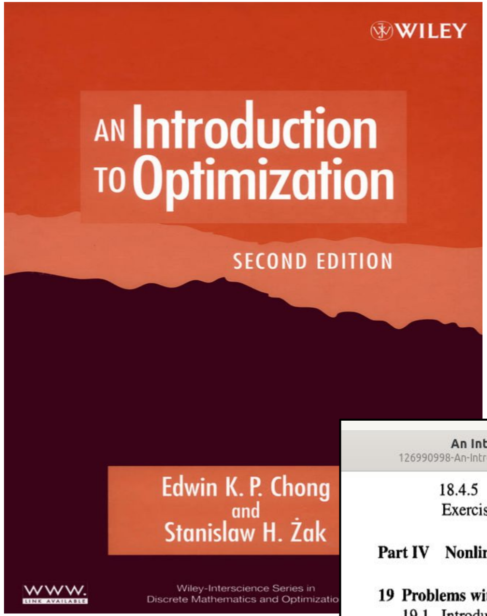
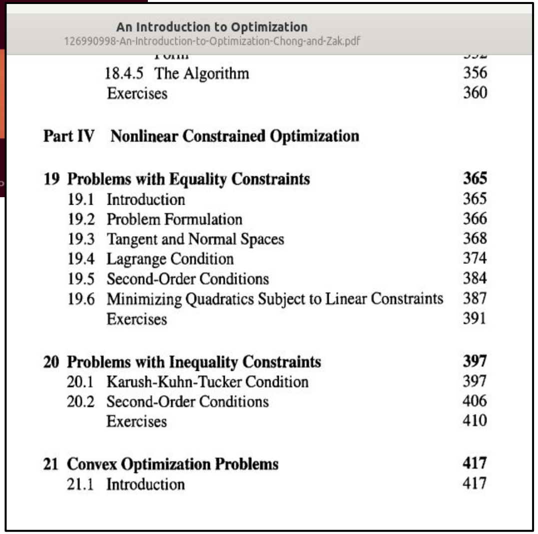
DSI207
Introduction to Optimization

Lecture 1-3

Overview

- (Linear programming/optimization)?
- Convex optimization problem
- Optimization algorithms (?)
- Denoising (image, audio? filter?)
- Learning representations (?)
- Sparse regression (linear?)
- Low-rank models
  (kernel methods? SVM? low-rank matrix approximations? Nyström method?)




An Introduction to Optimization
126990998-An-Introduction-to-Optimization-Chong-and-Zak.pdf

Administration

- Google Classroom
- Teaching Assistants?
- Email: roland.petrasch@gmail.com
- Chat: Google Hangouts, Line rpetrasch, Skype rpetrasch, Slack roland_petrasch
- Phone: 0972.320.460
- Grading:
  - A midterm exam (30%)
  - A final exam (30%)
  - Several (homework → classroom) assignments (40%), mostly at the end of the session, to be submitted until the end of the session, no late submissions possible

Overview

Books

E. Chong, S. Zak: An Introduction to Optimization. 2nd Ed, Wiley, 2001

S. Mallat: A Wavelet Tour of Signal Processing: The Sparse Way. Academic Press, 2008

M. J. Kochenderfer, T. A. Wheeler: Algorithms for Optimization.The MIT Press, 2019

I. Griva, S. G. Nash, A. Sofer: Linear and Nonlinear Optimization. 2008

Lecture 1: Logistic Regression

Mean

$$\mu = \frac{\sum_{i-1}^{n} x_i}{n}$$

Variance

$$s^2 = \frac{\sum_{i-1}^{n} (x_i - \bar{x})^2}{n-1} \text{ (for sample)}$$

$$\sigma^2 = \frac{\sum_{i-1}^{n} (x_i - \bar{x})^2}{N} \text{ (for population)}$$

Covariance

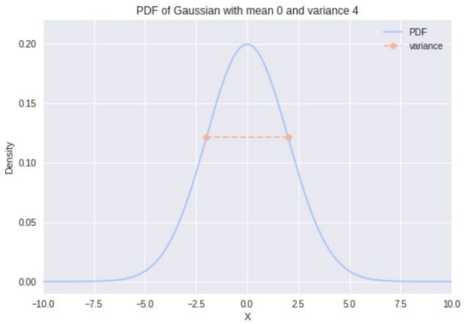
$$\mathrm{cov}(x,y) = \frac{\sum_{i-1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{n-1}$$

