Supervised Learning

Learning from Data and Linear Models

MLI

Module 1, Lecture 1

20th April 2021

[Adriano Koshiyama]

Supervised Learning

Learning from Data

MLI

Module 1, Lecture 1

20th April 2021

[Adriano Koshiyama]

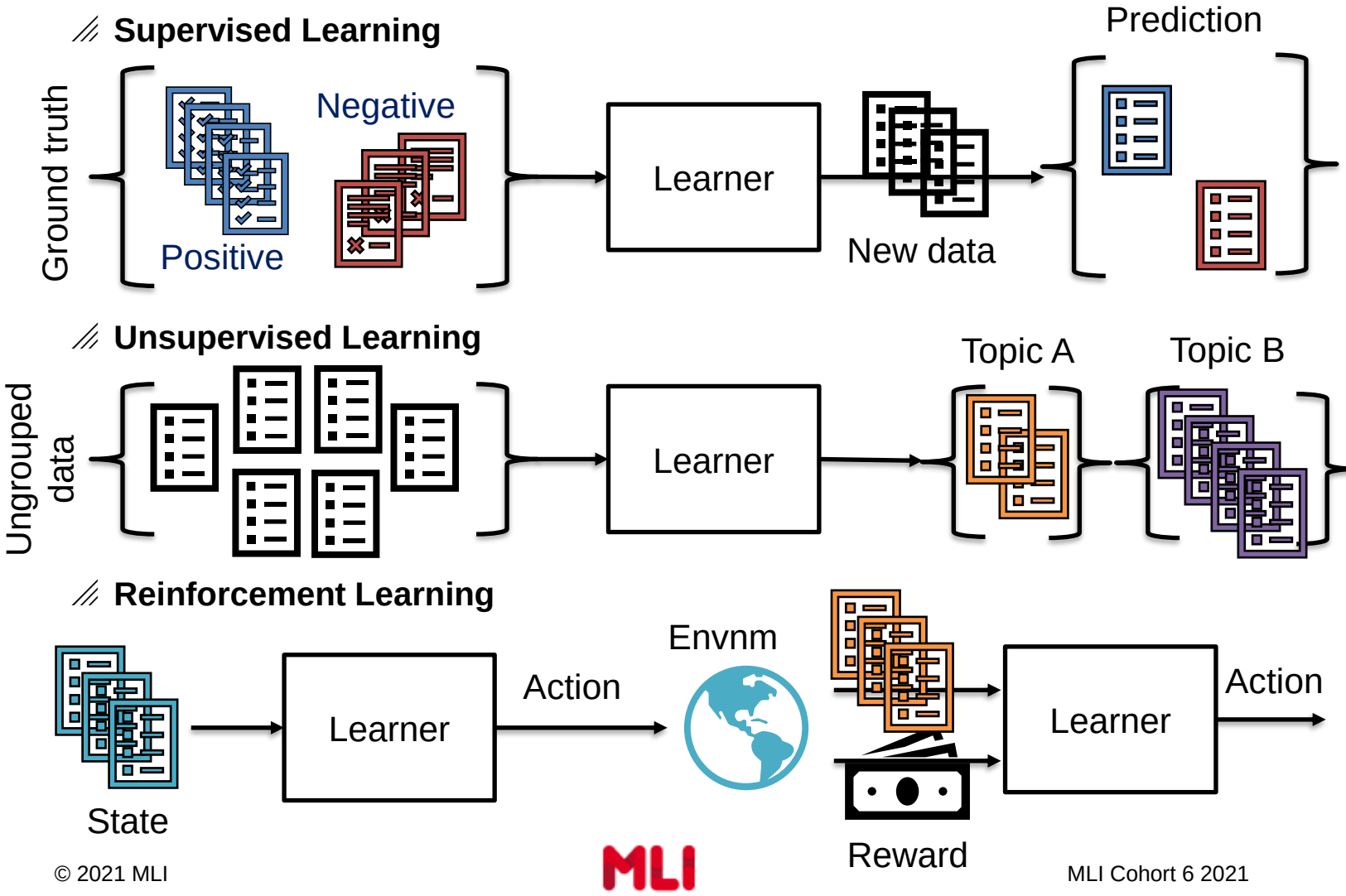
Overview

- **Supervised Learning**
  - Introduction to Machine Learning
  - What is Supervised Learning
  - Examples of Supervised Learning problems
  - Problems addressed by Supervised Learning
  - A non-comprehensive list of Supervised Learning models
- **Challenges in Supervised Learning**
  - Optimal Supervised Learning
  - Hypothesis spaces
  - Balancing Underfitting and Overfitting
  - Overfitting
    - Sources of Error
    - Guidelines for Modelling
- **Practical aspects in Supervised Learning**
  - Expected/Generalization Error Estimation
  - Cross-validation Schemes
  - Cross-validation and Covariance-Penalty
  - Packages and Implementations
  - Time series: what changes?

Supervised Learning

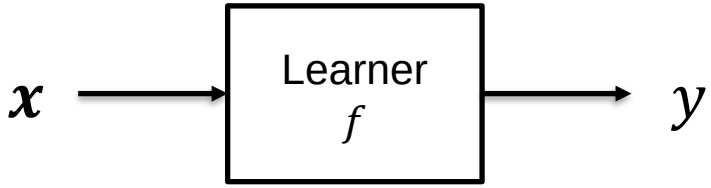
- Introduction to Machine Learning
- What is Supervised Learning
- Examples of Supervised Learning problems
- Problems addressed by Supervised Learning
- Other types of Supervised Learning
- A non-comprehensive list of Supervised Learning models

Introduction to Machine Learning



What is Supervised Learning (SL)

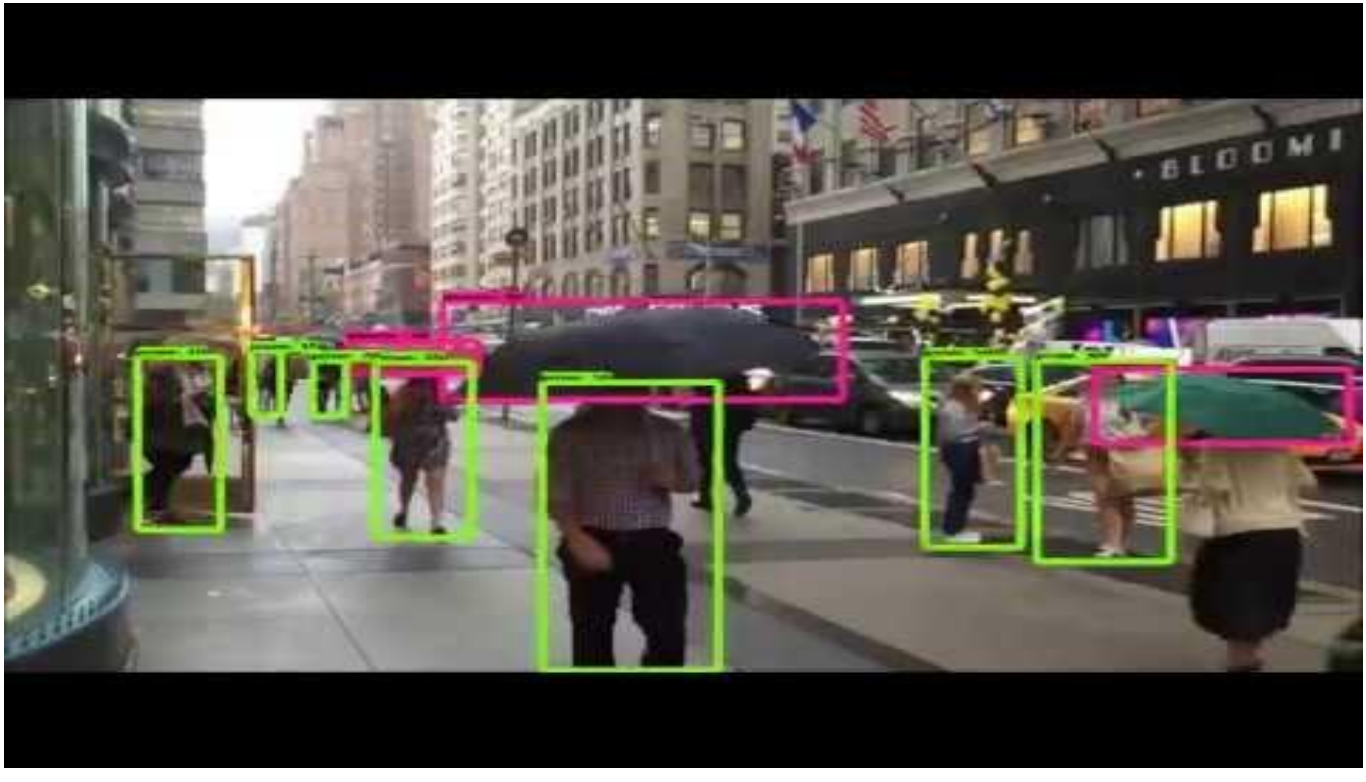
- Given a set of pairs $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$ where
  - Inputs/Independent/Features/RHS: $\boldsymbol{x_i} = (x_1, \ldots, x_{\boldsymbol{j}}, \ldots, x_J) \in \boldsymbol{S}^J$
  - Output/Dependent/Outcome/LHS: $y_i \in B$
  - D is the dataset with $i = 1, \ldots, n$ samples
- We wish to uncover the functional link $f: \boldsymbol{S}^J \to B, f(\boldsymbol{x}) \approx y$



- Since y_i was i.i.d. sampled with some noise from a fixed but unknown pdf $P(\boldsymbol{x}, y)$, we can formulate the **learning** problem as

