Machine Learning

A. Supervised Learning
A.3. Regularization

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Outline

1. The Problem of Overfitting

2. Model Selection

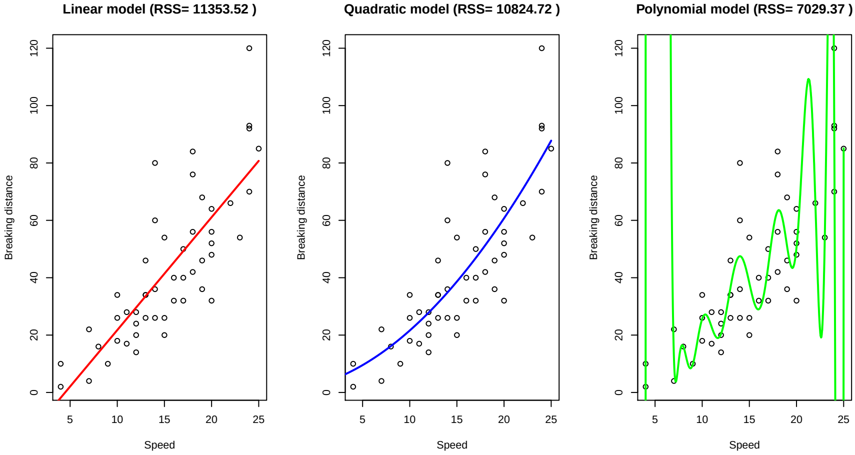
3. Regularization

4. Hyperparameter Optimization

Outline

1. The Problem of Overfitting
2. Model Selection
3. Regularization
4. Hyperparameter Optimization

Fitting of models



Underfitting/Overfitting

Underfitting: The model is not complex enough to explain the data well.This results in poor predictive performance.

Overfitting: The model is too complex, it describes the noise instead of the underlying relationship between the variables. Similarly to underfitting, this results in poor predictive performance.

Remark: Given n points (x_i, y_i) without repeated measurements (i.e. $x_i \neq x_j, \;\; i \neq j$), a polynomial of degree $n-1$ exists such that the RSS equals 0.

Outline

1. The Problem of Overfitting

2. Model Selection

3. Regularization

4. Hyperparameter Optimization

Model Selection Measures

Model selection means: we have a set of models, e.g.,

$$Y = \sum_{i=0}^{p-1} \beta_i X_i$$

indexed by p (i.e., one model for each value of p), make a choice which model **describes** the data best.

If we just look at **losses** / **fit measures** such as RSS, then

the larger p, the better the fit

or equivalently

the larger p, the lower the loss

as the model with p parameters can be **reparametrized** in a model with $p' > p$ parameters by setting

$$\beta'_i = \begin{cases} \beta_i, & \text{for } i \leq p \\ 0, & \text{for } i > p \end{cases}$$

Model Selection Measures

One uses **model selection measures** of type

$$\text{model selection measure} = \text{fit} - \text{complexity}$$

or equivalently

$$\text{model selection measure} = \text{loss} + \text{complexity}$$

The smaller the loss (= lack of fit), the better the model.

The smaller the complexity, the simpler and thus better the model.

The model selection measure tries to find a trade-off between fit/loss and complexity.

Model Selection Measures

Akaike Information Criterion (AIC): (maximize)

$$\text{AIC} := \log L - p$$

or (minimize)

$$\text{AIC} := -2 \log L + 2p = -2n \log(\text{RSS}/n) + 2p$$

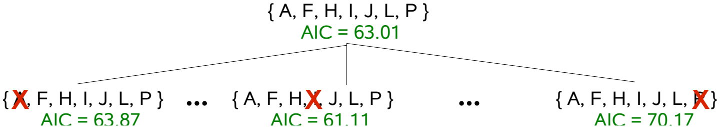
Bayes Information Criterion (BIC) /
Bayes-Schwarz Information Criterion: (maximize)

