Business Analytics

1. Prediction, 1.3 Regularized Loss Models

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
University of Hildesheim, Germany

Outline

1. Overfitting and Regularization
2. Prediction Functions
3. Sparse Predictors
4. Learning Algorithms

Outline

Overall Procedure

1. define a **prediction function** $\hat{y}$ that depends on some **model parameters** $\Theta \in \mathbb{R}^q$, e.g., for regression, a linear model:

$$\hat{y}(x;\Theta) := \beta_0 + \beta^T x = \beta_0 + \sum_{i=1}^{p} \beta_i x_i, \quad \Theta := (\beta_0, \beta_1, \ldots, \beta_p)$$

2. the **training error**

$$\ell(\Theta; \mathcal{D}^{\text{train}}) := \frac{1}{|\mathcal{D}^{\text{train}}|} \sum_{(x,y)\in\mathcal{D}^{\text{train}}} \ell(y, \hat{y}(x;\Theta))$$

is called **objective function**

$$f(\Theta; \mathcal{D}^{\text{train}}) := \ell(\Theta; \mathcal{D}^{\text{train}})$$

3. find the parameters Θ^* that **minimize the objective function** numerically.

Overall Procedure

1. define a **prediction function** $\hat{y}$ that depends on some **model parameters** $\Theta \in \mathbb{R}^q$, e.g., for regression, a linear model:

$$\hat{y}(x; \Theta) := \beta_0 + \beta^T x = \beta_0 + \sum_{i=1}^{p} \beta_i x_i, \quad \Theta := (\beta_0, \beta_1, \dots, \beta_p)$$

2. define a **regularization function**

$$R : \mathbb{R}^q \to \mathbb{R}_0^+, \quad \text{e.g., } R(\Theta) := ||\Theta||^2 = \sum_{i=1}^{q} \Theta_i^2$$

that penalizes complex models. Its combination with the **training error**

$$\ell(\Theta; \mathcal{D}^{\text{train}}) := \frac{1}{|\mathcal{D}^{\text{train}}|} \sum_{(x,y) \in \mathcal{D}^{\text{train}}} \ell(y, \hat{y}(x; \Theta))$$

is called **objective function**

$$f(\Theta; \mathcal{D}^{\text{train}}) := \ell(\Theta; \mathcal{D}^{\text{train}}) + \lambda R(\Theta), \quad \lambda \in \mathbb{R}_0^+$$

3. find the parameters Θ^* that **minimize the objective function**

Overfitting

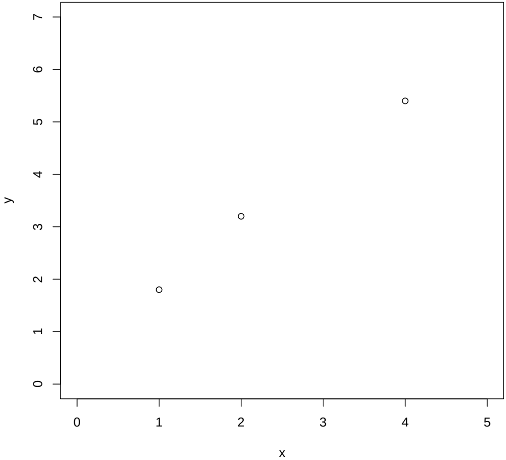
Example:
Assume the true data generating process is

$$Y = 1 + X_1 + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.1)$$

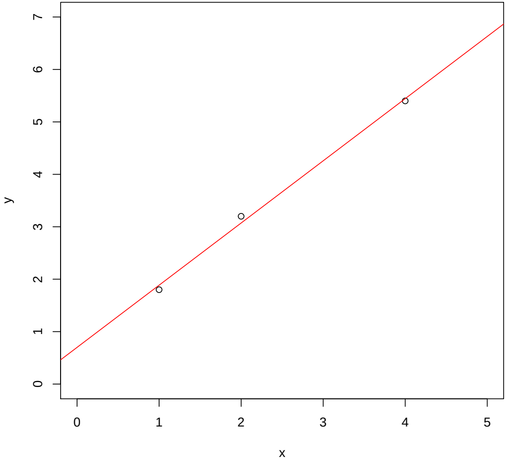
and we draw the following sample

x_1	y
1.0	1.8
2.0	3.2
4.0	5.4

Overfitting



Overfitting



The linear model with minimal training error is model #1:

