Machine Learning

A. Supervised Learning: Linear Models & Fundamentals

A.3. Regularization

Lars Schmidt-Thieme

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

Syllabus

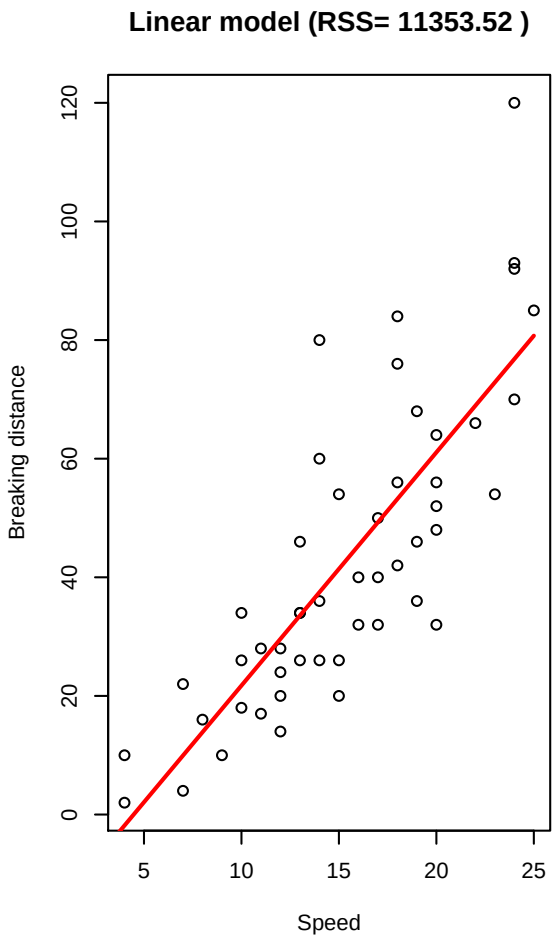
Fri. 25.10.	(1)	0. Introduction
		A. Supervised Learning: Linear Models & Fundamentals
Fri. 1.11.	(2)	A.1 Linear Regression
Fri. 8.11.	(3)	A.2 Linear Classification
Fri. 15.11.	(4)	A.3 Regularization
Fri. 22.11.	(5)	A.4 High-dimensional Data
		B. Supervised Learning: Nonlinear Models
Fri. 29.11.	(6)	B.1 Nearest-Neighbor Models
Fri. 6.12.	(7)	B.2 Neural Networks
Fri. 13.12.	(8)	B.3 Decision Trees
Fri. 20.12.	(9)	B.4 Support Vector Machines
		— *Christmas Break* —
Fri. 10.1.	(10)	B.5 A First Look at Bayesian and Markov Networks
		C. Unsupervised Learning
Fri. 17.1.	(11)	C.1 Clustering
Fri. 24.1.	(12)	C.2 Dimensionality Reduction
Fri. 31.1.	(13)	C.3 Frequent Pattern Mining
Fri. 7.2.	(14)	Q&A