$$\text{BIC} := \log L - \frac{p}{2} \log n$$

Variable Backward Selection

{ A, F, H, I, J, L, P }
AIC = 63.01

Variable Backward Selection



{ A, F, H, I, J, L, P }
AIC = 63.01

{ ~~A~~, F, H, I, J, L, P }
AIC = 63.87

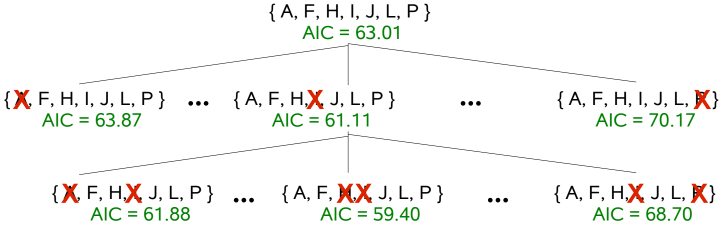
…

{ A, F, H, ~~I~~, J, L, P }
AIC = 61.11

…

{ A, F, H, I, J, L, ~~P~~ }
AIC = 70.17

Variable Backward Selection



{ A, F, H, I, J, L, P }
AIC = 63.01

{ ~~A~~, F, H, I, J, L, P }
AIC = 63.87

...

{ A, F, H, ~~I~~, J, L, P }
AIC = 61.11

...

{ A, F, H, I, J, L, ~~P~~ }
AIC = 70.17

{ ~~A~~, F, H, ~~I~~, J, L, P }
AIC = 61.88

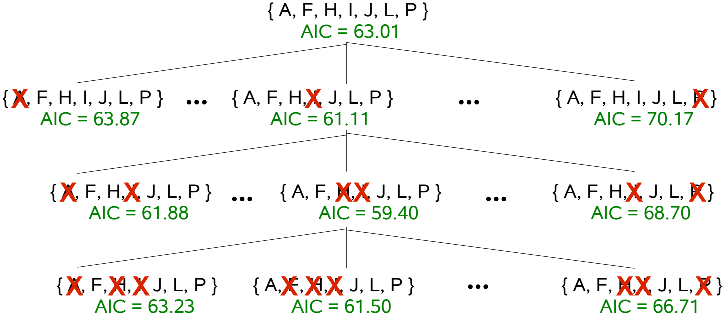
...

{ A, F, ~~H~~, ~~I~~, J, L, P }
AIC = 59.40

...

{ A, F, H, ~~I~~, J, L, ~~P~~ }
AIC = 68.70

Variable Backward Selection



{ A, F, H, I, J, L, P }
AIC = 63.01

{ ~~A~~, F, H, I, J, L, P }
AIC = 63.87

...

{ A, F, H, ~~I~~, J, L, P }
AIC = 61.11

...

{ A, F, H, I, J, L, ~~P~~ }
AIC = 70.17

{ ~~A~~, F, H, ~~I~~, J, L, P }
AIC = 61.88

...

{ A, F, ~~H~~, ~~I~~, J, L, P }
AIC = 59.40

...

{ A, F, H, ~~I~~, J, L, ~~P~~ }
AIC = 68.70

{ ~~A~~, F, ~~H~~, ~~I~~, J, L, P }
AIC = 63.23

{ A, ~~F~~, ~~H~~, ~~I~~, J, L, P }
AIC = 61.50

...

{ A, F, ~~H~~, ~~I~~, J, L, ~~P~~ }
AIC = 66.71

X removed variable

Outline

1. The Problem of Overfitting

2. Model Selection

3. Regularization

4. Hyperparameter Optimization

Shrinkage

Model selection operates by

- fitting models for a set of models with varying complexity and then picking the "best one" ex post,
- omitting some parameters completely (i.e. forcing them to be 0).

Shrinkage follows a similar idea:

- smaller parameters mean a simpler hypothesis/less complex model. Hence, small parameters should be prefered in general.
- a term is added to the model equation to penalize high parameters instead of forcing them to be 0.

Shrinkage

There are various types of shrinkage techniques for different application domains.

