Mechatronic Engineering program:
Python for machine learning and data science

Classification (revisited)

Ziemowit Dworakowski
AGH University of Krakow

1

What you should probably know already?

Outliers
(their sources, ways to deal with them)

Overlapping classes
(And how to deal with them)

Basic optimization methods
(Gradient, gradient with momentum, 1+1, grid search)

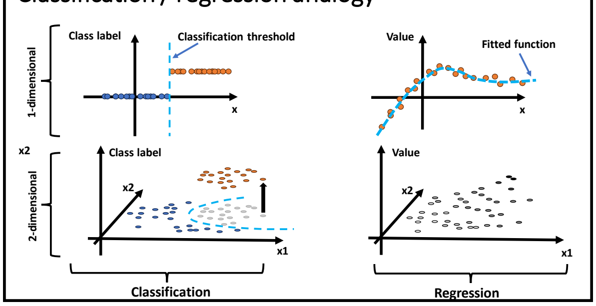
Overfitting
(And how to deal with it)

Metaparameter configuration procedure
(2nd order optimization)

Basic algorithms
(kNN, manually configured decision tree)

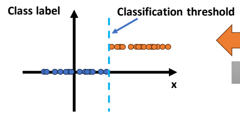
2

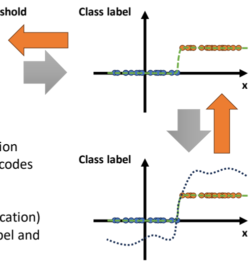
Classification / regression analogy



3

Classification / regression analogy



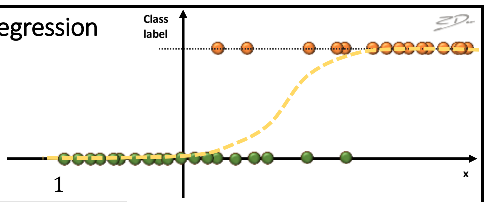


You can interpret classification as regression where your aim is to find a function that codes class labels.

To this end we can (similarily as in classification) strive to minimize error between class label and this function's output

4

Logistic regression



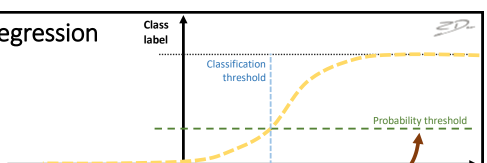
$$f(x) = \frac{1}{1 + e^{-(\beta_1 x + \beta_0)}}$$

This is again a sigmoid curve – here called a „logistic function".
β_1 and β_0 are parameters that need to be fit during training in such a way as to maximize **log likelihood** of correct label assignment.

We can use standard gradient-based optimization approach for that

5

Logistic regression



Once we have it, we can use it to classify new data by selection of acceptable probability threshold

$$f(x) = \frac{1}{1 + e^{-(\beta_1 x + \beta_0)}}$$

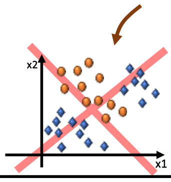
This is again a sigmoid curve – here called a „logistic function".
β_1 and β_0 are parameters that need to be fit during training in such a way as to maximize **log likelihood** of correct label assignment.

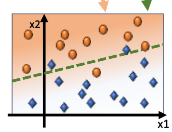
We can use standard gradient-based optimization approach for that

6

Logistic regression

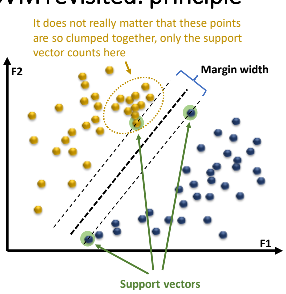
- LR is a classification equivalent of the linear regression
- It can work for multidimensional problems (we just have more parameters to learn)
- It can fit a „line classifier” in the „best way possible” (meaning that the classification has actually probabilistic interpretation
- It cannot solve nonlinearly separable classification problems





7

SVM revisited: principle

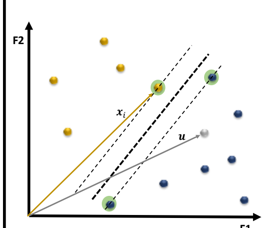


In regression SVM aimed to make marigin as small as possible and fit all the data **inside** the margin.