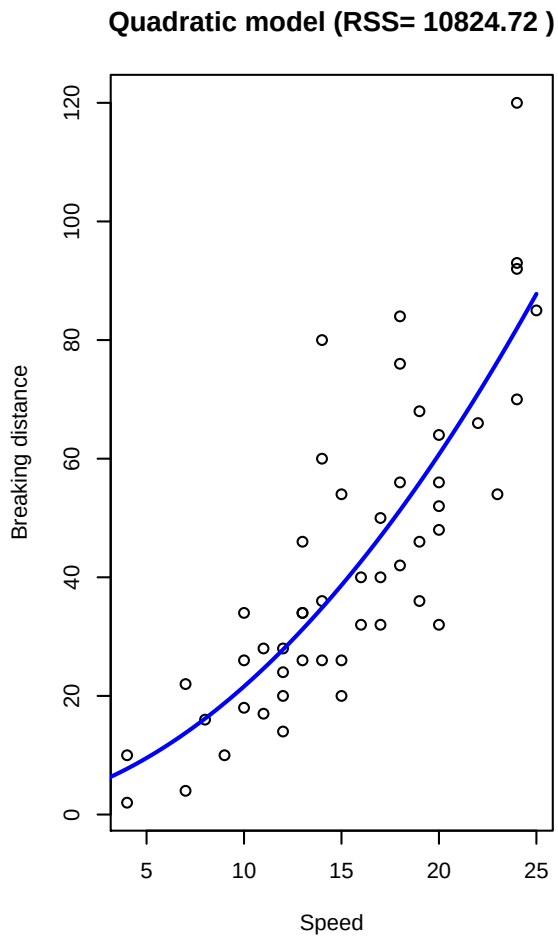
Outline

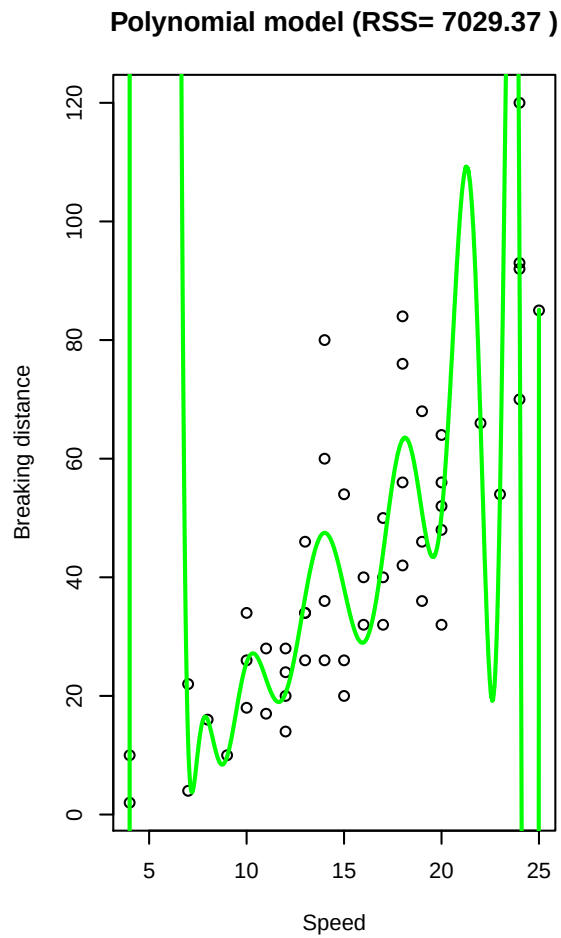
1. The Problem of Overfitting
2. Model Selection
3. Regularization
4. Hyperparameter Optimization

Outline

Fitting of models







Underfitting/Overfitting

- **Underfitting:**
  - the model is not complex enough to explain the data well.
  - results in poor predictive performance.
- **Overfitting:**
  - the model is too complex, it describes the
    - noise, inherent random variations of the data generating process, instead of the
    - signal, the underlying relationship between target and predictors.
  - results in poor predictive performance as well.
- Overfitting is easy: given N points (x_n, y_n) without repeated measurements (i.e. $x_n \neq x_m,\ n \neq m$), there exists a polynomial of degree $N-1$ with RSS equal to 0.

$$\hat{y}(x) := \sum_{n=1}^{N} y_n \prod_{\substack{m=1 \\ m \neq n}}^{N} \frac{x - x_m}{x_n - x_m}$$

Outline

Losses and Fit Measures

	loss	**fit/quality measure**
semantics	the smaller, the better	the larger, the better
goal	minimize	maximize
regression	$\text{RSS}(y, \hat{y}) := \|\|y - \hat{y}\|\|_2^2 := \sum_{n=1}^{N}(y_n - \hat{y}_n)^2$ $\text{L2}(y, \hat{y}) := \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2$ $\text{RMSE}(y, \hat{y}) := (\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2)^{\frac{1}{2}}$ $\text{MAE}(y, \hat{y}) := \frac{1}{N}\|\|y - \hat{y}\|\|_1 := \frac{1}{N}\sum_{n=1}^{N}\|y_n - \hat{y}_n\|$	$\log L_{\mathcal{N}}(y, \hat{y}) := \sum_{n=1}^{N} -\frac{1}{2\sigma_y^2}(y_n - \hat{y}_n)^2$
classification	$\text{MR}(y, \hat{y}) := \sum_{n=1}^{N}\mathbb{I}(y_n \neq \hat{y}_n)$	$\text{ACC}(y, \hat{y}) := \frac{1}{N}\sum_{n=1}^{N}\mathbb{I}(y_n = \hat{y}_n)$ $\log L_{\text{binomial}}(y, \hat{y}) := \sum_{n=1}^{N}\hat{y}_n\mathbb{I}(y_n = 1) + (1 - \hat{y}_n)\mathbb{I}(y_n = 0)$

Model Selection Measures

- **Model selection**:
  - given a set of models
    indexed by p, one model for each value of p

$$\hat{y}_p(x) = \sum_{m=0}^{p-1} \hat{\beta}_m x_m$$

  - 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

$$\hat{\beta}'_m = \begin{cases} \hat{\beta}_m, & \text{for } m \leq p \\ 0, & \text{for } m > p \end{cases}$$

Model Selection Measures

- One uses **model selection measures** of type

$$\text{model selection measure} = \text{fit} - \text{complexity} \quad \text{(max!)}$$

or equivalently

$$\text{model selection measure} = \text{loss} + \text{complexity} \quad \text{(min!)}$$

