$

Pick $d = \underset{i}{\text{argmin}} \text{MSE}[i]$

Can have several hyperparameters:

E.g. degree d

$$w_{\text{ridge}} = \operatorname{argmax}_w p(w|X, y) = \operatorname{argmax}_w p(y|X, w)p(w|X)/p(y|X) \\ = \operatorname{argmax}_w p(y|X, w)p(w)$$

$$= \operatorname{argmax}_w \sum_{i=1}^n \log \mathcal{N}(y_i | x_i^T w, \sigma^2) + \sum_{j=1}^d \log \mathcal{N}(w_j | 0, \tau^2)$$

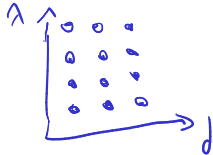
$$= \operatorname{argmin}_w \underbrace{\frac{1}{n} \sum_{i=1}^n (y_i - x_i^T w)^2}_{\text{fit}} + \underbrace{\lambda \|w\|_2^2}_{\text{regularizer}}$$

RIDGE

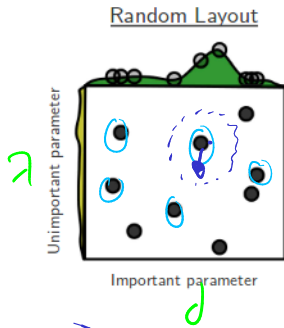
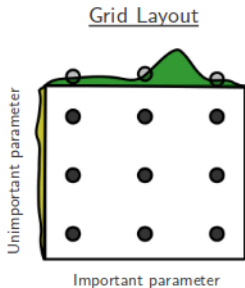
REG.

... and regularization constant λ

\rightsquigarrow Grid search



Grid search vs. random search:



(Stolen by Bengio 2012)

Alternative for continuous hyperparameters λ :

- Write function for test error
- do "automatic differentiation"
- follow the gradient to lower error

Hyperparameter tuning: 3rd attempt

Idea:

- ▶ split dataset \mathcal{D} into three disjoint sets $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{eval}}$ and $\mathcal{D}_{\text{test}}$
- ▶
 1. *train* on $\mathcal{D}_{\text{train}}$ (choose parameter w)
 2. *evaluate* on $\mathcal{D}_{\text{eval}}$ (choose hyperparameter d)
 3. *test* of $\mathcal{D}_{\text{test}}$ (calculate MSE for unseen data)
- ▶ choose d such that the MSE on the evaluation set is minimal for the parameter w chosen on the training set:

$$d_3 = \arg \min_d \text{MSE}(\mathcal{D}_{\text{eval}}, w_{ML}(\mathcal{D}_{\text{train}}, d), d)$$

- ▶ calculate the test error (the MSE on unseen data) on the test set:

$$\text{MSE}(\mathcal{D}_{\text{test}}, w_{ML}(\mathcal{D}_{\text{train}}, d_3), d_3)$$

Note:

- ▶ this is the gold standard (if we have enough data)!
- ▶ no overfitting of the parameter or of the hyperparameter

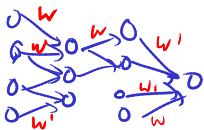
Why the final test ?

- estimate success of modelling
- model selection

E.g. Could have

- polynomial regression
- trigonometric regression
- neural network

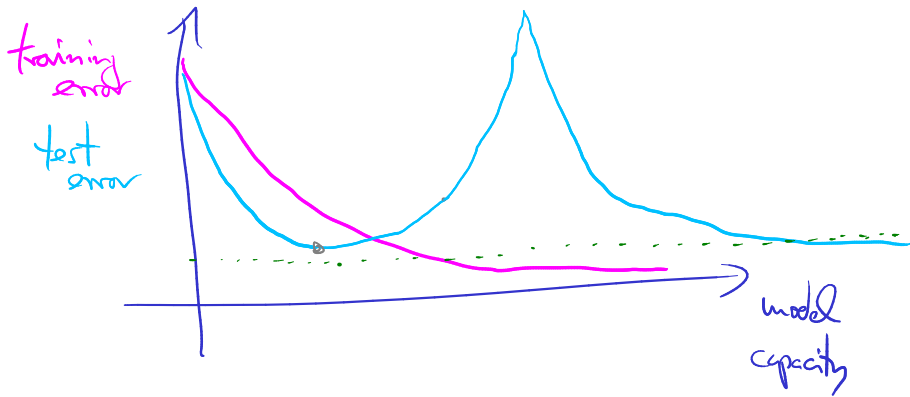
$$\sum_{n=1}^k w_n \sin(n \cdot \pi \cdot x)$$