Standard Deviation

$$s = \sqrt{\frac{\sum_{i-1}^{n} (x_i - \bar{x})^2}{n-1}} \text{ (for sample)}$$

Gaussian (Normal) distribution (PDF): X~N(μ,σ)

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



CDF (Cumulative dist. function)

$$\int_{\Omega} f(x)dx = 1$$

Lecture 1: Logistic Regression

- Linear regression / multiple regression ← linear models
- Regression line: R^2 (values = large effect = correlation), p-value = statistically significance, predictions
- Least square method is used for $f(x)=a+bx$ (min. the sum of squares of the residuals)
- Odds and logs(odds) function is used, e.g. the odds are 1:4 or 0.25 that my team wins = of 5 games, my team will win 1 and lose 4 ⇒ odds are not probabilities
  - Odds (ratio): sth happening/sth not happening
  - Probability: sth happening/(sth not happening+sth happening)
- Formula $p / (1 - p)$ is often used

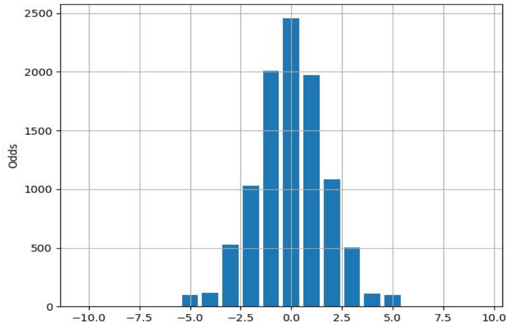
Lecture 1: Logistic Regression

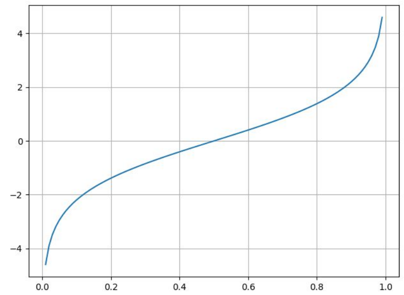
- Logs of the odds
  - Asymmetry makes it difficult to compare odds (for or against an outcome)
  - Using the log(odds) creates “symmetric values”: easier to interpret and use for statistics
- Odds can be calc. from the counts or the probability
  a. log(odds) = log(p/(1-p))
  b. log(p/(1-p)) is called the logit function
- Odds is a ratio (but not identical to the odds ratio)

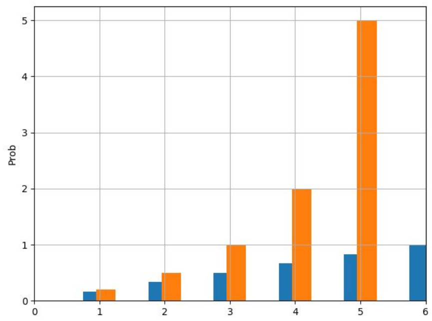
  See https://en.wikipedia.org/wiki/Odds_ratio

Lecture 1: Logistic Regression

- Exercise 1 “Odds” (Python):
  a) Calculate the odds and log(odds) for 5:3
  b) Plot the logit function (understand how it works)
  c) Create random numbers that add up to 100, calc. the log(odds)
     and put the values in a histogram. What do you get?
     Normal distribution / Gaussian bell curve?
  d) Calc the probability for throwing a die
     (with number of guesses as the independent variable)
     Why do you get an error “RuntimeWarning: divide by zero
     encountered” running the program?







Exercise 2 (Home Exercise)

Sport team wins n games and loses m games.

