Machine Learning Basics

Mladen Nikolić

Faculty of Mathematics
University of Belgrade

Everseen

Overview

Overview

What is machine learning?

- A discipline which deals with inducing algorithms from the data, instead of programming them explicitly

Critical notions

- Instead of algorithms, we talk about *models*
- Models express relations between different *variables* relevant for the task being solved
- Models are obtained from available *data* by some *learning algorithm*
- Models should *generalize* well, meaning that they should perform well on *unseen data*

Toy example (1)

- Automated detection of computer related articles
- How to detect them?

Toy example (1)

- Automated detection of computer related articles
- How to detect them?
- Based on terminology
- For instance "computer" and "file"
- Each article can be represented by frequencies of these words
- Points in 2D space!
- How to express discrimination rule between computer related ones and the others?

Toy example (2)

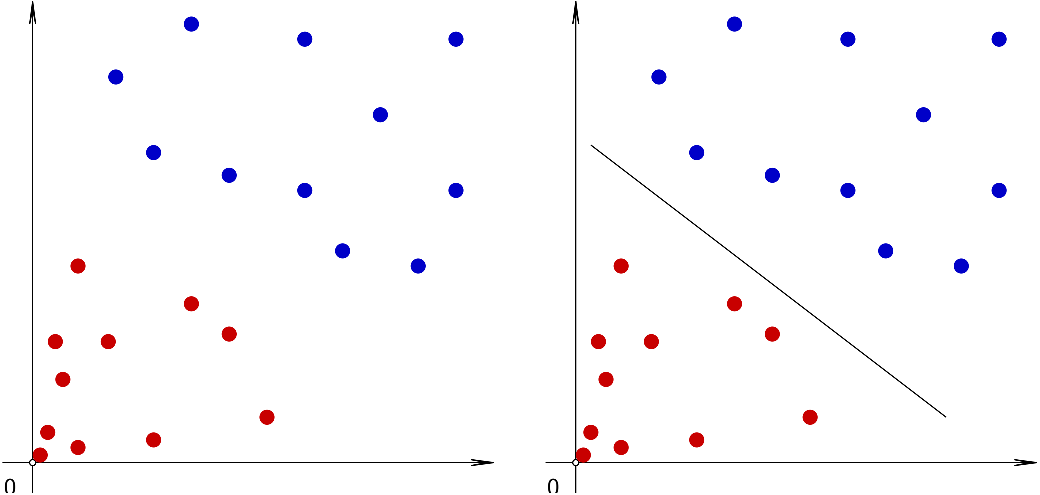


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Why is machine learning important?

- Numerous applications which move boundaries of technology and imagination
- Superhuman performance in some tasks (with a grain of salt)
- Deep theory of inductive inference
- Great interplay of theory and practice, of academia and industry
- Most applicable branch of artificial intelligence
- Probably most popular and fastest growing branch of computer science

Machine learning and artificial intelligence

- $\text{DL} \subsetneq \text{ML} \subsetneq \text{AI}$
- Logic based AI vs. probability based AI
- Logic formalizes deductive inference
- ML formalizes inductive inference
- Logic based approaches to AI expect formal definitions of inference rules and are applied to problems for which we are able to provide full formal descriptions
- ML aims at problems for which we can't provide formal descriptions (e.g., face recognition), but can provide examples instead
- Logic based approaches do not operate with uncertainty, while ML approach does (which is often great, but sometimes not)

Short history (1)

- 1943 – McCulloch and Pitts formulate threshold logic, first artificial neuron
- 1950 – Alan Turing contemplates about learning machines
- 1950 – Marvin Minsky builds a first neural network
- 1952 – Arthur Samuel makes first checkers playing programme
- 1957 – Frank Rosenblatt makes perceptron (in hardware)
- 1963 – Vapnik and Chervonenkis propose first support vector machine

Short history (2)

- 1967 – Cover and Hart propose k nearest neighbours algorithm with application to travelling salesman problem
- 1969 – Marvin Minsky and Seymor Papert criticize perceptron, leading to first neural network winter
- 1975 – Werbos formulates backpropagation algorithm
- 1981 – Dejong introduces explanation based learning for extraction of rules from data
- 1986 – Rumelhart, Hinton, and Williams reintroduce backpropagation

Short history (3)

- 1989 – Watkins proposes Q-learning
- 1989 – First selfdriving car
- 1992 – Boser, Guyon, and Vapnik propose to use kernles with SVM, starting the domination of SVM during nineties
- 1992 – Tesauro makes TD-Gammon, backgammon system which beats human champions
- 1995 – Tin Kam Ho proposes random decision forests
- 1997 – Hochreiter and Schmidhuber propose LSTM

Short history (4)

- 2006 – Hinton rebrands neural networks as *deep learning*
- 2011 – IBM's system Watson outcompetes human champions in Jeopardy!
- 2012 – Google Brain develops a system which can recognize cats in YouTube videos!!
- 2012 – AlexNet sets machine learning as a standard in computer vision
- 2016 – Google's Alpha Go defeats human world champion in the game of Go
- 2017 – Microsoft's speech recognition system beats human standard

Current day applications

- Algorithmic trading
- Bioinformatics
- Brain-machine interfaces
- Cheminformatics
- Computer vision
- Credit card fraud detection
- Computer vision
- Handwriting recognition
- Information retrieval
- Marketing
- Medical diagnostics
- Natural language processing
- Online advertising
- Recommender systems
- Robot control
- Social network analysis
- Speech recognition
- Tracking patient's health condition

Computer vision

- Face recognition
- Object detection
- 3D reconstruction
- Pose estimation
- Video captioning

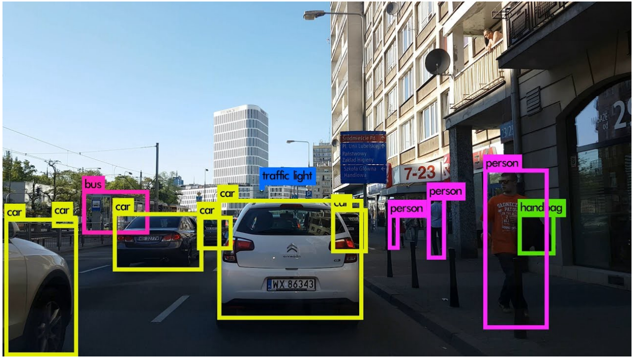


Figure: `https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b`

Autonomous driving/flight

- ALVINN drove 140km at the highway at the end of eighties with no human assistence
- In the past decade numerous companies started working on autonomous vehicle driving using neural networks, reinforcement learning...
- Autonomous flight of quadrotors, helicopters...

