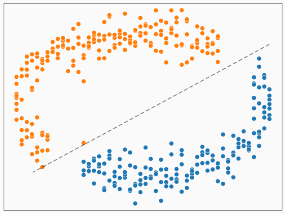
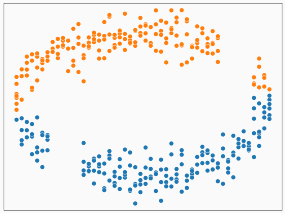
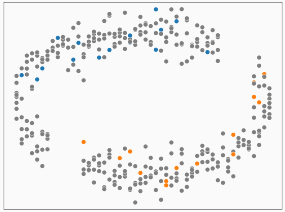
Introduction to statistical learning

Wojciech Wideł

What is statistical learning?







What is statistical learning?

- **Supervised**: the available dataset contains *inputs labelled with the corresponding outputs*; the goal is to build a model for predicting, or estimating, an output based on an input
- **Unsupervised**: there are inputs, but no supervising outputs; the goal is to build a model that transforms an input into something that can be used to solve a practical problem
- **Semisupervised**: the dataset contains both labelled and unlabelled inputs, usually with the latter constituting the vast majority; the goal is the same as in the case of the supervised learning
- **Reinforcement**: create a policy (model) that, given a description of the environment (its state), outputs an action to execute in that state; arises in problems with sequential decision making and long-term goals

Regression vs classification

- **Regression**: explained variable is continuous; the objective is to predict/estimate its values corresponding to observations
- **Classification**: explained variable is discrete; the objective is to categorize observations

Supervised learning, regression

$$Y = f(X) + \epsilon$$

- Y – observed quantitative response
- $X = (X_1, X_2, \ldots, X_p)$ – observed p predictors
- f – fixed but unknown function; systematic information that X provides about Y
- $\epsilon \sim (0, \sigma^2)$ – random error, independent of X

Goal: create $\widehat{f}$, an estimate of f.

Why estimate f?

- **Prediction**: the exact form of $\widehat{f}$ is irrelevant, what matters is the ability to perform accurate predictions for new inputs
- **Inference**: the goal is to understand the ways in which Y is affected as $X_1, \ldots, X_p$ change. Typical questions of interest include the following.
  - Which predictors are associated with the response?
  - What is the relationship between the response and each predictor?
  - Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is it more complicated?

How good can the estimate of f be?

With $\widehat{Y} = \widehat{f}(X)$ being an estimate of $Y = f(X) + \epsilon$, we have

$$\begin{aligned}
\mathbb{E}(Y - \widehat{Y})^2 &= \mathbb{E}[f(X) + \epsilon - \widehat{f}(X)]^2 \\
&= \mathbb{E}[(f(X) - \widehat{f}(X))^2 + 2\epsilon(f(X) - \widehat{f}(X)) + \epsilon^2] \\
&= \underbrace{(f(X) - \widehat{f}(X))^2}_{\text{reducible error}} + \underbrace{\sigma^2}_{\text{irreducible error}}
\end{aligned}$$

Intuitively, $\sigma^2 > 0$ because ϵ may contain unmeasurable variables useful in predicting Y or some unmeasurable variation.

How to estimate f?

Having access to training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, with $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, one can use

- **parametric methods**, which (1) make an assumption about the functional form of f and (2) apply some procedure to the training data that yields estimates of f's parameters, most commonly by minimizing some loss function, or
- **non-parametric methods**, which do not assume any parametric form for the relationship between predictors and response; predictions are derived from the training data.

How to estimate f? Methods examples

- Parametric: **linear regression**

$$f_\beta(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p,$$
$$L(\beta) := \frac{1}{n}\sum_{i=1}^{n}(y_i - f_\beta(x_i))^2, \widehat{\beta} := \underset{\beta}{\operatorname{argmin}}\, L(\beta)$$

- Non-parametric: **k-nearest neighbours regression**

$$\widehat{f}(x) = \frac{1}{k}\sum_{x_i \in N_k(x)} y_i,$$

where $N_k(x)$ denotes the set of k training observations closest to x with respect to some metric.

How to estimate f? Parametric vs non-parametric methods

- Parametric methods
  - The problem of estimating f is reduced down to estimation of a set of parameters
  - If the model is too simple, the fit will not be good
  - If the model is too complex, the risk of overfitting arises
- Non-parametric methods
  - Since no form is assumed, they have the potential to accurately fit a wider range of possible shapes for f
  - Large number of observations is needed in order to obtain an accurate estimate for f

How to estimate f? Gradient descent

Gradient descent: iterative algorithm for finding local minima of a differentiable function. In the ML context, applied to a loss function in search of model's parameters that minimize the loss.

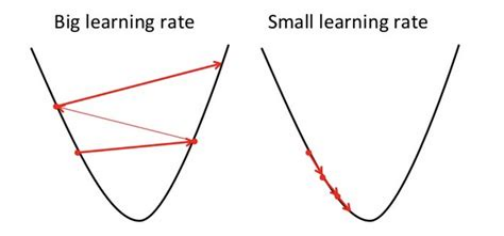
For a differentiable $f\colon \mathbb{R}^m \to \mathbb{R}$ and $a \in \mathbb{R}^m$, the **gradient** of f at a, denoted $\nabla f(a)$, is the vector of partial derivatives of f at a, namely

$$\nabla f(a) := \left(\frac{\partial f}{\partial x_1}(a), \ldots, \frac{\partial f}{\partial x_m}(a) \right)^T.$$

Algorithm:

- pick a random x^0 as the potential minimum of f
- $x^{n+1} \leftarrow x^n - \underbrace{\gamma \nabla f(x^n)}_{\text{learning step}}$, for $n \in \{0, 1, \ldots, N\}$; γ is called the **learning rate**

Notes on gradient descent



- If f is convex and $\gamma < 1$, GD will eventually converge to the global minimum
- If f is not convex, GD might get stuck in a critical point that is not a local minimum
- If the learning rate γ is too big, GD might oscillate around a minimum
- If the learning rate is too small, GD might take very long time to converge
- There are many modifications of GD out there (e.g., with adaptive γ)

Malfitting and the bias-variance trade-off

Assumptions: test data available, model accuracy metric $\mathcal{AM}$ selected.

