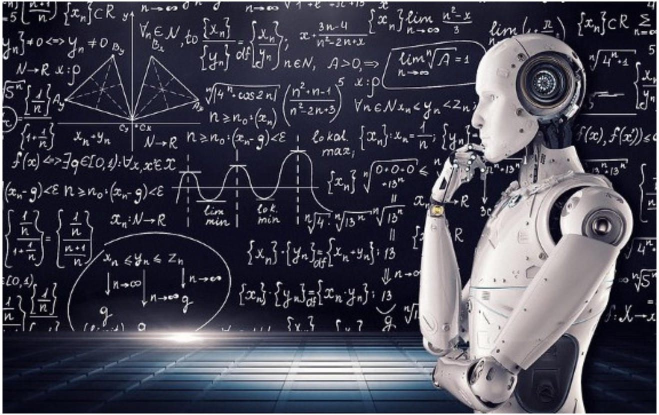
Common Machine Learning Algorithms



https://www.vpnsrus.com/

CONTENTS

LINEAR MODELS

LINEAR MODELS – FUNCTIONALITY

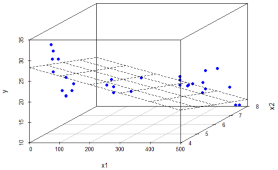
SUPERVISED PARAMETRIC WHITE-BOX

General idea Represent target as function of linear predictor $\boldsymbol{\theta}^T\mathbf{x}$

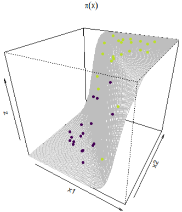
Hypothesis space

$\mathcal{H} = \left\{f : \mathcal{X} \to \mathbb{R} \mid f(\mathbf{x}) = \phi(\boldsymbol{\theta}^\top \mathbf{x})\right\}$, with suitable transformation $\phi(\cdot)$, e.g.,

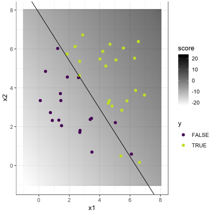
- Identity $\phi(\boldsymbol{\theta}^\top \mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$ $\Rightarrow$ **linear regression**
- Logistic sigmoid function $\phi(\boldsymbol{\theta}^T\mathbf{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^T\mathbf{x})} =: \pi(\mathbf{x} \mid \boldsymbol{\theta})$ $\Rightarrow$ **(binary) logistic regression**
  - Probability $\pi(\mathbf{x} \mid \boldsymbol{\theta}) = \mathbb{P}(y = 1 \mid \mathbf{x})$ of belonging to one of two classes
  - Separating hyperplane via decision rule (e.g., $\hat{y} = 1 \Leftrightarrow \pi(\mathbf{x}) > 0.5$)



Linear regression hyperplane



Logistic function for bivariate input



Separating hyperplane for bivariate logistic regression

LINEAR MODELS – FUNCTIONALITY

Empirical risk

- **Linear regression**
  - Typically, based on **quadratic** loss: $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - f\left(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}\right)\right)^2$

    $\Rightarrow$ corresponding to ordinary-least-squares (OLS) estimation
  - Alternatives: e.g., **absolute** or **Huber** loss (both improving robustness)
- **Logistic regression:** based on **Bernoulli/log/cross-entropy** loss

  $\Rightarrow \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} -y^{(i)} \log\left(\pi\left(\mathbf{x}^{(i)}\right)\right) - (1 - y^{(i)}) \log\left(1 - \pi\left(\mathbf{x}^{(i)}\right)\right)$

Optimization

- For **OLS**: analytically with $\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
  (with $\mathbf{X} \in \mathbb{R}^{n \times p}$: matrix of feature vectors)
- For **other loss functions**: numerical optimization

Hyperparameters None

LINEAR MODELS – PRO'S & CON'S

Advantages

+ **simple and fast** implementation
+ **analytical** solution
+ **cheap** computation
+ applicable for any **dataset size**, as long as number of observations $\gg$ number of features
+ flexibility **beyond linearity** with polynomials, trigonometric transformations etc.
+ intuitive **interpretability** via feature effects
+ statistical hypothesis **tests** for effects available

Disadvantages

— **nonlinearity** of many real-world problems
— further restrictive **assumptions**: linearly independent features, homoskedastic residuals, normality of conditional response
— **sensitivity** w.r.t. outliers and noisy data (especially with L2 loss)
— risk of **overfitting** in higher dimensions
— feature **interactions** must be handcrafted, so higher orders practically infeasible
— no handling of **missing** data

Simple method with good interpretability for linear problems, but with strong assumptions and limited complexity

LINEAR MODELS – PRACTICAL HINTS

Implementation

- **R:** `mlr3` learner `LearnerRegrLM`, calling `stats::lm()` / `mlr3` learner `LearnerClassifLogReg`, calling `stats::glm()`
- **Python:** `LinearRegression` from package `sklearn.linear_model`, package for advanced statistical parameters `statsmodels.api`

Regularization

- Quadratic penalty on model coefficients (a.k.a. **Ridge** regression)
  - Overall smaller, but still dense $\boldsymbol{\theta}$
  - Suitable with many influential features present, handling correlated features by shrinking their coefficients equally
- Absolute penalty on model coefficients (a.k.a. **Lasso** regression)
  - Actual variable selection
  - Suitable for sparse problems, ineffective with correlated features (randomly selecting one)
- Weighted combination of Ridge and Lasso: **elastic net**

REGULARIZED LINEAR MODELS

REGULARIZED LM – FUNCTIONALITY

SUPERVISED PARAMETRIC WHITE-BOX

General idea

- Unregularized LM: risk of **overfitting** in high-dimensional space with only few observations
- **Goal**: find compromise between model fit and generalization

Empirical risk

- Empirical risk function **plus complexity penalty** $J(\boldsymbol{\theta})$, controlled by shrinkage parameter $\lambda > 0$: $\mathcal{R}_{\text{reg}}(\boldsymbol{\theta}) := \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) + \lambda \cdot J(\boldsymbol{\theta})$.
- Popular regularizers
  - **Ridge regression**: L2 penalty $J(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$
  - **LASSO regression**: L1 penalty $J(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1$

Optimization

- **Ridge**: analytically with $\hat{\boldsymbol{\theta}}_{\text{Ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda \boldsymbol{I})^{-1} \mathbf{X}^\top \mathbf{y}$
- **LASSO**: numerically with, e.g., (sub-)gradient descent

Hyperparameters Shrinkage parameter λ

REGULARIZED LM – PRACTICAL HINTS

Choice of regularization parameter

- Standard hyperparameter optimization problem
- E.g., choose λ with minimum mean cross-validated error (default in R package `glmnet`)

Ridge vs. LASSO

- **Ridge**
  - Overall smaller, but still dense $\boldsymbol{\theta}$
  - Suitable with many influential features present, handling correlated features by shrinking their coefficients equally
- **LASSO**
  - Actual variable selection
  - Suitable for sparse problems, ineffective with correlated features (randomly selecting one)
- Neither overall better – best of both worlds: **elastic net** $\rightarrow$ weighted combination of Ridge and LASSO regularizers

Implementation

- **R:** `mlr3` learners `LearnerClassifGlmnet` / `LearnerRegrGlmnet`, calling `glmnet::glmnet()`
- **Python:** `LinearRegression` from package `sklearn.linear_model`, package for advanced statistical parameters `statsmodels.api`