

# Regression and Classification Basics

Análisis de datos y su interpretación

Master en Big Data y Data Science

José R. Dorronsoro

Dpto. de Ingeniería Informática, Escuela Politécnica Superior

Universidad Autónoma de Madrid

28049 Madrid, Spain

# Antes de empezar ...

- Contenido
  - ① Introducción al aprendizaje automático
  - ② Preprocesado de datos
  - ③ Clustering
  - ④ Redes Neuronales
  - ⑤ Máquinas de vectores soporte
  - ⑥ Conjuntos de clasificadores y árboles de decisión
  - ⑦ Deep Learning
- Evaluación:
  - Asistencia a clase: 10%
  - Evaluación continua: 50%
  - Examen final: 40%

# Outline

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
  - The Classification Model
  - Nearest Neighbor Classification
- 6 Logistic Regression
- 7 Practical Classification
  - Measuring Classifier Accuracy
  - Practical Issues

# Modeling Basics

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
  - The Classification Model
  - Nearest Neighbor Classification
- 6 Logistic Regression
- 7 Practical Classification
  - Measuring Classifier Accuracy
  - Practical Issues

# What Is Machine Learning (ML)?

- Lofty definition: make machines learn!!!
  - Have to make “machines” and “learn” more precise
- The machines of ML: mathematical input–output processes that lend themselves to some form of (numerical) parameterization
- The learning process: adjust the machine’s parameters until a goal is reached
- New thing: “goal”?
  - At first sight, get something done
  - Ultimately, to minimize some error measure
- Summing things up: a ML process tries to find a concrete mathematical/algorithmic **input–output parameterized transformation** that **minimizes an error measure** by iteratively **adjusting the transformation’s parameters**

## Where Lies ML?

- In the middle of a possibly long process chain
- Before ML starts we must
  - Go from **raw to organized** data: accesing, gathering, cleaning, formatting, ...
  - Go from **organized to** (potentially) **informative** data: extracting basic and derived features
- After ML finishes we must perform
  - Outcome **evaluation**: how good/actionable it is
  - Outcome **exploitation**: collect, organize, act
  - **Individual model maintenance**: monitor performance, tune hyper-parameters
  - **Modeling life cycle maintenance**: discard old models, introduce new ones and **communicate** our work/results
- ML is in the middle of the global process chain but also in the middle of some subchains

# Supervised/Unsupervised Models

- Model types: **supervised, unsupervised**
- Supervised models:
  - Targets  $y^p$  are known and the model tries to predict or estimate them
  - These known targets guide, or **supervise**, model building
  - Main emphasis here
- Unsupervised models:
  - There are no predetermined or supervising outputs
  - But nevertheless the model is supposed to learn relations or find structure in the data
  - Sometimes as a first step towards a supervised model

# Regression and Classification

- Problems (usually) to be solved by models: regression, classification
- Patterns come in pairs  $(x, y)$ 
  - $x$ : inputs, predictors, features, independent variables
  - $y$ : target, response, dependent variable; numerical in regression, class labels in classification
- **Regression:** the desired output  $y$  is regressed into the inputs  $x$  to derive a model  $\hat{y} = f(x)$ 
  - We want  $y \simeq \hat{y}$  so having  $y - \hat{y}$  "small" is the natural goal
- **Classification:** inputs are derived from several classes  $C_1, \dots, C_K$ , to which labels  $\ell_k$  are assigned
  - The model now assigns a label  $\ell(x)$  to an input  $x$
  - If  $x$  is derived from  $C_k$  we want to have  $\ell(x) = \ell_k$
  - Here having  $\ell(x) - \ell_k$  "small" may not make sense



# The Boston Housing Problem

- This is a first “toy” problem
- We want to estimate the median of house values over an area from some information about it which we believe relevant
- Features  $x$ : several real estate–related variables of Boston areas
  - CRIM: per capita crime rate by town
  - RM: average number of rooms per dwelling
  - NOX: nitric oxides concentration (parts per 10 million)
  - AGE: proportion of owner-occupied units built prior to 1940
  - LSTAT: % lower status of the population
  - ...
- Target  $y$ : MEDV, median value of owner-occupied homes in \$1,000's

# Wind Energy Forecasting

- This is a second, real regression problem
- We want to estimate the hourly energy production of a wind farm from NWP variables which we believe relevant
- The **features** are the NWP variables
  - U, V surface wind components
  - U, V 100-meter wind components
  - Temperature
  - Pressure
  - ...
- The **target** is the energy produced during the outgoing hour

# The ML Cycle in Wind Energy

- Raw data: historic wind energy production data plus NWP files from weather forecasters
  - Possibly huge files with special formats
  - We have to extract the relevant NWP information, organize them in a suitable way and pair it with the energy data
- The ML core: whatever set of (non-linear) regression algorithm which you may think useful
- After ML is finished
  - Collect, organize and save the different model outputs
  - Select one single model output or some combination (more ML) of them as your system's output
  - Compute uncertainty estimates
  - Combine your outputs with someone's else
  - And keep up the entire process

## How to Build Regression Models

- In general we have a sample  $S = \{x^p, y^p\}$ ,  $1 \leq p \leq N$ , with  $x^p$  the **features** and  $y^p$  the **targets**
- We want to build a model  $\hat{y} = f(x)$  so that  $\hat{y}^p = f(x^p) \simeq y^p$ ; i.e., we want to **regress**  $y$  to the  $x$
- The concrete  $f$  is chosen within a certain family  $\mathcal{F}$ 
  - Examples here: linear regression, multilayer perceptrons (MLPs), SVMs
  - And also: Random Forests (RF), Gradient Boosting (GB), nearest neighbor (NN)
- Natural option to ensure  $f(x^p) \simeq y^p$ : choose  $f$  to minimize the sample mean square error (MSE)

$$\hat{e}(f) = \hat{e}_S(f) = \frac{1}{2N} \sum_{p=1}^N (y^p - f(x^p))^2$$

- Thus, the model we select is  $\hat{f} = \hat{f}_S = \arg \min_{f \in \mathcal{F}} \hat{e}_S(f)$