In classification it does the opposite: it tries to make margin as wide as possible and keep all data **outside** of it

8

SVM revisited: How does it do that?



Let: $\boldsymbol{\omega}$ - be a vector perpendicular to the margin
$\boldsymbol{x_i}$ - be the i-th sample
$\boldsymbol{u}$ - be an unknown sample
y_i - be a class label (equal to either 1 or -1)
α_i - be a Lagrangian multiplier denoting whether a point should be treated as a support vector

In a nutshell, If we want to classify a new sample, we are using the following equation:

$$\sum_i \alpha_i y_i \vec{x}_i \circ \vec{u} + b \geq 0 \Rightarrow THEN \oplus$$

Note, that to classify a new point we calculate its dot product with all the support vectors – that will come in handy in a second…

9

If you want to know what is the source for this equation, here is a brief explanation. You don't need to learn that for the test though!

Decision rule:

$$\vec{\omega}\circ\vec{u}+b\geq 0\Rightarrow THEN\oplus$$

For the training samples:

$$\begin{cases}\vec{\omega}\circ\vec{x}_{+}+b\geq 1\\ \vec{\omega}\circ\vec{x}_{-}+b\leq -1\end{cases}\Rightarrow y_i(\vec{\omega}\circ\vec{x}_i+b)-1\geq 0$$

For samples „on the margin" (support vectors):

1 $$y_i(\vec{\omega}\circ\vec{x}_i+b)-1=0$$

Margin width:

$$(\vec{x}_{+}-\vec{x}_{-})\circ\frac{\vec{\omega}}{\|\omega\|}=\frac{2}{\|\omega\|}$$

We minimize $\frac{1}{2}\|\omega\|^2$

With respect to constraint 1

$$L=\frac{1}{2}\|\omega\|^2-\sum_i\alpha_i\left[y_i(\vec{\omega}\circ\vec{x}_i+b)-1\right]$$

$$\frac{\partial L}{\partial\omega}=\vec{\omega}-\sum_i\alpha_i y_i\vec{x}_i=0\Rightarrow\vec{\omega}=\sum_i\alpha_i y_i\vec{x}_i$$

$$\frac{\partial L}{\partial b}=-\sum_i\alpha_i y_i=0$$

After combination of the above equations, we have:

$$L=\sum_i\alpha_i-\frac{1}{2}\sum_i\sum_j\alpha_i\alpha_j y_i y_j\,\vec{x}_i\circ\vec{x}_j$$

New decision rule:

$$\sum_i\alpha_i y_i\,\vec{x}_i\circ\vec{u}+b\geq 0\Rightarrow THEN\oplus$$

10

SVM revisited: Nonlinearly separable data

Nonlinearly separable data

We can allow the algorithm to ignore a few points

(Sometimes ignoring just a few outliers makes the problem linearly separable)

We classify data accepting small error rate

„Soft margin" idea

We minimize:

$$\left[\frac{1}{2}\sum_{i=1}^{n}\max\left(0,1-y_i(\vec{\omega}\cdot\vec{x}_i-b)\right)\right]+\lambda\|\vec{\omega}\|^2$$

Previous **constraint** now is just a weighted part of the objective function

λ allows to balance importance of the margin width and influence of outliers

11

SVM revisited: Nonlinearly separable data

Nonlinearly separable data

„kernel trick"

Kernel $K(x_1,x_2)$ Defines „similarity measure" between two vectors of features. Examples of kernels include:

- **Polynomial functions:**

$$K(x_1,x_2)=(1+{x_1}^T x_2)^p$$

- **Radial-basis functions (RBF)** *(equivalent to weighted kNN model):*

$$K(x_1,x_2)=\exp(-\gamma\|x_1-x_2\|^2)$$

- **Sigmoids:** *(equivalent to single-layer MLP):*

$$K(x_1,x_2)=\tanh(\beta_0{x_1}^T x_2+\beta_1)$$

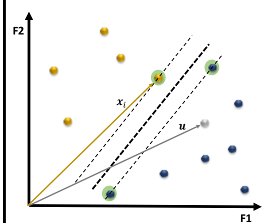
We can map the data nonlinearly into space with higher number of dimensions by using **Kernel functions** instead of dot products

(There is a chance that data that were not separable in euclidean space will be separable in our new kernel-based space)

We classify data using linear SVM

12

SVM revisited: How does it do that?



Let: $\boldsymbol{\omega}$ - be a vector perpendicular to the margin
$\boldsymbol{x_i}$ - be the i-th sample
$\boldsymbol{u}$ - be an unknown sample
y_i - be a class label (equal to either 1 or -1)
α_i - be a Lagrangian multiplier denoting whether a point should be treated as a support vector
b – is a marigin width parameter

In a nutshell, If we want to classify a new sample, we are using the following equation:

$$\sum_i \alpha_i y_i \, \vec{x}_i \circ \vec{u} + b \geq 0 \Rightarrow THEN \oplus$$

If we want to use nonlinear SVM, instead of dot product, we are using kernel function:

$$\sum_i \alpha_i y_i \, K(\vec{x}_i, \vec{u}) + b \geq 0 \Rightarrow THEN \oplus$$

A good visualization on how kernels affect data classification:
https://www.youtube.com/watch?v=Q7vT0--5VII

13

SVM revisited: How does it do that?

A good visualization on how kernels affect data classification:
https://www.youtube.com/watch?v=Q7vT0--5VII

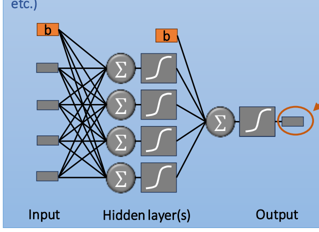
Very good lecture introduction to SVMs can be found here:
https://www.youtube.com/watch?v=_PwhiWxHK8o

Paper on SVM implementation in scikit is here (beware of challenge!):
https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

14

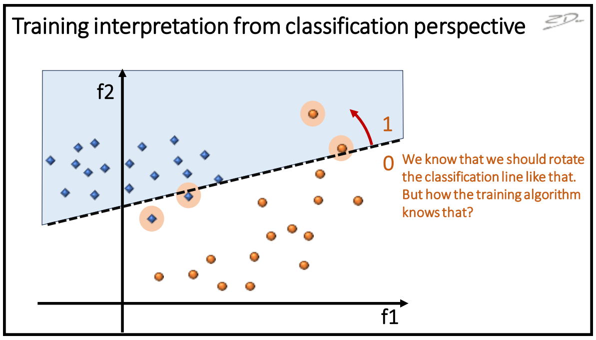
ANN revisited: MLP for classification

All of this works exactly as in regression (same structure, training methods, configuration, etc.)

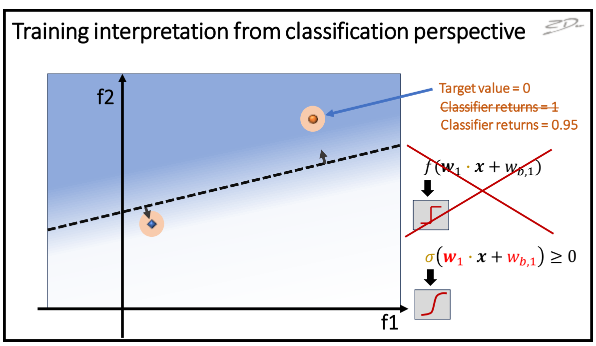


The only difference is here: we are using target data with labels (categorical binary variables) instead of numerical values