If training (i.e. optimizing w)
is costly: (e.g. neural networks)

"epochs"	Grid ptr.
1	64
16	16
32	4
64	1

Double descent



Regularization against overfitting:

$$= \operatorname{argmin}_w \underbrace{\frac{1}{n} \sum_{i=1}^n (y_i - x_i^T w)^2}_{\text{fit}} + \underbrace{\lambda \|w\|_2^2}_{\text{regularizer}}$$

$$(\lambda_1, \dots, \lambda_n) \cdot \begin{pmatrix} |w_0| \\ \vdots \\ |w_n| \end{pmatrix}$$

$$\downarrow$$
$$\sum_{i=1}^n \lambda_i |w_i|$$

Hyperparameter tuning: 3rd attempt - more thoughts

Idea:

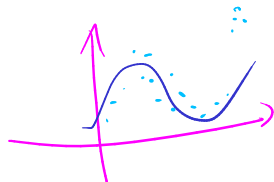
- ▶ split dataset \mathcal{D} into three disjoint sets $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{eval}}$ and $\mathcal{D}_{\text{test}}$
- ▶ 1. *train* on $\mathcal{D}_{\text{train}}$ (choose parameter w)
- ▶ 2. *evaluate* on $\mathcal{D}_{\text{eval}}$ (choose hyperparameter d)
- ▶ 3. *test* of $\mathcal{D}_{\text{test}}$ (calculate MSE for unseen data)

Problems

- ▶ we are not using all data for estimating the parameter
- ▶ we are not using all data for estimating the hyperparameter

Hack/Solution

- ▶ Cross-validation (see next slide)
- ▶ does improve the estimate



Cross-validation (2-fold)

Simpler setup w/o hyperparameter

- ▶ given data \mathcal{D}
- ▶ model with parameter w
- ▶ but no hyperparameter d

Goal

- ▶ estimate the test error

Solution

- ▶ split \mathcal{D} into two sets \mathcal{D}_1 and \mathcal{D}_2
 1. train w_1 on \mathcal{D}_1 , estimate MSE ε_1 on \mathcal{D}_2 for w_1
 2. train w_2 on \mathcal{D}_2 , estimate MSE ε_2 on \mathcal{D}_1 for w_2
- ▶ the cross-validation estimate of the test error is the average:

$$(\varepsilon_1 + \varepsilon_2)/2$$

More general:

Algorithm 10.1 (k -fold cross validation (CV))

Split your data \mathcal{D} into k disjoint subsets of approximately equal size:

$$\mathcal{D} = \mathcal{D}_1 \cup \dots \cup \mathcal{D}_k$$

For $i = 1$ to k

- 1. \mathcal{D}_i plays the role of the test set*
- 2. train parameter w on $\mathcal{D} \setminus \mathcal{D}_i$ (i.e. “ \mathcal{D} without \mathcal{D}_i ”, set minus)*
- 3. calculate MSE ε_i on \mathcal{D}_i*

The cross-validation estimate of the test error ε is the average validation error:

$$\varepsilon_{CV} = \frac{1}{k} \sum_{i=1}^k \varepsilon_i$$

For $k = n$ this is called leave-one-out cross validation (LOOCV).
= nr. of data points

Nested cross validation for the gold-standard

Setup

- ▶ given some data set \mathcal{D}
- ▶ model with parameter w and hyper-parameter d .

Goal:

- ▶ goal: estimate test error

Outer k -fold cross-validation

1. split \mathcal{D} into $\mathcal{D}_1, \dots, \mathcal{D}_k$
2. for i in $1 \dots k$:
 - 2.1 split \mathcal{D} into $\mathcal{D} \setminus \mathcal{D}_i$ and \mathcal{D}_i
 - 2.2 train w and d on $\mathcal{D} \setminus \mathcal{D}_i$
use **Inner k' -fold cross-validation** to estimate the MSE for the hyperparameter choice
 - 2.3 estimate ε_i on \mathcal{D}_i using w and d
3. estimate test error ε as the average of $\varepsilon_1, \dots, \varepsilon_k$

Bayesian model selection

Bayesian inference

Model specification

$$p(w) = \dots$$

prior

$$p(D|w) = \dots$$

likelihood

Inference given data

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

posterior

$$p(D) = \int p(D|w)p(w)dw$$

evidence

- ▶ The *posterior*
 - ▶ combines prior and likelihood via product rule (Bayes rule)
 - ▶ describes what the parameter might be after seeing data
- ▶ The *evidence*
 - ▶ integrates out the parameter via sum rule
 - ▶ is the expected likelihood under the prior