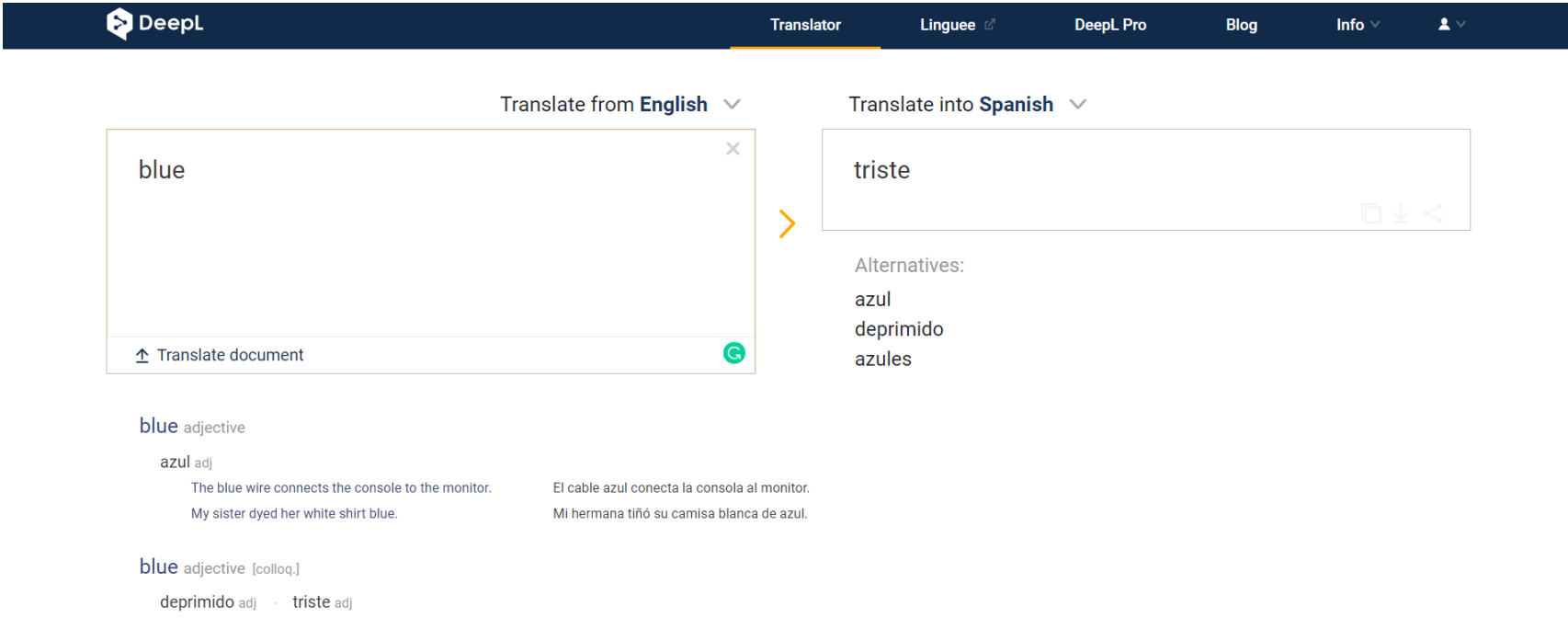
$$argmin_{f \in \Omega}\, \boldsymbol{L}(f) = \boldsymbol{E}\left[\left(y - f(\boldsymbol{x})\right)^2\right] = \int_{-\infty}^{\infty} \left(y - f(\boldsymbol{x})\right)^2 dP(\boldsymbol{x}, y)$$

Examples of SL problems



https://www.youtube.com/watch?v=_zZe27JYi8Y

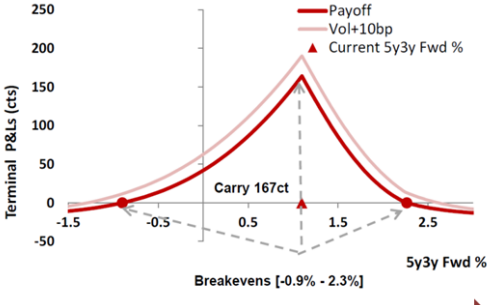
Examples of SL problems

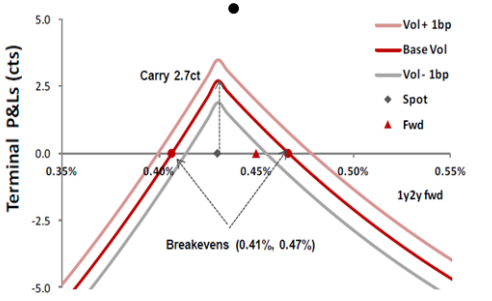


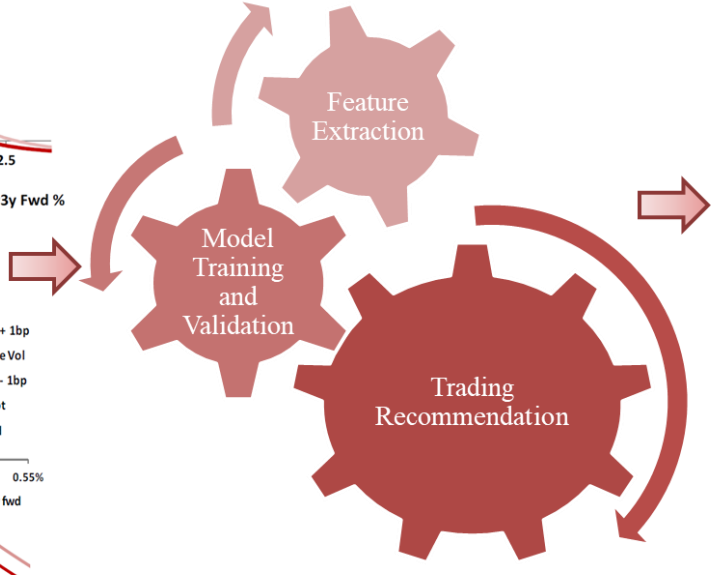
https://www.deepl.com/press.html

Examples of SL problems

Trade strategy ranking system







Long Position

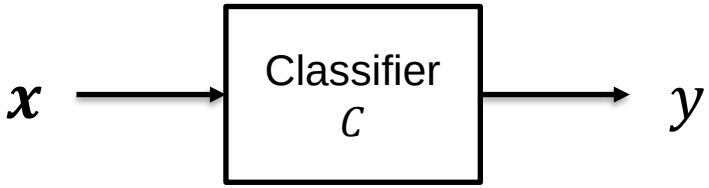
MCCS	Attractiveness Score (1y)	Expected Returns (1y)
EUR 5y5y5y	78%	21.22%
EUR 3y2y3y	62%	23.22%
EUR 4y1y5y	54%	17.65%
…	…	…
EUR 4y2y3y	12%	5.60%

Short Position

MCCS	Attractiveness Score (1y)	Expected Returns (1y)
EUR 1y1y1y	65%	25.67%
EUR 2y2y1y	55%	22.65%
EUR 3y1y1y	50%	26.32%
…	…	…
EUR 5y4y1y	15%	8.56%

Problems addressed by SL

- **Classification or Pattern Recognition**
  - Given a set of pairs $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$ where
    - Features: $\boldsymbol{x_i} = (x_1, \dots, x_j, \dots, x_J) \in \boldsymbol{S}^J$
    - Classes: $y_i \in \{0, 1, \dots, K\}$
    - D is the dataset with $i = 1, \dots, n$ patterns
  - We wish to find a classifier $C: \boldsymbol{S}^J \rightarrow \{0, 1, \dots, K\}$, $C(\boldsymbol{x}) \approx y \in \{[0,1], \dots\}$



  - Examples
    - Credit card fraud detection
    - Intrusion detection
    - Tail events determination
    - Credit rating and scoring

Problems addressed by SL

- **Regression or Curve Fitting (even Forecasting)**
  - Given a set of pairs $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$ where
    - Inputs/Independent variables: $\boldsymbol{x_i} = (x_1, \ldots, \boldsymbol{x_j}, \ldots, x_J) \in \boldsymbol{S}^J$
    - Output/Dependent variables: $y_i \in \mathbb{R}$
    - D is the dataset with $i = 1, \ldots, n$ samples, transactions, records
  - We wish to find a classifier $R: \boldsymbol{S}^J \to \mathbb{R},\ R(\boldsymbol{x}) \approx y$

  - Examples
    - Trade ranking system
    - Next earnings prediction
    - Conditional volatility forecasting
    - Returns or price forecasting

A non-comprehensive list of SL models

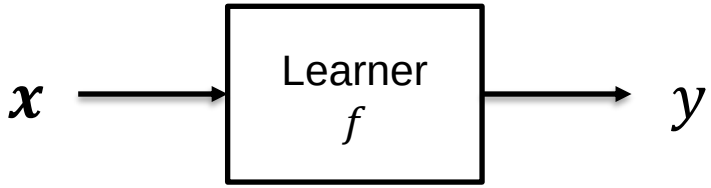
- **Linear**
  - Linear, Ridge and Lasso Regression
  - Linear Support Vector Machines
  - Linear Discriminant Analysis
- **Piecewise Linear and Weakly Nonlinear**
  - Naïve Bayes
  - k-Nearest Neighbours
  - Logistic, Ridge and Sparse/Lasso Regression
  - Quadratic Linear Discriminant Analysis
  - Decision Trees
- **Strongly Nonlinear (a.k.a. Machine Learning canon)**
  - Random Forest and Gradient Boosting Models
  - Kernel Ridge Regression
  - Kernel Support Vector Machines/Regression (SVM and SVR)
  - Neural Networks (Feed Forward, RBFs, LSTMs, etc.)

Challenges in Supervised Learning

- Optimal Supervised Learning
- Hypothesis spaces
- Balancing Underfitting and Overfitting
- Overfitting
  - Sources of Error
  - Guidelines for Modelling

Optimal Supervised Learning

- Given a set of pairs $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$ where
  - Inputs/Independent/Features/RHS: $\boldsymbol{x_i} = (x_1, \ldots, x_{\boldsymbol{j}}, \ldots, x_J) \in \boldsymbol{S}^J$
  - Output/Dependent/Outcome/LHS: $y_i \in B$
  - D is the dataset with $i = 1, \ldots, n$ samples
- We wish to uncover the functional link $f \in \Omega: \boldsymbol{S}^J \to B, f(\boldsymbol{x}) \approx y$



- Since D is composed of i.i.d. samples from a fixed but unknown probability density $P(\boldsymbol{x}, y)$, we can formulate the **learning** problem as:

$$argmin_{f \in \Omega}\, \mathcal{E}(f) = \boldsymbol{E}\left[\left(y - f(\boldsymbol{x})\right)^2\right] = \int_{-\infty}^{\infty} \left(y - f(\boldsymbol{x})\right)^2 dP(\boldsymbol{x}, y)$$

- Issue A: we don't know $P(\boldsymbol{x}, y)$ → can't compute the functional
- Issue B: searching for an arbitrary $f^* \in \Omega$ can be "dangerous"

Optimal Supervised Learning

- **Relaxations**
- **Solving Issue A**: we can replace the expected error $\mathcal{E}(f)$ by the **empirical error** $\mathcal{E}_{emp}(f)$, which in the case of regression, is:

$$\mathcal{E}_{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(\boldsymbol{x_i})\right)^2 \rightarrow_p^{n\to\infty} \boldsymbol{E}\left[\left(y - f(\boldsymbol{x})\right)^2\right] = \mathcal{E}(f)$$

- **Solving Issue B**: we can restrict our search for an optimal f^* inside a subset $\mathcal{H} \in \Omega$ know as the **hypothesis space**. Example:

$$\mathcal{H} = \{f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} : \boldsymbol{w} \in \mathbb{R}^J\} \text{ for linear regression}$$