- 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)

$$\text{AIC} := -2 \log L + 2p$$

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

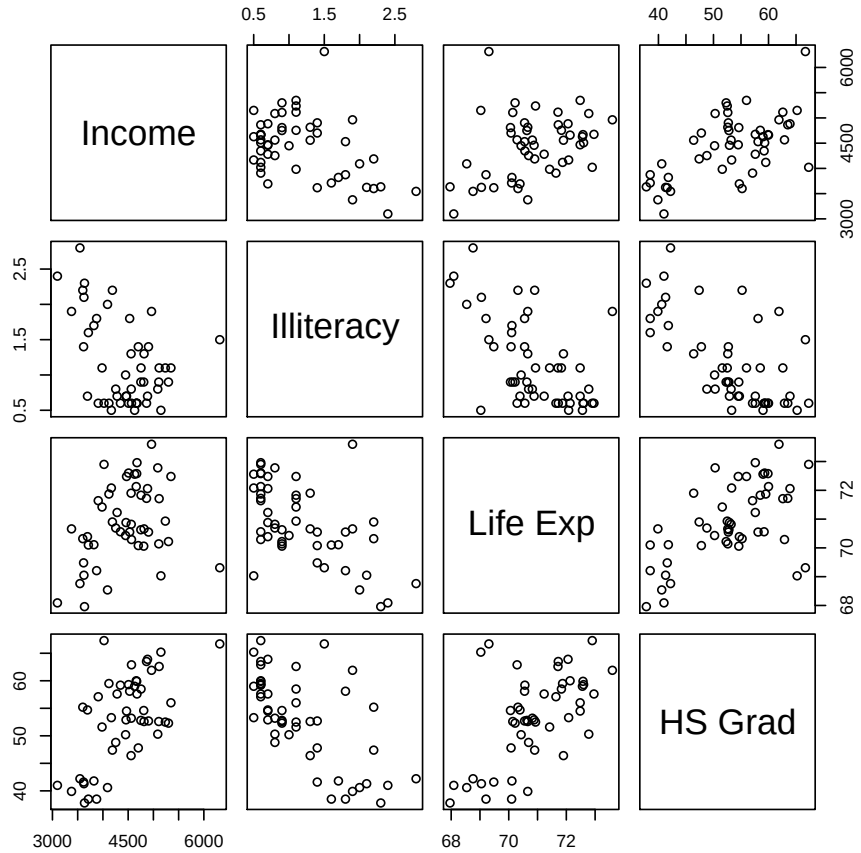
$$\text{BIC} := \log L - \frac{p}{2} \log N$$

where L denotes the likelihood
p the number of parameters
N the number of samples

Example: Predicting Murder Rate

sociographic data of the 50 US states in 1977:

- x_A land area in square miles
- x_F mean number of days with minimum temperature below freezing (1931–1960) in capital or large city
- x_H percent high-school graduates (1970).
- x_I illiteracy (percent of population, 1970),
- x_J income (per capita, 1974),
- x_L life expectancy (in years, 1969–71),
- x_P population (July 1, 1975)
- y_M murder rate per 100,000 population (1976)

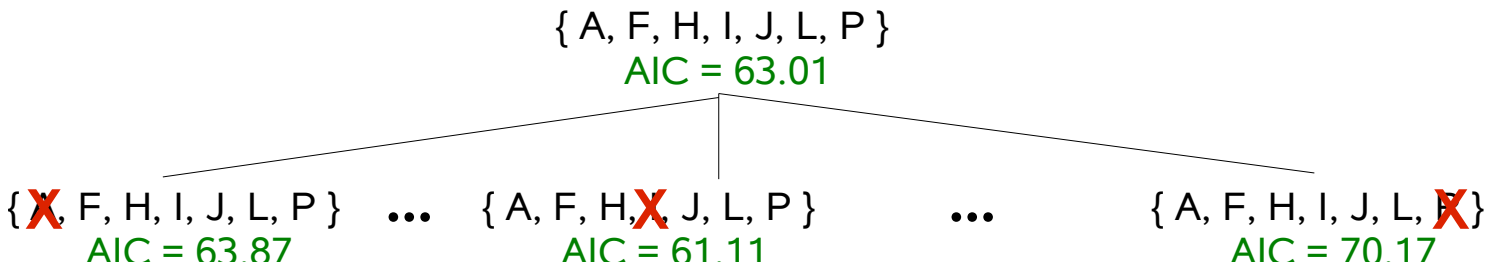


$$y_M = \beta_0 + \beta_A x_A + \beta_F x_F + \beta_H x_H + \beta_I x_I + \beta_J x_J + \beta_L x_L + \beta_P x_P$$