Game playing

- Human level backgammon player at the end of eighties
- Alfa Go defeats human champion in Go 4 to 1
- Neural network plays Atari games
- Neural network plays 3D shooters (e.g., Doom) better than humans

Natural language processing and speech recognition

- OCR and hand written text recognition
- Text classification
- Sentiment analysis
- Topic analysis
- Machine translation
- Speech recognition
- Dialog and recommendation systems

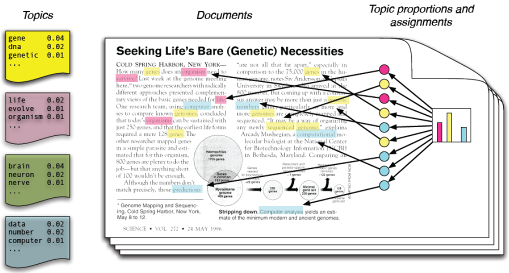


Figure: `https://rpubs.com/rain10241/63854`

Medical applications

- Tumor recognition and classification
- Predicting patient's future health state
- Therapy optimization (e.g., sepsis)

Network analysis

- Community detection
- Link recommendation in social networks
- Link detection in criminal and terrorist networks
- Targeted advertising

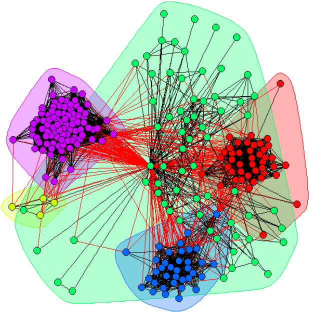


Figure: `http://francescopochetti.com/community-detection-social-networks/`

Overview

With respect to problem formulation

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised learning

- Model should establish relationship between *target variable* and *features*
- Model allows making predictions of target variable if feature values are known
- Input data consist of both feature values and target values
- Term supervision refers to availability of target values
- Task to be learned is defined by the data instead of being defined by the algorithm
- Typical tasks:
  - Regression
  - Classification

Regression

- Target variable is continuous
- Tasks like prediction of stock prices, steering angles, resource consumption, rainfall, algorithm runtime

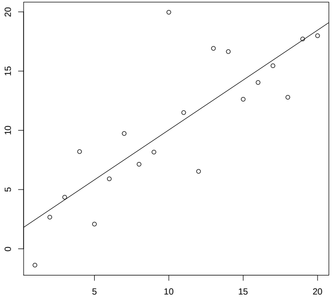


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Classification

- Target variable is categorical (finite and unordered value set)
- Tasks like face detection, object recognition, OCR, speech recognition

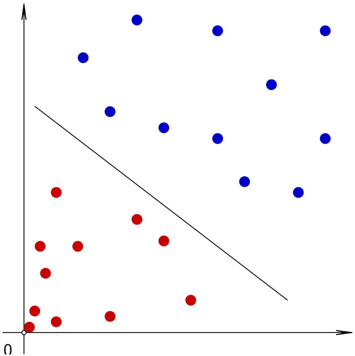


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Unsupervised learning

- Model should identify some relevant structure in the data
- Input data consists only of feature values, there are no target values
- Task to be learned is defined by the algorithm – for different kinds of tasks, different learning algorithms are formulated
- Typical tasks:
  - Clustering
  - Dimensionality reduction
  - Representation learning

Clustering

- Identification of groups of data
- Grouping can be defined based on proximity, density, shape, ...
- k means, DBSCAN, Gaussian mixture, agglomerative hierarchical clustering, ...
- Tasks like community detection in social networks, human genetic clustering, detection of different types of tissue in medical imaging, data reduction
- Interesting both in its own right and as a data preprocessing technique

Clustering illustration

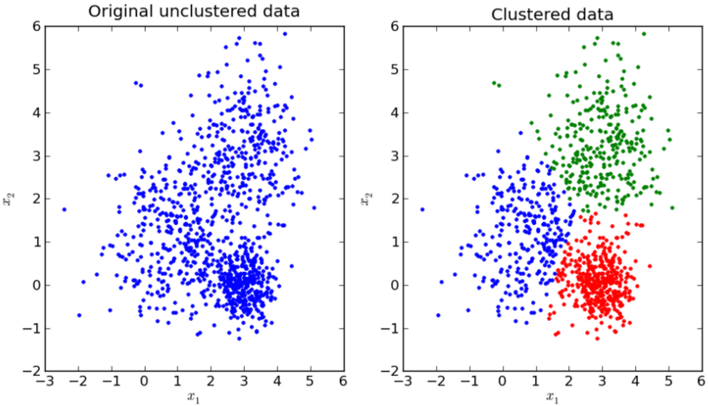


Figure: `https://towardsdatascience.com/k-means-data-clustering-bce3335d2203`

Dimensionality reduction

- Identification of subspaces (planes or manifolds) in which data lie
- PCA, autoencoders, *t*-SNE, ...
- Mostly used for data preprocessing and visualisation

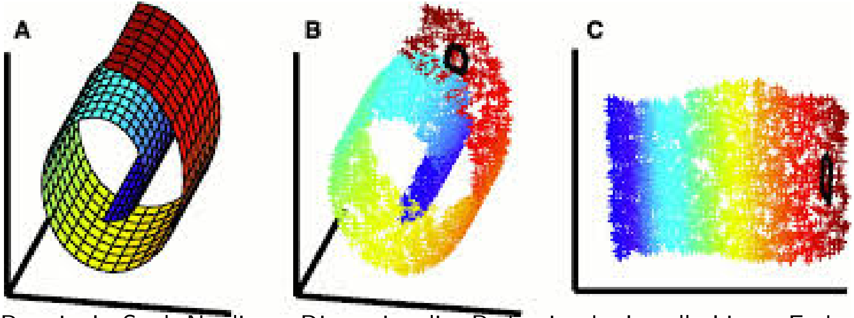


Figure: S. Roweis, L. Saul, Nonlinear Dimensionality Reduction by Locally Linear Embeddings

Representation learning

- Finding representations in data which facilitate exploitation of relevant information
- PCA, autoencoders, VAEs, GANs, word2vec,...
- Mostly used for natural language understanding, semantic image manipulation, improvement of other algorithms...

Reinforcement learning

- Probably the highest hype to utility ratio :)
- Agent takes actions in an environment, observes its state, and receives reward from the environment for actions taken
- Model should map states to actions, so that total obtained reward is maximal
- Since reward is given, it is not unsupervised learning
- Agent is not informed if the action taken in some state was the right one!!
- Therefore it is not supervised learning, either
- Credit assignment mechanism is needed to identify best actions based on total reward obtained
- Control tasks in robotics, autonomous vehicle driving, therapy optimisation, dialog systems, game playing

With respect to the kind of variable dependence modelled

- Generative: $p(\mathrm{x})$
- Discriminative: $p(y|\mathrm{x})$

Generative models

- Model joint probability $p(x)$
- Provide full description of the data if the probability model is right
- Can generate data
- Require a lot of data for training
- Costly to train and sometimes even to apply
- Can provide confidence intervals for y if the probability model is explicit
- Can be wrong if the probability model is far from reality

Discriminative models

- Model conditional probability $p(y|\mathrm{x})$
- Therefore cannot generate data (x, y), only y given x
- Less data is required
- Easier to train and use
- Can provide confidence intervals for y if the probability model is explicit
- Can be wrong if the probability model is far from reality