# Model Parameterization

- Usually individual models are selected through (ideally optimal) **parameter sets**
  - The parameters (weights)  $W \in R^M$  select a concrete  $f$  in  $\mathcal{F}$
- **Parametric** models have a fixed functional form  $f(x) = f(x; W)$
- Simplest example: linear regression, where  $M = d$  and  $W = (w_0, w)$

$$f(x; w_0, w) = w_0 + \sum_{j=1}^d w_j x_j = w_0 + w \cdot x$$

- **Semi-parametric** models also use weights but without a predefined functional form; MLPs but also RF or GBR
- **Non parametric** models do not use weights nor follow any broad functional form; NN models

# Model Estimation as Error Minimization

- For a parametric or semiparametric  $f(x; W)$  we can write  $\hat{e}_S(f) = \hat{e}_S(W)$
- The problem to solve becomes

$$\widehat{W}^* = \widehat{W}_S^* = \arg \min_W \hat{e}_S(f(\cdot; W)), \text{ i.e., } \hat{e}_S(\widehat{W}^*) \leq \hat{e}_S(W) \forall W$$

- In linear regression

$$\hat{e}(w_0, w) = \frac{1}{2N} \sum_p (y^p - w_0 - w \cdot x^p)^2$$

which ends up in a simple quadratic form

- The regression problem reduces to **minimize**  $\hat{e}_S(W)$ 
  - Something in principle well understood in mathematical optimization

# Regression Basics

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
  - The Classification Model
  - Nearest Neighbor Classification
- 6 Logistic Regression
- 7 Practical Classification
  - Measuring Classifier Accuracy
  - Practical Issues

# Regression Assumptions

- **Key assumption:**  $x$  and  $y$  are related as  $y = \phi(x) + n$  where
  - $\phi(x)$  is the **true** underlying function
  - $n$  is **additive noise** with 0 mean and finite variance  $\sigma_N^2$
- Our sample is just a particular instance of a deeper **sample generation process**
- Thus  $x, n$  are produced by **random variables**  $X, N$ 
  - And so is  $y$ , given by  $Y = \phi(X) + N$
- Moreover,  $X$  and  $N$  are **independent distributions** with densities  $q(x), \nu(n)$
- Thus,  $X$  and  $Y$  (or  $X$  and  $N$ ) have a joint density

$$p(x, y) = p(x, \phi(x) + n) = q(x) \nu(n) = q(x) \nu(y - \phi(x))$$



# The Best Regression Model

- It is easy to see that the best  $f$  is simply  $f(x) = E_y[y|x]$ , for

$$E_y[y|x] = E_n[\phi(x) + n] = \int (\phi(x) + n) \nu(n) dn = \phi(x)$$

- Have we finished? In theory yes; in practice, not at all!!!
  - We do not know  $\nu$  and, thus, cannot compute the required integrals
  - If for any  $x$  we would have  $M$  values  $y^j$ , we could try
$$\hat{\phi}(x) = \frac{1}{M} \sum_1^M y^j$$
  - But this doesn't happen either
- So we forget about using  $E[y|x]$  and get back to get an approximation  $f \simeq \phi$  from the sample

# Linear Models

- Assuming  $x \in R^d$ , the basic linear model is

$$f(x) = w_0 + \sum_1^d w_i x_i = w_0 + w \cdot x$$

- $w_0$  complicates notation; to drop it we center  $x$  and  $y$  so that  $E[x_i] = E[y] = 0$ ; then  $w_0 = 0$
- Then we are left with the simpler homogeneous model  $f(x) = w \cdot x$
- In practice we will always **normalize**  $x$ , for instance to have 0 mean and 1 standard deviation (std) on each feature
  - But not  $y$  if we may help it
- But: how do we find  $w$ ?

# 1-dimensional Linear Regression (LR)

- Assume that features  $X$  and target  $Y$  are **centered**, i.e., have 0 means
- For 1-dimensional patterns  $x$  the LR model then becomes

$$f(x) = w \cdot x$$

- And the error is then the function  $e(w)$

$$\hat{e}(w) = \frac{1}{2N} \sum_{p=1}^N (w \cdot x^p - y^p)^2 = \frac{1}{2N} \sum_p (\delta^p)^2$$

- The problem has obviously a minimum  $w^*$
- To find it we just solve  $\hat{e}'(w) = 0$

## Solving $\hat{e}'(w) = 0$

- To compute  $\hat{e}'(w)$  we have

$$\begin{aligned}\hat{e}'(w) &= \frac{1}{2N} \sum_p x^p \delta^p = \frac{1}{2N} \sum_p (w(x^p)^2 - x^p y^p) \\ &= w \left( \frac{1}{2N} \sum_p (x^p)^2 \right) - \frac{1}{2N} \sum_p x^p y^p\end{aligned}$$

- The optimal  $w^*$  solves  $\hat{e}'(w) = 0$  and is given by

$$w^* = \frac{\frac{1}{2N} \sum_p x^p y^p}{\frac{1}{2N} \sum_p (x^p)^2} = \frac{\frac{1}{2N} X \cdot Y}{\frac{1}{2N} X \cdot X} = \frac{\frac{1}{N} X \cdot Y}{\text{var}(x)}$$

where  $X$  and  $Y$  denote the  $N$  dimensional vectors  $(x^1, \dots, x^N)^t$ ,  $(y^1, \dots, y^N)^t$