- Putting everything together, we have:

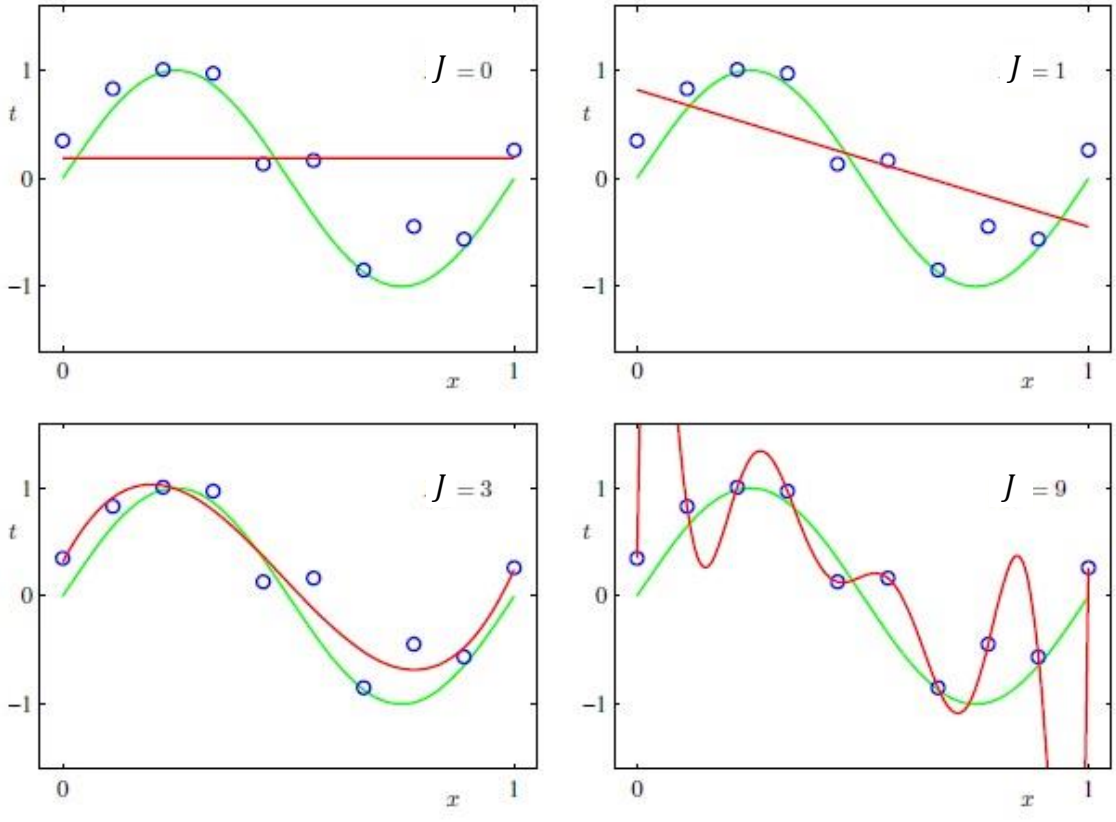
$$argmin_{f\in\Omega}\, \mathcal{E}(f) = \boldsymbol{E}\left[\left(y - f(\boldsymbol{x})\right)^2\right] \approx$$

$$argmin_{f\in\mathcal{H}}\mathcal{E}_{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(\boldsymbol{x_i})\right)^2$$

- Last issue… **How do we select a proper Hypothesis Space**?

Hypothesis Spaces

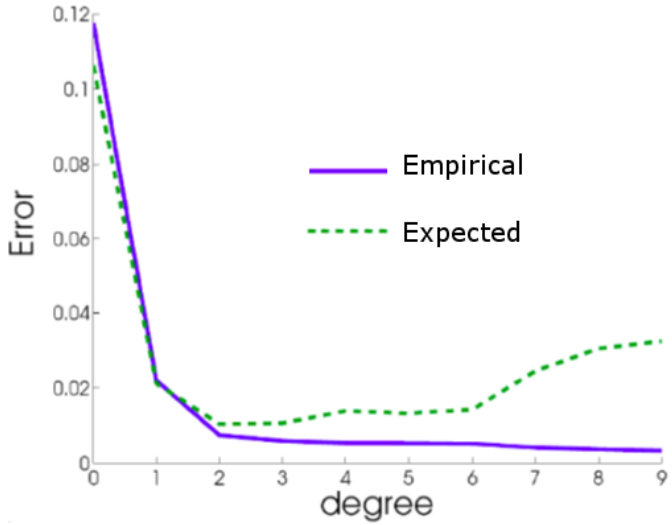
- Polynomials in action[1] : different degrees ($J = 0, 1, 3, 9$)



[1] Reproduced from Bishop, C. Pattern Recognition and Machine Learning. Springer-Verlag, New York, 2006.

Balancing Underfitting and Overfitting

- If we chart the empirical and expected error for every degree
  - **Empirical**: It always reduces as the polynomial degree increases, i.e., hypothesis space is larger
  - **Expected**: after degree 3 it starts to increase and explodes around 9
  - **Conclusion**: we cannot rely in the empirical error to identify the proper hypothesis space
  - This phenomena is known in Supervised Learning as **Overfitting**

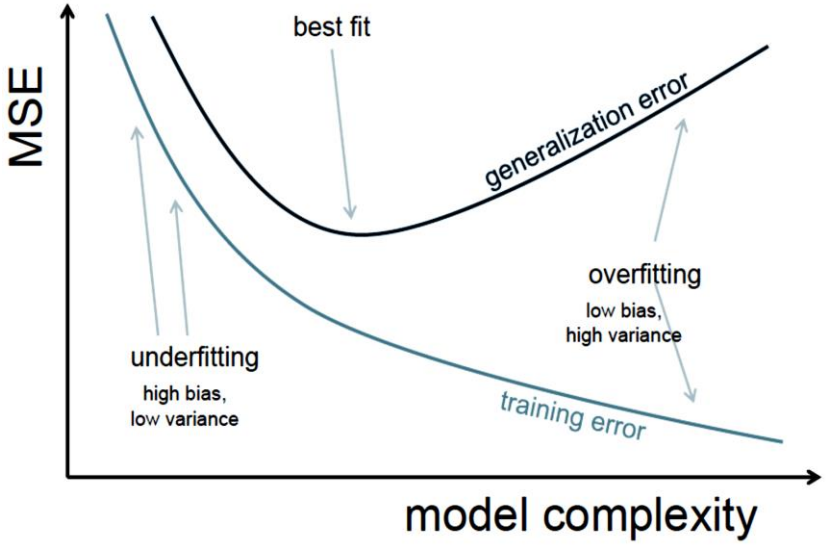


- In order **to tackle overfitting** we need to:
  - 1) **Understand** the **sources of error** occurring from modelling
  - 2) Have a **proper set of principles to follow** during modelling
  - 3) Improve model validation via **cross-validation or cov-penalty**

Overfitting: 1) Sources of Error †(1)

- Given unseen data (X_*, Y_*) and a prediction from our learning machine $\hat{f}(X_*)$, we can decompose its **expected out-sample error**
- Bias: $\boldsymbol{E}_Y\left[f(X_*) - \hat{f}(X_*)\right]$
- Variance: $\boldsymbol{E}_Y\left[\left(\hat{f}(X_*) - \boldsymbol{E}_Y[\hat{f}(X_*)]\right)^2\right]$
- Mean-squared Error (MSE)

$$\mathbf{MSE}\left(\hat{f}|X_*\right) = \boldsymbol{E}\left[\left(Y_* - \hat{f}(X_*)\right)^2\right]$$



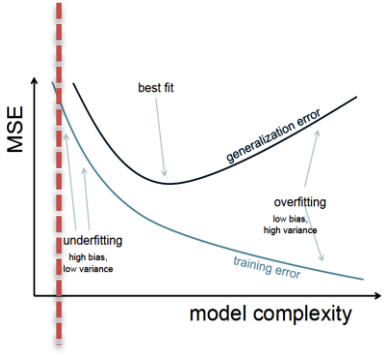
- **Proposition: Bias-Variance Trade-off**

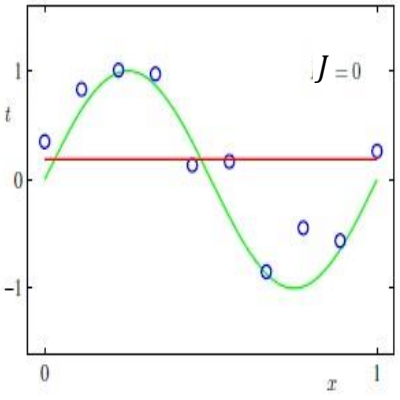
$$\mathcal{E}\left(\hat{f}|X_*\right) = \mathbf{MSE}\left(\hat{f}|X_*\right) = \boldsymbol{Var}[\varepsilon_*] + \boldsymbol{Bias}\left(\hat{f}\middle|X_*\right)^2 + \boldsymbol{Var}\left(\hat{f}\middle|X_*\right)$$

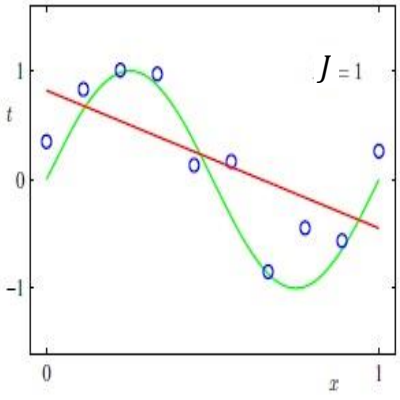
- Irreducible error: $\boldsymbol{Var}[\varepsilon_*]$
- Approximation error/Bias: $\boldsymbol{Bias}\left(\hat{f}\middle|X_*\right)^2$-- can be reduced by choosing a large $\mathcal{H}$
- Estimation error/Variance: $\boldsymbol{Var}\left(\hat{f}\middle|X_*\right)$ -- reduced by smaller $\mathcal{H}$ and larger n

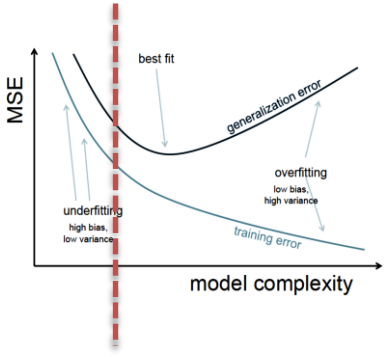
Overfitting: 1) Bias-Variance Trade-off

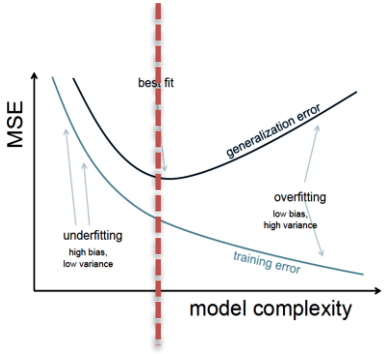
- Bias-variance trade-off in action:

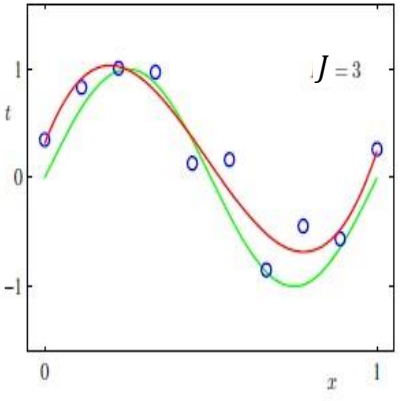


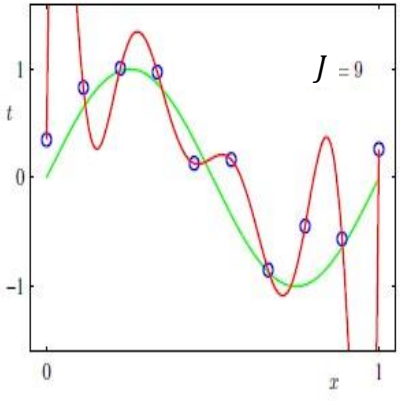


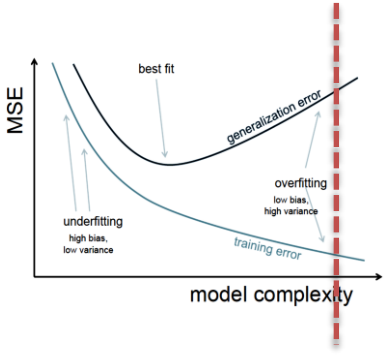








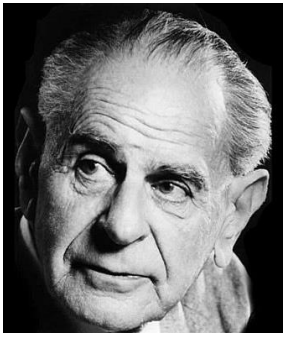




Overfitting: 2) Guideline for Modelling †(2)

- **A set of principles has to guide the modelling process, such as**:

- **Parsimonious**: “No more things should be presumed to exist than are absolutely necessary”, i.e., the fewer assumptions an explanation of a phenomenon depends on, the better the explanation



- **Falsifiability/Prediction Strength:** "[a good model] must accurately describe a large class of observations on the basis of a model that contains only a few arbitrary elements, and it must make definite predictions about the results of future observations", i.e., the simple model with lowest (prediction) error is best

- **Moral**: we need to combine these elements in order to be successful to design/search for a hypothesis space (**a.k.a. model selection**)

Practical aspects in Supervised Learning

- Expected/Generalization Error Estimation
- Cross-validation Schemes
- Cross-validation and Covariance-Penalty
- Packages and Implementations

Expected/Generalization Error Estimation

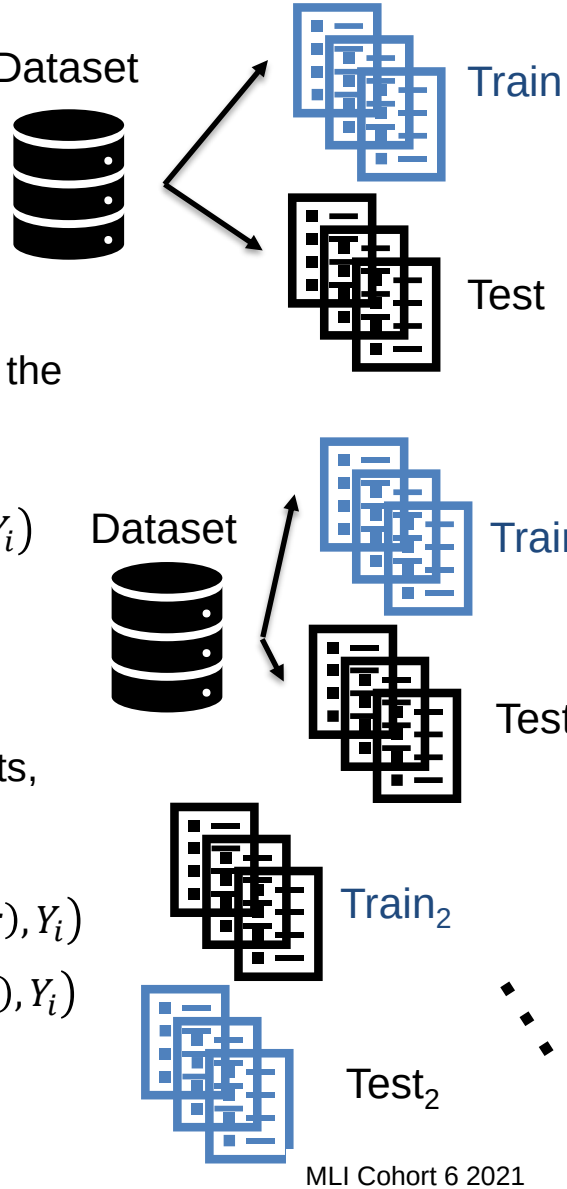
- We have seen that, from **Bias-Variance Trade-off**, we can decompose the expected error in regression by:

$$\mathcal{E}(\hat{f}|X_*) = \mathbf{MSE}(\hat{f}|X_*) = \boldsymbol{Var}[\varepsilon_*] + \boldsymbol{Bias}(\hat{f}|X_*)^2 + \boldsymbol{Var}(\hat{f}|X_*)$$

- Although in theory it is important to understand each source of error, in practice we are interested in the **aggregated Risk/Loss metric**
- Hence, the question can be reframed as:
  - Given a dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
  - A learning machine f that use the dataset and generate $\hat{f}$
  - A Risk/Loss Function $\mathcal{L}(\hat{f}(X_i), Y_i) \rightarrow a \subset \mathbb{R}$
- How can we reliably estimate the generalization risk of $\hat{f}$?
- **Two answers with many subcomponents**
  - Cross-validation schemes
  - Covariance-penalty methods

Cross-Validation Schemes

- **Single holdout or One-split**
  - Split the dataset in Training $Tr = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{H}$ and Testing $Ts = \{(\boldsymbol{x}_i, y_i)\}_{i=H}^{n}$ folders
  - Estimate/Train the learning machine using the Training set: $\hat{f}(.|Tr)$
  - Evaluate the learning machine against the Testing set: $\hat{\mathcal{E}}\left(\hat{f}|X_*\right) = \frac{1}{n-H}\sum_{H}^{n}\mathcal{L}\left(\hat{f}(X_i|Tr), Y_i\right)$
- **Issue:** depend on a single training set → high variance if algorithm is "unstable"
- **k-Fold-cross-validation**
  - Create a set of k-disjoint Train and Test sets, with non-overlapping Test Sets
  - Repeat the same process of Training and Testing, computing the partial: $OS_l\left(\hat{f}(X_i|Tr), Y_i\right)$
  - Evaluate as: $\hat{\mathcal{E}}\left(\hat{f}|X_*\right) = \frac{1}{k}\sum_{l=1}^{k} OS_k\left(\hat{f}(X_i|Tr), Y_i\right)$
- **Issue**: some correlation between the sets is introduced…



Cross-Validation and Covariance-Penalty

- Some theoretical results regarding one-split and k-fold-cv
- **One-split**
  - Unbiased estimator of expected error: $\mathbf{E}\left[\hat{\mathcal{E}}\left(\hat{f}|X_*\right)_{\boldsymbol{os}}\right] = \mathcal{E}\left(\hat{f}|X_*\right)$
  - Consistently estimates $\mathcal{E}\left(\hat{f}|X_*\right)$, i.e., $\hat{\mathcal{E}}\left(\hat{f}|X_*\right)_{\boldsymbol{os}} \rightarrow_p^{n\to\infty} \mathcal{E}\left(\hat{f}|X_*\right)$
- **k-fold-cv**
  - More efficient than One-split, since intra-fold correlation ρ tend to be *<1*

$$\mathbf{Var}\left[\hat{\mathcal{E}}\left(\hat{f}|X_*\right)_{\boldsymbol{k-cv}}\right] = \left(\rho + \frac{1-\rho}{k}\right) \cdot \mathbf{Var}\left[\hat{\mathcal{E}}\left(\hat{f}|X_*\right)_{\boldsymbol{os}}\right]$$

- **Covariance-penalty Lemma**:
  - In regression and under some more assumptions

$$\mathbf{MSE}\left(\hat{f}(.|Tr)|Ts\right) = \mathbf{MSE}\left(\hat{f}(.|Tr)|Tr\right) + \mathbf{2Cov}[\mathrm{Y}_i, \hat{f}(.|Tr)]$$

- reads: “Test MSE = Train MSE + How sensible $\hat{f}$ is to changes in Y_i”
- For linear regression

$$\widehat{\boldsymbol{MSE}}\left(\hat{f}(.|Tr)|Ts\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(\boldsymbol{x}_i)\right)^2 + \frac{2}{n}\boldsymbol{Var}[y_i] \cdot J$$

Typical Modelling Pipeline

Data and Task Setup

- Preparing queries and fetching data
- Defining the learning task (Classification, Regression, etc.)

Feature pre-processing and Engineering

- Scaling and transforming some features
- Creating new features, using z-scores, lagging, embedding, etc.

Model Selection

- Defining baselines, performance metrics and model evaluation
- Hypothesis space and hyperparameters to explore

Post-processing/Reporting

- Adding constraints to some models, feature importance, etc.
- Plotting results, compiling metrics and showing value

Productionizing and Deploying

- Stress-testing models and streamlining re-training and performance
- Setting up servers/hosts, liaising with potential users, etc.