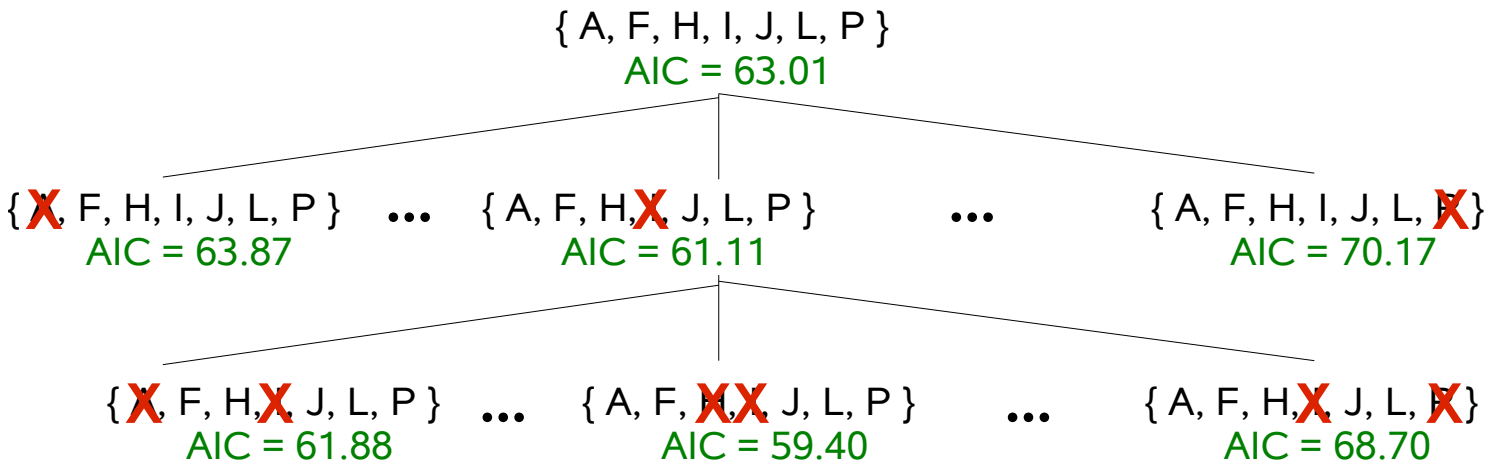
Variable Backward Selection

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

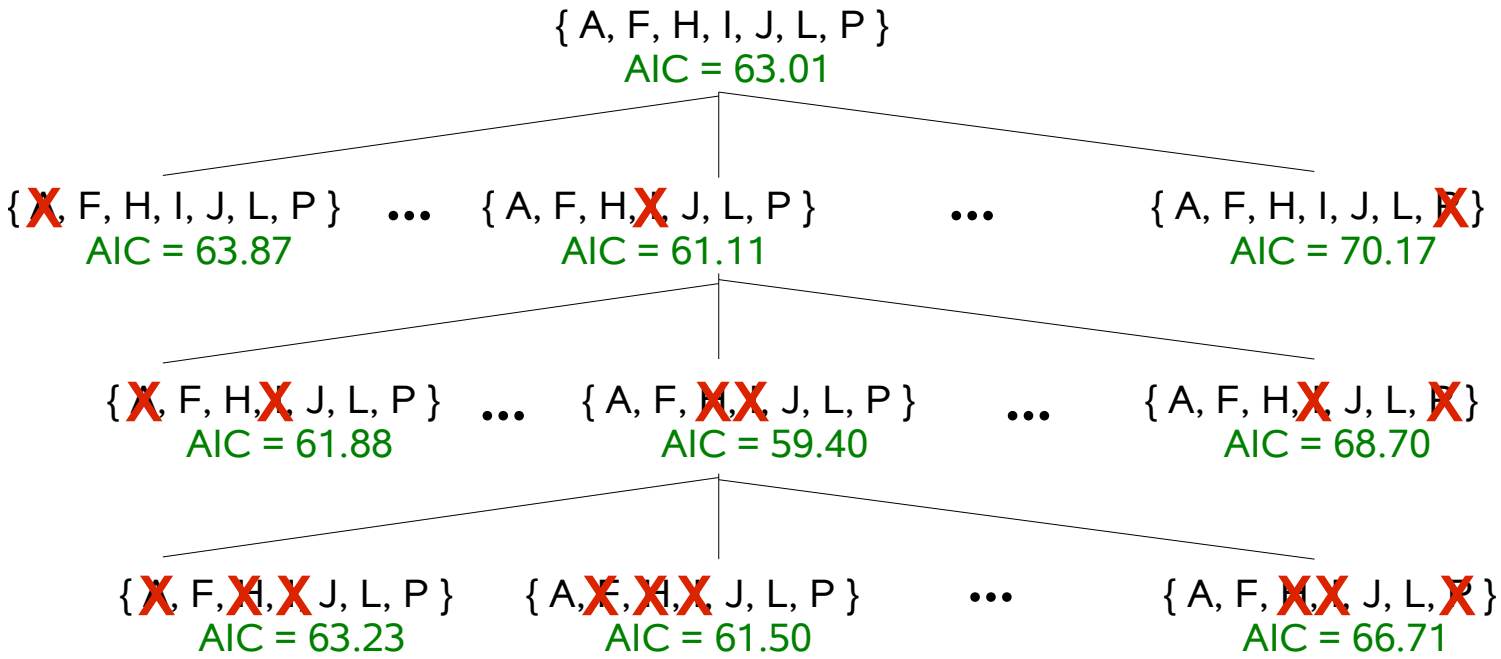
Variable Backward Selection



Variable Backward Selection



Variable Backward Selection



Outline

Shrinkage

- **Model selection** operates by
  1. fitting model instances for a set of models with varying complexity
  2. picking the "best one" in hindsight,
- **Variable Selection**
  - = model selection applied to models with different predictor subsets
  - for models $\hat{y}$ that factor through a linear combination of the predictors,

$$\hat{y}(x; \hat{\beta}) = f(\sum_{m=1}^{M} \hat{\beta}_m x_m) \quad \text{for a suitable } f$$