# General Linear Regression

- Assume again that  $X$  and  $Y$  are centered
- The LR model becomes now  $f(x) = \sum_1^d w_i x_i = w \cdot x$
- If  $Y$  is the  $N \times 1$  **target** vector and we organize the sample  $S$  in a  $N \times d$  **data matrix**  $X$ , the sample mse is given by

$$\begin{aligned}\hat{e}(w) &= \frac{1}{2N} \sum_p (w \cdot x^p - y^p)^2 = \frac{1}{2N} (Xw - Y)^t (Xw - Y) \\ &= \frac{1}{2N} (w^t X^t X w - 2w^t X^t Y + Y^t Y)\end{aligned}$$

- Now we have to solve  $\nabla \hat{e}(w) = 0$ , i.e.,  $\frac{\partial \hat{e}}{\partial w_i}(w) = 0$
- It is easy to see that

$$\nabla \hat{e}(w) = \frac{1}{N} X^t X w - \frac{1}{N} X^t Y = \hat{R} w - \hat{b}$$

## Solving the Linear Equations

- The optimal  $\hat{w}^*$  must verify  $\nabla \hat{e}(\hat{w}) = \hat{R} \hat{w} - \hat{b} = 0$ , where

$$\hat{R} = \frac{1}{N} X^t X, \quad \hat{b} = \frac{1}{N} X^t Y$$

- Over the original, non-centered data matrix we have

$$\hat{R} = \frac{1}{N} (X - \bar{X})^t (X - \bar{X});$$

i.e.,  $\hat{R}$  is the **sample covariance matrix**

- If  $\hat{R}$  is invertible, we just solve the linear system  $\hat{R} \hat{w} - \hat{b} = 0$
- And obtain the sample-dependent optimal  $\hat{w}^*$  as

$$\hat{w}^* = \hat{R}^{-1} \hat{b} = (X^t X)^{-1} X^t Y$$

# Finding Optimal Models

- For general regression models it may not be possible to solve analytically the equation  $\nabla \hat{e}(W) = 0$ 
  - For LR and big data, covariance matrices over large datasets or dimensions may not be computed
  - Numerical methods are needed

- The simplest numerical alternative is **gradient descent**:

- Starting from some random  $W^0$  we iteratively compute

$$W^{k+1} = W^k - \rho_k \nabla \hat{e}(W^k) = W^k - \frac{\rho}{N} (X^t X W^k - X^t Y)$$

- Component wise:  $w_i^{k+1} = w_i^k - \rho_k \frac{\partial \hat{e}}{\partial w_i}(W^k)$
- $\rho_k$  is the **learning rate**
- If  $W^k \rightarrow W^*$ , then  $\nabla \hat{e}(W^*) = 0$ 
  - Since our problems have obviously minima, this should be enough

## Measuring Model Fit

- First option: **Root Square Error**

$$RSE = \sqrt{\frac{1}{N} \sum (y^p - \hat{y}^p)^2} = \sqrt{\frac{1}{N} RSS}$$

- OK, but how good is this? We must always have a **base model** to benchmark our results
- Simplest “model”: the mean  $\bar{y} = \frac{1}{N} \sum_1^N y^p$ , with square error

$$\frac{1}{N} \sum (y^p - \bar{y})^2 = \frac{1}{N} TSS = \text{Var}(y)$$

- We can compare our model against our base computing

$$\frac{RSE^2}{\text{Var}(y)} = \frac{\sum (y^p - \hat{y}^p)^2}{\sum (y^p - \bar{y})^2} = \frac{RSS}{TSS}$$

- The widely used  $R^2$  coefficient is simply  $R^2 = 1 - \frac{RSS}{TSS}$



# Regularization

- Our regression solution  $\hat{w}^* = (X^t X)^{-1} X^t Y$  won't work if  $X^t X$  is not invertible
  - For instance, when some features are correlated
- We could fix this working instead with  $X^t X + \alpha I$  for some  $\alpha > 0$
- To make this practical, note that  $\hat{w}^* = (X^t X + \alpha I)^{-1} X^t Y$  minimizes

$$e_R(w) = \frac{1}{2N} \sum_p (y^p - w \cdot x_p^p)^2 + \frac{\alpha}{2} \|w\|^2,$$

- This is the **Ridge Regression** problem
  - Our first example of **regularization**, a key technique in Machine Learning
  - **All ML models must be regularized in some way**
- Important issue: how to find the right choice for  $\alpha$ ?

# Takeaways on Linear Regression

- ① We introduced **supervised** models
- ② We have reviewed the essentials of the **linear regression model** (always the first thing to try)
- ③ We have identified target noise as an unimprovable error bound
- ④ We have considered model estimation as a problem on **error minimization**
- ⑤ We have seen how to build linear models analytically and numerically
- ⑥ We have defined how to measure model fit
- ⑦ We have introduced regularization

# Bias, Variance and Cross Validation

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation**
- 4 Data and Model Analysis
- 5 Basic Classification
  - The Classification Model
  - Nearest Neighbor Classification
- 6 Logistic Regression
- 7 Practical Classification
  - Measuring Classifier Accuracy
  - Practical Issues