L1/Lasso Regularization: $\lambda \sum_{j=1}^{p} \left|\hat{\beta}_j\right| = \lambda \left\|\hat{\beta}\right\|_1$

L2/Tikhonov Regularization: $\lambda \sum_{j=1}^{p} \hat{\beta}_j^2 = \lambda \left\|\hat{\beta}\right\|_2^2$

Elastic Net: $\lambda_1 \left\|\hat{\beta}\right\|_1 + \lambda_2 \left\|\hat{\beta}\right\|_2^2$

Ridge Regression

Ridge regression is a combination of

$$\underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}_{} + \lambda \underbrace{\sum_{j=1}^{p} \beta_j^2}_{}$$

$$= \text{L2 loss} \qquad + \lambda \text{ L2 regularization}$$

Ridge Regression (Closed Form)

Ridge regression: minimize

$$\mathrm{RSS}_\lambda(\hat\beta) = \mathrm{RSS}(\hat\beta) + \lambda \sum_{j=1}^{p} \hat\beta_j^2 = \langle \mathbf{y} - \mathbf{X}\hat\beta, \mathbf{y} - \mathbf{X}\hat\beta \rangle + \lambda \sum_{j=1}^{p} \hat\beta_j^2$$

$$\Rightarrow \hat\beta = \left(\mathbf{X}^T\mathbf{X} + \lambda \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix} \right)^{-1} \mathbf{X}^T\mathbf{y}$$

with $\lambda \geq 0$ a **complexity parameter** / **regularization parameter**.

As solutions of ridge regression are not equivariant under scaling of the predictors, data is normalized before ridge regression:

$$x'_{i,j} := \frac{x_{i,j} - \bar{x}_{.,j}}{\hat\sigma(x_{.,j})}$$

Ridge Regression (Gradient Descent)

```
procedure RIDGE-REGR-GD($\hat{y} : \mathbb{R}^P \rightarrow \mathbb{R}, \hat{\beta}^{(0)} \in \mathbb{R}^{P+1}, \alpha, t_{\max} \in \mathbb{N}, X \in \mathbb{R}^{N \times P}$)
    for $t = 1, \ldots, t_{\max}$ do
        $\hat{\beta}_0^{(t)} := \hat{\beta}_0^{(t-1)} - \alpha \left( 2 \sum_{i=1}^{N} -(y_i - \hat{y}(X_i)) \right)$
        for $j = 1, \ldots, P$ do
            $\hat{\beta}_j^{(t)} := \hat{\beta}_j^{(t-1)} - \alpha \left( 2 \sum_{i=1}^{n} -X_{i,j} (y_i - \hat{y}(X_i)) + 2\lambda \hat{\beta}_j^{(t-1)} \right)$
        if converged then
            return $\hat{\beta}^{(t)}$
```

L2-Regularized Update Rule

$$\hat{\beta}_j^{(t)} := \underbrace{(1 - 2\alpha\lambda)}_{\text{shrinkage}} \hat{\beta}_j^{(t-1)} - \alpha \left(2 \sum_{i=1}^{n} -X_{i,j} (y_i - \hat{y}(X_i)) \right)$$

Tikhonov Regularization Derivation (1/2)

Treat the true parameters θ_j as random variables Θ_j with the following distribution (**prior**):

$$\Theta_j \sim \mathcal{N}(0, \sigma_\Theta), \quad j = 1, \ldots, p$$

Then the **joint likelihood of the data and the parameters** is

$$L_{\mathcal{D},\Theta}(\theta) := \left(\prod_{i=1}^{n} p(x_i, y_i \,|\, \theta)\right) \prod_{j=1}^{p} p(\Theta_j = \theta_j)$$

and the conditional joint log likelihood of the data and the parameters

$$\log L^{\text{cond}}_{\mathcal{D},\Theta}(\theta) := \left(\sum_{i=1}^{n} \log p(y_i \,|\, x_i, \theta)\right) + \sum_{j=1}^{p} \log p(\Theta_j = \theta_j)$$

and

$$\log p(\Theta_j = \theta_j) = \log \frac{1}{\sqrt{2\pi}\sigma_\Theta} e^{-\frac{\theta_j^2}{2\sigma_\Theta^2}} = -\log(\sqrt{2\pi}\sigma_\Theta) - \frac{\theta_j^2}{2\sigma_\Theta^2}$$

Tikhonov Regularization Derivation (2/2)

Dropping the terms that do not depend on θ_j yields:

$$\begin{aligned}\log L^{\text{cond}}_{\mathcal{D},\Theta}(\theta) &:= \left(\sum_{i=1}^{n} \log p(y_i \,|\, x_i, \theta)\right) + \sum_{j=1}^{p} \log p(\Theta_j = \theta_j) \\ &\propto \left(\sum_{i=1}^{n} \log p(y_i \,|\, x_i, \theta)\right) - \frac{1}{2\sigma_\Theta^2} \sum_{j=1}^{p} \theta_j^2\end{aligned}$$

This also gives a semantics to the complexity / regularization parameter λ:

$$\lambda = \frac{1}{2\sigma_\Theta^2}$$

but σ_Θ^2 is unknown. (We will see methods to estimate λ soon.)