$$\hat{y}(x_1) := 0.7 + 1.186\, x_1, \quad \mathrm{RMSE}(\hat{y}) = 0.093$$

Overfitting

Now lets assume we measure 3 further variables x_2, x_3 and x_4, not correlated with the target Y at all (**noise**):

x_1	x_2	x_3	x_4	y
1.0	1.0	0.0	0.0	1.8
2.0	0.0	1.0	0.0	3.2
4.0	0.0	0.0	1.0	5.4

Now, a linear model with minimal training error is model #2:

$$\hat{y}(x_1) := 0.7 + 1.186\,x_1 - 0.086\,x_2 + 0.128\,x_3 - 0.044\,x_4, \quad \text{RMSE}(\hat{y}) = 0$$

And another one is model #3:

$$\hat{y}(x_1) := 0.0 + 0.0\,x_1 + 1.8\,x_2 + 3.2\,x_3 + 5.4\,x_4, \quad \text{RMSE}(\hat{y}) = 0$$

These models **fit noise** or **overfit**.

Overfitting

How to **avoid overfitting**?

- do not include noisy variables

Overfitting

How to **avoid overfitting**?

- do not include noisy variables
  — but we do not know which ones are correlated with the target

Overfitting

How to **avoid overfitting**?

- do not include noisy variables
  — but we do not know which ones are correlated with the target

- employ model selection to find out which variables are noisy (**variable selection**)

Overfitting

How to **avoid overfitting**?

- do not include noisy variables
  — but we do not know which ones are correlated with the target

- employ model selection to find out which variables are noisy (**variable selection**)
  — possible, but usually slow and not very reliable

Overfitting

How to **avoid overfitting**?

- do not include noisy variables
  — but we do not know which ones are correlated with the target

- employ model selection to find out which variables are noisy (**variable selection**)
  — possible, but usually slow and not very reliable

- force all parameters to be small (**shrinking**)

Why Small Parameters Prevent Overfitting

Assume we force all parameters β_i to be $|\beta_i| \leq 2$.
Then model #3 is no longer allowed:

$$\hat{y}(x_1) := 0.0 + 0.0\,x_1 + 1.8\,x_2 + 3.2\,x_3 + 5.4\,x_4, \quad \text{RMSE}(\hat{y}) = 0$$

Model #4 is already much better:

$$\hat{y}(x_1) := 2.0 + 0.35\,x_1 - 0.55\,x_2 + 0.5\,x_3 + 2.0\,x_4, \quad \text{RMSE}(\hat{y}) = 0$$

Assume we force all parameters β_i to have $\sum_{i=0}^{p} |\beta_i| \leq 3$.
Model #5 is again much better:

$$\hat{y}(x_1) := 0.5 + 1.125\,x_1 - 0.175\,x_2 - 0.45\,x_3 + 0.4\,x_4, \quad \text{RMSE}(\hat{y}) = 0$$

Outline

Linear Model

$$\hat{y}(x) := \beta_0 + \beta^T x = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$$

Polynomial Model

of degree 2:

$$\hat{y}(x) := \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \sum_{i=1}^{p} \sum_{j=i}^{p} \beta_{i,j} x_i x_j$$

e.g.,

$$\hat{y}(x_1, x_2) := \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{1,1} x_1^2 + \beta_{2,2} x_2^2 + \beta_{1,2} x_1 x_2$$