    - dropping a variable x_m from the model is equivalent to
    - forcing its model parameter $\hat{\beta}_m$ to be 0.

Note: "Fitting a model instance" = "Learning model parameters",
for models having parameters such as linear regression, logistic regression etc.

Shrinkage

- **Variable Selection**
  - ...
    - forcing its model parameter $\hat{\beta}_m$ 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 objective function to
    - favor small parameters or equivalently
    - **penalize large parameters** or
    - **shrink them towards 0**

    instead of forcing them to be 0.

Shrinkage / Regularization Penalties

There are various types of shrinkage techniques for different problem settings.

L1/Lasso Regularization: $\lambda \sum_{m=1}^{M} |\hat{\beta}_m| = \lambda \|\hat{\beta}\|_1$

L2/Tikhonov Regularization: $\lambda \sum_{m=1}^{M} \hat{\beta}_m^2 = \lambda \|\hat{\beta}\|_2^2$

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

Ridge Regression

Ridge regression is a combination of

$$\underbrace{f(\hat{\beta}; \lambda, \mathcal{D})}_{\text{objective function}} := \underbrace{\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n(x_n; \hat{\beta}))^2}_{\text{L2 loss}} + \underbrace{\lambda \sum_{m=1}^{M} \hat{\beta}_m^2}_{\lambda \text{ L2 regularization}}$$

Learning Ridge Regression (Closed Form)

Ridge regression: minimize

$$f(\hat{\beta}; \lambda, \mathcal{D}) = \mathrm{L2}(\hat{\beta}) + \lambda \sum_{m=1}^{M} \hat{\beta}_m^2 = \frac{1}{N} \langle \mathbf{y} - \mathbf{X}\hat{\beta}, \mathbf{y} - \mathbf{X}\hat{\beta} \rangle + \lambda \langle \hat{\beta}, \hat{\beta} \rangle$$

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

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

Beware: ridge regression parameter estimates are not equivariant under scaling of the predictors

$\rightsquigarrow$ data should be normalized before parameter estimation:

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

Learning Ridge Regression (Gradient Descent)

learn-ridgereg-GD$(\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, \lambda, \mu, i_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:

$X := (x_1, x_2, \ldots, x_N)^T$

$y := (y_1, y_2, \ldots, y_N)^T$

$\hat{\beta}_0 := (0, \ldots, 0)$

$\hat{\beta} :=$ **minimize-GD**$($ $f(\hat{\beta}) := \frac{1}{N}(y - X\hat{\beta})^T(y - X\hat{\beta}) + \lambda \hat{\beta}^T \hat{\beta}$, $\hat{\beta}_0, \mu, i_{\max}, \epsilon)$

return $\hat{\beta}$

Learning Ridge Regression (Gradient Descent; elementary operations)

learn-ridgereg-GD$(\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, \lambda, \mu, i_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:

$X := (x_1, x_2, \ldots, x_N)^T$

$y := (y_1, y_2, \ldots, y_N)^T$

$\hat{\beta} := 0_M$

$\ell := \frac{1}{N}||y - X\hat{\beta}||^2$

for $t = 1, \ldots, i_{\max}$:

  $\hat{\beta} := \hat{\beta} - \mu(-\frac{2}{N} \cdot X^T(y - X\hat{\beta}) + 2\lambda\hat{\beta})$

  $\ell^{\text{old}} := \ell$

  $\ell := \frac{1}{N}||y - X\hat{\beta}||^2$

  if $\ell - \ell^{\text{old}} < \epsilon$:

    return $\hat{\beta}$

raise exception "not converged in $i_{\max}$ iterations"

L2-Regularized Update Rule

$$\hat{\beta}^{(t)} := (1 - 2\mu\lambda)\hat{\beta}^{(t-1)} + \frac{2\mu}{N} X^T(y - X\hat{\beta}^{(t-1)})$$

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

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

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

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

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:

$$\log L^{\text{cond}}_{\mathcal{D},\Theta}(\theta) := \left(\sum_{n=1}^{N} \log p(y_n \mid x_n, \theta)\right) + \sum_{j=1}^{p} \log p(\Theta_j = \theta_j)$$

$$\propto \left(\sum_{n=1}^{N} \log p(y_n \mid x_n, \theta)\right) - \frac{1}{2\sigma^2_\Theta} \sum_{j=1}^{p} \theta_j^2$$