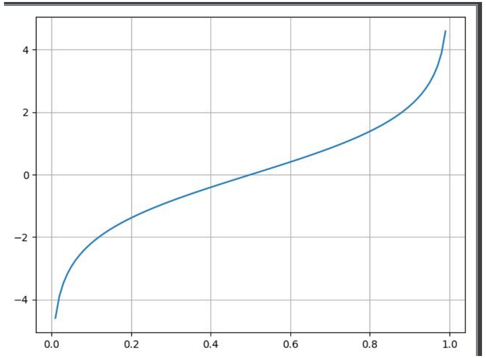
a) Calculate the odds and log(odds) for 6 matches (and all combinations)

1. Cout: n/m
2. Probability: p/(1-p)

b) Plot the logit function

Range 0..1..+infinity → -infinity…0...+infinity

c) Create the log(odds) for random numbers that add up to 20 (games) and put the values in a histogram. What do you get?

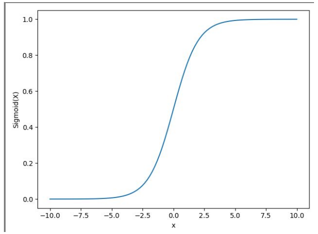


Lecture 1: Logistic Regression

- Logistic regression
  - Predicts whether something is true or false or has a rank/grade, e.g. obese / not obese, test passed / failed, addicted / not addicted …
  - Normally used for classification
- Binary dependent variable → two possible outcomes (for binary LR)
- Giving the value of the explanatory (independent) variable
  → prediction in form of a probability = value between 0 and 1
  (more than one independent variable might exist)
- LR provides prob. and classify new samples (cont./discrete measurements)
  → machine learning
- LR does not have residuals ⇒ least squares cannot be used
  and R^2 cannot be calculated → maximum likelihood

Lecture 1: Logistic Regression

- Linear regression: x and y values can be any (real) number → calculations
- Logistic regression: y "values" are for classifications
  = probability (0..1) → calculations?
  - Solution: y-axis is transformed from probability to the log(odds)
  - Result: y values can range from -infinity to +infinity
- Logit function: log (p / (1-p)) or logit(p)
- Inverse of logit function is the sigmoid function:
  For probability p: sigmoid(logit(p)) = p
  (sigmoid function maps arbitrary real values back to the range [0, 1]): $f(x) = 1 / 1 + e^{-(x)}$
- Exercise 3 "Sigmoid" (Python):
  Plot the sigmoid function



Lecture 1: Logistic Regression

Types of Logistic Regression:

- Binary Logistic Regression: Target variable
  has two possible outcomes, e.g.
  spam / not spam, cancer / no cancer
- Multinomial Logistic Regression: Target variable
  has > two nominal categories, e.g. cat, dog, sheep
- Ordinal Logistic Regression: target variable
  has > two ordinal categories, e.g. product rating low, medium, high

Lecture 2: Logistic Regression

- Exercise 4 “Logistic Regression” (Python):
  a. Train and test with the university admission data
  b. Try out different test data percentages and show the confusion matrix
  c. Predict for new candidates

- Exercise 5 “Logistic Regression (Home Exercise)” (Python):
  a. Train and test your own data (with different train/test data sets, e.g. 90/10, 80/20, 70/30)
  b. Show the confusion matrix
  c. Predict for new value for independent variable

  Why the result vector consists only of 0s and 1s? Where are the probabilites?

