Datascience 4.0

Introduction to Machine learning in Python

contact info on next slide

September 16, 2018

Before we start.. Who we are?

- Last year Nemanja Micovic
  `nemanja_micovic@matf.bg.ac.rs`
- This year me, Milan Cugurovic
  `milan_cugurovic@matf.bg.ac.rs`
- There are a lot of as at:

Faculty of Mathematics Machine Learning Group

- Based at Faculty of Mathematics
- Includes researchers, students and particioners from various research institutions and leading IT companies
- Led by proffessor Mladen Nikolic [1]
- Lectures/talks and practical sessions are held on every 2 weeks
- MATF Machine Learning Group: `machinelearning.math.rs`
    - Feel free to visit our sessions and join in!

[1] web: poincare.matf.bg.ac.rs/∼ *nikolic*, *email* : *nikolic@matf.bg.ac.rs*

Table of contents

Table of contents

About Machine learning

- Field of Artificial Intelligence
- Very active research field today
- Has acomplished amazing results
- Built on multiple mathematical disciplines

About Machine learning

Famous definition by Tom M. Mitchell

- A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance measure P** if its performance at tasks in T, as measured by P, improves with experience E

What is the goal of Machine learning?

- To create models able to generalize
- To give a theoretical base of generalization
- To solve a whole class of problems difficult for deterministic algorithms

Some result of Machine learning

- 1992 - TD-Gammon, computer program develoepd by Gerald Tesauro able to play backgammon
- 2011 - IBM's Watson wins in quiz *Jeopardy!*
- 2012 - Google X creates system able to recognize cats on video recordings
- 2015 - Classification error for images reduced to 3.6% (5-10% is the error made by humans)
- 2016 - Google creates AlphaGo, agent able to play Go who beats the world champion 4:1
- 2017 - AlphaGo plays against its 2016 version and wins 100/100 games
- October 2017 - AlphaGo Zero learnt to play the game of Go simply by playing games against itself, starting from scratch (40 days)

Applications of Machine learning

- Autonomous driving
- Bioinformatics
- Social networks
- Algorithm porfolio
- Playing video games
- Image classification
- Recognizing handwritting
- Natural language processing
- Generating optimization algorithms [Andrychowicz et al., 2016]
- Generating images

- Computer vision
- Detecting credit card frauds
- Data mining
- Medical assistance and assesment
- Marketing
- Targeted marketing
- Controlling robots
- Economy
- Speach recognition
- Recommendation systems

But why is it so successful and popular today?

- There is serious amount of mathematics behind [Murphy, 2012, Bishop, 2006, Hastie et al., 2001, Shalev-Shwartz and Ben-David, 2014, Vapnik, 1995]

$$
\begin{gathered}
P(\sup_{f\in\mathcal{F}}(R(f)-E(f))>\varepsilon) \leq \\
2P(\sup_{f\in\mathcal{F}}(E'(f)-E(f))>\varepsilon/2) = \\
2P\left(\sup_{l\in\mathcal{L}_{z_1,\dots z_N,z'_1,\dots,z'_N}}\left(\frac{1}{N}\sum_{i=N+1}^{2N} l_i-\frac{1}{N}\sum_{i=1}^{N} l_i\right)>\varepsilon/2\right) \leq \\
2\sum_{l\in\mathcal{L}_{z_1,\dots z_N,z'_1,\dots,z'_N}} P\left(\frac{1}{N}\sum_{i=N+1}^{2N} l_i-\frac{1}{N}\sum_{i=1}^{N} l_i>\varepsilon/2\right) = \\
2\sum_{l\in\mathcal{L}_{z_1,\dots z_N,z'_1,\dots,z'_N}} P\left(\frac{1}{N}\sum_{i=N+1}^{2N} l_i-R(f)+R(f)-\frac{1}{N}\sum_{i=1}^{N} l_i>\varepsilon/2\right) \leq \\
2\sum_{l\in\mathcal{L}_{z_1,\dots z_N,z'_1,\dots,z'_N}}\left(P\left(\frac{1}{N}\sum_{i=N+1}^{2N} l_i-R(f)>\varepsilon/2\right)+P\left(R(f)-\frac{1}{N}\sum_{i=1}^{N} l_i>\varepsilon/2\right)\right) \leq \\
2\sum_{l\in\mathcal{L}_{z_1,\dots z_N,z'_1,\dots,z'_N}}(\exp(-N\varepsilon^2/2)+\exp(-N\varepsilon^2/2)) = \\
4\sum_{l\in\mathcal{L}_{z_1,\dots z_N,z'_1,\dots,z'_N}}\exp(-N\varepsilon^2/2) =
\end{gathered}
$$

Figure: *M. Nikolic, Machine learning*

But why is it so successful and popular today?

