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**

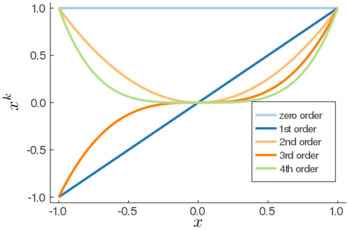
$$f(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_k x_i^k, \quad \text{for } i = 1, \ldots, n.$$

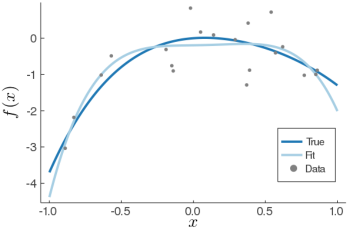
$$\mathrm{y} = \mathrm{X}\beta + \varepsilon,$$

where ith row of X is

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

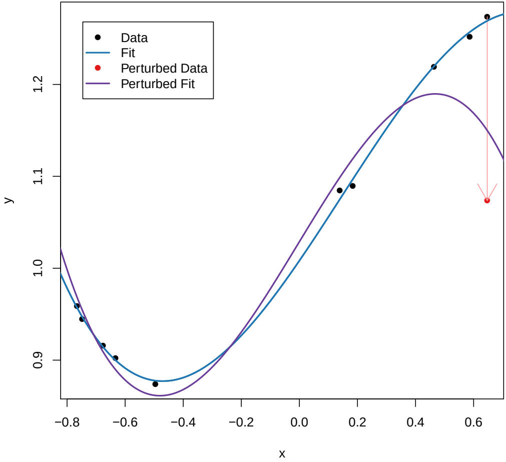
- Still **linear in β**. Least squares: $\hat{\boldsymbol{\beta}} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y}$.





Polynomial regression is global

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



Spline regression

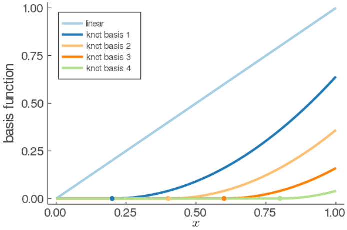
- Splines are local basis functions.
- **Truncated quadratic splines** with **knot locations** $\kappa_1, ..., \kappa_m$:

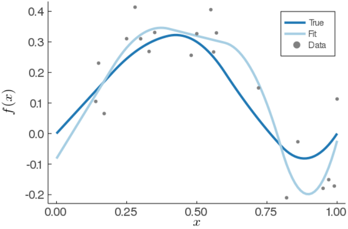
$$b_j(x) = \begin{cases} (x - \kappa_j)^2 & \text{if } x > \kappa_j \\ 0 & \text{otherwise} \end{cases}$$

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

where ith row of X is

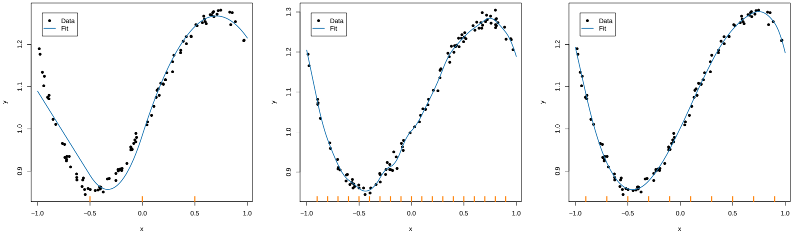
$$(1, x_i, b_1(x_i), ..., b_m(x_i)) .$$





Spline regression

- See R notebook `SplinesR.Rmd`.