Take a look at

https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

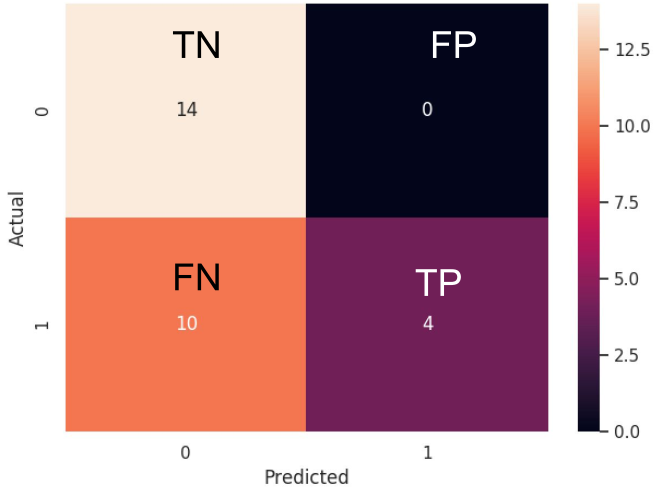
Lecture 2: Logistic Regression

- Inverse of the logit function: sigmoid function = s-shape function for logistic regression
- Formula for the sigmoid function:
  $\sigma(x) = 1/(1 + \exp(-x))$ with $\exp(x) = e^x$
  $\Rightarrow \sigma(x) = e^x/(e^x + 1)$
- Logistic function $f(x)=L/(1+e^{-k(x-x0)})$ with
  x_0 = x value of sigmoid's midpoint, L = curve's maximum value, k = logistic growth rate (steepness)
- For a probability p: sigmoid(logit(p)) = p
- Sigmoid function maps arbitrary real values back to the range [0, 1]
  The larger the value, the closer to 1 (for L=1)

Lecture 2: Logistic Regression

- LR equation: $y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})$
  $b_0 + b_1*x$ with the coefficient vector b (or β) → linear model
- LR is a machine learning technique:
  For big sets split 75% - 80% of the data into training set while 20% - 25% into test set (or use K-fold cross-validation)
- Algorithms, e.g. maximize log likelihood with gradient descent

Lecture 2: Logistic Regression



- Confusion matrix: table used to describe the performance of a classification model, e.g. In logistic regression
- Set of data for which the true values are known (training / actual data) and set test data (prediction) are used.
- Both, actual and predicted classification are compared, e.g. for 10 data points the predicted classifier is 0, but the actual value is 1 (false negative = FN), Likewise there are 4 true positive (TP) (see the example diagram)
- TN, FN, FP, TP, actual yes/no, and predicted yes/no are used for metrics
    - Accuracy = (TP+TN)/total, example: (14+4) / 28 = 0.64 (64%)
    - True Positive Rate = TP/actual yes (or 1 or true), example: 4 / 14 = 0.29 (29%)
    - True Negative Rate = TN/actual no, examples: 14 / 14 = 1 (100%)

Lecture 3: Logistic Regression

- LR equation: $y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})$ or (more general) $1 / (1 + e^{-z})$ with z = prediction function, e.g. mx+b or $b_0 + b_1*x$
  b0 + b1*x with the coefficient vector b (or β) → linear model
- Predictors $x_0 \ldots x_n$ (with $x_0=1$) and one binary (Bernoulli) response variable Y (in the case of binary log. regression)
- $S(z) = 1 / (1 + e^{-z})$ calculates the probability
- Decision boundary

  $p \geq 0.5$, class=1

  $p < 0.5$, class=0
- Predictions

  Example: $P(\text{class} = 1) = 1 / (1 + e^{-z}) = 0.4 \Rightarrow$ Fail

Lecture 4: Logistic Regression

Cost function

- The predicted values should be “close” to the actual values
  $\hat{y}^{(i)} \approx y^{(i)}$ with $^{(i)}$ training data set
- Loss (error) or cost function $L(\hat{y}, y)$
  - If $y = 1$: $-\log(\hat{y})$
  - If $y = -0$: $-\log(1-\hat{y})$
- cost function $J(a,b)$ = average of the loss function
  - $J(a,b) = 1/m \, Sum(L(\hat{y}^{(i)}, y^{(i)})$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

Lecture 4: Logistic Regression

Exercise 6:

a) Train a model with the mice-obesity data and print the results incl the confusion matrix. Show also the intercept/coefficient

b) Predict for new mice

c) Plot the raw data, test and training data and the new prediction in one scatter plot

d) Show the cost (calc. the loss first)

e) Calc. prediction using the sigmoid and the intercept/coeff. and compare them with the predictions made by the library function

Lecture 4: Logistic Regression

Exercise 7 (Homework exercise):

a) Train a model with your own data and print the results incl the confusion matrix. Show also the intercept/coefficient

b) Predict for data (independent variable)

c) Plot the raw data, test and training data and the new prediction in one scatter plot

d) Show the cost (calc. the loss first)

e) Calc. prediction using the sigmoid and the intercept/coeff. and compare them with the predictions made by the library function

Write your own (!) Python program. Do not (!) just copy the code.