- Don't be afraid, today we talk ML introduction without so much math stuff
- Big amounts of data
- We also have graphical cards with thousands of processors
  - They allow us to get extremely high levels of parallelization
- Industry and academia complement each other
  - Our meeting here today is the evidence of that :)

Types of machine learning

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Table of contents

Supervised learning

- Our main focus today
- We are given attributes $x_1, x_2, ...x_n$
- Using them, we need to predict target variable y
- We want to create a model that will approximate $f(x_1, x_2, ..., x_n) = y$
- So we need to create a function $f' \approx f$

Regression

- Target variable y is continuous
- Trying to predict temperature (y) using pressure (x)

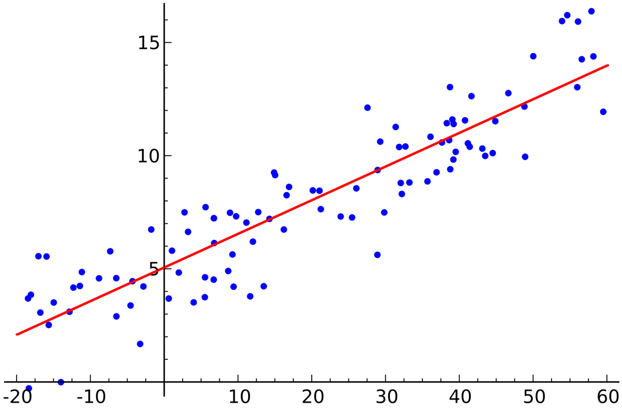


Figure: Linear regression (wikipedia)

Classification

- Target variable y is discreete
- Trying to predict gender (y) using weight (x_1) and height (x_2)

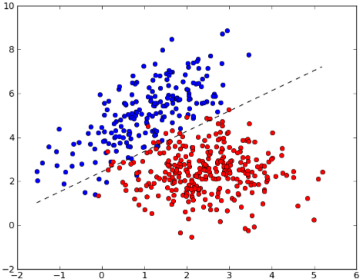


Figure: Classification example 1

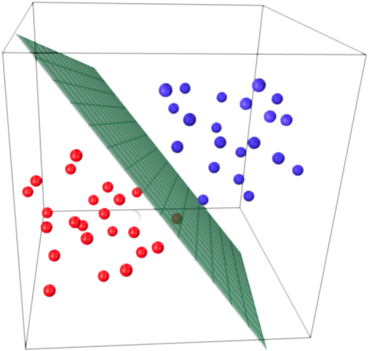


Figure: Classification example 2 (Sachin Joglekar's blog)

Linear regression

- We construct the model in the following form:

$$f_w(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

$$f_w(x) = w_0 + \sum_{i=1}^{n} w_ix_i$$

- We calculate model accuracy using the following formula[2]:

$$Loss(w) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - f_w(X_i))^2$$

[2] Which is usually called *Mean squared error*

Linear regression - minimization problem

- We have lots of different models
- Every tuple $(w_0, w_1, \ldots, w_n)$ defines a different model
- What is the *best*[3] one?
- Model that makes the smallest mistake on the data we have is *generally* great for us!
- But how do we find such model?

[3] Using term *best* is tricky here, but let's stick with it for now.

Linear regression - minimization problem

- Actually, that's not so difficult to do, we can derive the following equation with a bit of algebra
- Let's assume for simplicity that we have only one attribute x
- x_i is the i-th dataset element
- y_i is the target value for i-th dataset element

$$w = (X^\top X)^{-1} X^\top Y$$

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \\ 1 & x_N \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix}$$

Linear regression - minimization problem

So what's the problem then?

- Matrix multiplication - lowest complexity so far $O(n^{2.373})$ [Gall, 2014]
- Matrix inverse - lowest complexity so far $O(n^{2.373})$
- Storing big[4] matrix $n \times m$ in memory: $O(nm)$

[4] non sparse, we can store sparse matrices more efficiently

Linear regression - gradient descent

- How does our error function generally look?
- Which point has the smaller error, A or B?

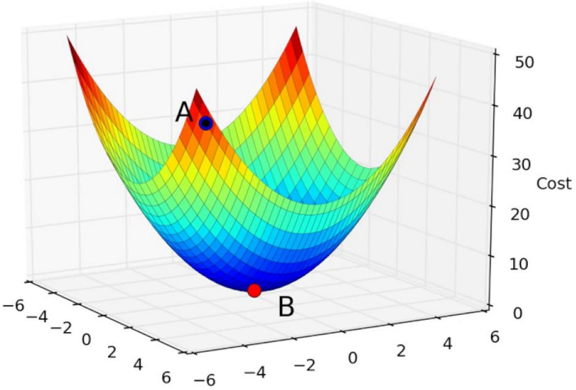


Figure: An example of the error function

Linear regression - gradient descent

Can we somehow *descend* into the function minimum?

- Yes!

But how?