The distinction is not clear

- What if we model $p(x_1, x_2, x_3 | x_4, x_5, x_6)$?
- It is conditional, so it does not model dependencies between x_4, x_5, x_6, but jointly models x_1, x_2, x_3
- It's somewhere in between

Overview

Loss and risk

- There is a relationship between x and y
- We are aware of that relationship via sample $\mathcal{D} = \{(\mathrm{x}_i, y_i) \mid i = 1, \ldots, N\}$
- Find "the best" function f such that $y \approx f(\mathrm{x})$
- Let *loss function* L quantify the discrepancy between y and $f(\mathrm{x})$
- Loss is averaged over the training set to obtain *error function* or *empirical risk*

$$E(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\mathrm{x}_i))$$

- *Emprirical risk minimization principle*: find the model which minimizes E

Model representation

- Considering all possible models is infeasible, so a *model representation* is assumed
- We assume that the model $f_{\mathrm{w}}(\mathrm{x})$ is determined by a vector of *model parameters* w, so the error function can be written $E(\mathrm{w}, \mathcal{D})$

ERM for classification

- What should we minimize?

ERM for classification

- What should we minimize?
- Minimize the number of training errors
- Indicator function:

$$I(F) = \begin{cases} 1 & \text{if } F \\ 0 & \text{if } \neg F \end{cases}$$

- Loss: $L(u, v) = I(u \neq v)$
- Optimization problem:

$$\min_{\mathrm{w}} \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq f_{\mathrm{w}}(\mathrm{x}_i))$$

Regression

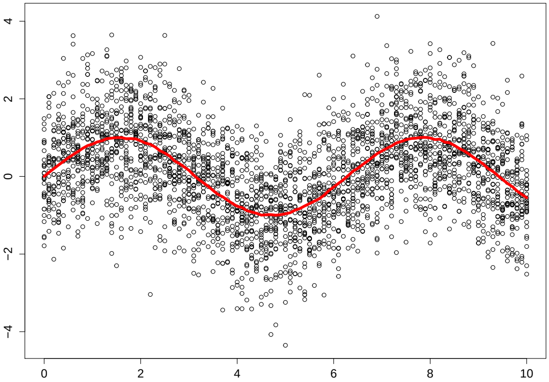


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Regression

- Regression function: $r(\mathrm{x}) = \mathbb{E}(y|\mathrm{x}) = \int y \; p(y|\mathrm{x})dy$
- If $r(\mathrm{x}) = f_{\mathrm{w}}(\mathrm{x})$ for some w, then it is a minimizer of

$$\mathbb{E}[(y - f_{\mathrm{w}}(\mathrm{x}))^2]$$

- In general, the minimum is attained for the function closest[1] to $r(\mathrm{x})$

[1] With respect to ℓ_2 norm

ERM for regression

- Loss:
$$L(u, v) = (u - v)^2$$
- Optimization problem:
$$\min_{\mathrm{w}} \frac{1}{N} \sum_{i=1}^{N} (y_i - f_{\mathrm{w}}(\mathrm{x}_i))^2$$

How well can we fit a model?

- Consider regression problem
- Simple linear regression: $f_{\mathrm{w}}(x) = w_0 + w_1 x$
- Polynomial linear regression: $f_{\mathrm{w}}(x) = \sum_{i=0}^{n} w_i x^i$

Simple linear regression

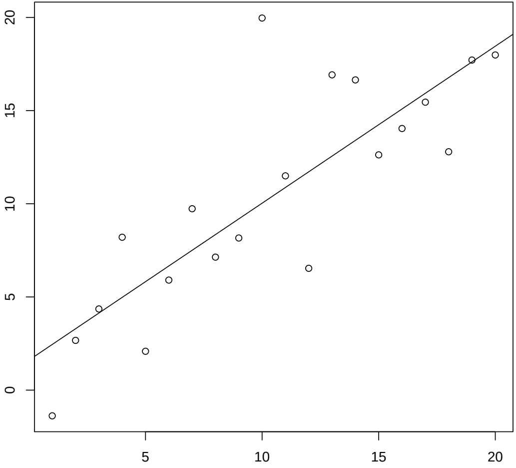


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Polynomial linear regression

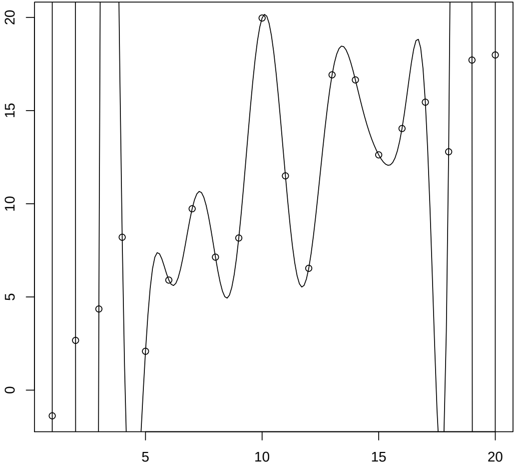


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Training and testing data

- Obviously, we cannot trust the error $E(\mathrm{w}, \mathcal{D})$ as an estimate of future model error on unseen data
- Therefore, fitting and evaluation are always performed on separate data sets – training and testing data
- Still, what's going on?

Overfitting

- Good fit of the model on the training data does not mean good generalization
- Compare to rote learning
- Caused by model flexibility (often called complexity)
- Controlling model flexibility is of paramount importance for good generalization
- One of central topics of machine learning and source of it's deepest theory

How to make models less flexible?

- Restrict model representation (e.g. linear models)?

How to make models less flexible?

- Restrict model representation (e.g. linear models)?
- Possible, but that approach may be too rigid

How to make models less flexible?

- Restrict model representation (e.g. linear models)?
- Possible, but that approach may be too rigid
- Given a very flexible model representation, can flexibility be tuned based on model's performance?

Regularization (1)

- Minimization of regularized empirical risk:

$$\min_{\mathrm{w}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f_{\mathrm{w}}(\mathrm{x}_i)) + \lambda \Omega(\mathrm{w})$$

- Frequent choice of *regularization term* is squared ℓ_2 norm

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

- Regularization term penalizes the magnitude of the parameters, making the model less adaptable to the data
- *Regularization meta-parameter* λ tunes model flexibility/complexity

Regularization (2)

- In a more general sense, regularization is any modification of optimization problem that restricts model flexibility and makes it less susceptible to overfitting
- In an even more general sense, regularization is any modification of a mathematical problem which makes it less sensitive to changes in input parameters

Regularization example – classification models

- Linear classification model:

$$f_{\mathrm{w}}(\mathrm{x}) = w_0 + w_1 x_1 + w_2 x_2$$

- Polynomial classification model:

$$f_{\mathrm{w}}(\mathrm{x}) = \sum_{i=1}^{n} \sum_{j=0}^{i} w_{ij} x_1^j x_2^{i-j}$$