The parameters θ that maximize the joint likelihood of the data and the parameters are called **Maximum Aposteriori Estimators (MAP estimators)**.

L2-Regularized Logistic Regression (Gradient Descent)

$$\log L^{\text{cond}}_{\mathcal{D}}(\hat{\beta}) \;=\; \sum_{i=1}^{n} y_i \langle x_i, \hat{\beta}\rangle - \log(1 + e^{\langle x_i, \hat{\beta}\rangle}) - \lambda \sum_{j=1}^{P} \hat{\beta}_j^2$$

```
procedure LOG-REGR-GA($L^{\text{cond}}_{\mathcal{D}} : \mathbb{R}^{P+1} \to \mathbb{R}, \hat{\beta}^{(0)} \in \mathbb{R}^{P+1}, \alpha, t_{\text{max}} \in \mathbb{N}, \epsilon \in \mathbb{R}^{+}$)
    for $t = 1, \ldots, t_{\text{max}}$ do
        $\hat{\beta}_0^{(t)} := \hat{\beta}_0^{(t-1)} + \alpha \sum_{i=1}^{n} \left(y_i - p\left(Y = 1 | X = x_i; \hat{\beta}^{(t-1)}\right)\right)$
        for $j = 1, \ldots, P$ do
            $\hat{\beta}_j^{(t)} := \hat{\beta}_j^{(t-1)} + \alpha \sum_{i=1}^{n} x_{i,j} \left(y_i - p\left(Y = 1 | X = x_i; \hat{\beta}^{(t-1)}\right)\right) - 2\lambda \hat{\beta}_j^{(t-1)}$
        if $L^{\text{cond}}_{\mathcal{D}}(\hat{\beta}^{(t-1)}) - L^{\text{cond}}_{\mathcal{D}}(\hat{\beta}^{(t)})) < \epsilon$ then
            return $\hat{\beta}^{(t)}$
    error "not converged in $t_{\text{max}}$ iterations"
```

L2-Regularized Logistic Regression (Newton)

Newton update rule:

$$\hat{\beta}^{(t)} := \hat{\beta}^{(t-1)} + \alpha H^{-1} \nabla_{\hat{\beta}} p\left(Y = 1 | X = x_i; \hat{\beta}^{(t-1)}\right)$$

$$p_i = p\left(Y = 1 | X = x_i; \hat{\beta}^{(t-1)}\right)$$

$$\nabla_{\hat{\beta}} L_{\mathcal{D}}^{cond} = \begin{pmatrix} \sum_{i=1}^{n} -(y_i - p_i) \\ \sum_{i=1}^{n} -x_{i,1}(y_i - p_i) -2\lambda\hat{\beta}_1 \\ \vdots \\ \sum_{i=1}^{n} -x_{i,P}(y_i - p_i) -2\lambda\hat{\beta}_P \end{pmatrix}$$

$$H = \sum_{i=1}^{n} -p_i(1 - p_i) x_i x_i^T -2\lambda \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Outline

1. The Problem of Overfitting

2. Model Selection

3. Regularization

4. Hyperparameter Optimization

What is Hyperparameter Optimization?

Many learning algorithms $\mathcal{A}_\lambda$ have hyperparameters λ (learning rate, regularization). After choosing them, $\mathcal{A}_\lambda$ can be used to map the training data D_{train} to a function $\hat{y}$ by minimizing some loss $\mathcal{L}(x;\hat{y})$.

Identifying good values for the hyperparameters λ is called **hyperparameter optimization**.