Polynomial Model

of degree 2:

$$\hat{y}(x) := \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \sum_{i=1}^{p} \sum_{j=i}^{p} \beta_{i,j} x_i x_j$$

of degree 3:

$$\hat{y}(x) := \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \sum_{i=1}^{p} \sum_{j=1}^{p} \beta_{i,j} x_i x_j + \sum_{i=1}^{p} \sum_{j=i}^{p} \sum_{k=j}^{p} \beta_{i,j,k} x_i x_j x_k$$

e.g.,

$$\hat{y}(x_1, x_2) := \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{1,1} x_1^2 + \beta_{2,2} x_2^2 + \beta_{1,2} x_1 x_2 \\ + \beta_{1,1,1} x_1^3 + \beta_{2,2,2} x_2^3 + \beta_{1,1,2} x_1^2 x_2 + \beta_{1,2,2} x_1 x_2^2$$

Polynomial Model

of degree 3:

$$\hat{y}(x) := \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \sum_{i=1}^{p} \sum_{j=1}^{p} \beta_{i,j} x_i x_j + \sum_{i=1}^{p} \sum_{j=i}^{p} \sum_{k=j}^{p} \beta_{i,j,k} x_i x_j x_k$$

of degree d:

$$\hat{y}(x) := \sum_{J \in \Delta_{p,d}} \beta_J x^J \qquad \text{where } x^J := \prod_{i=1}^{p} x_i^{J_i}, \quad J \in \Delta_{p,d}$$

$$\Delta_{p,d} := \{J \in \mathbb{N}^p \mid \sum_{i=1}^{p} J_i \leq d\}$$

Factorized Polynomial Models

of degree d:

$$\hat{y}(x) := \sum_{J \in \Delta_{p,d}} \beta_J x^J$$

with

$$\beta_J := \sum_{k=1}^{K} \phi_k^J, \quad \phi_k \in \mathbb{R}^p$$

Kernels

$$\hat{y}(x) := \beta_0 + \sum_{i=1}^{N} \alpha_i y_i k(x_i, x), \quad \text{with } (x_1, y_1), \ldots, (x_N, y_N) \in \mathcal{D}^{\text{tra}}$$

and a **kernel** $k : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}_0^+$, e.g.,

polynomial kernel

$$k(x, x') := (1 + x^T x')^d, \quad d \in \mathbb{N} \text{ degree}$$

radial basis function kernel

$$k(x, x') := e^{\gamma x^T x'}, \quad \gamma \in \mathbb{R}^+$$

Models — Many Fancy Names

prediction function	loss	regularization	model name
linear	L2	—	regression, least squares
linear	L2	L2	ridge regression
kernel	L2	—	kernel regression
linear	L2	L1	lasso
linear	L2	L1+L2	elastic net
linear/kernel	ϵ-insensitive	L2	support vector regression
⋮	⋮	⋮	⋮
linear	hinge	—	perceptron
linear/kernel	hinge	L2	support vector machine
linear/kernel	squared hinge	L2	L2 support vector machine
logistic(linear)	loglikelihood	—	logistic regression
logistic(linear)	loglikelihood	L2	logistic ridge regression
⋮	⋮	⋮	⋮

Outline

Sparse predictors

Predictor vectors $X\mathbb{R}^p$ are called **sparse**, if on average only a few of its components, say $p_{\text{nz}} < p$ are non-zero.

Sparse predictors

Predictor vectors $X\mathbb{R}^p$ are called **sparse**, if on average only a few of its components, say $p_{\text{nz}} < p$ are non-zero.

Examples:

- the products a customer bought in an online shop.
- the categories a document belongs to.
- ...

Sparse predictors

Predictor vectors $X\mathbb{R}^p$ are called **sparse**, if on average only a few of its components, say $p_{\text{nz}} < p$ are non-zero.

Examples:

- the products a customer bought in an online shop.
- the categories a document belongs to.
- ...

Sparse predictors

- can be **stored more compact** in $O(p_{\text{nz}}) < O(p)$ by storing only indices and values of non-zero components:

$$x = (5, 0, 0, 3, 4, 0, 0, 0, 0, 0) \in \mathbb{R}^{10} \leftrightarrow x = ((1, 5), (4, 3), (5, 4)) \in (\mathbb{N}\times\mathbb{R}$$

Sparse predictors

Predictor vectors $X\mathbb{R}^p$ are called **sparse**, if on average only a few of its components, say $p_{\text{nz}} < p$ are non-zero.

