

Machine Learning

Lecture 2 - Regularized non-linear regression and classification

Mattias Villani

Department of Statistics
Stockholm University

Department of Computer and Information Science
Linköping University



mattiasvillani.com



@matvil



mattiasvillani

Lecture overview

■ Regularized regression

- ▶ Polynomial regression
- ▶ Spline regression
- ▶ L1 and L2 regularization
- ▶ Beyond L1 and L2

■ Regularized classification

- ▶ k -NN classification
- ▶ Classification trees
- ▶ Evaluating a classifier's performance

Polynomial regression

Polynomial regression

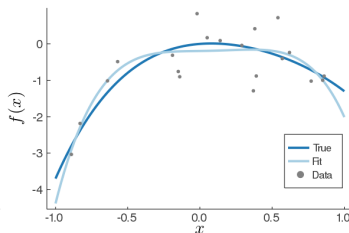
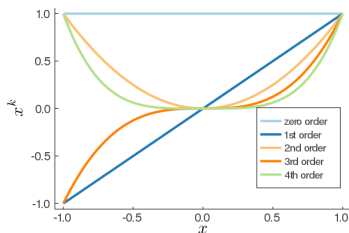
$$f(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_k x_i^k, \quad \text{for } i = 1, \dots, n.$$

$$y = X\beta + \varepsilon,$$

where i th row of X is

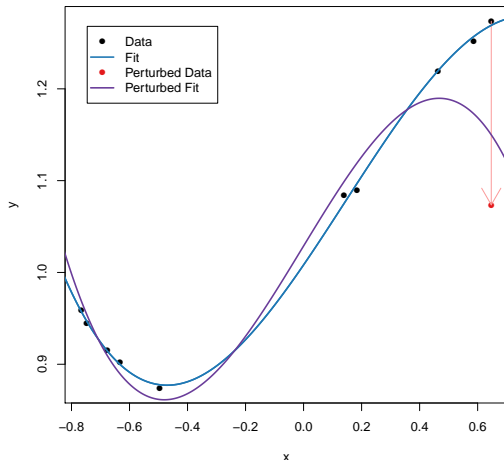
$$(1, x_i, x_i^2, \dots, x_i^k).$$

Still **linear in β** . Least squares: $\hat{\beta} = (X^T X)^{-1} X^T y$.



Polynomial regression is global

- Global fitting method: changes in a data point affect fit elsewhere:



Spline regression

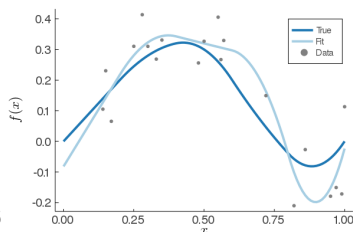
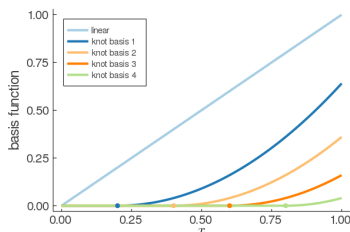
- Splines are local basis functions.
- **Truncated quadratic splines** with **knot locations** $\kappa_1, \dots, \kappa_m$:

$$b_j(x) = \begin{cases} (x - \kappa_j)^2 & \text{if } x > \kappa_j \\ 0 & \text{otherwise} \end{cases}$$

$$y = X\beta + \varepsilon,$$

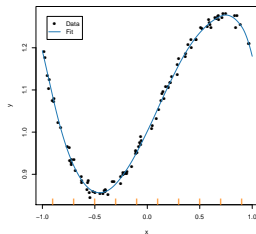
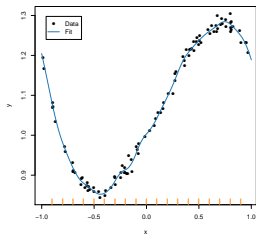
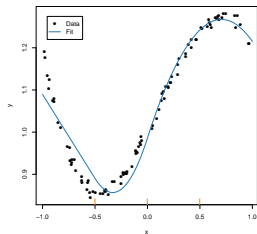
where i th row of X is

$$(1, x_i, b_1(x_i), \dots, b_m(x_i)).$$



Spline regression

■ See R notebook `SplinesR.Rmd`.