Regularization example – data points

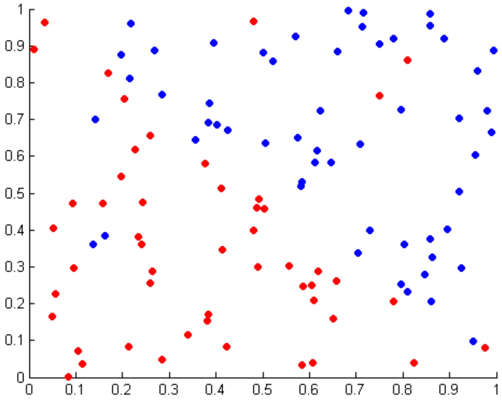
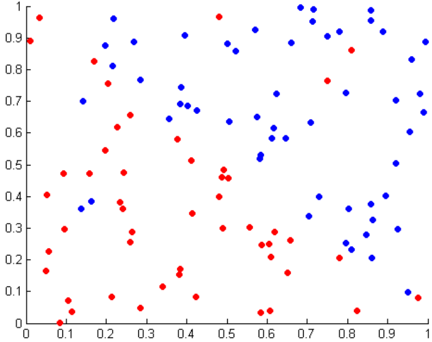


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Regularization example – linear classifier prediction



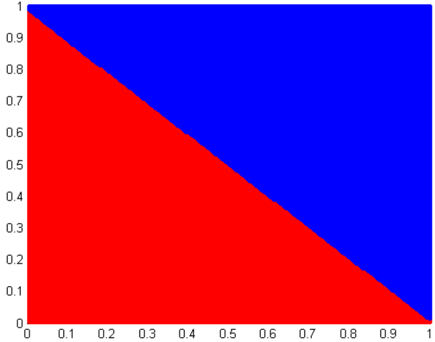


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Regularization example – polynomial classifier prediction

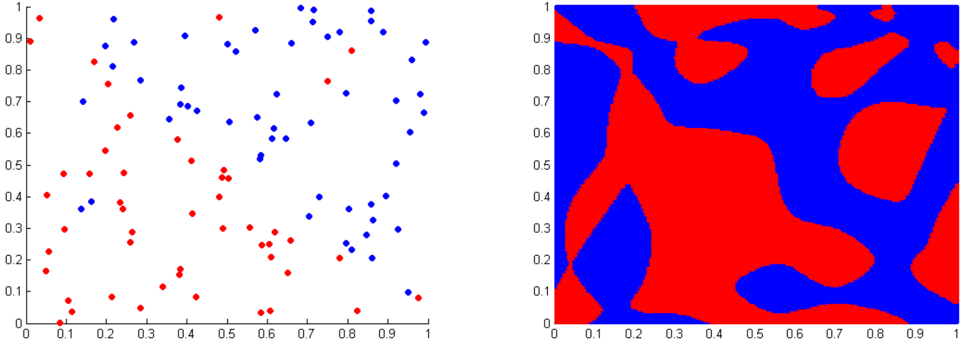
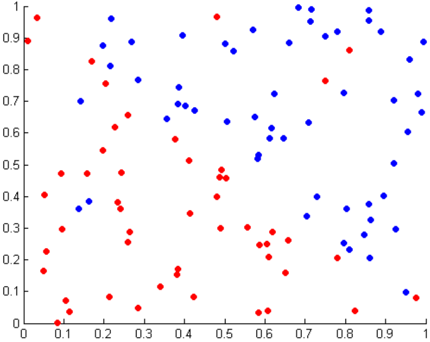


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Regularization example – regularized poly. prediction



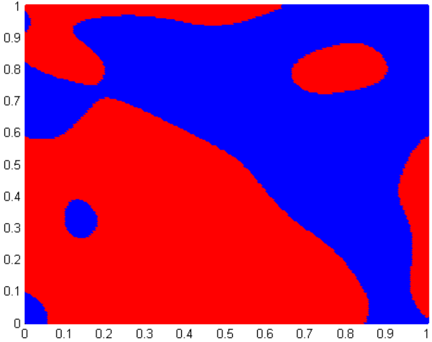


Figure: P. Janičić, M. Nikolić, Artificial intelligence, (in Serbian).

Regularization example – regularized poly. prediction

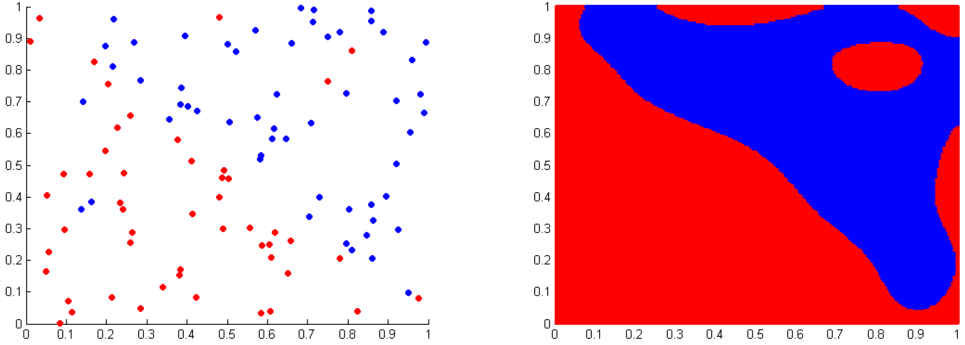


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Regularization example – regularized poly. prediction

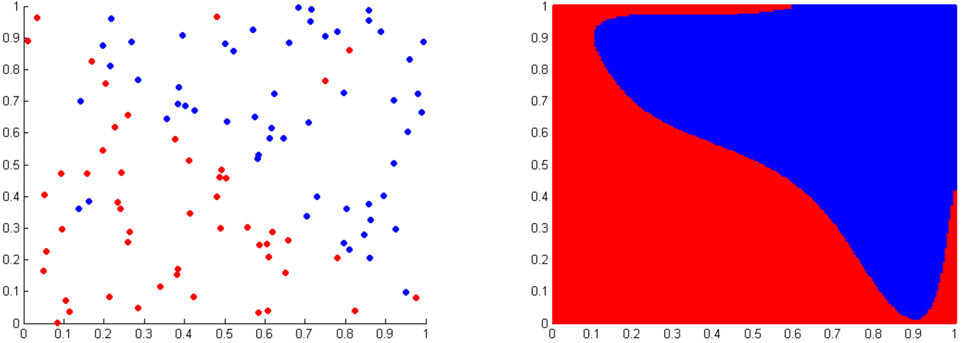


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Regularization example – regularized poly. prediction