Packages and Implementations

- There are a plethora of software, packages, implementations, etc. available for machine learning. Here we present a few of the main ones:
- **Python**
  - scikit-learn: http://scikit-learn.org/
  - keras: https://keras.io/
  - pytorch: https://pytorch.org/
  - sciblox (similar to caret): https://pypi.org/project/sciblox/
  - shogun: http://shogun-toolbox.org/
- **R**
  - mlr: https://mlr-org.github.io/mlr/
  - caret: http://topepo.github.io/caret/index.html
- **General**
  - tensorflow: https://www.tensorflow.org/

Additional Reading

- ***Main Reference of this Lecture***
  - *Friedman, J., Hastie, T., & Tibshirani, R. (2009). The elements of statistical learning. vol. 2. Springer.*
    - *Chapter 1 - Introduction*
    - *Chapter 2 - Overview of Supervised Learning*
    - *Chapter 7 - Model Assessment and Selection*
  - *Efron, B., & Hastie, T. (2016). Computer age statistical inference (Vol. 5). Cambridge University Press.*
    - *Chapter 2 - Frequentist Inference*
    - *Chapter 12 - Cross-validation and Cp Estimates of Prediction Error*
  - Koshiyama, Adriano and Firoozye, Nick and Treleaven, Philip, Algorithms in Future Capital Markets (January 29, 2020). Available at SSRN: https://ssrn.com/abstract=3527511 or http://dx.doi.org/10.2139/ssrn.3527511

Supervised Learning

Linear Models

MLI
Module 1, Lecture 1
20th April 2021

[Adriano Koshiyama]

Overview

- **Linear Modelling**
  - What is a Linear Model?
  - Why use a Linear Model?
  - Applications of Linear Models
  - When Linear Modelling Fail?
- **Workhorses: Linear and Logistic Regression**
  - Multiple Linear Regression and Logistic Regression
    - Model
    - Estimation
  - Statistical vs Machine Learning View
- **Improving Linear Models: Feature Maps, Selection and Regularization**
  - Explicit Feature Maps
  - Explicit Feature Selection
  - Regularization – Implicit Feature Selection
    - Ridge and Lasso Regression
    - Why it works?

Linear Modelling

- What is a Linear Model?
- Why use a Linear Model?
- Applications of Linear Models
- When Linear Modelling Fail?

What is a Linear Model?

- Typically, if the phenomena y_i can be expressed by a linear model, it can be write down as:

$$y_i = \boldsymbol{\beta}^T \boldsymbol{x}_i + \varepsilon_i, \qquad \varepsilon_i \sim P$$

hence, y_i can be closely described by a **linear combination** of features $\boldsymbol{x}_i$

- A model that is **linear in the parameters**, i.e., given a feature map $\phi: S^n \to \mathcal{R}^N$, y_i can still be expressed

$$y_i = \boldsymbol{\alpha}^T \phi(\boldsymbol{x}_i) + \varepsilon_i, \qquad \varepsilon_i \sim P$$

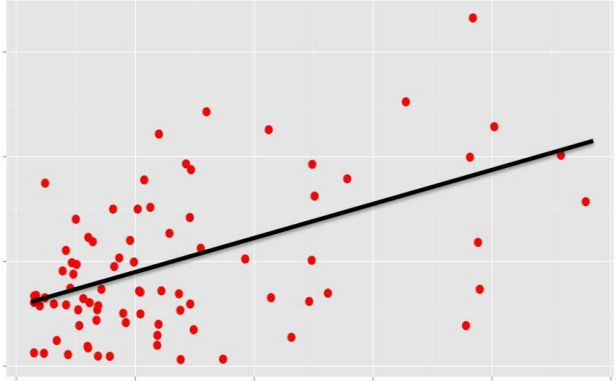
typically a polynomial regression

- A model that **is nonlinear in the parameters and variables**, but that can be "linearized" by a mapping:

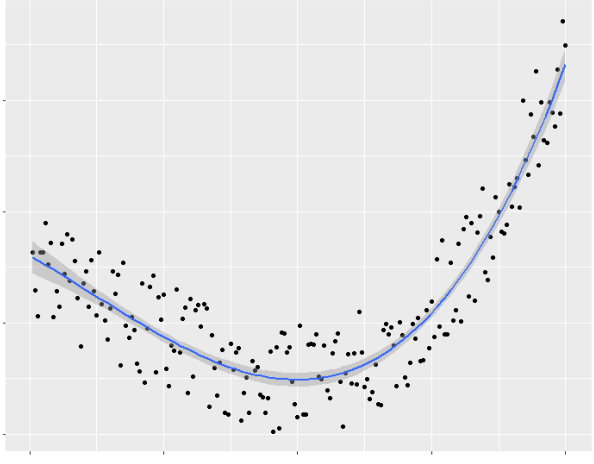
$$\varphi(y_i) = \boldsymbol{\omega}^T \phi(\boldsymbol{x}_i) + \varepsilon_i, \qquad \varepsilon_i \sim P$$

e.g. log-linear, some GLMs, etc.

E.g. Linear in params and variables



E.g. Linear in params only



Why use a Linear Model?

Criteria	Params & Variables	Params only	Linear after a transform	Nonlinear
Theoretical Analysis	Easy	Fair	Fair	Hard
Intuitive	Very	Fairly	Fairly	Not much
Interpretable	Very	Hard	Fairly	Not much
Statistical Analysis	Easy	Easy	Easy	Medium
Training Time	Fast	Medium	Medium	Slow
“Online” Behaviour	Stable	Might be unstable	Might be unstable	May be unstable
Prediction	Poor	Fair	Fair	Good
Overfitting	Hard	Medium	Medium	Easy

Applications of Linear Models

- Statistical Arbitrage

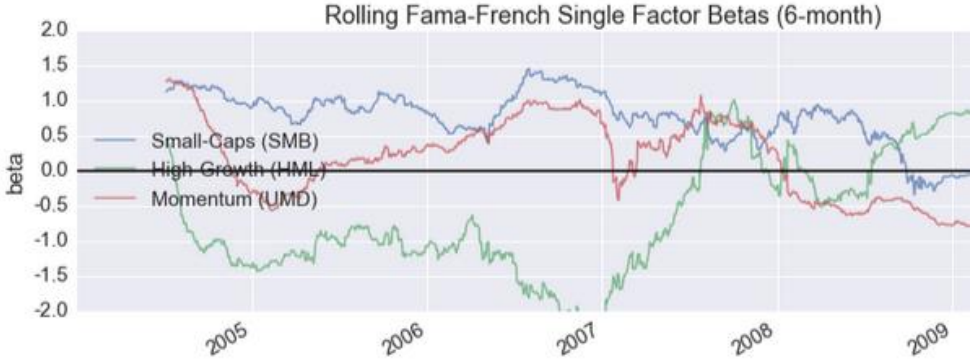
$$IG_t = \beta \cdot HY_t + \varepsilon$$

where $\beta = \rho(IG_t, HY_t)\frac{\sigma_{IG_t}}{\sigma_{HY_t}}$, represents the leverage factor: how much you should short Investment Grade (IG_t), and go long High-Yield (HY_t)

- Fama-French Decomposition

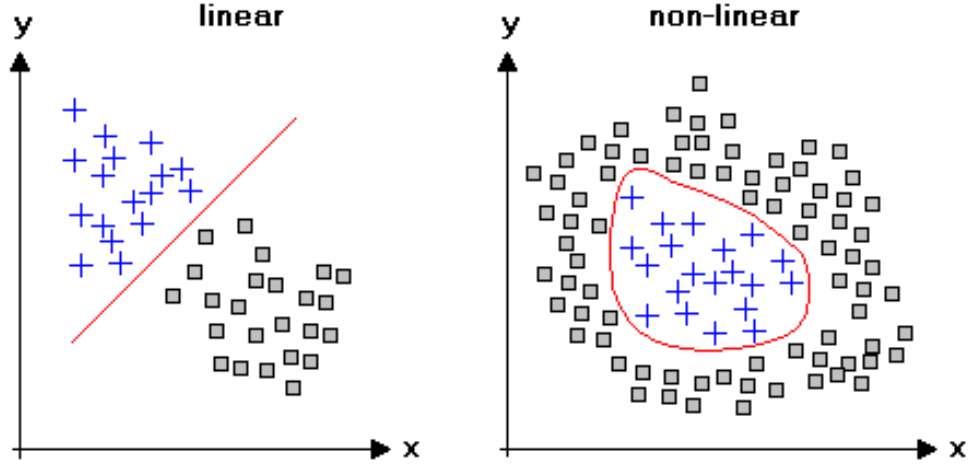
$$r_t = r_f + \beta_M(r_m - r_f) + \beta_{SMB} r_{SMB} + \beta_{HML} r_{HML} + \beta_{MD} r_{MD} + \varepsilon_t$$

- r_t - asset returns
- r_f - risk-free rate
- r_m- "market" returns
- r_{SMB}, r_{HML}, r_{MD} - Fama-French factors

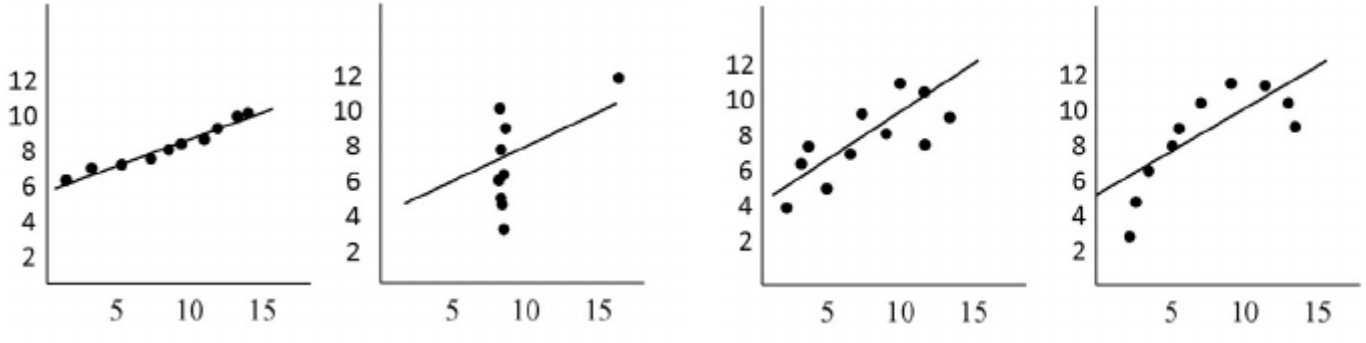


When Linear Modelling Fail?