- Calculate gradient of the error function with respect to w
- This vector *points* into the direction of the fastest function growth

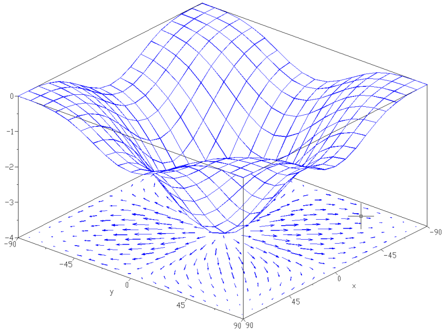


Figure: Blue arrows represent function gradients

Linear regression - gradient descent

Gradient descent algorithm:

- Repeat until convergence
  - $w_j := w_j - \mu \frac{\partial}{\partial w_j} Loss(w), \ j \in \{1, 2, ..., n\}$

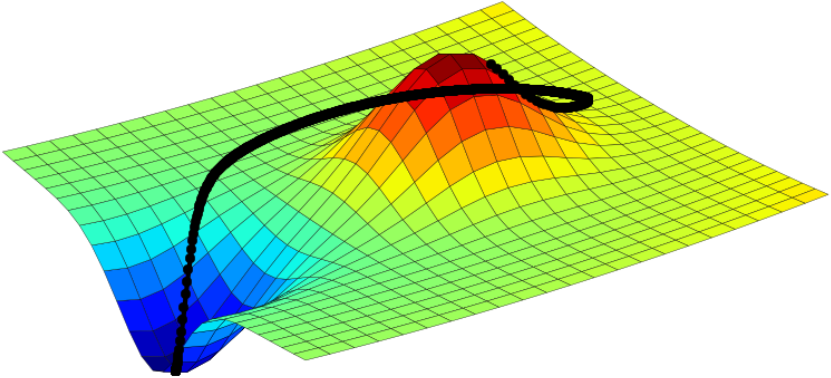


Figure: An example of the steps made by the gradient descent algorithm (github.com/joshdk)

Linear regression - (R)MSE

- Mean

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

- Mean squared error (MSE)

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - f_w(x_i))^2$$

- Root mean squared error (RMSE)

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f_w(x_i))^2}$$

Linear regression - R^2

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(f_w(x_i) - y_i)^2}{\sum_{i=1}^{N}(\bar{y} - y_i)^2}$$

- It could be interpreted as change in the target value y which our model is failed to explain by changing the variable x.
- Errors on which we define variance and R^2:

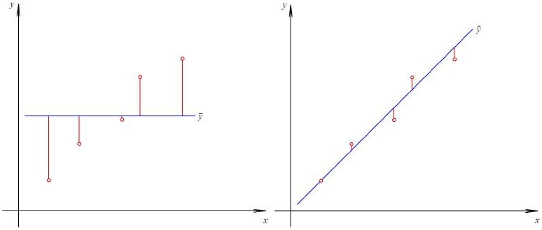


Figure: M. Nikolic, Machine learning

Linear regression - R^2

- Coefficient of determination, mostly called R^2
- It is the **proportion of the variation** in the **dependent** variable that is predictable from the **independent** variable(s)
- We can say that it determines how much of variability has our model **managed to explain**
- What is the minimum of R^2?
- What is the maximum of R^2?

Linear regression - coding time

- Let's code linear regression in `scikit-learn`

Linear regression - overfitting

- Analyze the following images[5]

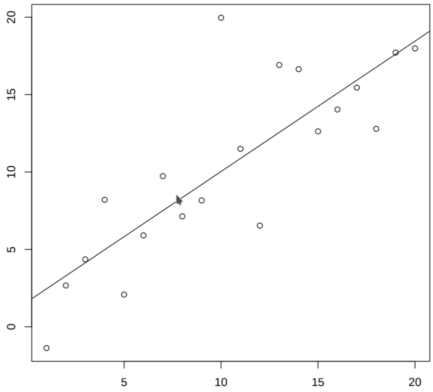


Figure: Linear regression 1

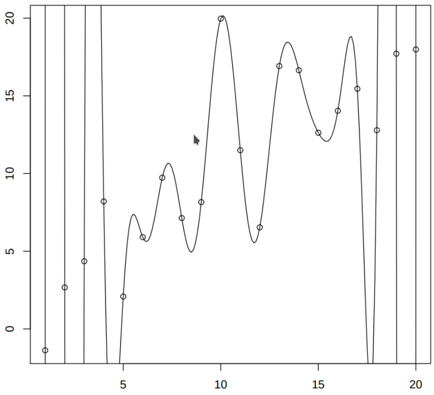


Figure: Linear regression 2

[5]Images taken from book *P. Janičić, M. Nikolić, Artificial Intelligence*

Linear regression - underfitting and overfitting

- Given 3 models, which one do you prefer in respect to black points (dataset samples)?