Natural cubic splines

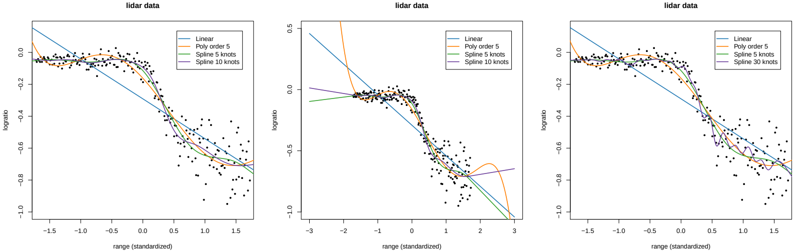
- Twice differentiable at the knots.
- Linear fit beyond the boundary.

```r
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

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

  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**

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

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

$$f(x_1, x_2, \ldots, x_p) = \|\mathrm{x} - \boldsymbol{\kappa}\|_2^2 \log \|\mathrm{x} - \boldsymbol{\kappa}\|_2 .$$

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

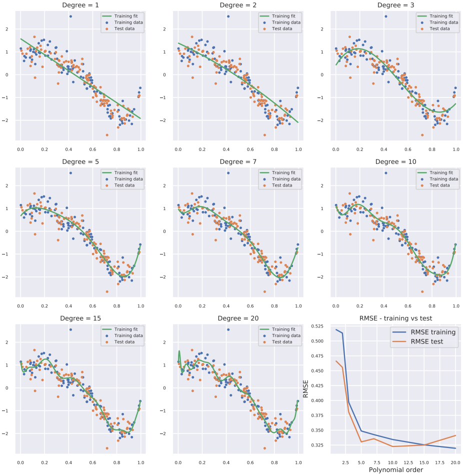
$$(\mathrm{y} - \mathrm{X}\boldsymbol{\beta})^T (\mathrm{y} - \mathrm{X}\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_2^2$$

  where $\|\boldsymbol{\beta}\|_2^2 = \boldsymbol{\beta}^T\boldsymbol{\beta} = \sum_{j=1}^{p} \beta_j^2$ is **L_2-regularization**.
- Hyperparameter $\lambda > 0$ determined by predictive performance.
- Ridge regression estimator

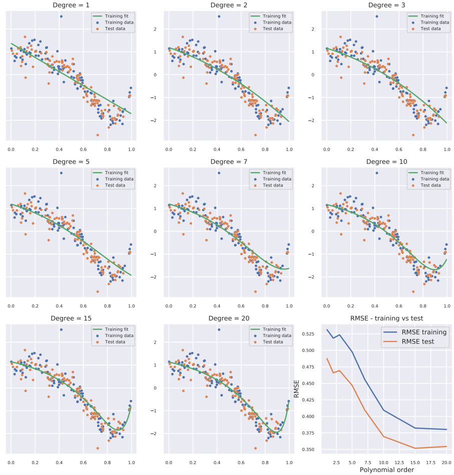
$$\hat{\boldsymbol{\beta}}_{\mathrm{ridge}}(\lambda) = (\mathrm{X}^T\mathrm{X} + \lambda I_n)^{-1}\mathrm{X}^T\mathrm{y}.$$

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

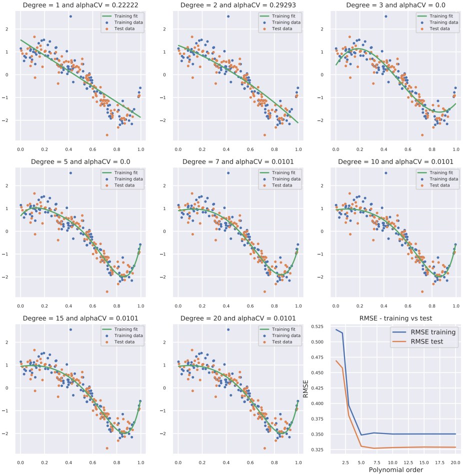
Polynomial regression without regularization



Polynomial regression, L$_2$-regularization, $\lambda = 1$



Polynomial regression, L$_2$-regularization, λ_{opt}



Lasso regression

- Lasso estimator minimizes L1-penalized sum of squares

$$(\mathrm{y} - \mathrm{X}\boldsymbol{\beta})^T (\mathrm{y} - \mathrm{X}\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1$$

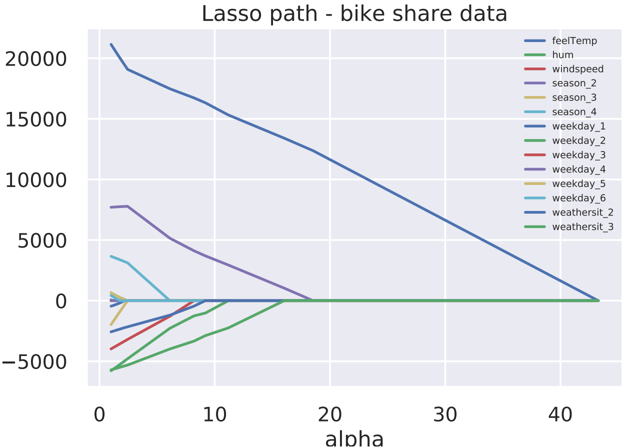
  where $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$ is L_1-**regularization**.

- Lasso can shrink weights exactly to zero ⇒ **variable selection**.

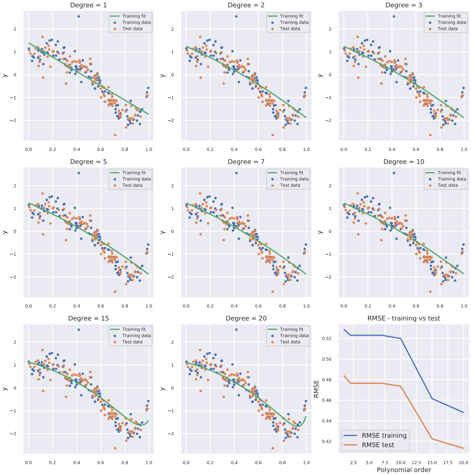
- No explicit formula for $\hat{\boldsymbol{\beta}}_{\text{lasso}}(\lambda)$.

- **LARS** is a super-fast algorithm for computing the entire **Lasso path**.

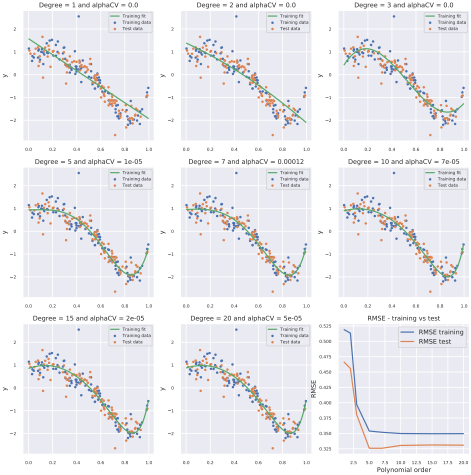
Lasso regression - bike sharing data



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

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

- Interpret a regularization penalty as a log prior

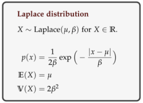
$$\log p(\boldsymbol{\beta}|\mathrm{y},\mathrm{X}) = \underbrace{\log p(\mathrm{y}|\boldsymbol{\beta},\mathrm{X})}_{\text{log-likelihood/sum of squares}} + \underbrace{\log p(\boldsymbol{\beta})}_{\text{penalty}} + \text{constant}$$

- Ridge is equivalent to using a Normal prior