Examples:

- the products a customer bought in an online shop.
- the categories a document belongs to.
- ...

Sparse predictors

- can be **stored more compact** in $O(p_{\text{nz}}) < O(p)$ by storing only indices and values of non-zero components:

  $$x = (5,0,0,3,4,0,0,0,0,0) \in \mathbb{R}^{10} \leftrightarrow x = ((1,5),(4,3),(5,4)) \in (\mathbb{N}\times\mathbb{R}$$

- can be **multiplied faster** with a dense or sparse vector in $O(p_{\text{nz}}) < O(p)$:

  $$\beta^T x = \sum_{i=1}^{p} \beta_i x_i \leftrightarrow \beta^T x = \sum_{i=1}^{|x|} \beta_{x_{i,1}} x_{i,2}$$

Outline

1. Overfitting and Regularization

2. Prediction Functions

3. Sparse Predictors

4. Learning Algorithms

Objective Function

Learning a model means to find the parameters $\hat{\theta}$ with a **minimum of the objective function** f:

$$\hat{\Theta} := \arg\min_{\Theta} f(\Theta) := \frac{1}{|\mathcal{D}^{\text{train}}|} \sum_{(x,y)\in\mathcal{D}^{\text{train}}} \ell(y, \hat{y}(x;\Theta)) + \lambda R(\Theta)$$

with $\lambda \in \mathbb{R}_0^+$ fixed.

Objective Function

Learning a model means to find the parameters $\hat{\theta}$ with a **minimum of the objective function** f:

$$\hat{\Theta} := \arg\min_{\Theta} f(\Theta) := \frac{1}{|\mathcal{D}^{\text{train}}|} \sum_{(x,y)\in\mathcal{D}^{\text{train}}} \ell(y, \hat{y}(x;\Theta)) + \lambda R(\Theta)$$

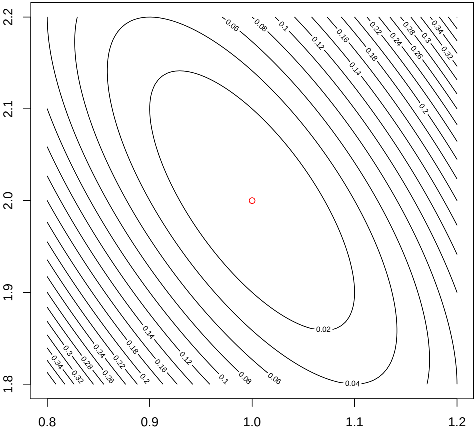
with $\lambda \in \mathbb{R}_0^+$ fixed.

- only for regression and ridge regression this is an unconstrained quadratic problem that easily can be solved as a system of linear equations

$$(X^T X + \lambda I)\hat{\beta} = X^T y$$

- in all other cases a solution needs to be found numerically.

Objective Function



$$\hat{\Theta} := \arg\min_{\Theta} f(\Theta) := \frac{1}{|\mathcal{D}^{\text{train}}|} \sum_{(x,y)\in\mathcal{D}^{\text{train}}} \ell(y, \hat{y}(x;\Theta)) + \lambda R(\Theta)$$

Gradient Descent

$$\text{choose } \Theta^{(0)} \in \mathbb{R}^p$$

$$\Theta^{(t+1)} := \Theta^{(t)} - \eta^{(t)} \frac{\partial f}{\partial \Theta}(\Theta^{(t)}), \quad t = 0, 1, 2, \ldots$$

$$\text{stop once } ||\frac{\partial f}{\partial \Theta}(\Theta^{(t)})|| < \epsilon$$

with

- $\eta^{(t)} \in \mathbb{R}^+$ called **step size** / **learning rate**.
- $\epsilon \in \mathbb{R}^+$ called **minimum gradient norm** / **stopping criterion**.