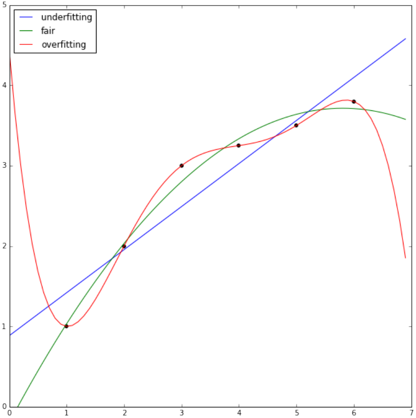


Figure: Examples of underfitting and overfitting

Linear regression - underfitting and overfitting

Underfitting

- A situation in which our model is not flexible enough in order to capture the essence of a phenomena

Overfitting

- A situation in which our model is too flexible and it fits too well towards the training data we feed it

Linear regression - underfitting and overfitting

- If $MSE_{train} = 0$, are we *always* happy?

Linear regression - underfitting and overfitting

- If $MSE_{train} = 0$, are we *always* happy?
  - Not really, we probably have overfitted quite a bit

Linear regression - underfitting and overfitting

- If $MSE_{train} = 0$, are we *always* happy?
  - Not really, we probably have overfitted quite a bit
- If $MSE_{test} = 0$, are we *mostly* happy?

Linear regression - underfitting and overfitting

- If $MSE_{train} = 0$, are we *always* happy?
  - Not really, we probably have overfitted quite a bit
- If $MSE_{test} = 0$, are we *mostly* happy?
  - Indeed we are, if we have a decent representable test set

Linear regression - underfitting and overfitting

- If $MSE_{train} = 0$, are we *always* happy?
  - Not really, we probably have overfitted quite a bit
- If $MSE_{test} = 0$, are we *mostly* happy?
  - Indeed we are, if we have a decent representable test set
- If we have underfitting, which one will be bigger, MSE_{train} or MSE_{test}?

Linear regression - underfitting and overfitting

- If $MSE_{train} = 0$, are we *always* happy?
  - Not really, we probably have overfitted quite a bit
- If $MSE_{test} = 0$, are we *mostly* happy?
  - Indeed we are, if we have a decent representable test set
- If we have underfitting, which one will be bigger, MSE_{train} or MSE_{test}?
  - They will both be rather large

Linear regression - underfitting and overfitting

- If $MSE_{train} = 0$, are we *always* happy?
  - Not really, we probably have overfitted quite a bit
- If $MSE_{test} = 0$, are we *mostly* happy?
  - Indeed we are, if we have a decent representable test set
- If we have underfitting, which one will be bigger, MSE_{train} or MSE_{test}?
  - They will both be rather large
- If we have overfitting, which one will be bigger, MSE_{train} or MSE_{test}?

Linear regression - underfitting and overfitting

- If $MSE_{train} = 0$, are we *always* happy?
  - Not really, we probably have overfitted quite a bit
- If $MSE_{test} = 0$, are we *mostly* happy?
  - Indeed we are, if we have a decent representable test set
- If we have underfitting, which one will be bigger, MSE_{train} or MSE_{test}?
  - They will both be rather large
- If we have overfitting, which one will be bigger, MSE_{train} or MSE_{test}?
  - $MSE_{test} > MSE_{train}$

Linear regression - underfitting and overfitting

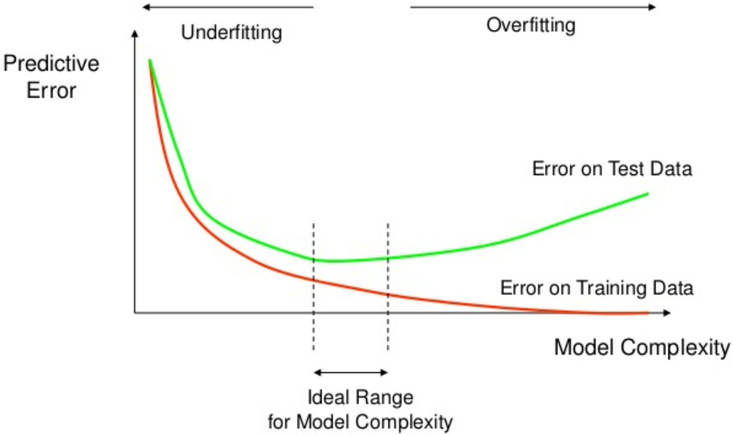


Figure: Graph showing the difference between underfitting and overfitting

Linear regression - How to battle underfitting?

- Take a more flexible model
- Instead of $f_w(x) = w_0 + w_1x_1 + w_2x_2$ take $g_w(x) = w_0 + w_1w_2x_1 + w_1^2x_1 + w_2^2x_2$
- Usually easier to solve then overfitting
- There is a wide variety of flexible models, and we can always complicate things[6]

[6] As in life and mathematics...

Linear regression - How to battle overfitting?