## Sample Dependence

- Important: **everything is sample dependent** for if we change  $S$  we get a different model
- We get sample-dependent weights  $\widehat{W} = \widehat{W}_S$  and model  $\widehat{f}_S(x) = \widehat{f}(x; \widehat{W}_S)$
- We must control their dependence on the concrete  $S$  sample used to build it
- Moreover, we must apply our model on new, **unseen** samples
- We must have a sample generating procedure that ideally gives homogeneous samples and a robust model building methodology
- Both together should (reasonably) guarantee that, for two  $S, S'$ ,

$$\widehat{f}_S(x) \simeq \widehat{f}_{S'}(x)$$

## Sample Bias and Variance

- With several **independent** samples  $S_1, \dots, S_M$ , it is natural to use as our best final model the averages of the  $\hat{f}_{S_m}(x)$  models, i.e.,

$$\frac{1}{M} \sum_1^M \hat{f}_{S_m}(x) \simeq E_S[\hat{f}_S(x)] = \hat{f}_N(x)$$

- The expectation  $E_S[\hat{f}_S(x)]$  is taken over all possible samples  $S$  of size  $N$
- The **variance** of the  $\hat{f}_S(x)$  estimates is then

$$V_N(x) = E_S [(\hat{f}_S(x) - \hat{f}_N(x))^2]$$

# Bias Versus Variance

- Ideally we would like to have a model such that

$$\hat{f}_N(x) - \phi(x) \simeq 0,$$

i.e., a model with small **bias**

- This should be achievable with rich, highly flexible models
- Or with essentially no regularization
- But we would also like to have a model such that

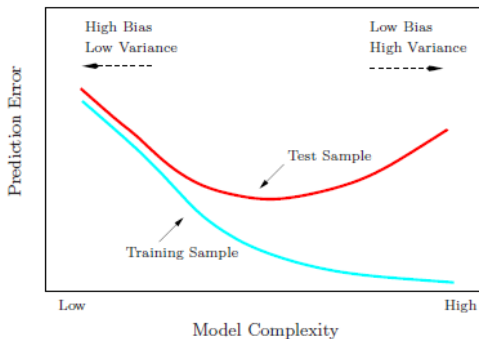
$$V_N(x) \simeq 0,$$

i.e., a model with small **variance**  $V_N(x)$

- This should be achievable with simple models with few parameters
- Or with more severe regularization
- But obviously both goals are contradictory to a large extent

# The Bias–Variance Tradeoff

- There is thus a **tradeoff** between bias (low for complex models) and variance (low for simple models)



Taken from *Hastie et al.*, p. 38

## Evaluating Expected Performance

- It is obvious that before we start applying a model, we should have a reasonably accurate idea of its performance in practice
- I.e., we want to estimate the model's **generalization performance**
- Estimating the generalization performance **only over the sample  $S$  used for training results in misleading error values**
- The preceding suggests to have  $M$  independent subsamples  $S_m$  and
  - Compute  $\hat{f}_M(x) = \frac{1}{M} \sum_m \hat{f}_{S_m}(x) \simeq \hat{f}_N(x)$
  - Get the error estimate  $\hat{e} = \frac{1}{N} \sum_p (y^p - \hat{f}_M(x^p))^2$  over a new, **unseen** sample  $S' = \{(x^p, y^p)\}$
- But since usually we only have a single  $S$ , we apply **Cross Validation (CV)** to get our first realistic generalization error estimates



# Cross Validation

- In Cross Validation (CV) we
  - Randomly split the sample  $S$  in  $M$  subsets  $S_1, \dots, S_M$
  - Work with  $M$  **folds**: pairs  $(S_m, S_m^c)$ , with

$$S_m^c = S - S_m = \cup_{i \neq m} S_i$$

- Build  $M$  different models **using the  $S_m^c$  as training subsets**
  - Compute their errors  $e_m$  on the folds' **validation subsets  $S_m$**
  - Use these errors' average as a first estimate of the true model performance
- CV can and **must be used** in any model building procedure
- Most data science packages have tools to simplify this
- We will also use CV to find an **optimal model hyper-parameter  $\alpha$**  in Ridge Regression

## Grid Hyper-parameter Selection

- Build  $M$  **folds**: pairs  $(S_m, S_m^c)$  and use  $S_m^c$  as training and  $S_m$  as the validation subsets
- Fix a hyper-parameter range  $[0, A]$ 
  - $\alpha = 0$ : no penalty and, thus, small bias and high variance
  - $\alpha = A$ : large penalty and, thus, small variance but high bias
- Select an  $L + 1$  point **grid**

$$G = \left\{ 0, \frac{A}{L}, \frac{2A}{L}, \dots, \frac{\ell A}{L}, \dots, \frac{LA}{L} = A \right\}$$

- At each  $\alpha_\ell = \frac{\ell}{L}A$ ,  $0 \leq \ell \leq L$ 
  - Train  $M$  models on the  $S_m^c$  using the hyper-parameter  $\alpha_\ell$
  - Average their  $M$  validation errors  $e_m$  on the  $S_m$  to get the error  $e(\alpha_\ell)$  at  $\alpha_\ell$
- Finally choose the (hopefully) optimal hyper-parameter  $\alpha^*$  as

$$\alpha^* = \arg \min_{0 \leq \ell \leq L} e(\alpha_\ell)$$

## Takeaways on Bias, Variance and CV

- ① We have stressed that **any model estimation is sample-dependent** and that this has to be controlled
- ② We have introduced the **bias** and **variance** as the two key components of any model error
- ③ We have discussed **bias-variance trade-off**
- ④ We have introduced **Cross Validation** here as a tool to estimate a **model's generalization performance**
- ⑤ We have also introduced **Cross Validation** as a tool to estimate a **model's hyper-parameters**

# Data and Model Analysis

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis**
- 5 Basic Classification
  - The Classification Model
  - Nearest Neighbor Classification
- 6 Logistic Regression
- 7 Practical Classification
  - Measuring Classifier Accuracy
  - Practical Issues

## And So What?

- Key question: what are models for?
  - First answer: to be used to derive new predictions
  - Better answer: to extract knowledge and to make inference on the underlying problem
- In this light, LR models are simple, perhaps not too powerful, but certainly useful
  - They are the first tool to apply in (almost) any problem analysis
- Some questions are easier to answer for them:
  - Which variables do influence the target and which do not?
  - What are the strongest predictive variables?
  - Are there related/redundant variables?
  - Is the relationship actually linear?