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

$$\lambda = \frac{1}{2\sigma^2_\Theta}$$

but σ^2_Θ 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)

learn-reglogreg-GA$(\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, \lambda, \mu, t_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:

$\ell := \frac{1}{N} \log L_{\mathcal{D}}^{\text{cond}}(\hat{\beta}) := \frac{1}{N} \sum_{n=1}^{N} y_n \langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle}) + \lambda \hat{\beta}^T \hat{\beta}$

$\hat{\beta} := \text{maximize-GA}(\ell, 0_M, \mu, t_{\max}, \epsilon)$

return $\hat{\beta}$

L2-Regularized Logistic Regression (Gradient Descent)

learn-logreg-GA$(\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, \lambda, \mu, t_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:

$X := (x_1, x_2, \ldots, x_N)^T$

$y := (y_1, y_2, \ldots, y_N)^T$

$\hat{\beta} := 0_M$

$\ell := \frac{1}{N} \sum_{n=1}^{N} y_n \langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})$

for $t = 1, \ldots, t_{\max}$:

$\hat{y} := (1/(1 + e^{-\hat{\beta}^T x_n})_{n \in 1:N}$

$\hat{\beta} := \hat{\beta} + \mu \cdot (\frac{1}{N} X^T (y - \hat{y}) - 2\lambda\hat{\beta})$

$\ell^{\text{old}} := \ell$

$\ell := \frac{1}{N} \sum_{n=1}^{N} y_n \langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})$

if $\ell - \ell^{\text{old}} < \epsilon$:

return $\hat{\beta}$

raise exception "not converged in $t_{\max}$ iterations"

L2-Regularized Logistic Regression (Newton)

Newton update rule:

$$\hat{\beta}^{(t+1)} := \hat{\beta}^{(t)} + \mu (H^{(t)})^{-1} \nabla_{\hat{\beta}} (L_{\mathcal{D}}^{\text{cond}})^{(t)}$$

$$(\nabla_{\hat{\beta}} L_{\mathcal{D}}^{\text{cond}})^{(t)} = \begin{pmatrix} \sum_{n=1}^{N} \left(y_n - \hat{y}_n^{(t)} \right) \\ \sum_{n=1}^{N} x_{n,1} \left(y_n - \hat{y}_n^{(t)} \right) - 2\lambda \hat{\beta}_1^{(t)} \\ \vdots \\ \sum_{n=1}^{N} x_{n,M} \left(y_n - \hat{y}_n^{(t)} \right) - 2\lambda \hat{\beta}_M^{(t)} \end{pmatrix}$$

$$H^{(t)} = \sum_{n=1}^{N} -\hat{y}_n^{(t)} \left(1 - \hat{y}_n^{(t)} \right) x_n x_n^T - 2\lambda I$$

Outline

What is Hyperparameter Optimization?

- Most models and learning algorithms have parameters that cannot be learned by minimizing the objective function, because either
  - the objective function would be minimized for a trivial value, e.g., $\lambda = 0$, or
  - the parameters affect the learning algorithm, e.g., learning rate.

- These parameters are called **hyperparameters** λ and they parametrize a **learning algorithm** $\mathcal{A}_\lambda$.
  - choose suitable hyperparameters λ
  - use $\mathcal{A}_\lambda$ to map the training data $\mathcal{D}_{\text{train}}$ to a prediction function $\hat{y}$ by minimizing some loss $\mathcal{L}(\mathcal{D}, \hat{y})$ over the training data.

What is Hyperparameter Optimization?

- Identifying good values for the hyperparameters λ is called **hyperparameter optimization**.
  - hyperparameter optimization is a second level optimization

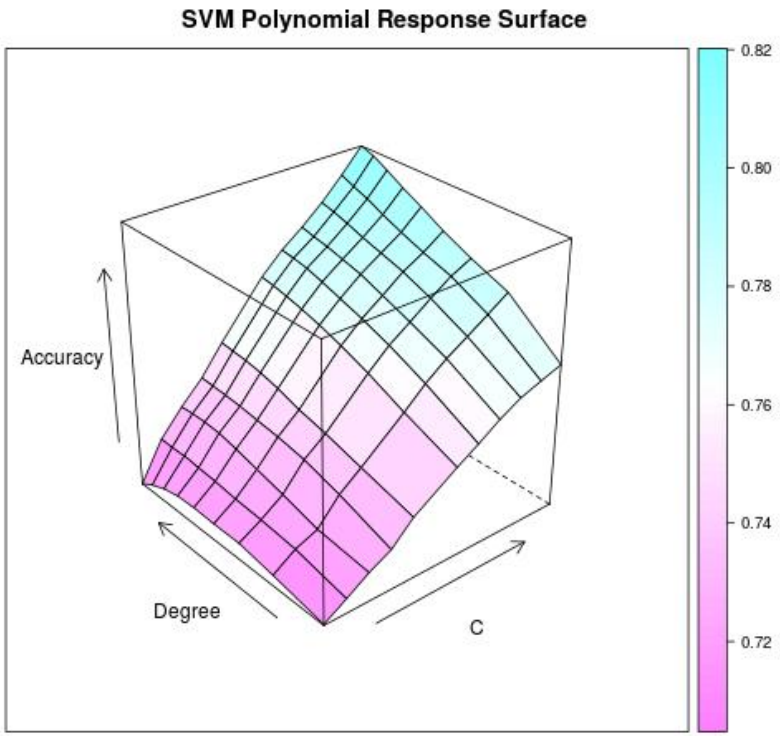
$$\arg\min_{\lambda \in \Lambda} \mathcal{L}(\mathcal{D}_{\text{valid}}, \mathcal{A}_\lambda(\mathcal{D}_{\text{train}})) = \arg\min_{\lambda \in \Lambda} \Psi(\lambda)$$