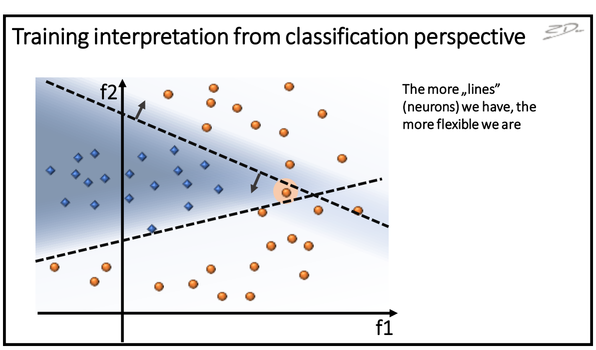
15


Training interpretation from classification perspective
f2
1
0
We know that we should rotate the classification line like that. But how the training algorithm knows that?
f1

16


Training interpretation from classification perspective
f2
Target value = 0
~~Classifier returns = 1~~
Classifier returns = 0.95
$\cancel{f(w_1 \cdot x + w_{b,1})}$
$\sigma(w_1 \cdot x + w_{b,1}) \geq 0$
f1

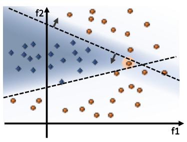
17

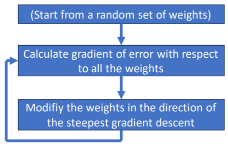

Training interpretation from classification perspective
f2
The more „lines” (neurons) we have, the more flexible we are
f1

18

ANN training: (Vanilla) gradient descent

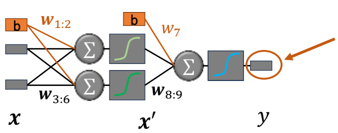
(standard, without additions)





19

Consider a simple Multilayer Perceptron (MLP) network:



$y = \sigma(w_8 \cdot x'_1 + w_9 \cdot x'_2 + w_7)$

$x'_1 = \sigma(w_3 \cdot x_1 + w_5 \cdot x_2 + w_1)$

$x'_2 = \sigma(w_2 \cdot x_1 + w_6 \cdot x_2 + w_2)$

You can combine this to get a $f(\boldsymbol{w}, \boldsymbol{x})$ function equivalent for a network

We also have a target t associated with each input vector $\boldsymbol{x}$. Using that, we can calculate error d:

$$d = \frac{1}{2}(y - t)^2$$

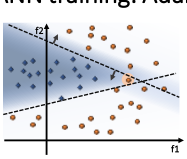
Now we can calculate gradient of d with respect to weights $\boldsymbol{w}$:

$$\nabla d = \left[\frac{\partial d}{\partial w_1}, \frac{\partial d}{\partial w_2}, \frac{\partial d}{\partial w_3}, \dots, \frac{\partial d}{\partial w_9}\right]$$

The gradient now tells us how to adjust weights

20

ANN training: Adam optimizer



Adaptive moment estimation (ADAM)

We now have a method that will be gradually dragging us towards a better classification accuracy.

What can we do to improve on it further?

1) We can accelerate optimization using averaged gradients (momentum): consecutive steps in the same direction cause the algorithm to speed up)
2) We can make the optimization more robust by including an average of square gradients of the past (oscillations tend to be stopped rapidly)
3) We can just use a sign of the derivative for each weight instead of its actual value (speedup for saddle points)

21

ANN training: Adam optimizer

Adam metaparameters:

- β_1 - Decay rate for averaged gradients (default: 0.9)
- β_2 - Decay rate for averaged gradient squares (default: 0.999)
- α - Learning rate (default: 0.001)
- ε - Small utility constant (default: 10^{-8}, don't change)

** If you want to know more (including math), this is a good (basic) article on Adam optimizer here:*
https://www.geeksforgeeks.org/intuition-of-adam-optimizer/

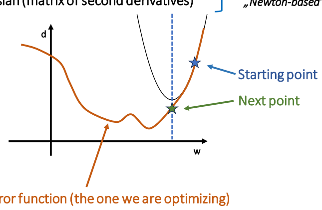
** If you want to actually learn the method at the source, this article introduces Adam optimizer (it may be challenging for beginners though):*
https://arxiv.org/pdf/1412.6980.pdf