## Issues with LR

- Before building any model we must perform a prior data analysis to keep under control important issues:
  - **Collinearity**: predictor variables that are redundant
  - **Outliers**: points  $(x^p, y^p)$  with a “normal” pattern  $x$  but an unlikely target value  $y^p$
  - **High-leverage points**: points  $(x^p, y^p)$  with an unlikely pattern  $x^p$  and a reasonable target value  $y^p$
- And after a model is built we must check if its results agree with its assumptions
  - **Linearity** of the response–predictor relationships: if not, the LR will be poor
  - **No correlation of error terms**, i.e. our basic model assumption does hold
  - **No heteroscedasticity**, i.e., no non-constant variance of error terms, that varies on several  $x$  regions

## Detecting and Handling Data Issues

- Before **any** model is built we **must** try detect possible data inconsistencies and/or redundancies
- Feature collinearity: look at least at the correlation matrix
- Analyze feature–target scatterplots; if possible, look also at the two–predictor scatterplots (though there are  $d(d - 1)/2$  of them)
- Outliers: will cause  $(x^p, y^p)$  to be far from the line fit or the residual to be out of range
  - Can detect them with box plots
- High-leverage points:  $x^p$  outside the main  $x$  range; harder to spot in multidimensional models
- We consider all this over the Boston Housing dataset

## Housing: First Conclusions on the Data

- Collinearity: some predictor variables may be redundant
  - AGE-DIS: proportion of units built prior to 1940 and weighted distances to five employment centres
  - RAD-TAX: accessibility to radial highways and full-value property-tax rate
  - NOX-INDUS
- Outliers: points  $(x^p, y^p)$  with a normal pattern  $x$  but an unlikely target value  $y^p$ 
  - ???
- High-leverage points (HLPs): perhaps at variables
  - ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
  - CHAS: 1 if tract bounds Charles river; 0 otherwise
  - B:  $1000(Bk - 0.63)^2$ , with  $Bk$  the proportion of blacks by town
  - But have to look at HLPs as D-dimensional points and not features



## Detecting and Handling Model Issues

- After the model is built we check whether it supports the basic LR assumptions
- Non-linearity: a residual plot should not have any structure
- Uncorrelated error terms: residuals do not change rather smoothly
- Error histograms should be symmetric and sharp at 0
- Heteroscedasticity: residual plots do not show a “funnel” like structure
- **Always address these possible problems:** if not, we may be fooling ourselves with an untenable model
- Let's build LR models over the Boston Housing data

# Housing: First Conclusions on the Linear Model

- Recall the first things to look at after LR model building:
  - Linearity of the response-predictor relationships?
  - No correlation of residuals?
  - No heteroscedasticity?
- Linearity of the response-predictor relationships: not bad
  - If perfect fit,  $y$  and  $\hat{y}$  in diagonal; here in near diagonal
- Correlation of residuals only for large targets
  - Perhaps we should think about two separate models
- No heteroscedasticity, i.e., constant variance of residuals
  - No funnel appears in target-residual representation but there is still a bias
- Build a second model?

# Takeaways on Data and Model Analysis

- ① Before any model building we must analyze and understand our data
- ② We must understand the assumptions our model implies on the data
  - If they aren't true the model won't be very good
- ③ This must be checked after the model is built
- ④ LR models are simple but their assumptions are of interest to any other model
- ⑤ LR are the first models to build, to have a benchmark and to better understand the problem and its data
- ⑥ And
  - Always tune the hyperparameters for our models
  - Always try out many different models
  - Always explore several feature representations for our data

# Basic Classification

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification**
  - The Classification Model
  - Nearest Neighbor Classification
- 6 Logistic Regression
- 7 Practical Classification
  - Measuring Classifier Accuracy
  - Practical Issues

# A First Problem: Pima Indian Diabetes

- We want to diagnose whether a person may have diabetes from some clinical measures
- Features  $x$ : clinical measures
  - 'numPregnant'
  - 'bloodPress'
  - 'massIndex'
  - 'age' ...
- Target  $y$ : 0 (no diabetes), 1 (diabetes)
- Clear goal but perhaps too radical
- Better: try to estimate the probability of having diabetes depending on the features

## Classification Setup

- We have random patterns  $\omega$  from  $M$  classes,  $C_1, \dots, C_M$
- Over each pattern we “measure”  $d$  features  $x = x(\omega) \in \mathbb{R}^d$ 
  - $x$  inherits the randomness in  $\omega$  and becomes a random variable
- A  $\omega$  has a **prior probability**  $\pi_m$  of belonging to  $C_m$
- Inside each class  $C_m$  there is a **conditional class density**  $f(x|m)$  that “controls” the appearance of a given  $x$
- The  $\pi_m$  and  $f(x|m)$  determine the **posterior probability**  $P(k|m)$  that  $x$  comes from class  $C_m$
- **Intuition:** we should assign  $x$  to the class with the largest  $P(k|m)$ , that is, work with the classifier

$$\delta(x) = \arg \max_m P(m|x)$$

# Computing Posterior Probabilities I

- **Bayes rule:**  $P(B|A) = \frac{P(A \cap B)}{P(A)}$
- This requires to work with probabilities, not densities:  
 $P(\{x\}) = P(m \cap \{x\}) = 0!!!$
- But we can use the approximation