- **Overfitting/high variance**: model achieves very good values of $\mathcal{AM}$ on the training set but very poor ones on the test set.
  The variance is an error of the model due to its sensitivity to small fluctuations in the training set.
- **Underfitting/high bias**: model achieves poor values of $\mathcal{AM}$ on the training set.
  The bias is an error arising from erroneous assumptions in the model (e.g., model assumes linear relationship between features and response, but in fact it is far from linear).

Goal: create a model with low bias and low variance.

Causes of malfitting

- Overfitting
  - Model is too complex for the data
  - Too many features used but a small number of training observations

  In consequence, model learns the training data, not a general relationship.
- Underfitting
  - Model is too simple for the data (e.g., linear in the case of a non-linear relationship)
  - Features do not provide enough information (too few features used or non-informative ones)

  In consequence, model is unable to learn a meaningful relationship.

Overfitting prevention: regularization

Idea: modify the loss function by adding a penalizing term whose value is higher when the model is more complex.

- **L1/lasso regularization**, example of linear regression:

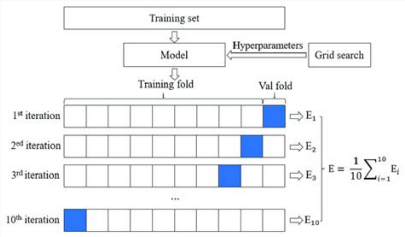
$$L_1(\beta) := \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \alpha\sum_{j=1}^{n}|\beta_j|$$

  For sufficiently large values of the **tuning parameter** $\alpha > 0$, the l_1 penalty forces some of the coefficients estimates to be exactly equal to zero. Thus, it performs variable selection, possibly increasing explainability of the model.

- **L2/ridge regularization**, example of linear regression:

$$L_2(\beta) := \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \alpha\sum_{j=1}^{n}\beta_j^2$$

Overfitting prevention: cross-validation



k-fold cross-validation:

- Partition training data D randomly into k sets, say, $D_1, \ldots, D_k$.
- For $i \in \{1, \ldots, k\}$, train the model on D_i, compute its predictions accuracy $\mathcal{A}_i$ on $D \setminus D_i$.
- Use the average accuracy $\frac{1}{k} \sum_{i=1}^{k} \mathcal{A}_i$ as the estimate for the accuracy on the test set.
- Pick the model with the highest average accuracy.

Data: preprocessing numerical features

- **Missing values treatment**
  - Replace, e.g., with the feature's median or mean.
  - Reconstruct, e.g., interpolate from neighbours in the case of time series.
  - Create a predictive model to predict missing values.
  - Replace variable with missing with indicator variables, e.g., with missing in *sex*, introduce *declared_sex_male* and *declared_sex_female*.
  - Remove corresponding observations from the dataset.
- **Outliers treatment** (investigate for possible errors in data sources!)
  - If not many, exclude them from training.
  - Otherwise, clip the values, e.g., to the 5th - 95th percentile range.
- **Scaling** is important, since it usually **speeds up the convergence of gradient descent** (and sampling in the Bayesian context); treat it as a hyperparameter (e.g., standardization, $\tilde{x}^{(j)} := (x^{(j)} - \bar{x}^{(j)})/sd(x^{(j)})$ vs normalization, $\tilde{x}^{(j)} := (x^{(j)} - \min x^{(j)})/(\max x^{(j)} - \min x^{(j)})$).

Data: feature engineering

Goal: create informative features from the available data, ones that will help the model make more accurate predictions. Use domain knowledge, be creative.

- Try **non-linear transformations** of numerical features, e.g., x^2, x^3, $\sqrt[3]{x}$ or, in the case of non-negative features, $\ln(1+x)$, $\sqrt{x+2/3}$, $1/(1+x)$.
- Try **combinations of features**: products, quotients, concatenations (of categorical variables).
- **One–hot encoding**: for a categorical variable x assuming values in {val1, ..., valk}, introduce binary variables x_is_val1, ..., x_is_valk (can be achieved in Python using `pandas.get_dummies`).
- **Binning** (or **bucketing**): if the exact value of a numerical variable seems not to matter, the variable can be converted into categorical ones, e.g., instead of age, could use age_below_18, age_in_18_25, ..., age_over_80.

Summary

1. Select potential classes of models to be used (regression vs classification, predictive vs explanatory, parametric vs non-parametric).
2. Explore data in search of relationships.
3. Preprocess data (missing, outliers).
4. Feature engineering (transformations, encodings, scalings).
5. Model selection (cross-validation).
6. Analyse results produced by the best model. Are they satisfactory? Is there any bias? Does the model struggle with specific data? If yes, go back to step 4.

References

- A. Burkov, *The Hundred-Page Machine Learning Book*
- G. James, D. Witten, T. Hastie and R. Tibshirani, *An Introduction to Statistical Learning*