22

ANN training: Limited memory BFGS (LM-BFGS)

(Broyden–Fletcher–Goldfarb–Shanno)

Instead of focusing on gradient, we can also use a Hessian (matrix of second derivatives) — *So called: „Quasi-Newton" or „Newton-based" methods*



23

ANN training: Limited memory BFGS (LM-BFGS)

(Broyden–Fletcher–Goldfarb–Shanno)

Instead of focusing on gradient, we can also use a Hessian (matrix of second derivatives) — *So called: „Quasi-Newton" or „Newton-based" methods*

This allows often to outperform first-order-based methods (like Adam) provided that the input space is small (not many data points, not many dimensions)

LM-BFGS is one of the many algorithms from this family – available in popular libraries for ML in Python

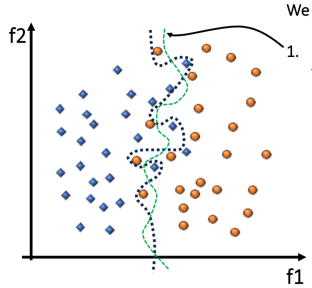
A paper with a nice overview on second-order optimization algorithms :
https://cs.nyu.edu/~overton/mstheses/skajaa/msthesis.pdf

24

ANN training: **Regularization**

„How to prevent overfitting in such a way so we don't have to stop training early?"

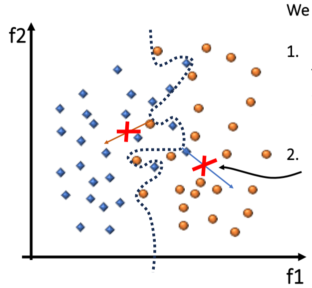
25



We can stop this proces in two ways:

1. By iteratively adding random noise to weights – not allowing the algorithm to actually memorize data samples

26

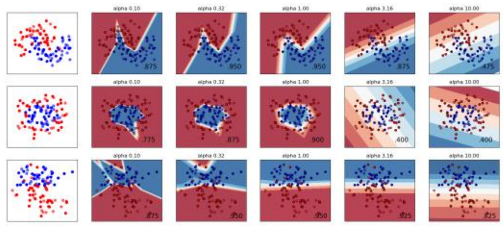


We can stop this proces in two ways:

1. By iteratively adding random noise to weights – not allowing the algorithm to actually memorize data samples
2. By penalizing high weights (indirectly: not allowing high model complexity)

27

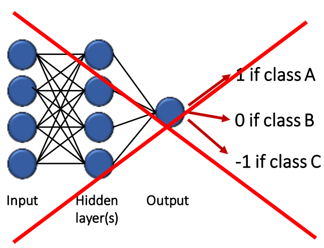
ANN training: **Regularization**



Source: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron

28

Classifier outputs



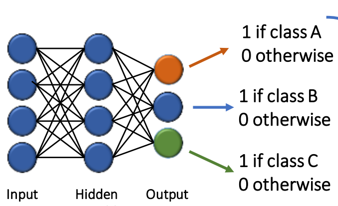
This tells the classifier, that class A is in relation to class C such that A is reversed C

Causes problems in training

Note 1: this example uses a neural network, but the reasoning is valid for all typical classifiers
Note 2: The scheme here is simplified. No biases and no activation functions.

29

Classifier outputs

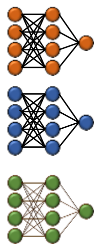


In multiclass classification we can either use multiple outputs from a classifier (each focusing on one class)

Note 1: this example uses a neural network, but the reasoning is valid for all typical classifiers
Note 2: The scheme here is simplified. No biases and no activation functions.

30

Classifier outputs



1 if class A
0 otherwise

1 if class B
0 otherwise

1 if class C
0 otherwise

Or we can use multiple classifiers
(each focusing on one class)

Note 1: this example uses a neural network, but the reasoning is valid for all typical classifiers
Note 2: The scheme here is simplified. No biases and no activation functions.