$$\begin{aligned} P(m|x) &\simeq P(m|B_r(x)) = \frac{P(C_m \cap B_r(x))}{P(B_r(x))} = \frac{P(B_r(x)|m)P(C_m)}{P(B_r(x))} \\ &= \frac{\pi_m P(B_r(x)|m)}{P(B_r(x))} \\ &= \pi_m \frac{\int_{B_r(x)} f(y|m) dy}{\int_{B_r(x)} f(z) dz} \end{aligned}$$

where we assume that features  $x$  are measured independently from classes  $m$

## Computing Posterior Probabilities II

- Remember the Fundamental Theorem of Calculus:

if  $F(x) = \int_a^x f(y)dy$ ,

$$\lim_{\epsilon \rightarrow 0} \frac{1}{2\epsilon} \int_{x_0 - \epsilon}^{x_0 + \epsilon} f(y)dy = \frac{dF}{dx}(x_0) = f(x_0)$$

- In  $d$  dimensions it becomes

$$g(w) = \lim_{r \rightarrow 0} \frac{1}{|B_r(w)|} \int_{B_r(w)} g(z)dz$$

- Putting everything together, we arrive

$$\begin{aligned} P(m|x) &= \lim_{r \rightarrow 0} P(m|B_r(x)) = \pi_m \lim_{r \rightarrow 0} \frac{\int_{B_r(x)} f(y|m)dy}{\int_{B_r(x)} f(z)dz} \\ &= \pi_m \lim_{r \rightarrow 0} \frac{\frac{1}{|B_r(x)|} \int_{B_r(x)} f(y|m)dy}{\frac{1}{|B_r(x)|} \int_{B_r(x)} f(z)dz} = \frac{\pi_m f(x|m)}{f(x)} \end{aligned}$$



# The Obviously Optimal Classifier

- As a consequence we would decide according to a **classifier** function  $\delta$

$$\begin{aligned}\delta_B(x) &= \arg \max_m P(m|x) = \arg \max_m \frac{\pi_m f(x|m)}{f(x)} \\ &= \arg \max_m \pi_m f(x|m)\end{aligned}$$

- Again we have arrived to the optimal solution of the classification problem, the **Bayes Classifier**  $\delta_B$  (although we cannot measure it yet)
- But ... this doesn't look too practical for we do not know either  $\pi_m$  or (much harder)  $f(x|m)$

# Approximating the Bayes Classifier

- To define  $\delta_B$  we need to know the prior probabilities  $\pi_m$  and the prior densities  $f(x|m)$
- A reasonable choice for  $\pi_m$  is  $\hat{\pi}_m = \frac{N_m}{N}$ , where  $N_m$  is the number of patterns of  $C_m$  in the sample
- But effective multidimensional density estimates are rather difficult, because of the **curse of dimensionality**
  - Densities generalize histograms
  - Good histograms need accurate counts of elements nearby
  - But in high dimensions there won't be nearby elements!!
- Options:
  - Restrict possible density models: logistic regression
  - Assume no model and apply a Nearest Neighbor (NN) strategy

# Nearest Neighbor Classification

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification**
  - The Classification Model
  - Nearest Neighbor Classification**
- 6 Logistic Regression
- 7 Practical Classification
  - Measuring Classifier Accuracy
  - Practical Issues

# The $k$ -NN Classifier

- Very simple: at any  $x$  consider the subset  $N_k(x)$  of its  $k$  closest sample points and
  - Let  $n_m(x)$  the number of elements of class  $m$  in  $N_k(x)$
  - Notice that  $0 \leq n_m(x) \leq k$
  - Define  $\delta_{kNN}(x) = \arg \max_m n_m(x)$
- That is,  $\delta_{kNN}$  assigns  $x$  to the class that has more patterns in  $N_k(x)$
- We can partially justify this definition from a Bayesian point of view
- Assume that  $B_r(x)$  is the smallest ball that contains  $N_k(x)$  and consider the approximations
  - $|B_r(x)| f(x|m) \simeq \int_{B_r(x)} f(z|m) dz = P(C_m \cap B_r(x)) \simeq \frac{n_m(x)}{N_m}$
  - Similarly,  $|B_r(x)| f(x) \simeq \int_{B_r(x)} f(z) dz = P(B_r(x)) \simeq \frac{k}{N}$
  - And  $\pi_m \simeq \frac{N_m}{N}$

## $k$ -NN and the Bayes Classifier

- We then have

$$\begin{aligned} P(m|x) &= \pi_m \frac{f(x|m)}{f(x)} = \pi_m \frac{|B_r(x)| f(x|m)}{|B_r(x)| f(x)} \\ &\simeq \frac{N_m}{N} \frac{n_m(x)}{N_m} \frac{1}{\frac{n(x)}{N}} = \frac{n_m(x)}{n(x)} \end{aligned}$$

- Therefore  $\delta_{kNN}$  should be close to  $\delta_B$ , for

$$\begin{aligned} \delta_{kNN}(x) &= \arg \max_m n_m(x) = \arg \max_m \frac{n_m(x)}{n(x)} \\ &\simeq \arg \max_m P(m|x) \end{aligned}$$