Natural cubic splines

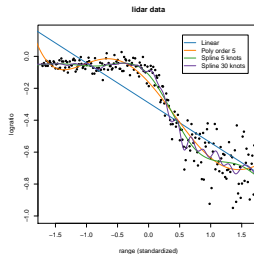
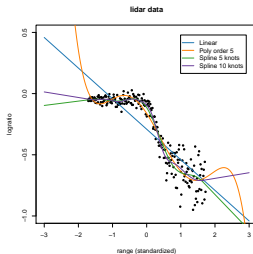
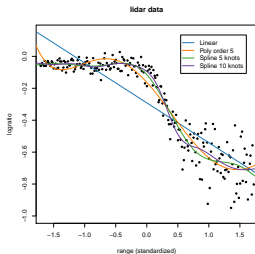
- Twice differentiable at the knots.
- Linear fit beyond the boundary.

```
library(SemiPar) # for the lidar data
library(splines) # for natural cubic splines, ns(), and B-splines, bs().

# linear model
lm(logratio ~ range, data = lidar)

# 5th degree polynomial model
lm(logratio ~ poly(range, 5), data = lidar)

# Natural cubic spline with 10 knots
lm(logratio ~ ns(range, knots = seq(-1.5, 1.5, length = 10)), data = lidar)
```



Additive models and interactions

- **Additive model** has no interactions

$$y = f_1(x_1) + f_2(x_2) + \dots + f_p(x_p) + \varepsilon$$

where function $f_j(x_j)$ is a spline for covariate x_j .

- Interactions in linear regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \varepsilon$$

- General **interactions**

$$y = f(x_1, x_2, \dots, x_p) + \varepsilon$$

e.g. thin plate splines with multi-dim knots $\kappa = (\kappa_1, \dots, \kappa_p)^\top$

$$f(x_1, x_2, \dots, x_p) = \|x - \kappa\|_2^2 \log \|x - \kappa\|_2.$$

- Partial interactions

$$y = f_1(x_1) + f_2(x_2) + f_{23}(x_2, x_3) + \varepsilon$$

- R code (mgcv package for gam, sp package for meuse data)

```
gam(y ~ s(longitude, latitude) + dist, data = meuse))
```


Ridge regression - L2-regularization

- Many features $\Rightarrow \hat{\beta} = (X^T X)^{-1} X^T y$ has high variance.
- $\hat{y} = X \hat{\beta}$ can **overfit** the data. Poor prediction on test data.
- **Regularization**: “soft restrictions” on estimators.
- **Ridge estimator** minimizes L2-penalized sum of squares

$$(y - X\beta)^T (y - X\beta) + \lambda \|\beta\|_2^2$$

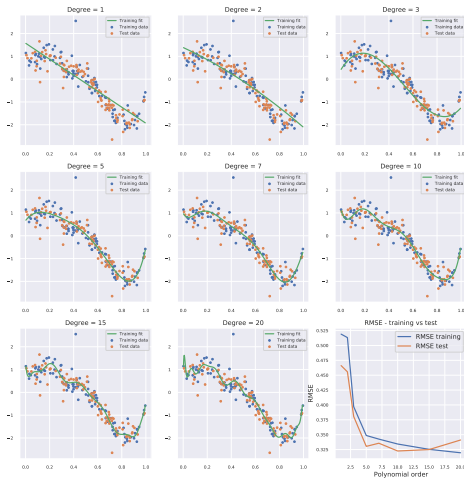
where $\|\beta\|_2^2 = \beta^T \beta = \sum_{j=1}^p \beta_j^2$ is **L₂-regularization**.

- Hyperparameter $\lambda > 0$ determined by predictive performance.
- Ridge regression estimator

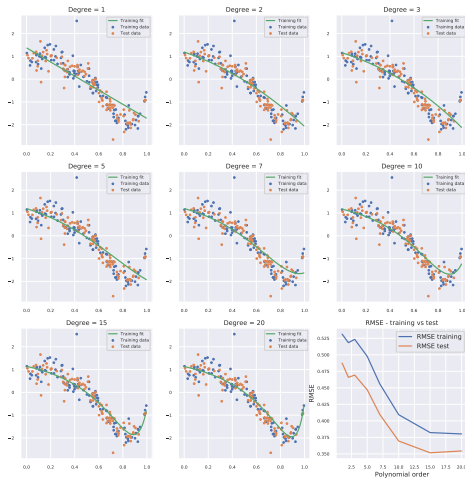
$$\hat{\beta}_{\text{ridge}}(\lambda) = (X^T X + \lambda I_n)^{-1} X^T y.$$

- $\hat{\beta}_{\text{ridge}}(\lambda)$ **shrinks** $\hat{\beta}$ toward zero as $\lambda \rightarrow \infty$.
- **Bias-variance trade-off**.