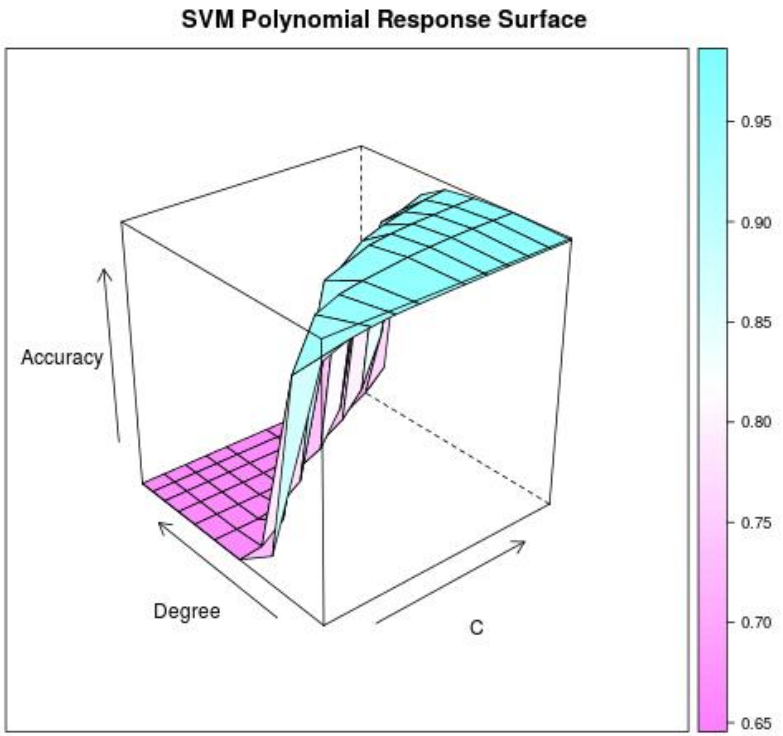
where

- Ψ is the **hyperparameter response function** and
- $\mathcal{D}_{\text{valid}}$ a **validation data** (aka **calibration data** and **holdout data**).

Why Hyperparameter Optimization

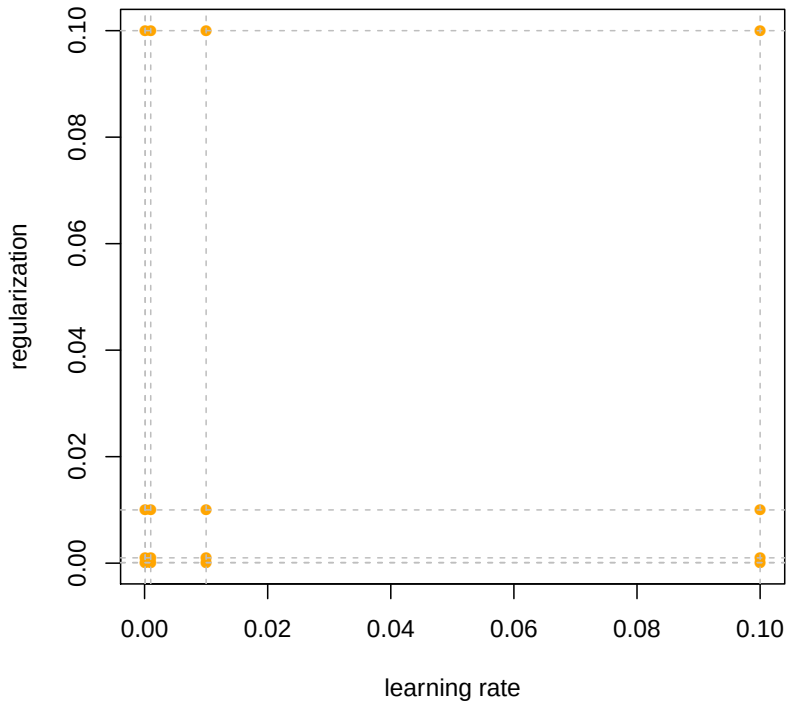
- So far only model parameters were optimized.
- Values for hyperparameters (such as regularization λ and learning rate μ) came "out of the blue".
- Hyperparameters can have a big impact on the prediction quality.