## By the Way: $k$ -NN Regression

- Sometimes the relation between features  $x$  and targets  $y$  doesn't justify a strong model  $y = \phi(x) + n$
- $k$ -NN Regression relies on a reasonable assumption: **Predictors that are close should give predictions that are also close**
- In  $k$ -NN Regression we fix a number  $k$  of neighbors to be considered and for an input  $x$  set

$$\hat{y} = Y_{kNN}(x) = \frac{1}{k} \sum_{x^p \in N_k(x)} y^p$$

where  $N_k(x)$  denotes again the  $k$  sample points closest to  $x$

- **Weighted variants:** for instance,  
$$Y_k^w(x) = \frac{1}{C_k(x)} \sum_{x^p \in N_k(x)} \frac{1}{\|x^p - x\|} y^p$$
  - $C_k(x) = \sum_{x^p \in N_k(x)} \frac{1}{\|x^p - x\|}$  is a normalizing constant

## Some $k$ -NN Issues

- **Q1: How do we choose  $k$ ?** Using CV, of course
- There are no closed form solution and we have to balance again the bias–variance tradeoff
  - Small variance with large  $k$ : if  $k = N$ ,  $k$ -NN returns the mean
  - Small bias with small  $k$ : if  $k = 1$  a very close point should give a very close prediction
  - But also large variance: the nearest point to  $x$  in another sample may belong to another class
- **Q2: Is  $k$ -NN always meaningful?**
- We have to modify our first assumption: Predictors that are close should give predictions that are also close, **provided that there are enough of them close by**
  - In fact, if  $x$  is away from the sample, the average over  $N_k(x)$  may be meaningless

# The Curse of Dimensionality

- This consideration reflects the **curse of dimensionality**:  
Even for low dimensions and large samples, the sample space is essentially empty
- Thus, for most problems, **there never will be enough close points**
- As a consequence, to get  $k$  observations we may go too far away from  $x$  and the average will not be meaningful
- Therefore, unless we deal with violently non-linear problems, a simple model such as linear or logistic regression (later) may be better than  $k$ -NN for moderate dimensions



# Logistic Regression (LR)

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
  - The Classification Model
  - Nearest Neighbor Classification
- 6 Logistic Regression**
- 7 Practical Classification
  - Measuring Classifier Accuracy
  - Practical Issues

## Linear Regression for Classification?

- $k$ -NN Classifier is simple but also crude; have to look elsewhere
- Building a regression model with targets some coding of class labels usually doesn't make sense
- But for a binary 0–1 response, it can be shown that the  $w_0 + w \cdot x$  obtained using linear regression is in fact an estimate of  $P(1|x)$ 
  - We may thus fix a threshold  $\delta_0$  and decide 0 if  $w_0 + w \cdot x < \delta_0$  and 1 otherwise
  - However, we may end up with probability estimates less than 0 or bigger than 1!!!
- We know that our goal should be to estimate  $P(j|m)$ ; let's try to attain it!

# Logistic Regression (LR)

- We assume

$$P(1|x) = \frac{1}{1 + e^{-(w_0 + w \cdot x)}}$$

- Then  $0 \leq P(1|x) \leq 1$  for any  $x$
- We then have

$$P(0|x) = 1 - P(1|x) = \frac{e^{-(w_0 + w \cdot x)}}{1 + e^{-(w_0 + w \cdot x)}} = \frac{1}{1 + e^{w_0 + w \cdot x}}$$

- Notice that if  $w_0 + w \cdot x = 0$ ,  $P(1|x) = P(0|x) = 0.5$
- The ratio  $\frac{P(1|x)}{P(0|x)} = e^{w_0 + w \cdot x}$  is called the **odds** of  $x$  and its log the **log odds** or **logit**
- Thus, the basic assumption in LR is that the **logit is a linear function**  $w_0 + w \cdot x$  of  $x$
- We have the model  $f(x; w)$ ; we need a loss function  $L(w)$  to minimize for which we use the sample's **likelihood**

## Sample's Likelihood

- Assume a sample  $S = \{(x^p, y^p)\}$ , with  $y^p$  either 1 or 0 and the  $(x^p, y^p)$  independent
- If the  $y^p$  labels are derived from a LR model with weights  $w_0, w$ , the probability of obtaining them in the sample  $S$  is

$$\begin{aligned} P(S, w_0, w) &= \left\{ \prod_{y^p=1} P(1|x^p) \right\} \left\{ \prod_{y^p=0} P(0|x^p) \right\} \\ &= \prod_1^N P(1|x^p)^{y^p} P(0|x^p)^{1-y^p} \end{aligned}$$

because

- If  $y^p = 1$ ,  $P(1|x) = P(1|x^p)^{y^p} P(0|x^p)^{1-y^p}$  and
- If  $y^p = 0$ ,  $P(0|x) = P(1|x^p)^{y^p} P(0|x^p)^{1-y^p}$

## Max Log-Likelihood Estimation

- The log-likelihood of  $w_0, w$  given  $S$  is then

$$\begin{aligned}\ell(w_0, w; S) &= \log P(S; w_0, w) \\&= \sum_p \{y^p \log p(1|x^p) + (1 - y^p) \log p(0|x^p)\} \\&= \sum_p y^p \log \frac{p(1|x^p)}{p(0|x^p)} + \sum_p \log p(0|x^p) \\&= \sum_p y^p (w_0 + w \cdot x^p) - \sum_p \log(1 + e^{w_0 + w \cdot x^p})\end{aligned}$$