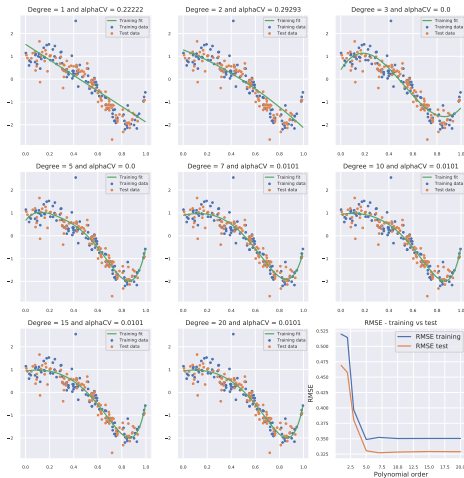
Polynomial regression without regularization



Polynomial regression, L_2 -regularization, $\lambda = 1$



Polynomial regression, L_2 -regularization, λ_{opt}



Lasso regression

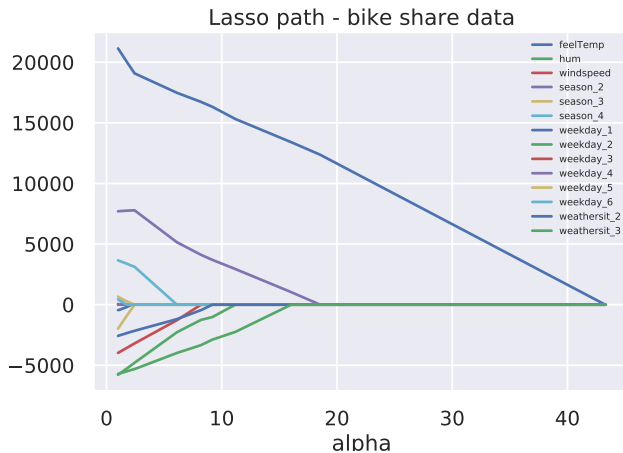
- **Lasso** estimator minimizes L1-penalized sum of squares

$$(y - X\beta)^T (y - X\beta) + \lambda \|\beta\|_1$$

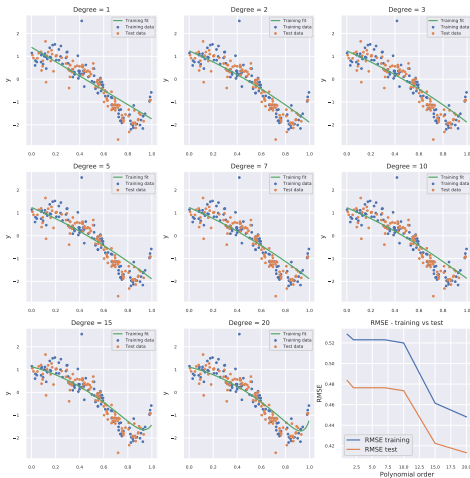
where $\|\beta\|_1 = \sum_{j=1}^p |\beta_j|$ is **L₁-regularization**.

- Lasso can shrink weights exactly to zero \Rightarrow **variable selection**.
- No explicit formula for $\hat{\beta}_{\text{lasso}}(\lambda)$.
- **LARS** is a super-fast algorithm for computing the entire **Lasso path**.