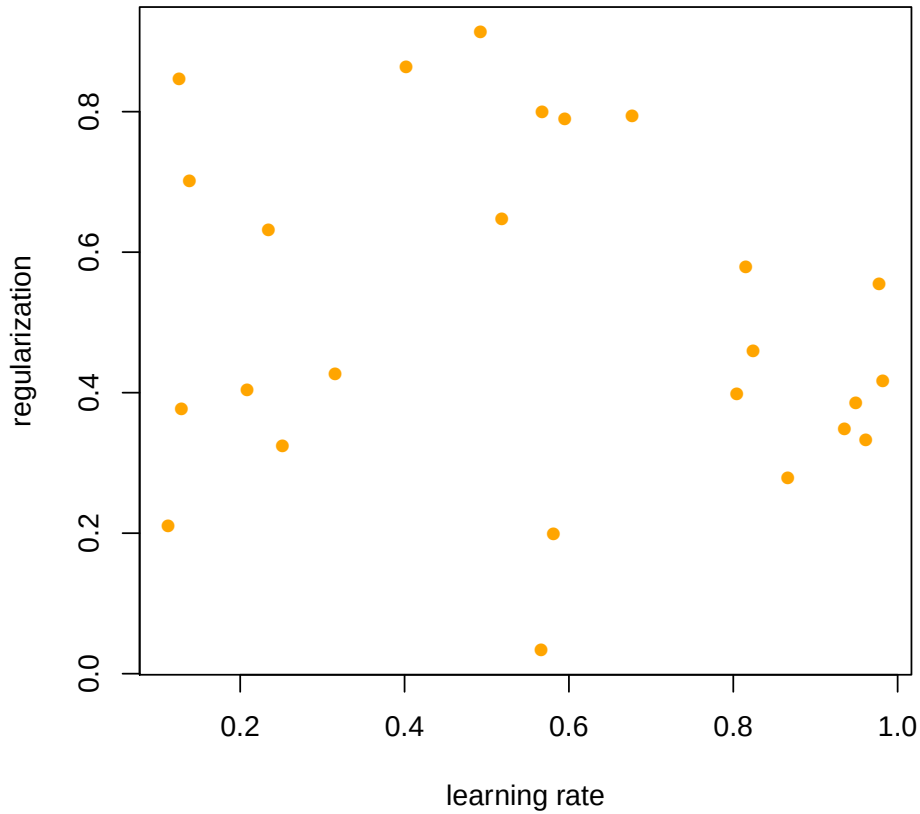
Grid Search

- Assume we have Q hyperparameters $\lambda_1, \ldots, \lambda_Q$
- Choose for each hyperparameter λ_q a set of values Λ_q.
- $\Lambda := \prod_{q=1}^{Q} \Lambda_q$ is then a grid of hyperparameters.
- Choose the hyperparameter combination $\lambda \in \Lambda$ with best performance on $\mathcal{D}_{\text{valid}}$.



Random Search

- Instead of trying hyperparameter combinations on a grid, try random hyperparameter combinations λ for Λ (within a reasonable range).
- Usually slightly better results than grid search.



What is the Validation 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 usefulness for estimating the test error.
- Therefore, one splits the training data again in
  - **(proper) training data** and
  - **validation data**.
- The validation data figures as test data during the training process.

Cross Validation

Instead of a single split into

training data, (validation data,) and test data

K-fold cross validation splits the data in K parts (of roughly equal size)

$$\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_K, \quad \mathcal{D}_k \text{ pairwise disjoint}$$

and averages performance over K learning problems

$$\mathcal{D}^{(k)}_{\text{train}} := \mathcal{D} \setminus \mathcal{D}_k, \quad \mathcal{D}^{(k)}_{\text{test}} := \mathcal{D}_k, \quad k = 1, \ldots, K$$

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 **underfitting** can be overcome by using more complex models, e.g., having
  - variable interactions as in polynomial models.
- The problem of **overfitting** can be overcome by
  - **model selection** / variable selection as well as by
  - (parameter) **shrinkage**.
- Applying L2-regularization to Linear and Logistic Regression requires only few changes in the learning algorithms.
- Shrinkage introduces a **hyperparameter** λ that cannot be learned by direct loss minimization.
- Estimating the best hyperparameters can be considered as a **meta-learning problem**. They can be estimated e.g. by
  - **Grid Search** or
  - Random Search — both using **validation data**.

Further Readings

- [James et al., 2013, chapter 3], [Murphy, 2012, chapter 7], [Hastie et al., 2005, chapter 3].

References

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, volume 27. Springer, 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.