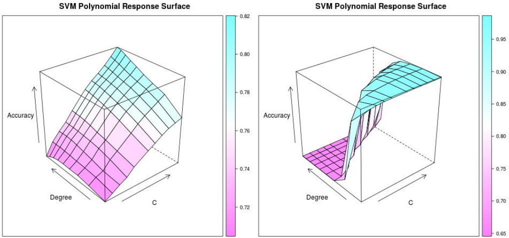
Hence, hyperparameter optimization is a second order optimization

$$\text{argmin}_{\lambda\in\Lambda}\frac{1}{|D_{\text{calib}}|}\sum_{x\in D_{\text{calib}}}\mathcal{L}\left(x;\mathcal{A}_\lambda\left(D_{\text{train}}\right)\right)=\text{argmin}_{\lambda\in\Lambda}\Psi(\lambda)$$

where Ψ is the **hyperparameter response function** and D_{calib} a **calibration set**.

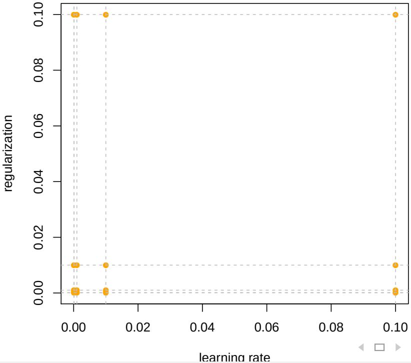
Why Hyperparameter Optimization

- So far only model parameters were optimized.
- Hyperparameters (such as learning rate α and regularization λ) were omitted.
- Hyperparameters can have a big impact on the prediction quality.



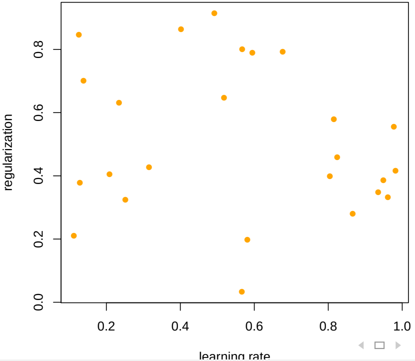
Grid Search

- Choose for each hyperparameter a set of values $\Lambda_1, \ldots, \Lambda_q$.
- $\Lambda = \prod_{i=1}^{q} \Lambda_i$ is then the combination of all hyperparameters in all Λ_is.
- Then choose the hyperparameter $\lambda \in \Lambda$ with best performance on D_{calib}.



Random Search

- Instead of choosing hyperparameters on a grid, choose random hyperparameters λ for Λ (within a reasonable space).
- Provides better results than grid search in cases of insensitive hyperparameters.



What is the Calibration Data?

Whenever a learning process depends on a hyperparameter, the hyperparameter can be estimated by picking the value with the lowest error.

If this is done on test data, one actually uses test data in the training process ("train on test"), thereby lessen its usefullness for estimating the test error.

Therefore, one splits the training data again in

- (proper) training data and
- **calibration data**.

The calibration data figures as test data during the training process.

Cross Validation

Instead of a single split into

training data, (validation data,) and test data

cross validation splits the data in k parts (of roughly equal size)

$$D = D_1 \cup D_2 \cup \cdots \cup D_k, \quad D_i \text{ pairwise disjunct}$$

and averages performance over k learning problems

$$D^{(i)}_{\text{train}} = D \setminus D_i, \quad D^{(i)}_{\text{test}} = D_i \quad i = 1, \ldots, k$$

Common is 5- and 10-fold cross validation.

n-fold cross validation is also known as **leave one out**.

Cross Validation

How many folds to use in k-fold cross validation?

$k = n$ / leave one out:

- approximately unbiased for the true prediction error.
- high variance as the n training sets are very similar.
- in general computationally costly as n different models have to be learnt.

$k = 5$:

- lower variance.
- bias could be a problem,
  due to smaller training set size the prediction error could be overestimated.

Summary

- The problem of overfitting can be overcome by model selection or shrinkage.
- Applying L2-Regularization for Linear and Logistic Regression needs only few changes in the learning algorithm
- Estimating the best hyperparameters can be considered as a meta-learning problem. They can be estimated e.g. by Grid Search and Random Search.

Further Readings

- [JWHT13, chapter 3], [Mur12, chapter 7], [HTFF05, chapter 3].

References

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin.
The elements of statistical learning: data mining, inference and prediction, volume 27.
2005.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani.
An introduction to statistical learning.
Springer, 2013.

Kevin P. Murphy.
Machine learning: a probabilistic perspective.
The MIT Press, 2012.