- When it is used to model nonlinear or nonlinearly separable problems



- When coefficients, t-stats, etc. is used to replace visualization/theory:



Workhorses: Linear and Logistic Regression

- Multiple Linear Regression and Logistic Regression
  - Model
  - Estimation
- Statistical vs Machine Learning View

Multiple Linear Regression - Model

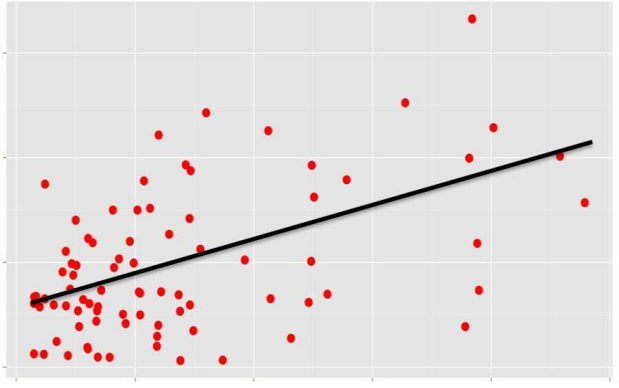
- General model

$$y_i = \boldsymbol{\beta}^T \boldsymbol{x}_i + \varepsilon_i, \qquad \varepsilon_i \sim P$$

by concatenating $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^T$ and $\boldsymbol{y} = [y_1, \dots, y_n]^T$ it is rewritten as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_n]^T$$

- **Main Assumption**: the dependent variable y_i can be described by a linear combination of independent variables $\boldsymbol{\beta}^T \boldsymbol{x}_i$ with additive noise ε_i
- **Statistical Assumptions for Inference**
- Homoscedasticity: $\boldsymbol{V}[\varepsilon_i] = \sigma^2 \ \ \forall i$
- No serial autocorrelation: $\boldsymbol{Cov}[\varepsilon_i, \varepsilon_j] = 0, j \neq i$
- Normality: $\varepsilon_i \sim N(0, \sigma^2)$
- Hence:
  - $y_i \mid \boldsymbol{x}_i \sim N(\boldsymbol{\beta}^T \boldsymbol{x}_i, \sigma^2)$



Multiple Linear Regression – Estimation

- How do we identify $\boldsymbol{\beta}$? "Natural" way: **MSE Minimization**

$$\min_{\boldsymbol{\beta}} \boldsymbol{MSE}(\boldsymbol{\beta}) = \left|\left|\boldsymbol{y} - \boldsymbol{X\beta}\right|\right|_2^2 = (\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta})$$

First condition for unconstrained minimization $\nabla_{\boldsymbol{\beta}}\, \boldsymbol{MSE}(\boldsymbol{\beta}) = \mathbf{0}$, hence

$-\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{X}^T\boldsymbol{X\beta} = \mathbf{0}$, a.k.a. the "normal equations"

yielding the solution of $\boldsymbol{\beta} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$

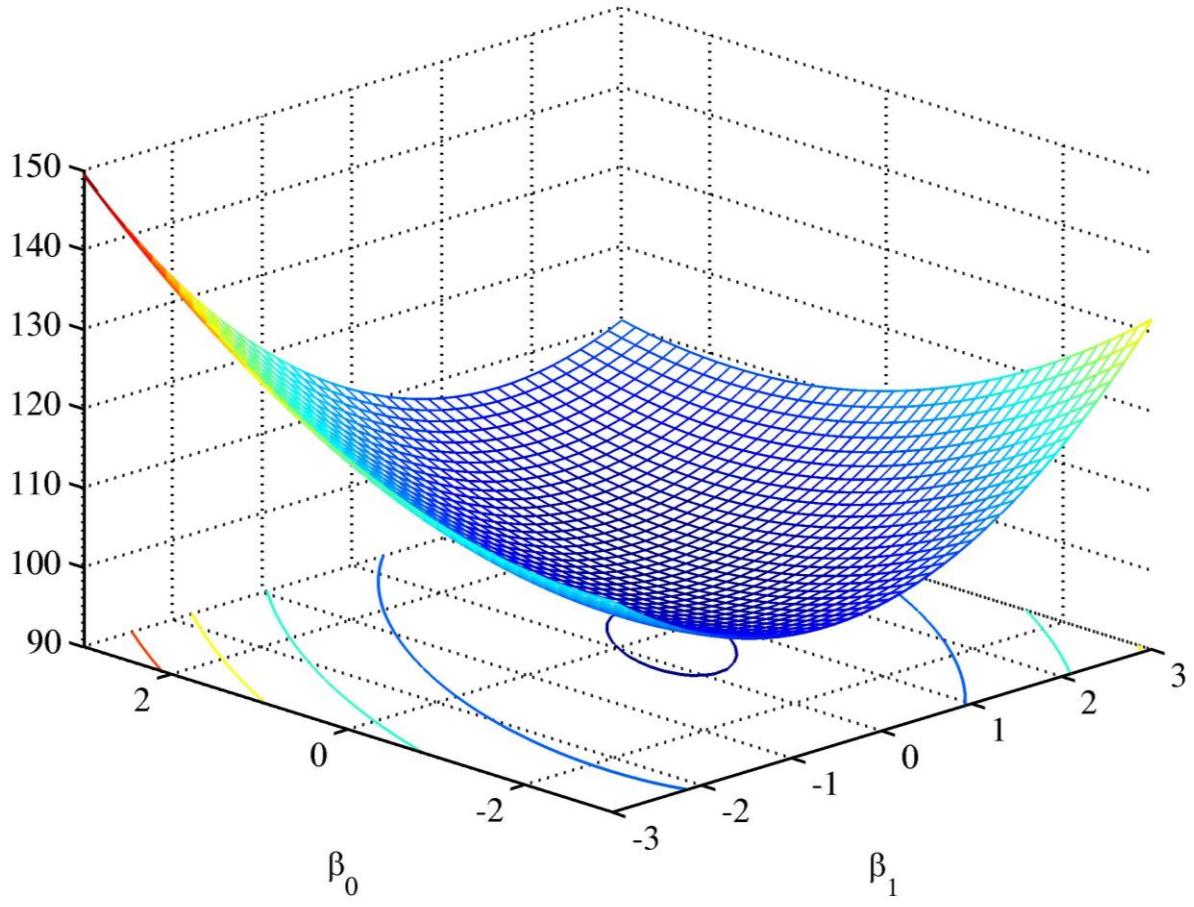
- Second condition: $\boldsymbol{H}[\boldsymbol{MSE}(\boldsymbol{\beta})] = \boldsymbol{X}^T\boldsymbol{X}$ is positive definite & absent of $\boldsymbol{\beta}$

- More general way: **Maximum Likelihood Estimation**
  - Using the statistical assumptions, we can write the log-likelihood

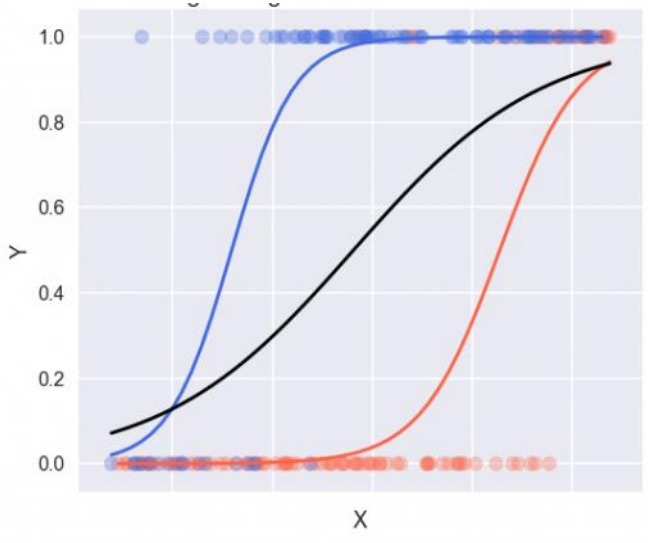
$$\max_{\boldsymbol{\beta}} \ell(\boldsymbol{y}, \boldsymbol{X};\, \boldsymbol{\beta}, \sigma^2) = -n\log\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2\sigma^2}\sum_i^n \left(y_i - \boldsymbol{\beta}^T x_i\right)^2$$

hence, under Gaussian noise, MSE minimization and MLE estimation points towards the **same objective function**

Multiple Linear Regression – Estimation



Logistic Regression - Model



- Model

$$(y_i == 1) \approx g_{\boldsymbol{\beta}}(\boldsymbol{x_i}) = \frac{1}{1 + \exp(-\boldsymbol{\beta^T x_i})}$$

hence, $y_i \in \{0,\ 1\}$ can be described by a **nonlinear transformation of a linear combination** of the input variables

- Why opt for such **nonlinear mapping**, instead of just $g_{\boldsymbol{\beta}}(\boldsymbol{x_i}) = \boldsymbol{\beta^T x_i}$?
  Assuming that $y_i \sim Bern(p_i)$, and $p_i \approx g_{\boldsymbol{\beta}}(\boldsymbol{x_i})$, we have
  - 1. Heteroscedasticity

$$\boldsymbol{V}[y_i] = p_i(1 - p_i) \approx g_{\boldsymbol{\beta}}(\boldsymbol{x_i})\left(1 - g_{\boldsymbol{\beta}}(\boldsymbol{x_i})\right) = \boldsymbol{\beta^T x_i}(1 - \boldsymbol{\beta^T x_i})$$

  - 2. Restriction to a subset

$$y_i \in \{0, 1\} \text{ but } g_{\boldsymbol{\beta}}(\boldsymbol{x_i}) = \boldsymbol{\beta^T x_i} \in (-\infty, +\infty)$$

  - 3. Non-normality

$$\varepsilon_i = \left(y_i - \boldsymbol{\beta^T x_i}\right) = \left(1 - \boldsymbol{\beta^T x_i}\right)y_i + \left(0 - \boldsymbol{\beta^T x_i}\right)(1 - y_i)$$

Logistic Regression – Estimation

- **Maximum Likelihood Estimation** (under i.i.d. assumptions)

Likelihood function: $\max_{\boldsymbol{\beta}} L(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\beta}) = \prod_{i=1}^{n} g_{\boldsymbol{\beta}}(\boldsymbol{x_i})^{y_i} \left(1 - g_{\boldsymbol{\beta}}(\boldsymbol{x_i})\right)^{1-y_i}$

using the log-likelihood (log-loss) and computing the derivative to β_j

$$\frac{d\ell(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\beta})}{d\beta_i} = \sum_{i=1}^{n} \left[y_i \frac{g'_{\boldsymbol{\beta}}(\boldsymbol{x_i})}{g_{\boldsymbol{\beta}}(\boldsymbol{x_i})} - (1 - y_i) \frac{g'_{\boldsymbol{\beta}}(\boldsymbol{x_i})}{\left(1 - g_{\boldsymbol{\beta}}(\boldsymbol{x_i})\right)} \right]$$

use the fact that $g'_{\boldsymbol{\beta}}(\boldsymbol{x_i}) = g_{\boldsymbol{\beta}}(\boldsymbol{x_i}) \left(1 - g_{\boldsymbol{\beta}}(\boldsymbol{x_i})\right)$ to finish expression

- Since no analytical solution is available, we have to resort to iterative schemes, being the Gradient Descent, the most common for this model
  - Algorithm

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \gamma \nabla \ell(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\beta}, n)$$

with γ being the learning rate/step size (usually quite small); the algorithm stops when $\left|\left|\boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}_t\right|\right| < \epsilon_1$ or $\left|\left|\nabla \ell(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\beta}, n)\right|\right| < \epsilon_2$