- We can thus estimate the optimal  $\hat{w}_0^*, \hat{w}^*$  as

$$\hat{w}_0^*, \hat{w}^* = \arg \min_{w_0, w} -\ell(w_0, w; S)$$

## Newton–Raphson Solution

- Since  $\ell$  is a convex differentiable function of  $(w_0, w)$ , it is enough to solve  $\nabla\ell(w_0, w) = 0$ 
  - However, it doesn't admit a closed form solution
- We apply the Newton–Raphson iterative method, here equivalent to the general Newton method for function minimization
- Starting with an initial random  $w^0$ , Newton's iterations are

$$w^{k+1} = w^k - (\mathcal{H}_\ell(w^k))^{-1} \nabla\ell(w^k)$$

- $\mathcal{H}_\ell(w^k)$  denotes the Hessian of  $\ell$  at  $w^k$ , which may or may not be invertible
  - Everything is fine if  $w^0$  is close enough to the optimum  $w^*$  but far away things may get tricky
- Logistic regression is typical of many of the model building methods used in Machine Learning

# Learning in ML

- The general approach to **learning** is the following:
  - A model  $f(x; w)$  is chosen
  - Given a sample  $S = \{(x^1, y^1), \dots, (x^N, y^N)\}$ , we define a sample dependent loss function

$$L(w) = L(w_0, w; S) = L(y^1, \dots, y^N, f(x^1; w), \dots, f(x^N; w))$$

- $L(w)$  is minimized starting at some  $w^0$  and applying iterations

$$w^{k+1} = w^k - \rho_k G(w^k, S)$$

with  $\rho_k$  a **learning rate** and  $G$  some sample-dependent vectorial function

- When  $G(w) = \nabla L(w)$  we have **gradient descent**
- When the entire sample  $S$  is used at each iteration, we speak of **batch learning**
- When only single patterns  $(x^p, y^p)$  are used, we speak of **on-line learning**
- Several such procedures will appear here in the coming weeks

# Measuring Classifier Accuracy

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
  - The Classification Model
  - Nearest Neighbor Classification
- 6 Logistic Regression
- 7 Practical Classification
  - Measuring Classifier Accuracy
  - Practical Issues



# The Confusion Matrix I

- Consider a two class problem with labels  $y = 0, 1$
- We will call patterns with label 1 **positive** and those with label 0 **negative**
  - Usually the positive patterns are the interesting ones: sick people, defaulted loans, ...
- Let  $\hat{y} = \hat{y}(x)$  the label predicted at  $x$ ; we say that  $x$  is
  - **True Positive (TP)** if  $y = \hat{y} = 1$
  - **True Negative (TN)** if  $y = \hat{y} = 0$
  - **False Positive (FP)** if  $y = 0$  but  $\hat{y} = 1$
  - **False Negative (FN)** if  $y = 1$  but  $\hat{y} = 0$
- The standard way of presenting these data is through the **confusion matrix**

## The Confusion Matrix II

- Standard layout

	P' (Predicted)	N' (Predicted)
P (Actual)	True Positive	False Negative
N (Actual)	False Positive	True Negative

- Other layouts:
  - Positives (with label 1) at bottom (as done in `confusion_matrix` of `sklearn`)
  - Predicted values in rows, real values in columns

# Classifier Metrics

- The classifier **accuracy** is  $acc = \frac{TP+TN}{N}$
- $acc$  is the first thing to measure but it may not be too significant: if the number  $N_0$  of negatives is  $\gg N_1$ , the number of positives
  - The classifier  $\delta(x) = 0$  will have a high accuracy  $N_0/N \simeq 1$
  - But it will also be useless!!
- First variant: Precision, Recall
  - **Recall**:  $TP/(TP + FN)$ , i.e., the fraction of positives detected
  - **Precision**:  $TP/(TP + FP)$ , i.e., the fraction of true alarms issued
- Recall measures how many positive cases we recover; precision the effort we need for that
- Ideal classifier: high recall, high precision

# Practical Issues

- 1 Machine Learning Modeling Basics
- 2 Basic Regression
- 3 Bias, Variance and Cross Validation
- 4 Data and Model Analysis
- 5 Basic Classification
  - The Classification Model
  - Nearest Neighbor Classification
- 6 Logistic Regression
- 7 Practical Classification**
  - Measuring Classifier Accuracy
  - Practical Issues**

## What's New from Regression?

- Some things change from regression, some don't
- We should check feature correlations: they will affect most models
- Important: **positive and negative-class feature histograms**
  - Scatter plots  $(x_i, y)$  are usually less informative
- The **bias-variance trade-off** is subtler in classification
- Accuracy, recall, precision are the usual model quality measures
- We use CV with **stratified folds** to estimate generalization performance
- We also use CV for hyperparameter estimation, for regularization will also be needed
  - In LR we should minimize  $-\ell(w_0, w; S) + \frac{\alpha}{2} \|w\|^2$

## How to Handle Posterior Probabilities

- If possible, we don't want labels as model outputs but **posterior probabilities**
- Most models give them as pairs

$$(\hat{P}(0|x), \hat{P}(1|x)) = (\hat{P}(0|x), 1 - \hat{P}(0|x))$$

- In principle we would decide 1 if  $\hat{P}(1|x) > 0.5$  and viceversa, but this may be too crude
- It may be advisable to set a threshold  $\theta$  and decide 1 if  $\hat{P}(1|x) > 1 - \theta$  and 0 if  $\hat{P}(1|x) < \theta$
- For **imbalanced** problems where  $\pi_0 \gg \pi_1$  (usually the interesting ones) we would have  $\hat{P}(0|x) \simeq 1$  for most  $x$ 
  - In this case we may choose another  $\theta \simeq 1$  and **suggest** 1 if  $\hat{P}(0|x) < \theta$

# Takeaways on Basic Classification

- ① We have introduced the classification problem as one of computing posterior probabilities
- ② We have found the optimal Bayes classifier and approximated it by  $k$ -NN
- ③ We have introduced several measures of classifier performance
- ④ We have introduced Logistic Regression and the numerical minimization of its (minus) log-likelihood
- ⑤ We have reviewed some practical issues of classification