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

- Lasso is equivalent to posterior mode under Laplace prior

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



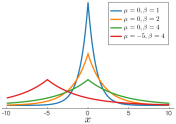
Laplace distribution

$X \sim \text{Laplace}(\mu,\beta)$ for $X \in \mathbb{R}$.

$$p(x) = \frac{1}{2\beta}\exp\left(-\frac{|x-\mu|}{\beta}\right)$$

$\mathbb{E}(X) = \mu$

$\mathbb{V}(X) = 2\beta^2$



Model complexity

- **Fitted values** for linear models

$$\hat{\mathrm{y}} = \mathrm{X}\hat{\boldsymbol{\beta}} = \underbrace{\mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T}_{H}\mathrm{y} = \mathrm{H}\mathrm{y}.$$

- **Hat matrix**

$$\underset{n\times n}{\mathrm{H}} = \mathrm{X}(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T.$$

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

$$\hat{\mathrm{y}} = \mathrm{H}\mathrm{y}.$$

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

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:

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

- 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**, $\tau_1, \ldots, \tau_p$, for each feature.
  - Total shrinkage on jth feature: $\lambda\tau_j$.
- Example: **Horseshoe regularization**.

Logistic regression

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

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

- **Log odds is linear** in covariates

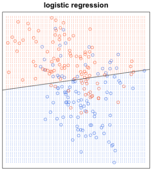
$$\log \frac{\Pr(y = 1|\mathrm{x})}{\Pr(y = 0|\mathrm{x})} = \mathrm{x}^\top \beta$$

- Decision boundary for binary classification: the x that solve

$$\Pr(y = 1|\mathrm{x}) = \Pr(y = 0|\mathrm{x})$$

- Logistic regression is a **linear classifier**

$$\frac{\exp(\mathrm{x}^\top \beta)}{1 + \exp(\mathrm{x}^\top \beta)} = \frac{1}{1 + \exp(\mathrm{x}^\top \beta)} \Longleftrightarrow \exp(\mathrm{x}^\top \beta) = 1 \Longleftrightarrow \mathrm{x}^\top \beta = 0$$



Non-linear logistic regression

- **Linear logistic regression**

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

- **Non-linear logistic regression**

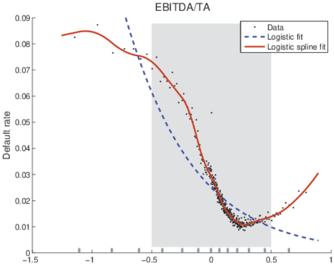
$$\Pr(Y_i = 1|\mathrm{x}) = \frac{\exp\left(f\left(\mathrm{x}_i\right)\right)}{1 + \exp\left(f\left(\mathrm{x}_i\right)\right)}$$

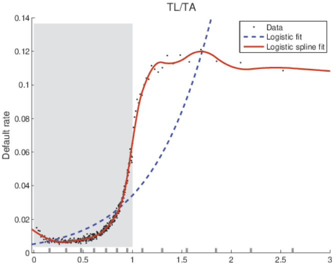
where $f(\mathrm{x})$ is some potentially non-linear function, e.g.

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

Predicting firm bankruptcy

- Data from $\approx 250,000$ 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.



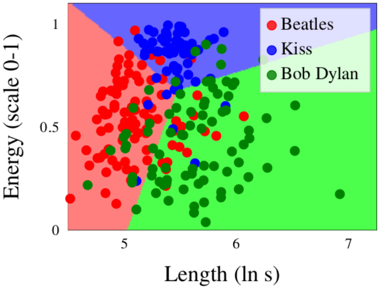


Multi-class logistic regression

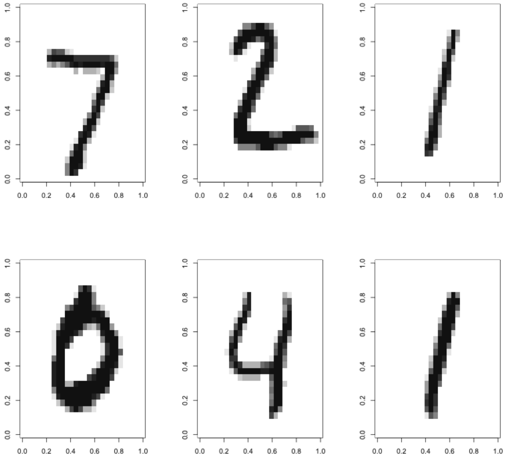