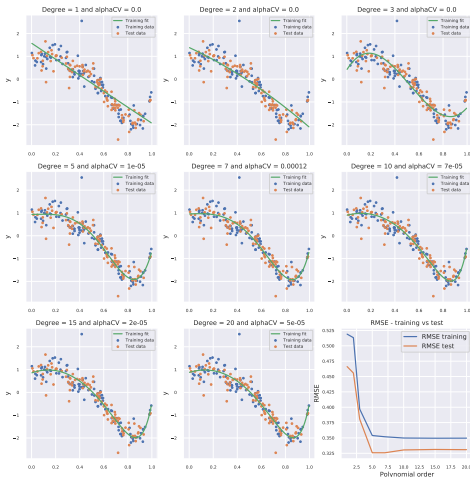
Lasso regression - bike sharing data



Polynomial regression, L_1 -regularization, λ large



Polynomial regression, L_1 -regularization, λ_{opt}



Bayesian interpretation of regularization

- Bayesian inference is based on the **posterior distribution**

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

- Interpret a **regularization penalty as a log prior**

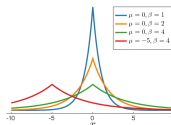
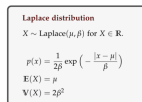
$$\log p(\beta|y, X) = \underbrace{\log p(y|\beta, X)}_{\text{log-likelihood/sum of squares}} + \underbrace{\log p(\beta)}_{\text{penalty}} + \text{constant}$$

- **Ridge** is equivalent to using a Normal prior

$$\beta_i | \sigma^2 \stackrel{\text{iid}}{\sim} \text{Normal}\left(0, \frac{\sigma^2}{\lambda}\right)$$

- **Lasso** is equivalent to posterior mode under Laplace prior

$$\beta_i | \sigma^2 \stackrel{\text{iid}}{\sim} \text{Laplace}\left(0, \frac{\sigma^2}{\lambda}\right)$$



Model complexity

- **Fitted values** for linear models

$$\hat{y} = X \hat{\beta} = \underbrace{X(X^T X)^{-1} X^T}_H y = Hy.$$

- **Hat matrix**

$$H_{n \times n} = X(X^T X)^{-1} X^T.$$

- The i th row of H : how \hat{y}_i depends on the n data points.
- A **linear smoother** is any fitting method of the form

$$\hat{y} = Hy.$$

- Ex: poly reg, splines, nearest neighbor, ridge regression ...
- **Degrees of freedom**, $\text{tr}(H)$, measures complexity.
- Sanity check: linear regression with p covariates: $\text{tr}(H) = p$.

Beyond Ridge and Lasso

- **Ideal shrinkage:**

- ▶ hard shrinkage of coefficients on unimportant features
- ▶ no/small shrinkage of coefficients on important features

- Ridge: shrinks all coefficients similarly.

- Lasso: can allow some features to have larger coefficients.

- **Elastic-net** combines L1 and L2 penalties:

$$\lambda \left[(1 - \alpha) \|\boldsymbol{\beta}\|_2^2 / 2 + \alpha \|\boldsymbol{\beta}\|_1 \right]$$

- Ridge, Lasso and elastic net apply **global shrinkage**.

- Better variable selection with **global-local shrinkages**

- ▶ **Global shrinkage**, λ , is baseline shrinkage for all features
- ▶ **Local shrinkage**, τ_1, \dots, τ_p , for each feature.
- ▶ Total shrinkage on j th feature: $\lambda \tau_j$.

- Example: **Horseshoe regularization**.

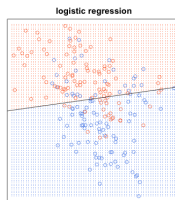
Logistic regression

- **Binary logistic regression** for $y \in \{0, 1\}$ or $y \in \{-1, +1\}$

$$\Pr(y = 1|x) = \frac{\exp(x^\top \beta)}{1 + \exp(x^\top \beta)}.$$

- **Log odds is linear** in covariates

$$\log \frac{\Pr(y = 1|x)}{\Pr(y = 0|x)} = x^\top \beta$$



Non-linear logistic regression

■ Linear logistic regression

$$\Pr(y = 1|x) = \frac{\exp(x^\top \beta)}{1 + \exp(x^\top \beta)}.$$

■ Non-linear logistic regression

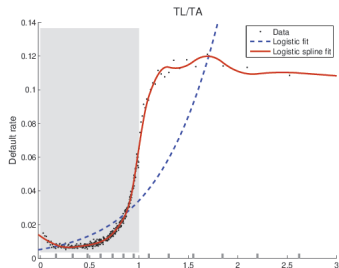
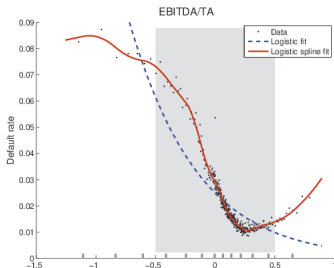
$$\Pr(Y_i = 1|x) = \frac{\exp(f(x_i))}{1 + \exp(f(x_i))}$$

where $f(x)$ is some potentially non-linear function, e.g.