Regularization

- It allows us to control model complexity
- Term λ controls the *intensity* of regularization
- There are multiple options to pick from for function Ω
- Interesting tutorial: https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/
- We modify the minimization problem into:

$$\min_{w} \frac{1}{N} \sum_{i=1}^{N} (y_i - f_w(x_i))^2 + \lambda \Omega(w)$$

Linear regression - Ridge regularization

- Very commong regularization function
- It forces optimization algorithms not to increase model coefficients too much
- If coefficients get increased, then the sum of their squares rise a lot

$$\Omega(w) = \|w\|_2^2 = \sum_{i=1}^{n} w_i^2$$

- Using ridge, we obtain the following minimization problem

$$\min_w \frac{1}{N} \sum_{i=1}^{N} (y_i - f_w(x_i))^2 + \lambda \sum_{i=1}^{n} w_i^2$$

Linear regression - coding time

- Let's code ridge regression in `scikit-learn` [7]

[7] Challenge: code Lasso Regression, which performs L1 regularization, i.e. adds penalty equivalent to absolute value of the magnitude of coefficients

K-Nearest neighbours (kNN)

kNN

- Simple yet sometimes powerful classification algorithm
- `K` inside name comes from parameter k
- k determines the number of neighbours we check when classifying an instance

kNN

- How much is k?

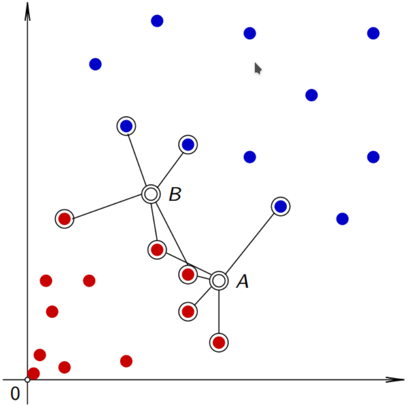


Figure: kNN example (image taken from [Janičić and Nikolić, 2017])

kNN

- How much is k?
  - 5

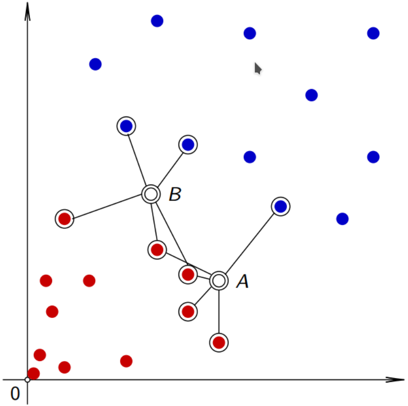


Figure: kNN example (image taken from [Janičić and Nikolić, 2017])

kNN

- How much is k?
  - 5
- What is the class of A?

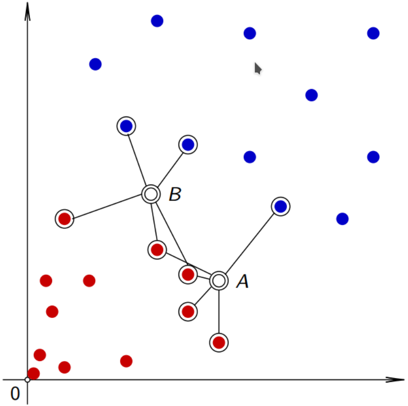


Figure: kNN example (image taken from [Janičić and Nikolić, 2017])

kNN

- How much is k?
  - 5
- What is the class of A?
  - Red

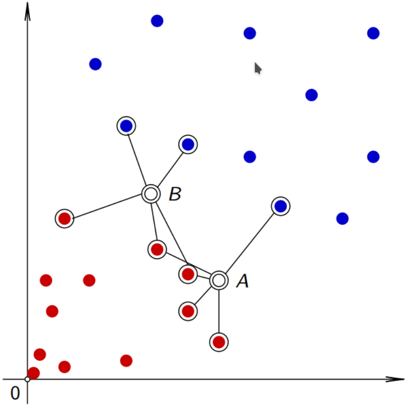


Figure: kNN example (image taken from [Janičić and Nikolić, 2017])

kNN

- How much is k?
  - 5
- What is the class of A?
  - Red
- What is the class of B?

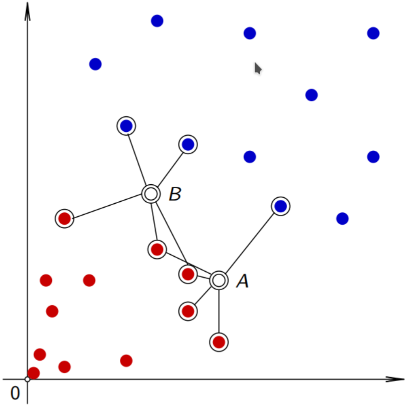


Figure: kNN example (image taken from [Janičić and Nikolić, 2017])

kNN

- How much is k?
  - 5
- What is the class of A?
  - Red
- What is the class of B?
  - Red