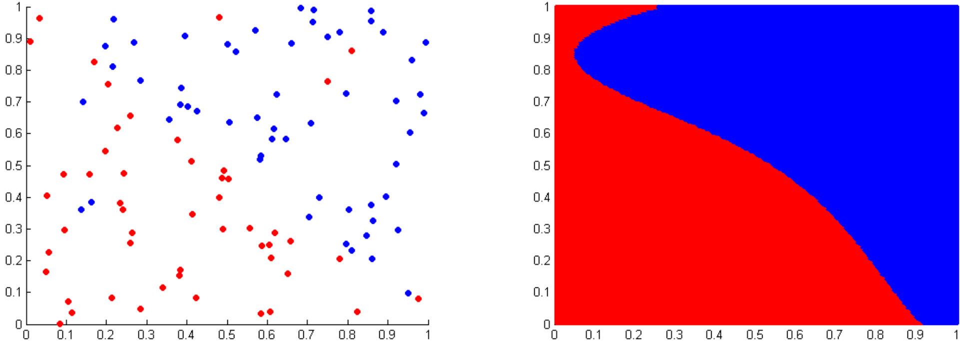


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Regularization example – regularized poly. prediction

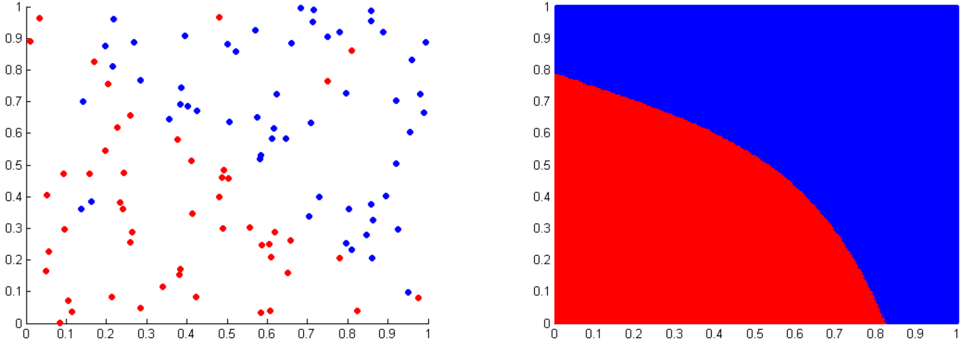


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Regularization example – regularized poly. prediction

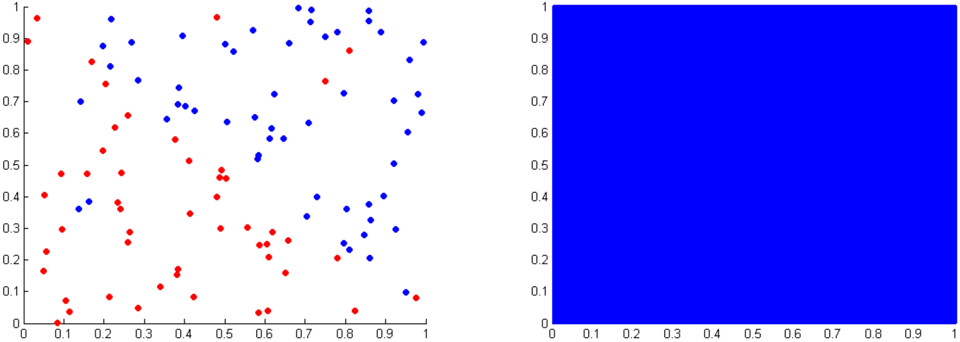


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

How to minimize error function?

- Anlytically by setting gradients to zero - often impossible.
- Numerically by iteratively moving towards lower values of the function
- What is the direction of steepest descent?
- Differentiable error functions allow for use of gradients (direction of steepest ascent)
- Cautious move in opposite direction leads to decrease of error function value

Gradient

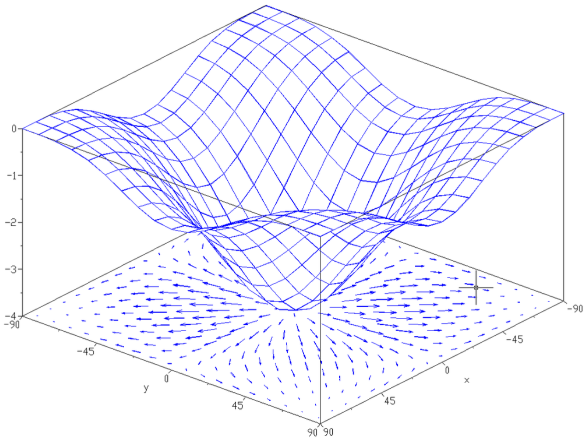


Figure: math.wikia.com/wiki/Gradient

Gradient descent

- Repeat until convergence:

$$\mathrm{w}_{k+1} = \mathrm{w}_k - \mu_k \nabla E(\mathrm{w}_k, \mathcal{D})$$

- How to select step size μ_k?
- Fixed step size is often used, but there are better choices

Gradient descent

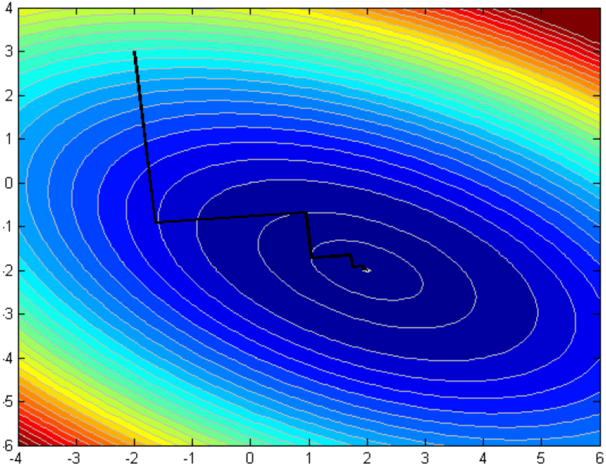


Figure: Y. Li, Course materials.

Overview

Model

- Assume linear model:

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

- Let's go probabilistic!

$$p_{\mathrm{w}}(y|\mathrm{x}) = \mathcal{N}(\mathrm{w} \cdot \mathrm{x}, \sigma^2)$$

Illustration

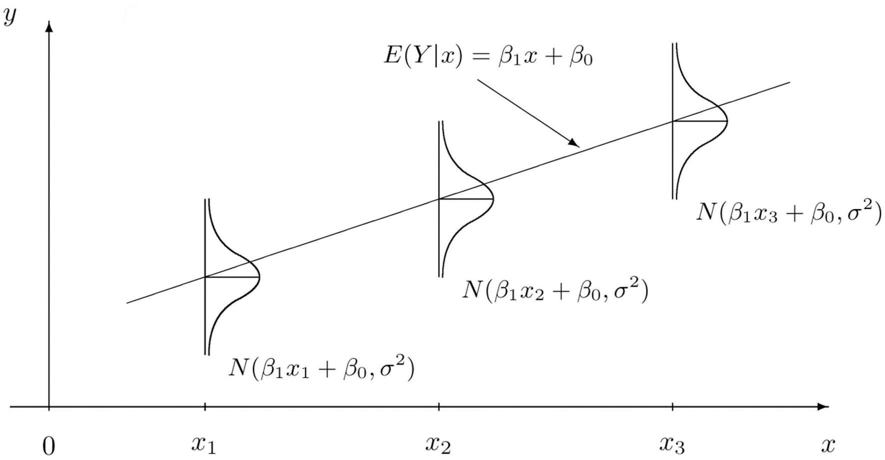


Figure: D. Shafer, Z. Zhang, Introductory Statistics, 2012.