Bayesian inference with hyper-parameters

Model specification

$$p(\theta) = \dots \quad \text{hyper-prior (level 2)}$$

$$p(w|\theta) = \dots \quad \text{prior (level 1)}$$

$$p(D|w) = \dots \quad \text{likelihood}$$

Level 1: Inference of parameters

$$p(w|D, \theta) = \frac{p(D|w)p(w|\theta)}{p(D|\theta)} \quad \text{posterior}$$

$$p(D|\theta) = \int p(D|w)p(w|\theta)dw \quad \text{evidence}$$

Level 2: Inference of hyper-parameters

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \quad \text{hyper-posterior}$$

$$p(D) = \int p(D|\theta)p(\theta)d\theta \quad \text{hyper-evidence}$$

From Murphy, Machine Learning, A probabilistic perspective

5.3. Bayesian model selection

157

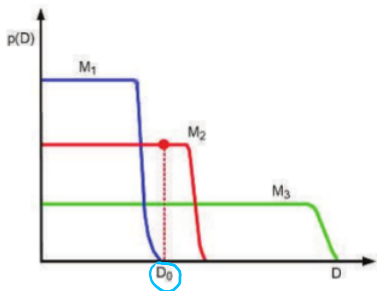


Figure 5.6 A schematic illustration of the Bayesian Occam's razor. The broad (green) curve corresponds to a complex model, the narrow (blue) curve to a simple model, and the middle (red) curve is just right. Based on Figure 3.13 of (Bishop 2006a). See also (Murray and Ghahramani 2005, Figure 2) for a similar plot produced on real data.

McKay: More precise derivation of Occam's razor phenomenon.

Bayesian inference with hyper-parameters and models

Model specification

$p(\mathcal{H}) = \dots$	model-prior (level 3)
$p(\theta \mathcal{H}) = \dots$	hyper-prior (level 2)
$p(w \theta, \mathcal{H}) = \dots$	prior (level 1)
$p(D w, \mathcal{H}) = \dots$	likelihood

Level 1, 2, 3: Inference of parameters, hyper-parameters and model given data

$p(w D, \theta, \mathcal{H}) = \frac{p(D w, \mathcal{H})p(w \theta, \mathcal{H})}{p(D \theta, \mathcal{H})}$	posterior
$p(\theta D, \mathcal{H}) = \frac{p(D \theta, \mathcal{H})p(\theta \mathcal{H})}{p(D \mathcal{H})}$	hyper-posterior
$p(\mathcal{H} D) = \frac{p(D \mathcal{H})p(\mathcal{H})}{p(D)}$	model-posterior

Bayesian inference with hyper-parameters and models

Level 1: Inference of parameters

$$p(w | D, \theta, \mathcal{H}) = \frac{p(D | w, \mathcal{H})p(w | \theta, \mathcal{H})}{p(D | \theta, \mathcal{H})} \quad \text{posterior}$$

$$p(D | \theta, \mathcal{H}) = \int p(D | w, \mathcal{H})p(w | \theta, \mathcal{H})dw \quad \text{marginal likelihood}$$

Level 2: Inference of hyper-parameters

$$p(\theta | D, \mathcal{H}) = \frac{p(D | \theta, \mathcal{H})p(\theta)}{p(D | \mathcal{H})} \quad \text{hyper-posterior}$$

$$p(D | \mathcal{H}) = \int p(D | \theta, \mathcal{H})p(\theta | \mathcal{H})d\theta$$

Level 3: Inference of model

$$p(\mathcal{H} | D) = \frac{p(D | \mathcal{H})p(\mathcal{H})}{p(D)} \quad \text{model-posterior}$$

$$p(D) = \sum_{\mathcal{H}} p(D | \mathcal{H})p(\mathcal{H})$$

Model selection for regression

Instead of a model for $p(D)$ we specify a model for $p(y | X)$ with location matrix X and targets y .

Bayesian model selection for regression

Model specification

$p(\mathcal{H}) = \dots$	model-prior
$p(\theta \mathcal{H}) = \dots$	hyper-prior
$p(w \theta, \mathcal{H}) = \dots$	prior
$p(y X, w, \mathcal{H}) = \dots$	likelihood (this is different)

Level 1, 2, 3: Inference of parameters, hyper-parameters and model given data

$p(w y, X, \theta, \mathcal{H}) = \frac{p(y X, w, \mathcal{H})p(w \theta, \mathcal{H})}{p(y X, \theta, \mathcal{H})}$	posterior
$p(\theta y, X, \mathcal{H}) = \frac{p(y X, \theta, \mathcal{H})p(\theta \mathcal{H})}{p(y X, \mathcal{H})}$	hyper-posterior
$p(\mathcal{H} y, X) = \frac{p(y X, \mathcal{H})p(\mathcal{H})}{p(y X)}$	model-posterior