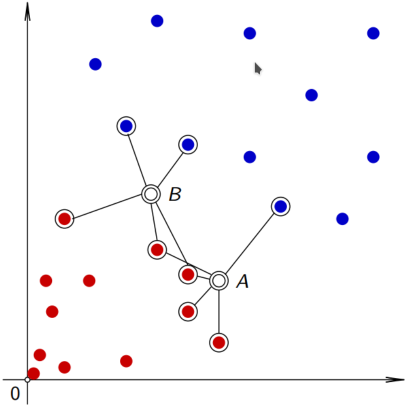


Figure: kNN example (image taken from [Janičić and Nikolić, 2017])

kNN

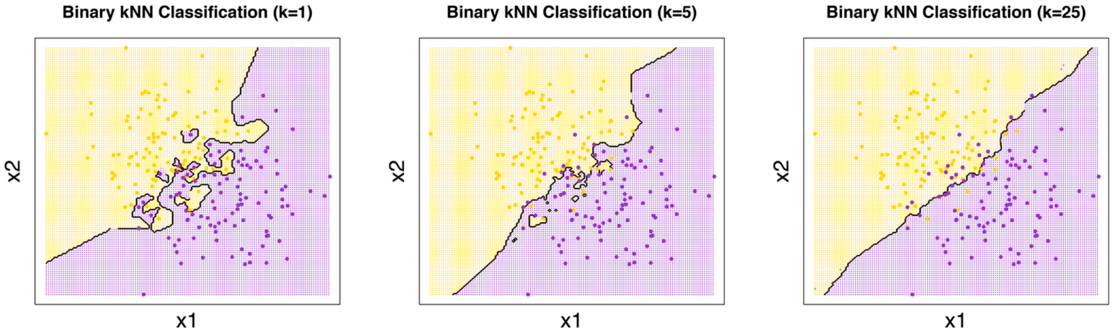


Figure: kNN example (image taken from Burton DeWilde's blog)

kNN

- In linear regression, we represented our model with coefficients w

kNN

- In linear regression, we represented our model with coefficients w
- What is the model in the kNN classifier?

kNN

- In linear regression, we represented our model with coefficients w
- What is the model in the kNN classifier?
  - There is no such thing, we only need to know k

kNN

- In linear regression, we represented our model with coefficients w
- What is the model in the kNN classifier?
  - There is no such thing, we only need to know k
- How do we train the model?

kNN

- In linear regression, we represented our model with coefficients w
- What is the model in the kNN classifier?
  - There is no such thing, we only need to know k
- How do we train the model?
  - We don't, but every time we must calculate neighbours for a new instance

kNN - distances

- There are multiple functions we can use to calculate distances
- Assume we are given points $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$
- Minkowski

$$\left(\sum_{i=1}^{n}(|x_i - y_i|)^q\right)^{\frac{1}{q}}$$

- Manhattan (q = 1) [8]

$$\sum_{i=1}^{n}|x_i - y_i|$$

- Euclidean distance (q = 2)

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

[8] Taxi metrics :)

kNN - Curse of dimensionality

- Our intuition is bad for high dimensionals spaces
  - $lim_{n \to \infty} \frac{V_{0.99}^n}{V_1^n} = lim_{n \to \infty} \frac{0.99^n}{1^n} = 0$
  - In hd spaces, all dots are far away from center of ball
  - if a point is classified on the basis of close neighbors, close neighbors are expected
- When dimensionality increases, the volume of space increases really fast
- This can make our dataset very sparse (even if it isn't sparse, then our algorithm is very computationally expensive)
- Essentially, we number of dataset instances required increases exponentially with the dimensionality
- This is very bad for kNN

Classification - important metrics

- TP (true positive): those that are positive and our model was correct
- TN (true negative): those that are negative and our model was correct
- FP (false positive): those that are negative and our model was wrong
- FN (false negative): those that are negative and our model was wrong
- *Accuracy*

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

  - Example: only two classes, in one class, 99 instances, in the other only 1 instance. We can achieve impressive score 0.99 just by saying that all instances are from the first class.
    (Credit card detection, tumor detection etc.)

Classification - important metrics

- *Precision*

$$Precision = \frac{TP}{TP + FP}$$

  - The proportion of positive instances in all instances that were declared positive
- *Recall* score

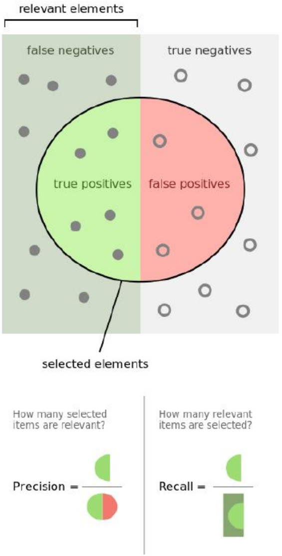
$$Recall = \frac{TP}{TP + FN}$$

  - The proportion of found positive instances in all positive instances
- F_1 score

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

  - If all instances is positive - Recall is max
    If no positive instances - Precision is max

Classifcation - Precision/Recall ilustration



kNN - coding time

- Let's code kNN Classifier in `scikit-learn` [9]

[9] Any idea for kNN Regression?