- **Polynomials, Splines**
- **Deep neural nets**
- **Gaussian Processes**

Predicting firm bankruptcy

- Data from $\sim 250,000$ Swedish corporations.
- **Features**: profits, liquidity, debt + macro variables.
- “Big data” \Rightarrow **non-linearities** are visible by the eye.
- Model: **logistic regression** with **additive splines**.
- Substantially improved predictive performance with splines.

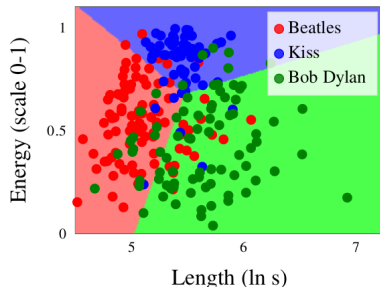


Multi-class logistic regression

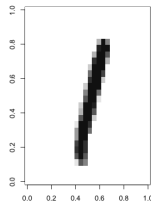
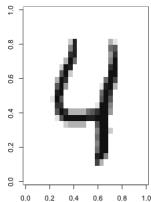
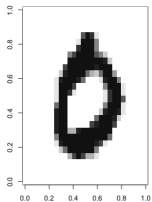
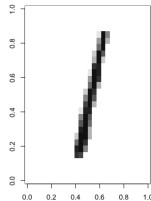
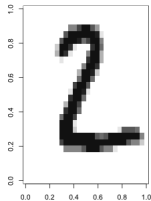
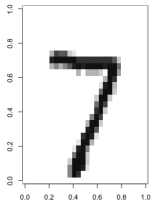
- **Multi-class logistic regression** for $y \in \{1, 2, \dots, C\}$

$$\Pr(y = c|x) = \frac{\exp(x^\top \beta_c)}{\sum_{j=1}^C \exp(x^\top \beta_j)}.$$

- Non-identified: likelihood invariant to scaling $\beta_c \rightarrow k\beta_c$ for all $c = 1, \dots, C$ for scalar k .
 - ▶ Statistics: set $\beta_1 = 0$. First class is the reference.
 - ▶ ML: no restrictions, use regularization to identify.



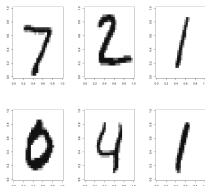
Classifying handwritten digits



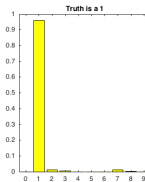
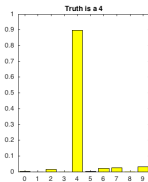
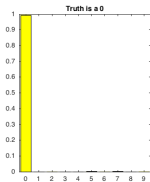
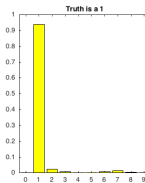
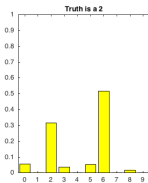
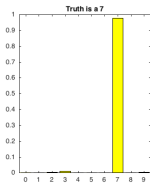
Classifying handwritten digits

- **Raw data**: gray intensity (0-255) in the $28 \times 28 = 784$ pixels.
- **Training** data: 60000 images. **Test** data: 10000 images.
- **Simple features**: 784 pixel intensity covariates.
- **Multi-class** problem: predict class $c \in \{0, 1, \dots, 9\}$.
- **Multinomial regression**

$$\Pr(y = c | \mathbf{x}) = \frac{\exp(\mathbf{x}^\top \beta_c)}{\sum_{j=1}^C \exp(\mathbf{x}^\top \beta_j)}.$$



Classifying handwritten digits



Handwritten digits 10000 training examples

		Truth									
		0	1	2	3	4	5	6	7	8	9
Decision	0	958	0	8	3	1	7	10	0	7	9
	1	0	1116	3	1	1	5	3	23	9	9
	2	1	2	920	21	5	5	9	22	7	2
	3	0	2	10	915	0	34	1	1	14	10
	4	1	0	16	0	908	9	10	12	11	46
	5	11	3	3	31	2	795	15	1	31	12
	6	6	4	20	2	11	16	909	0	13	1
	7	1	0	15	13	4	5	0	938	9	22
	8	2	8	34	16	2	12	1	5	859	5
	9	0	0	3	8	48	4	0	26	14	893