Bayesian model selection for Linear Regression

Level 1:

$$\begin{aligned}w &\sim \mathcal{N}(0_d, \tau^2 I_d) \\ y | X, w &\sim \mathcal{N}(Xw, \sigma^2 I_n)\end{aligned}$$

We infer posterior:

$$w | X, y \sim \mathcal{N}(w_n, V_n) \quad \text{posterior}$$

with posterior mean w_n and posterior covariance V_n .

$$\begin{aligned}V_n &= \left(\frac{1}{\sigma^2} X^T X + \frac{1}{\tau^2} I_d \right)^{-1} \\ w_n &= (X^T X + \frac{\sigma^2}{\tau^2} I_d)^{-1} X^T y\end{aligned}$$

Note that on this slide σ^2 and τ^2 is assumed to be known (constant).

Level 2: Let's add random variables for the hyper-parameters!

Bayesian model selection for Linear Regression

Level 1:

$$\theta \sim \dots$$

hyper-prior

$$w | \theta \sim \mathcal{N}(0_d, \tau^2 I_d)$$

prior

$$y | X, w, \theta \sim \mathcal{N}(Xw, \sigma^2 I_n)$$

likelihood

where $\theta = (\sigma^2, \tau^2, d)$ collects all hyperparameter.

We infer posterior:

$$w | X, y, \theta \sim \mathcal{N}(w_n, V_n | \theta)$$

posterior

with posterior mean w_n and posterior covariance V_n which do dependent on θ .

Level 2: We need the evidence from level 1:

$$p(y | X, \theta) = \int p(y | X, w, \theta) p(w | \theta) dw = \mathcal{N}(\dots)$$

This is also called *marginal likelihood*.

Bayesian inference for level 2

Level 2: Infer hyper-posterior:

$$p(\theta | y, X) = \frac{p(y | X, \theta)p(\theta)}{p(y | X)}$$

This gets quickly too complicated! (the integral become to difficult)

Instead we only maximize the **Marginal likelihood**

$$p(y | X, \theta) = \int p(y | X, w, \theta)p(w | \theta)dw = \mathcal{N}(\dots)$$

with respect to the hyper-parameter θ .

- ▶ this is called “Type II maximum likelihood” and “empirical Bayes” and “generalized maximum likelihood” and “evidence approximation”

Marginal likelihood for linear regression

We can derive the form of the logarithm of the marginal likelihood:

$$\begin{aligned}\log p(y|X, \theta) &= \log \int p(y|X, w, \theta) p(w|\theta) dw \\&= \log \int \mathcal{N}(y|Xw, \sigma^2 I_n) \mathcal{N}(w|0_d, \tau^2 I_d) dw \\&= \log \int \mathcal{N}(y|0_n, \sigma^2 I_n + \tau^2 XX^T) \mathcal{N}(w|\dots, \dots) dw \\&= \log \mathcal{N}(y|0_n, \sigma^2 I_n + \tau^2 XX^T) \int \mathcal{N}(w|\dots, \dots) dw \\&= \log \mathcal{N}(y|0_n, \sigma^2 I_n + \tau^2 XX^T) \\&= -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log|\sigma^2 I_n + \tau^2 XX^T| - \frac{1}{2} y^T (\sigma^2 I_n + \tau^2 XX^T)^{-1} y\end{aligned}$$

Notes:

- ▶ $\mathcal{N}(w|\dots, \dots)$ does not depend on y .
- ▶ The third equality follows from some fancy rule for the Gaussians (see next slide).

Another product formula for Gaussians

copied from Bishop's book Sec. 2.3.3, page 93

Given a marginal Gaussian for x and a conditional Gaussian for y given x ,

$$x \sim \mathcal{N}(\mu, \Lambda^{-1})$$
$$y|x \sim \mathcal{N}(Ax + b, L^{-1})$$

we have the following marginal Gaussian for y and conditional Gaussian for x given y :

$$y \sim \mathcal{N}(A\mu + b, L^{-1} + A\Lambda^{-1}A^T)$$
$$x|y \sim \mathcal{N}(\Sigma(A^T L(y - b) + \Lambda\mu), \Sigma)$$

where $\Sigma = (\Lambda + A^T L A)^{-1}$

End of Section 10

Literature on Bayesian model selection / linear regression:

- ▶ Chapter 3.5 in Bishop's book "Pattern Recognition and Machine learning"

▶ Murphy 5.3

▶ McKay