Logistic regression

Logistic regression

- Classification algorithm
- By design, similar to linear regression
- We want to approximate $p(y|x)$

$$f_w(x) = w_0 + \sum_{i=1}^{n} w_i x_i$$

- $f_w(x)$ is not in interval $[0, 1]$

Logistic regression - sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

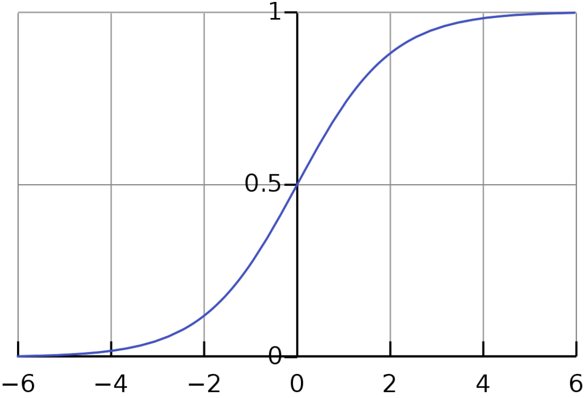


Figure: Graph of the sigmoid function

Logistic regression - loss function

- We define the loss function as following (check [Bishop, 2006, Murphy, 2012] for details)

$$Loss(w) = -\sum_{i=1}^{N} \log p_w(y_i|x_i) = -\sum_{i=1}^{N} [y_i \log f_w(x_i) + (1 - y_i) \log(1 - f_w(x_i))]$$

- Function defined as: $L(u, v) = -u \log v - (1 - u) \log(1 - v)$ is a loss function![10] Why?

[10] Her name is binary crossentropy!

Logistic regression - minimization problem

- We end up with the following minimization problem

$$\min_{w} - \sum_{i=1}^{N} [y_i \log f_w(x_i) + (1 - y_i) \log(1 - f_w(x_i))]$$

- Which is a convex function with a global minimum

Logistic regression - examples

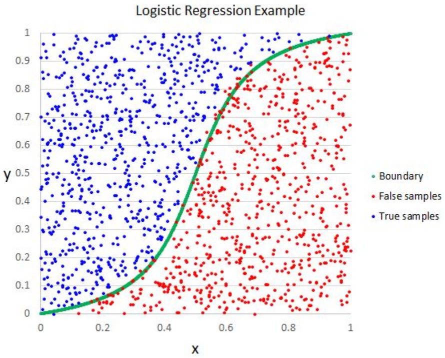


Figure: Example taken from www.helloacm.com

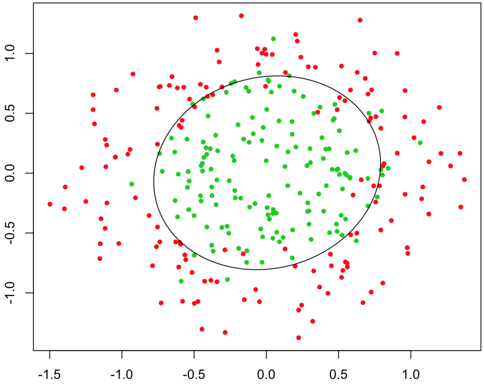


Figure: Example taken from statsblogs.com

Linear regression - coding time

- Let's code logistic regression in `scikit-learn`

Table of contents

Neural Networks - First Look

- Most popular and most appropriate machine learning methods
- As I said before:
  - Medical diagnostic
  - Image recognition and Object detection
  - Autonomous driving
  - NLP (Natural Language Processing)
  - Go, FlappyBird ...
- Five main types:
  - Feed Forward neural network (basic)
  - Convolutional neural network (images)
  - Recurent neural network (memory)
  - Recursive neural network (tree)
  - Graph neural network (graph)
- The reason for their re-popularity (computational power!)

Before we dive in...

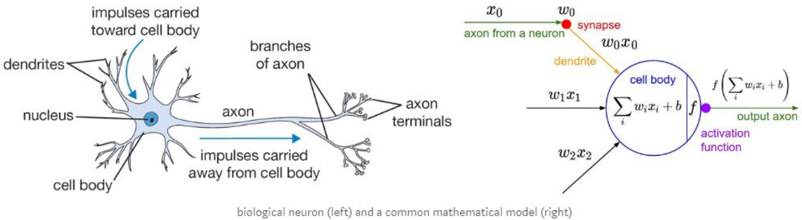
- Biological motivation and connection
  - Neuron - the basic computational unit of the brain
  - Around 86 billion neurons can be found in the human nervous system
  - They are connected with approximately $10^{14} - 10^{15}$ synapses
  - Human 'training process' - a couple of years :) (AlphaGo 40 days right now)
- Mathematical strength
  - Universal approximation theorem
  - Let $\phi(\cdot)$ be a nonconstant, bounded, and monotonically-increasing continuous function. Let I_m denote he m-dimensional unit hypercube $[0,1]^m$.The space of continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\epsilon > 0$ and any function $f \in C(I_m)$, there exists an integer N, real constants v_i, b_i in and real vectors $w_i \in {}^m$, where $i = 1, 2, ..., N$, such that we may define:

    $$F(x) = \sum_{i=1}^{N} v_i \phi(w_i^T x + b_i)$$

    as an approximate realization of the function f, where f is independent of ϕ; That is $|F(x) - f(x)| < \epsilon$, for all $x \in I_m$.
  - **I am apologize for previous theorem**, in other words...