Maximal likelihood principle (1)

- How to choose w?
- Probability of observing the training set is (assuming IID)

$$\prod_{i=1}^{N} p_{\mathrm{w}}(y_i | \mathrm{x}_i)$$

where

$$p_{\mathrm{w}}(y|\mathrm{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mathrm{w} \cdot \mathrm{x})^2}{2\sigma^2}\right)$$

- As a function of w it is called *likelihood*
- We are interested in *maximal likelihood estimate* of the parameters – the parameter values under which the data is most likely

Maximal likelihood principle (2)

- It is more suitable to minimize negative log likelihood of the parameters:

$$NLL(\mathrm{w}) = -\log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathrm{w}\cdot \mathrm{x}_i)^2}{2\sigma^2}\right)$$

$$NLL(\mathrm{w}) = \frac{N}{2}\log 2\pi + N\log\sigma + \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \mathrm{w}\cdot \mathrm{x}_i)^2$$

- Learning problem:

$$\min_{\mathrm{w}} \sum_{i=1}^{N}(y_i - \mathrm{w}\cdot \mathrm{x}_i)^2$$

Solution (1)

- Matrix formulation

$$\min_{w} \|y - Xw\|_2^2$$

- It is a convex problem!
- Ideally

$$Xw = y$$

  so, ideally,

$$w = X^{-1}y$$

  but, in general, X is not quadratic, nor invertible
- Let's set derivatives of error function to 0

$$E(w) = \|y - Xw\|^2 = (y - Xw)^T(y - Xw)$$

$$\nabla E(w) = 2X^T(y - Xw) = 0$$

$$w = (X^TX)^{-1}X^Ty$$

Solution (2)

- Interestingly $(X^TX)^{-1}X^T$ behaves like one would expect nonexistent X^{-1} to behave:

$$\underbrace{(X^TX)^{-1}X^T}_{\text{pseudoinverse}}X = I$$

- If the matrices are too big for inversion or even storing in memory, gradient based methods can be used

Interpretability

- Magnitude of parameters reflects their relative importance (if the features vary in the same range)
- Sign of a parameter reflects the direction of correlation of the corresponding feature and the target variable
- For example, model $y = 2x_1 - 0.1x_2 + 3$ suggests that feature x_1 affects y much more strongly than feature x_2 and that x_1 affects it positively and x_2 negatively

What about polynomials?

- Linearity means linearity in parameters, not in features!!
- This is a linear model:
$$f_{\mathrm{w}}(\mathrm{x}) = w_0 + \sum_{i=1}^{n} w_i x^i$$
- It does not seem linear in coordinate system (x), but it clearly is in coordinate system $(1, x, x^2, \ldots, x^n)$

Illustration

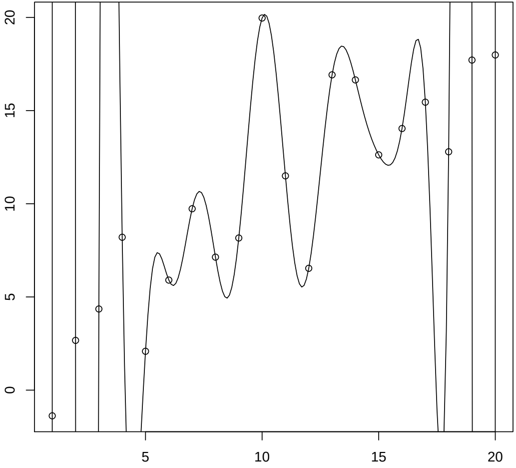


Figure: P. Janičić, M. Nikolić, Artificial intelligence (in Serbian).

Interactions

- Linear model expresses independent contributions of features to target variable, which is not realistic
- One solution is to include interactions (products of features):

$$f_{\mathrm{w}}(\mathrm{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} w_{ij} x_i x_j$$

- The model is still linear, but the contribution of x_i can be dependent on x_j if it benefits the prediction

Ridge regression

- If the features are linearly dependent, matrix $X^T X$ is not invertible
- If they are highly correlated, it is ill-conditioned
- Therefore, regularized problem is considered

$$\min_{w} \sum_{i=1}^{N} (y_i - w \cdot x_i)^2 + \lambda \|w\|_2^2$$

- The solution

$$w = (X^T X + \lambda I)^{-1} X^T y$$

- Adding λI makes it a full rank matrix

Overview

Going binary

- Assume classification task and let $y \in \{0, 1\}$
- Linear model approximates values $\{0, 1\}$ very badly:

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

- But it can be squashed to the interval $(0, 1)$ using *sigmoid function*:

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

- The model:

$$f_{\mathrm{w}}(\mathrm{x}) = \sigma(\mathrm{w} \cdot \mathrm{x})$$

Sigmoid function

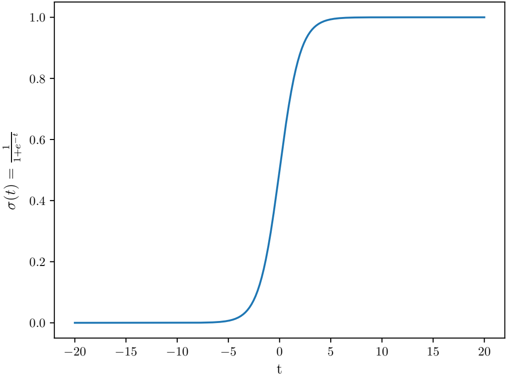


Figure: M. Nikolić, A. Zečević, Machine learning (in Serbian).

Going probabilistic

- This can be seen as probability

$$p_{\mathrm{w}}(y = 1|\mathrm{x}) = \sigma(\mathrm{w} \cdot \mathrm{x})$$

- Therefore

$$p_{\mathrm{w}}(y|\mathrm{x}) = \sigma(\mathrm{w} \cdot \mathrm{x})^{y}(1 - \sigma(\mathrm{w} \cdot \mathrm{x}))^{1-y}$$

- How to choose parameters w?

Maximal likelihood principle

- Likelihood function:

$$\prod_{i=1}^{N} p_{\mathrm{w}}(y_i|\mathrm{x}_i) = \prod_{i=1}^{N} \sigma(\mathrm{w}\cdot\mathrm{x}_i)^{y_i}(1-\sigma(\mathrm{w}\cdot\mathrm{x}_i))^{1-y_i}$$

- Negative log likelihood:

$$NLL(\mathrm{w}) = -\sum_{i=1}^{N}[y_i \log\sigma(\mathrm{w}\cdot\mathrm{x}) + (1-y_i)\log(1-\sigma(\mathrm{w}\cdot\mathrm{x}))]$$

- Loss used is called *crossentropy* and is very common in classification tasks:

$$L(p,q) = -\sum_{j} p(j)\log q(j)$$

Learning problem

- Learning problem is to minimize regularized negative log likelihood:

$$NLL(\mathrm{w}) = -\sum_{i=1}^{N}[y_i \log \sigma(\mathrm{w} \cdot \mathrm{x}_i) + (1 - y_i) \log(1 - \sigma(\mathrm{w} \cdot \mathrm{x}_i))] + \lambda \|\mathrm{w}\|_2^2$$

- The problem is convex!
- Usually optimized by Newton's method, but let's go for gradient descent!