Gradient Descent

$$
\begin{aligned}
&\text{choose } \Theta^{(0)} \in \mathbb{R}^p \\
&\Theta^{(t+1)} := \Theta^{(t)} - \eta^{(t)} \frac{\partial f}{\partial \Theta}(\Theta^{(t)}), \quad t = 0, 1, 2, \ldots \\
&\text{stop once } \left|\left| \frac{\partial f}{\partial \Theta}(\Theta^{(t)}) \right|\right| < \epsilon
\end{aligned}
$$

with

- $\eta^{(t)} \in \mathbb{R}^+$ called **step size** / **learning rate**.
- $\epsilon \in \mathbb{R}^+$ called **minimum gradient norm** / **stopping criterion**.

$$
\frac{\partial f}{\partial \Theta}(\Theta) = \frac{1}{|\mathcal{D}^{\text{train}}|} \sum_{(x,y) \in \mathcal{D}^{\text{train}}} \frac{\partial \ell}{\partial \hat{y}}(y, \hat{y}(x; \Theta)) \frac{\partial \hat{y}}{\partial \Theta}(x; \Theta) + \lambda \frac{\partial R}{\partial \Theta}(\Theta)
$$

Gradient Descent

Example: logistic regression.

$$\ell(y,\hat{y}) = -y \log \hat{y} - (1-y)\log(1-\hat{y})$$

$$\frac{\partial \ell}{\partial \hat{y}}(y,\hat{y}) = -y\frac{1}{\hat{y}} - (1-y)\frac{-1}{1-\hat{y}} = \frac{\hat{y}-y}{\hat{y}\,(1-\hat{y})}$$

$$\hat{y}(x;\Theta) = \text{logistic}(\Theta^T x)$$

$$\frac{\partial \hat{y}}{\partial \Theta_j}(x;\Theta) = \text{logistic}(\Theta^T x) \cdot (1-\text{logistic}(\Theta^T x))x_j$$

$$R(\Theta) = \Theta^T\Theta$$

$$\frac{\partial R}{\partial \Theta_j}(\Theta) = 2\Theta_j$$

$$\rightsquigarrow \quad \frac{\partial f}{\partial \Theta}(\Theta) = \frac{1}{|\mathcal{D}^{\text{train}}|}\sum_{(x,y)\in\mathcal{D}^{\text{train}}}(\hat{y}(x;\Theta)-y)x + 2\lambda\Theta$$

Newton Algorithm

$$\text{choose } \Theta^{(0)} \in \mathbb{R}^p$$

$$\text{solve } \left(\frac{\partial^2 f}{\partial \Theta^2}(\Theta^{(t)})\right) d^{(t)} = -\frac{\partial f}{\partial \Theta}(\Theta^{(t)})$$

$$\Theta^{(t+1)} := \Theta^{(t)} - \eta^{(t)} d^{(t)}$$

$$\text{stop once } ||d^{(t)}|| < \epsilon$$

with

- $\eta^{(t)} \in \mathbb{R}^+$ called **step size** / **learning rate**.
- $\epsilon \in \mathbb{R}^+$ called **minimum gradient norm** / **stopping criterion**.

Stochastic Gradient Descent

Rewrite the objective as a big sum:

$$f(\Theta) = \sum_{i=1}^{n} f_i(\Theta),$$

$$f_i(\Theta) := \ell(y_i, \hat{y}(x_i; \Theta) + \frac{\lambda}{n} R(\Theta)$$

then minimize a summand at a time:

choose $\Theta^{(0)} \in \mathbb{R}^p$

pick uniformly at random $i^{(t)} \in \{1, \ldots, n\}$

$$\Theta^{(t+1)} := \Theta^{(t)} - \eta^{(t)} \frac{\partial f_{i^{(t)}}}{\partial \Theta}(\Theta^{(t)}), \quad t = 0, 1, 2, \ldots$$

stop once $||\Theta^{(t)} - \Theta^{(t-t_0)}|| < \epsilon$

Stochastic Gradient Descent

- Stochastic Gradient Descent (SGD) is as simple to derive and implement as full gradient descent.
- SGD **often converges much faster** than full gradient descent as parameters are updated more quickly.
- For stopping, lack of progress on several iterations (t_0) has to be observed.
- Often the regularization term is not spread uniformaly over all summand functions, but in a clever way s.t. f_i depends on as few Θ_j as possible (**sparse parameter updates**).