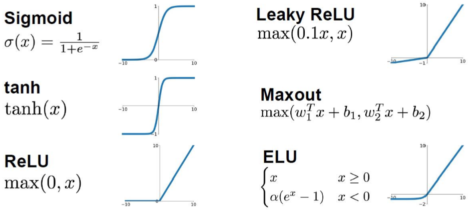
Main parts of Feed Forward Neural Network

- Neuron:



biological neuron (left) and a common mathematical model (right)

- You see f, f is an activation function:



Layers

- Neurons (with its activation functions) are stored in layers[11]
- Input layer - No computation is done here within this layer, they just pass the information to the next layer
- Hidden layer**s** - they perform computations and then transfer the weights (signals or information) from the input layer to the following layer (another hidden layer or to the output layer)
- Output layer - Here we finally use an activation function that maps to the desired output format (e.g. softmax for classification)

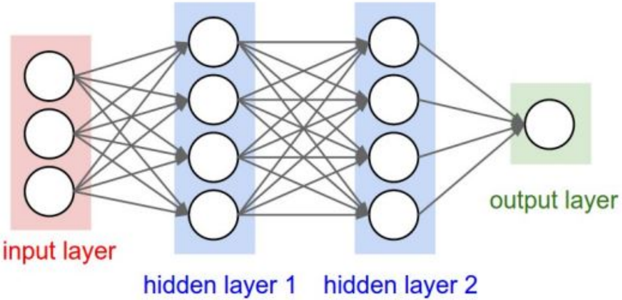
$$softmax((x_1, \dots, x_C)) = \left(\frac{e^{x_1}}{\sum_{i=1}^{C} e^{x_i}}, \dots, \frac{e^{x_C}}{\sum_{i=1}^{C} e^{x_i}} \right)^{12}$$

[11] A block of nodes

[12] MNIST interpretation, choose one out of ten digits

Fully image

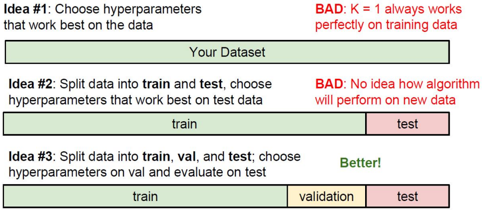
- Feed Forward Neural Network



- Idea of Dropuot

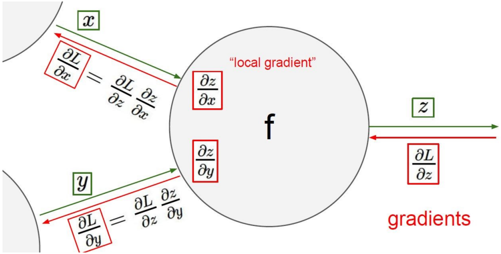
Should I learn something?

- Give me the data!



[13]

- Ok, when I have the data, I learn via backprop.



[13] Check on the internet the term 'cross-validation'

Feed Forward Neural Network - coding time

- Let's code MNIST[14] Feed Forward Neural Network in `keras`[15] [16]

[14]The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits

[15]https://keras.io/

[16]Search for the Colvolutional Neural Network on the internet! Can you solve MNIST problem using CNN?

Questions?

Thank you for your attention!

Bibliography I

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. (2016).
Learning to learn by gradient descent by gradient descent.
ArXiv e-prints.

Bishop, C. M. (2006).
Pattern Recognition and Machine Learning (Information Science and Statistics).
Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Gall, F. L. (2014).
Powers of tensors and fast matrix multiplication.
CoRR, abs/1401.7714.

Hastie, T., Tibshirani, R., and Friedman, J. (2001).
The Elements of Statistical Learning.
Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Bibliography II

Janičić, P. and Nikolić, M. (2017).
Artificial Intelligence.
Matematički fakultet.

Murphy, K. P. (2012).
Machine Learning: A Probabilistic Perspective.
The MIT Press.

Shalev-Shwartz, S. and Ben-David, S. (2014).
Understanding Machine Learning: From Theory to Algorithms.
Cambridge University Press, New York, NY, USA.

Vapnik, V. N. (1995).
The Nature of Statistical Learning Theory.
Springer-Verlag New York, Inc., New York, NY, USA.