Gradient

$$
\begin{aligned}
\frac{\partial NLL(\mathrm{w})}{\partial w_j} &= -\sum_{i=1}^{N}\left[y_i \frac{\partial}{\partial w_j}\log\sigma(\mathrm{w}\cdot\mathrm{x}_i) + (1-y_i)\frac{\partial}{\partial w_j}\log(1-\sigma(\mathrm{w}\cdot\mathrm{x}_i))\right] + \lambda\frac{\partial}{\partial w_j}\sum_{i=1}^{n} w_i^2 \\
&= -\sum_{i=1}^{N}\left[y_i \frac{\sigma(\mathrm{w}\cdot\mathrm{x}_i)(1-\sigma(\mathrm{w}\cdot\mathrm{x}_i))}{\sigma(\mathrm{w}\cdot\mathrm{x}_i)} x_{ij} - (1-y_i)\frac{\sigma(\mathrm{w}\cdot\mathrm{x}_i)(1-\sigma(\mathrm{w}\cdot\mathrm{x}_i))}{1-\sigma(\mathrm{w}\cdot\mathrm{x}_i)} x_{ij}\right] + 2\lambda w_j \\
&= -\sum_{i=1}^{N}[y_i(1-\sigma(\mathrm{w}\cdot\mathrm{x}_i))x_{ij} - (1-y_i)\sigma(\mathrm{w}\cdot\mathrm{x}_i)x_{ij}] + 2\lambda w_j \\
&= \sum_{i=1}^{N}[\sigma(\mathrm{w}\cdot\mathrm{x}_i) - y_i]\, x_{ij} + 2\lambda w_j
\end{aligned}
$$

Gradient descent updates

- Elementwise:

$$w_j \leftarrow w_j - \mu \sum_{i=1}^{N} [\sigma(\mathrm{w} \cdot \mathrm{x}_i) - y_i]\, x_{ij} + 2\lambda w_j$$

- Matrix form:

$$\mathrm{w} \leftarrow \mathrm{w} - \mu \mathrm{X}^T [\sigma(\mathrm{Xw}) - \mathrm{y}] + 2\lambda \mathrm{w}$$

Unregularized vs. regularized

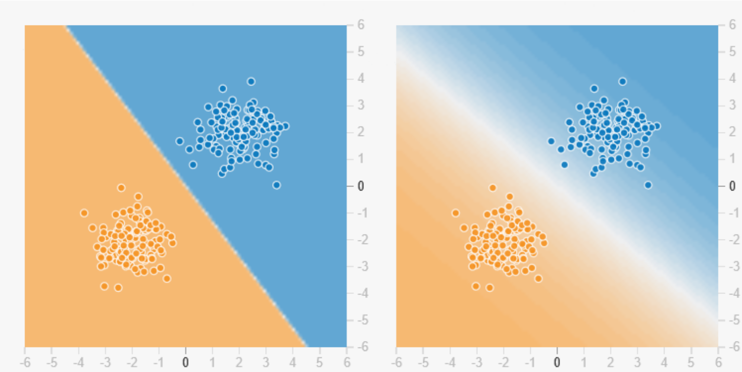


Figure: `https://playground.tensorflow.org`

Overview

What is preprocessing?

- Sometimes algorithms are not directly applicable to the data due to its form
- Sometimes they are applicable, but their performance may be worse due to the form of the data
- In such cases data needs to be transformed to a more desirable form, which is called *preprocessing*

Some often used techniques

- Coding of categorical features
- Missing value imputation
- Standardization/normalization
- Outlier removal
- Dimensionality reduction (e.g., PCA)
- Decorrelation (e.g., PCA)
- Aggregation of feature values or instances
- Feature selection

One hot encoding

- How can we use a linear model if a categorical variable is present?
- Consider a feature – country of birth with C possible outcomes
- Terrible way to represent its values would be to, say, sort countries alphabetically and assign them indices in the sorted sequence
- Meaningful way would be to introduce C variables such that for i-th country all are 0 except the i-th variable, which is 1

$$(0, 0, 1, 0, 0)$$

Missing value imputation

- Sometimes, values of some variables are not observed, but the values of others are
- One way of dealing with this problem is removing such instances, but they could be numerous
- Such data also contains information which should be used
- Two simple approaches:
  - Imputation of the mean of observed values of the variable
  - Prediction of missing values by a regression model based on other variables

Variables of different scale

- Features are often measured at wildly different scales (e.g., savings and age)
- If interpretability is of value, model parameters cannot be compared to determine relative importance of such variabels
- Regularization will act differently on parameters corresponding to different variables
- Numerical/optimization stability may be an issue

Standardization

- In response to the previous problem, some kind of feature scaling is virtually always applied
- One such approach is standardization – each feature is *centered* by removing the mean and divided by its standard deviation

What does evaluation consist of?

- Evaluation metrics – metrics in which we express the quality of the model
- Evaluation techniques – procedures used to compute metrics in a proper way

Classifcation metrics

- Often derived from confusion matrix

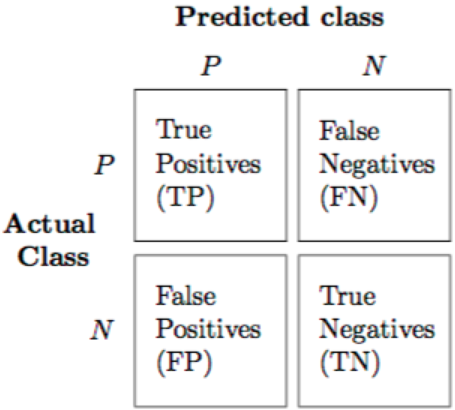


Figure: MLxtend documentation

Accuracy

- Fraction of correctly classified instances among all classified instances

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

- Sensitive to class imbalance
- Consider detection of a rare disease

AUC

- Area under the (receiver operator characteristic) curve
- The name is as ugly as a related interpretation (we focus on a nice one)
- Assume that a binary classifier assigns a score to each class – lower scores to class 0 and higher scores to class 1 (there should be a threshold)
- Pick instances x_0 from class 0 and x_1 from class 1 at random

$$AUC = P(f_w(x_0) < f_w(x_1))$$

- 0.5 is random guessing and < 0.5 means you are doing something very wrong :)
- Insensitive to class imbalance