- **Multi-class logistic regression** for $y \in \{1, 2, \ldots, C\}$

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

- Non-identified: likelihood invariant to scaling $\beta_c \to k\beta_c$ for all $c = 1, \ldots, C$ for scalar k.
  - Statistics: set $\boldsymbol{\beta}_1 = 0$. First class is the reference.
  - ML: no restrictions, use regularization to identify.



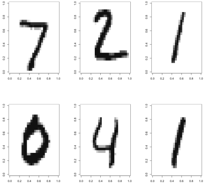
Classifying handwritten digits



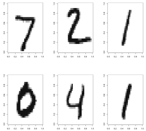
Classifying handwritten digits

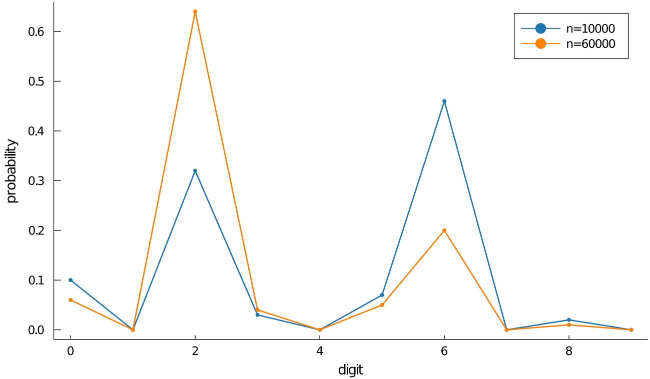
- **Raw data**: gray intensity (0-255) in the $28 \times 28 = 784$ pixels.
- **Training** data: 60000 images. **Test** data: 10000 images.
- **Simple features**: 784 pixel intensity covariates.
- **Multi-class** problem: predict class $c \in \{0, 1, ..., 9\}$.
- **Multinomial regression**

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



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

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

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

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

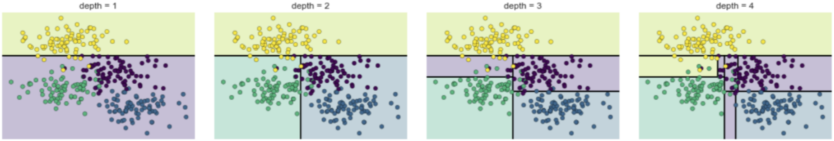
Classification trees

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

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

- **Predicted class** for $\mathrm{x}_\star \in R_\ell$: **majority vote**

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



Classification trees

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

$$\frac{1}{n}\sum_{i=1}^{n} L\left(\hat{y}(\mathrm{x}_i), y_i\right) + \eta\,|T|$$

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

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

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

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

Pimp up your generalized linear model

- Poisson regression for count data

$$
\begin{aligned}
y_i | \mathrm{x}_i, \boldsymbol{\beta}_i &\sim \mathrm{Pois}(\lambda_i) \\
\lambda_i &= \exp(\mathrm{x}_i^\top \boldsymbol{\beta})
\end{aligned}
$$

- Learn $\boldsymbol{\beta}$ using the log-likelihood as the loss function.
- Generalizations:
  - replace linear term $\mathrm{x}_i^\top \boldsymbol{\beta}$ with **non-linear** (splines, trees etc).
  - replace the (inverse) **link function** $\exp()$ with other function.
  - use other members of **exponential family**, e.g gamma distribution for positive continuous regression response.
  - truncated, censored, missing data etc.
- Machine learning is really just about **flexible regularized** models with a (probabilistic) **prediction**/decision focus.