ML vs Stats Perspectives

- Overall, both models have their roots in Statistics, and therefore, they go beyond pure curve-fitting/prediction when the assumptions are met
- From a **Statistical View/Appraisal** these are the relevant elements
  - **t-stat coefficients**: $t_{\beta_j} = \frac{\widehat{\beta_j}}{\widehat{se}\ (\beta_j)}$, with $\widehat{se}\ (\beta_j)$ as the standard error of β_j
  - **F-test**: comparing MSE or log-loss of full model vs null model
  - **Assumptions**: checking whether $\varepsilon_1, \ldots, \varepsilon_n$ follow all assumptions
  - **Full sample** R^2, R^2_{adj}, etc., that is comparing to regression to the mean
- A **Machine Learning View/Appraisal** is typically focused on
  - **Cross-validated metrics**: MSE, MAD, etc. with average in the test set
  - **Baseline comparisons**: to check whether some signal is being tracked
  - **Statistical comparisons**: to uncover best predictive model
  - **Explainability**: coefficients, partial dependency charts, etc. are used more to provide insights, rather than as the main goal

Feature Mapping and Selection

- Improving Linear Models
- Explicit Feature Maps
- Explicit Feature Selection
- Regularization – Implicit Feature Selection
  - Ridge and Lasso Regression
  - Why it works and remaining questions

Improving Linear Models

Criteria	Params & Variables	Params only	Linear after a transform	Nonlinear
Theoretical Analysis	Easy	Fair	Fair	Hard
Intuitive	Very	Fairly	Fairly	Not much
Interpretable	Very	Hard	Fairly	Not much
Statistical Analysis	Easy	Easy	Easy	Medium
Training Time	Fast	Medium	Medium	Slow
“Online” Behaviour	Stable	Might be unstable	Might be unstable	May be unstable
Prediction	Poor	Fair	Fair	Good
Overfitting	Hard	Medium	Medium	Easy

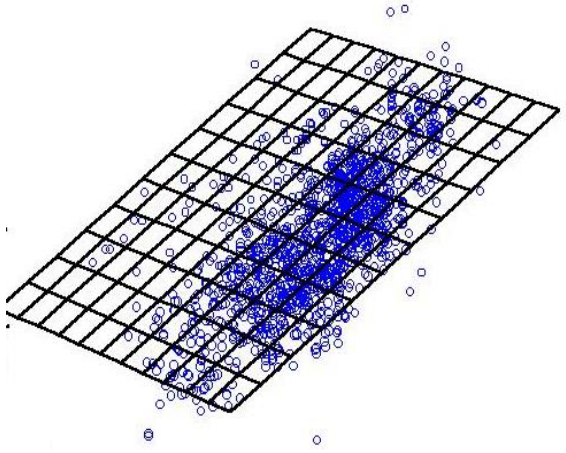
Improving Linear Models

- There are **two main strands** in which we can turn a nimble Linear Model in a more “nonlinear” machine in predictive terms

Map // Selection	Explicit	Implicit
Explicit	Basis Functions with Stepwise	Regularization
Implicit	Kernelization with Stepwise	Kernel Methods with Regularization

- **Implicit** Selection are widely adopted in Machine Learning
- **Explicit** Mapping is not very systematic, being more problem-dependent, but some recipes are universal (interactions, z-scores, etc.)
- **Implicit** Mapping with **Implicit** Feature selection is **Lecture 3**
- **Implicit** mapping with **Explicit** feature selection is not generally used

Explicit Feature Maps



- Overall, the main idea is to extend the model, including new features engineered using some **nonlinear mapping** (a.k.a. basis function)

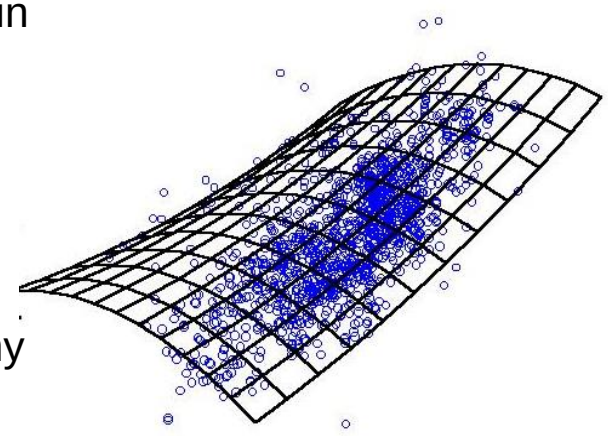
$$y_i = \boldsymbol{\alpha}^T \phi(\boldsymbol{x}_i) + \varepsilon_i$$

with $\phi: S^J \rightarrow \mathcal{R}^M$, usually $M > J$; hence more coefficients have to be fitted

- Since the model is linear in the parameters $\boldsymbol{\alpha}^T$, we can call $\phi(\boldsymbol{x}_i) = \boldsymbol{z_i}$ and use the usual solution $\boldsymbol{\alpha} = \left(\boldsymbol{Z^T Z}\right)^{-1} \boldsymbol{Z^T y}$, with $\boldsymbol{Z} = [\boldsymbol{z_1}, \ldots, \boldsymbol{z_n}]^{\boldsymbol{T}}$
- Some commonly used feature maps, adopted in many situations
  - Polynomial: ϕ: $x_i \rightarrow (1, x_i, x_i^2, x_i^3, \ldots)$
  - Veronese/ANOVA/Interaction mapping

ϕ: $x_{i1}, x_{i2} \rightarrow (1, x_{i1}, x_{i2}\, x_{i1}^2, x_{i2}^2, x_{i1} x_{i2}, \ldots)$

- In fact, log, sqrt, one-hot encoding, etc. and any other transformation

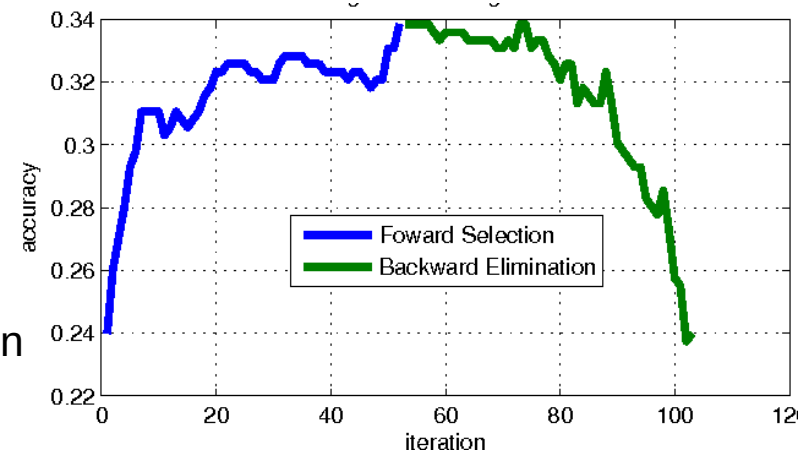


Explicit Feature Selection

- Selecting features aid in the process of
  - estimating model parameters (more degrees of freedom)
  - reduce the chances of overfitting (less variance)
- **Goal**: given a dataset $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$ with $\boldsymbol{x_i^{(J)}} \in \mathcal{R}^J$, find a subset $\boldsymbol{x_i^{(k)}} \in \mathcal{R}^k \subset \mathcal{R}^J$ such that $\mathcal{L}\left(\hat{f}\left(\boldsymbol{x_i^{(k)}}\right)\right) \lesssim \mathcal{L}\left(\hat{f}\left(\boldsymbol{x_i^{(J)}}\right)\right)$
- **Problem: $2^J - 1$ possible subsets…**
- **Greedy** alternatives: Forward Selection and Backward Elimination
  1. Start with a model $\hat{f}$ empty/full of variables
  2. For all the variables not selected/eliminated
     1. Add/remove one variable at each time
     2. Compute a loss criteria $\mathcal{L}$
  3. Add/remove a variable that minimized the loss criteria
  4. Return to step 2

Explicit Feature Selection

- Some usual performance criteria
  - Accuracy: $[\hat{f}(\boldsymbol{x}_i) == y_i]\frac{1}{n}$
  - RMSE: $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(\boldsymbol{x}_i)\right)^2}$



- And with some additional assumption
  - Nested F-test

$$F = \frac{SSE_{Reduced} - SSE_{Full}}{SSE_{Full}/(n - J + 2)}, remove\ if\ F_{stat} > F\left(\alpha; 1;\ n\ -(J-2)\right)$$

  - AIC (for $\boldsymbol{\beta}^*$ which the log-likelihood $\ell(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\beta}^*)$ is maximized)

$$AIC = 2 \cdot J - 2 \cdot \ell(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\beta}^*)$$

  - Mallow Cp

$$Mallow\ C_p = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(\boldsymbol{x}_i)\right)^2 + \frac{2}{n}\boldsymbol{Var}[y_i] \cdot J$$

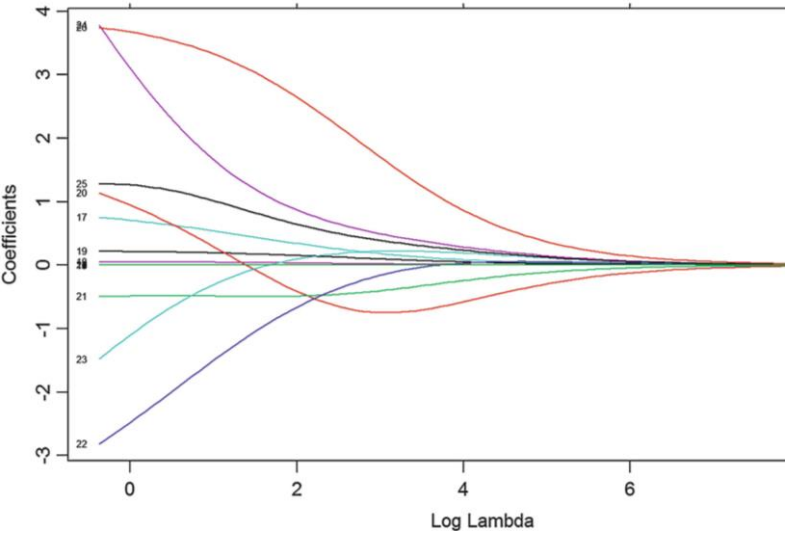
Implicit Feature Selection - Regularization