Precision and recall

- Often used in information retrieval and ranking

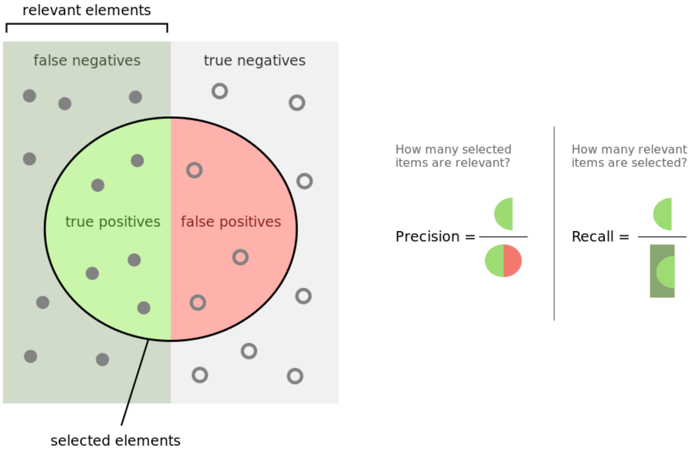


Figure: https://en.wikipedia.org/wiki/Precision_and_recall

Regression metrics

- Often derived from model residuals

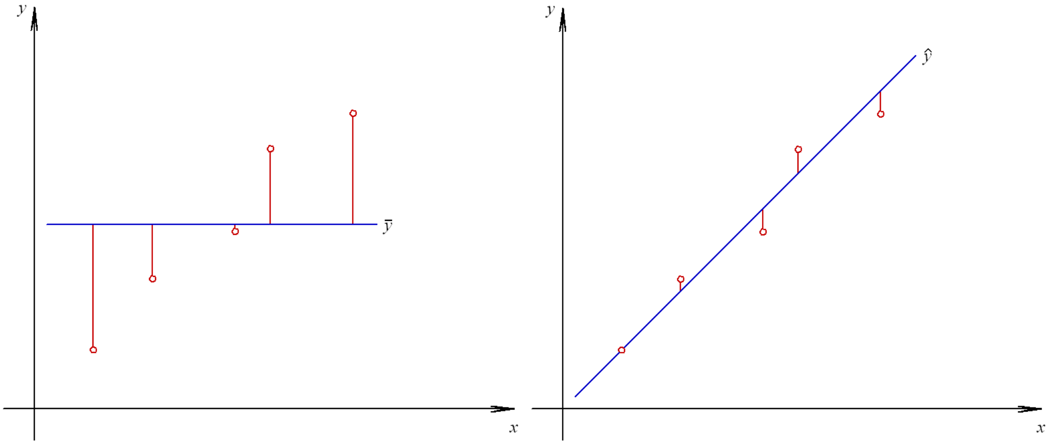


Figure: P. Janičić, M. Nikolić, Veštačka inteligencija (in Serbian).

Root mean squared error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f_{\mathrm{w}}(\mathrm{x}))^2}$$

- Like standard deviation, but not with respect to the mean, but with respect to the model
- Expressed in same units as the original variable
- Used to estimate the magnitude of the error
- Particularly useful if we know what magnitude of the error is acceptable in particular application

Coefficient of determination R^2

$$R^2 = 1 - \frac{MSE}{Var} = 1 - \frac{\sum_{i=1}^{N}(y_i - f_{\mathrm{w}}(\mathrm{x}_i))^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

- Measures the portion of variance of target variable explained by the model
- In range $(-\infty, 1]$
- If < 0 your training set is probably biased
- More suitable in comparisons due to fixed scale

Main tenant of model evaluation

- Data used for model evaluation should by no means be used in its training!

Training/testing split

- Split data into two sets – training and test set
- Use training set to train a model
- Use test set to compute the error of the model
- Used for a single training run (if there is such a thing), not for meta-parameter tuning!

x_1	x_2	x_3	y
1	9	0	8
0	6	2	1
1	3	1	5
4	9	7	6
1	1	6	7
7	2	3	4
2	9	9	9
3	3	4	6
7	2	1	7
6	5	1	5

Training/validation/testing split

- Split data into training, validation, and test sets
- Tune the model by training it for different meta-parameter settings on the training and checking its performance on the validation set
- Estimate the error of the best model on the test set

x_1	x_2	x_3	y
1	9	0	8
0	6	2	1
1	3	1	5
4	9	7	6
1	1	6	7
7	2	3	4
2	9	9	9
3	3	4	6
7	2	1	7
6	5	1	5

Pitfalls of preprocessing and evaluation

- No information from the test set should be used in training, therefore:
  - For missing value imputation on the test set, use exclusively means computed on the training set
  - For standardization, use exclusively means and standard deviations computed on the training set
- Test set selection should follow practically realistic scenarios

Overview

Will it work?

PROBABLY NOT!

If it's quite bad

- Bad data (go check your data :))
- Underfitting
- Overfitting

Underfitting vs. overfitting

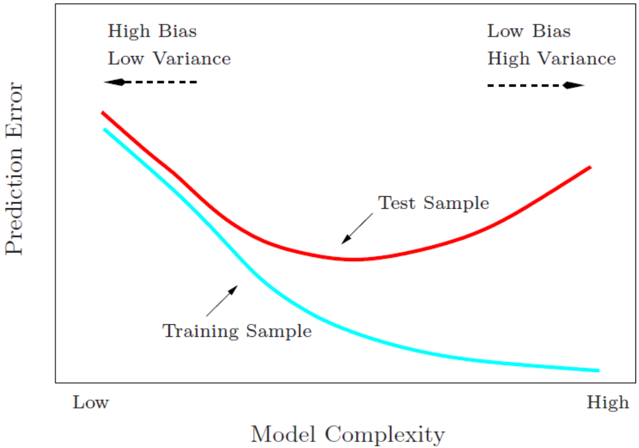


Figure: T. Hastie, R. Tibshirani, J. Friedman, "Elements of Statistical Learning", 2001.

What if we have an underfitting problem?

- Use more flexible models (even try to overfit)
- Consider model properties – maybe the one you used is not suited to the problem
- Use lower regularization parameter values
- Construct new features

What if we have an overfitting problem?

- Use less flexible models (even try to underfit)
- Use feature selection
- Use higher regularization parameter values
- Use more data

Word of caution

- High training error and high test error indicate lack of flexibility which leads to underfitting
- But not necessarily – e.g., that could also happen due to large learning step
- Low training error and high test error indicate too flexible model, which leads to overfitting
- But not necessarily – e.g., that could also happen due to bad preprocessing (stratification)
- It's tricky, be cautious

Sometimes it's all about features

- If the features are not informative enough, no learning algorithm can help
- Check which classes get mixed-up and check if existing features should be able to differentiate between them
- Check for correlations between features and target variable
- Consider using deep neural networks over raw data, instead of hand crafting the features

Getting more into details

- Check different error metrics, they tell you different things
- If the model is interpretable, check if it makes sense
- Check for patterns in instances for which the predictions are wrong
- Inspect instances for which the model provides wrong answers with high confidence
- Try to visualize errors – might be easy for images, hard for high dimensional vectorial data

Mladen Nikolić
nikolic@math.rs
Machine Learning and Applications Group at the Faculty of Mathematics
machinelearning.math.rs