Handwritten digits 60000 training examples

		Truth									
		0	1	2	3	4	5	6	7	8	9
Decision	0	966	0	8	1	1	7	9	2	4	6
	1	0	1121	1	1	0	2	3	13	7	7
	2	2	2	957	13	5	4	4	21	7	0
	3	0	2	9	947	0	29	1	3	12	10
	4	0	0	12	1	940	5	5	9	8	32
	5	6	1	3	19	1	816	9	1	24	9
	6	4	4	13	1	7	12	926	0	10	1
	7	1	0	9	10	2	2	0	954	5	13
	8	1	4	17	11	2	10	1	3	892	4
	9	0	1	3	6	24	5	0	22	5	927

AI is getting better over time - handwritten digits

Time: 1998 — — — — — — — — → Today

	Logistic	K-nearest	SVM	3-layer NN	ConvNet
Error rate:	12%	5%	1.4%	1.53%	0.4%

Detecting fraudulent banknotes

- Dataset with 1372 photographed banknotes. 610 fake.
- Raw data: $400 \times 400 = 160000$ gray-scale pixels.
- The 160000 pixel variables are condensed to four **features**:
 - ▶ variance of Wavelet Transformed image
 - ▶ skewness of Wavelet Transformed image
 - ▶ kurtosis of Wavelet Transformed image
 - ▶ entropy of image
- **Deep learning** on raw images (more later!).



Detecting fraudulent banknotes

- 1000 images for training. Predictions on 372 test images.
- Logistic regression

$$\Pr(\text{Fraud} = \text{True} | \text{features}) = \frac{\exp(\mathbf{x}^\top \beta_c)}{\sum_{j=1}^C \exp(\mathbf{x}^\top \beta_j)}$$

- **Decision**: signal fraud if $\Pr(\text{Fraud} = \text{True} | \text{Features}) > 0.5$.
- **Confusion matrix**

		Truth	
		No fraud	Fraud
Decision	No Fraud	208	1
	Fraud	3	160

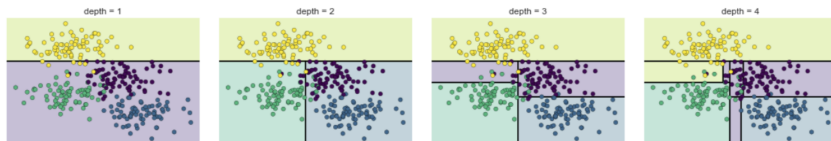
Classification trees

- **Multi-class classification trees.** Probability of class c in rectangle R_ℓ with n_ℓ observations

$$\hat{p}_{\ell,c} = \frac{1}{n_\ell} \sum_{i: x_i \in R_\ell} \mathbb{I}(y_i = c), \quad c = 1, \dots, C.$$

- **Predicted class** for $x_\star \in R_\ell$: **majority vote**

$$\hat{y}(x_\star) = \arg \max_c \hat{p}_{\ell,c}$$



Classification trees

- **Prune** the tree by collapsing non-terminal nodes to minimize

$$\frac{1}{n} \sum_{i=1}^n L(\hat{y}(x_i), y_i) + \eta |T|$$

- **Mis-classification rate** as loss function:

$$\frac{1}{n} \sum_{i=1}^n L(\hat{y}(x_i), y_i) = \frac{1}{n} \sum_{\ell=1}^{|T|} \sum_{i: x_i \in R_\ell} I(y_i \neq \hat{y}(x_i)) = \frac{1}{n} \sum_{\ell=1}^{|T|} n_\ell (1 - \hat{p}_{\ell, \hat{y}(x_i)})$$

- Negative **Log-likelihood (cross-entropy)** as loss function:

$$\sum_{i=1}^n L(\hat{y}(x_i), y_i) = \sum_{\ell=1}^{|T|} n_\ell \left(- \sum_{c=1}^C \hat{p}_{\ell, \hat{y}(x_i)} \log \hat{p}_{\ell, \hat{y}(x_i)} \right)$$