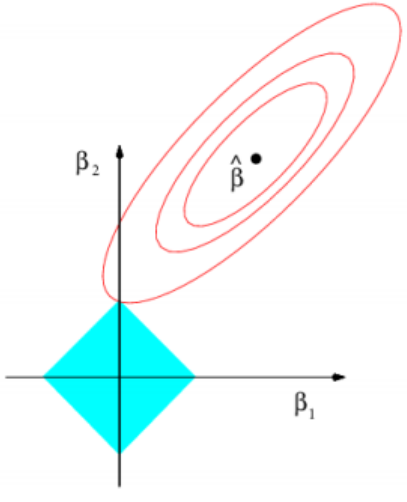
- Instead of removing via an algorithm the variables, refitting the model at every step, why not integrate the problem in the objective function:

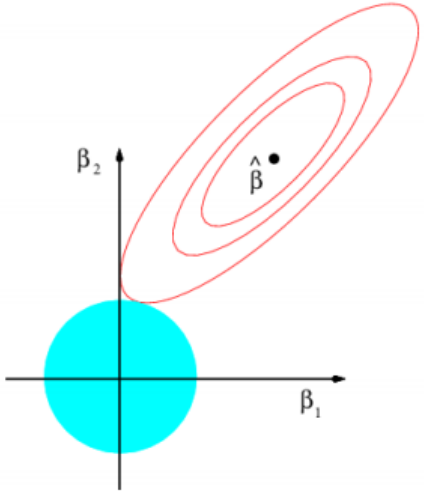
$$\min_{\boldsymbol{\beta}(\lambda)} MSE_p[\boldsymbol{\beta}(\lambda)] = \left|\left|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\right|\right|_2^2 + \lambda\left|\left|\boldsymbol{\beta}\right|\right|_p$$

which for different choices of the ℓ_p-norm (ℓ_1-Lasso, ℓ_2-Ridge, etc.) and values of λ we can expect different levels of **shrinkage/sparsity** of the solution; for $\lambda = 0$ we recover the traditional MSE minimization.

Con: now we have an additional hyperparameter to fine-tune!







Ridge Regression Estimation

- If we consider the Ridge Regression case, we have

$$\min_{\boldsymbol{\beta}(\lambda)} \boldsymbol{MSE}[\boldsymbol{\beta}(\lambda)] = \left|\left|\boldsymbol{y} - \boldsymbol{X\beta}\right|\right|_2^2 + \lambda\left|\left|\boldsymbol{\beta}\right|\right|_2^2, \text{ for a given } \lambda \geq 0$$

- we can rewrite this expression as

$$\min_{\boldsymbol{\beta}(\lambda)} \boldsymbol{MSE}[\boldsymbol{\beta}(\lambda)] = \boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X\beta} + \lambda\boldsymbol{\beta}^T\boldsymbol{I}_{JxJ}\boldsymbol{\beta}$$

- First order condition: $\boldsymbol{\nabla}_{\boldsymbol{\beta}(\lambda)}\, \boldsymbol{MSE}[\boldsymbol{\beta}(\lambda)] = 0$, then

$$-2\boldsymbol{X}^T\boldsymbol{y}^T + \boldsymbol{X}^T\boldsymbol{X\beta} + \lambda\boldsymbol{I}_{JxJ}\boldsymbol{\beta} = \boldsymbol{0} \rightarrow -2\boldsymbol{X}^T\boldsymbol{y} + (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_{JxJ})\boldsymbol{\beta} = \boldsymbol{0}$$

- Yielding the solution, akin to Linear Regression, but with a new param

$$\boldsymbol{\beta} = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_{JxJ}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- Second order condition: $\boldsymbol{H}[\boldsymbol{MSE}[\boldsymbol{\beta}(\lambda)]]$ is positive semi-definite

$$\boldsymbol{H}\left[\boldsymbol{MSE}[\boldsymbol{\beta}(\lambda)]\right] = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_{JxJ}\right)^T = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_{JxJ}\right)$$

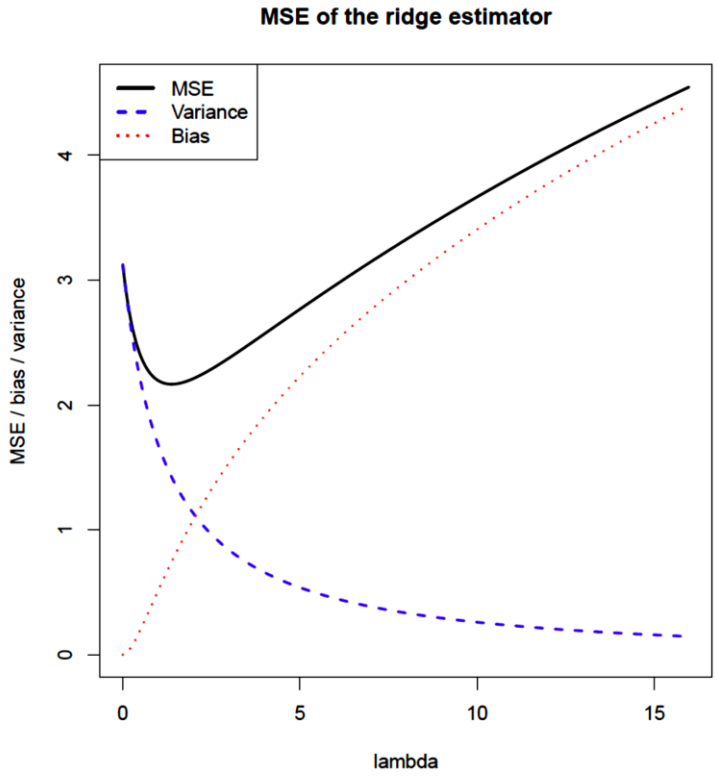
since $\boldsymbol{X}^T\boldsymbol{X}$ is positive semi-definite, $\boldsymbol{I}_{JxJ}$ is positive definite, the hessian matrix will be positive semi-definite for a suitable choice of λ (usually $\lambda \geq 0$)

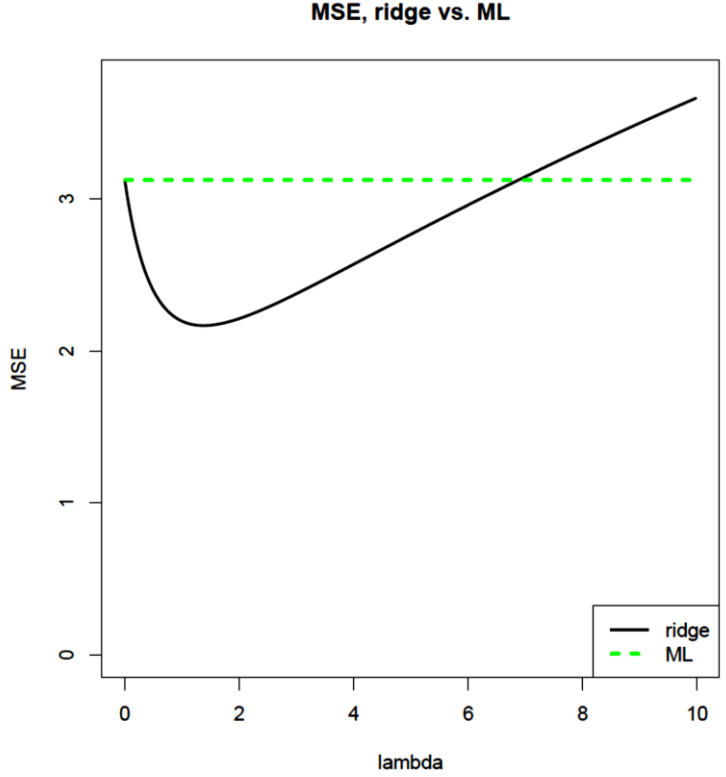
PS: remember to scale $\boldsymbol{X}$ columns (like having zero mean and unit std)

Some reasons why regularization works

- Theorem [Theobald, 1974]: By bias-variance trade-off, there exist $\lambda > 0$ such that $\boldsymbol{MSE}[\boldsymbol{\beta}(\lambda)] < \boldsymbol{MSE}[\boldsymbol{\beta}(0)]$

  Hence, for some choices of λ, Ridge outperforms Linear Regression





Additional Reading

- **Main References**
  - Friedman, J., Hastie, T., & Tibshirani, R. (2009). The elements of statistical learning. vol. 2. Springer.
    - Section 2.3 - Linear Models and Least Squares
    - Chapter 3 - Linear Methods for Regression, in particular
      - 3.2. - Linear Regression
      - 3.3.3 - Forward-Stagewise Regression
      - 3.4.1. - Ridge Regression
    - Section 4.3. - Linear Discriminant Analysis
    - Section 4.4. - Logistic Regression
    - Section 5.1. - Introduction to Basis Expansions
  - Efron, B., & Hastie, T. (2016). Computer age statistical inference (Vol. 5). Cambridge University Press.
    - Section 7.1. - James-Stein Estimator
    - Section 7.3. - Ridge Regression
- **Proof of Bias-Variance Trade-off for Ridge Regression**
  - Theobald, C. M. (1974). Generalizations of mean square error applied to ridge regression. Journal of the Royal Statistical Society. Series B (Methodological), 103-106.
- **Foundational Reference on Generalized Linear Models**
  - McCullagh, P., & Nelder, J. A. (1989). Generalized linear models (Vol. 37). CRC press
- **Seminar reference on Linear Methods in general**
  - Draper, N. R., & Smith, H. (2014). *Applied regression analysis* (Vol. 326). John Wiley & Sons.

Further Reading

- Theobald, C. M. (1974). Generalizations of mean square error applied to ridge regression. *Journal of the Royal Statistical Society. Series B (Methodological)*, 103-106.
- Draper, N. R., & Smith, H. (2014). *Applied regression analysis* (Vol. 326). John Wiley & Sons.
- Terasvirta, T., Tjostheim, D., & Granger, C. W. (2010). Modelling nonlinear economic time series. *OUP Catalogue*.
- McCullagh, P., & Nelder, J. A. (1989). *Generalized linear models* (Vol. 37). CRC press.
- De Gooijer, J. G. (2017). *Elements of nonlinear time series analysis and forecasting*. Springer.

email: faculty@mlinstitute.com

DISCLAIMER

The material in this presentation is based on information that we consider reliable, but we do not warrant that it is accurate or complete, and it should not be relied on as such. Opinions expressed are current opinions only. We are not soliciting any action based upon this material. Neither the author, his employers, any operating arm of his employers nor any affiliated body can be held liable or responsible for any outcomes resulting from actions arising as a result of delivering this presentation. This presentation does not constitute investment advice nor should it be considered as such.

The views expressed in this presentation represent those of the lecturer in his or her individual private capacity and should not be taken to be the views of any employer or any affiliated body, including the MLI, or of the lecturer as an employee of any institution or affiliated body. Either he/she or his/her employers may or may not hold, or have recently held, a position in any security identified in this document.