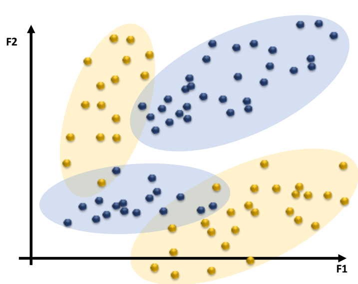
31

(Because they operate based on randomly selected training dataset. Had the dataset been different, the results would also change!)

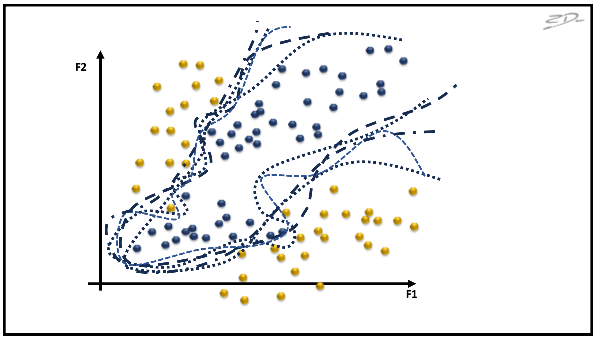
Classifiers and regressors should **always** be treated as non-deterministic.

How to be reliable in such a context?
(How to ensure, that we are not just lucky or unlucky in how data are drawn?)

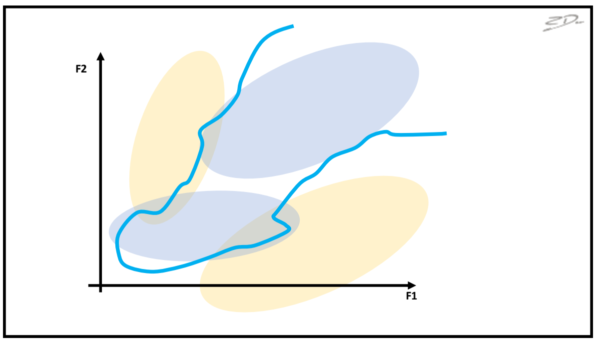
32



33

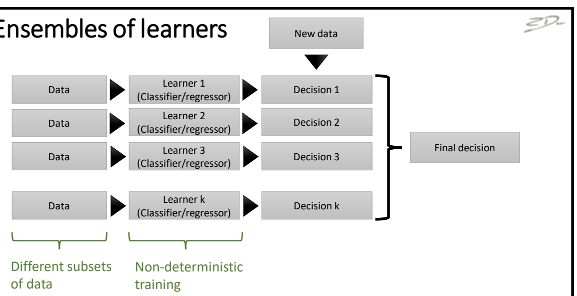


34



35

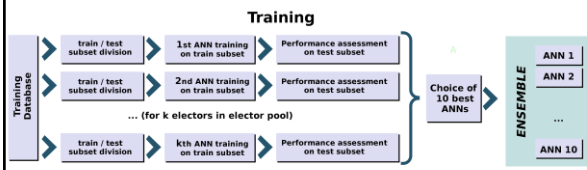
Ensembles of learners



The more **diverse** and **accurate** the learners are, the better

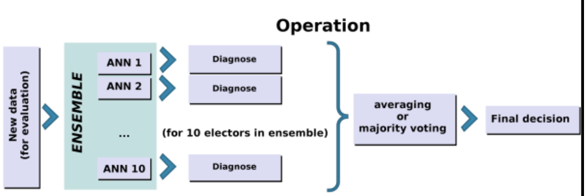
36

Ensembles of learners: Example



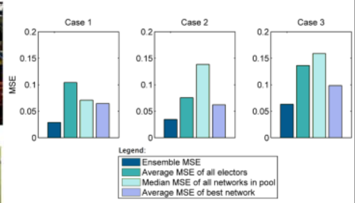
37

Ensembles of learners: Example



38

Ensembles of learners: Example



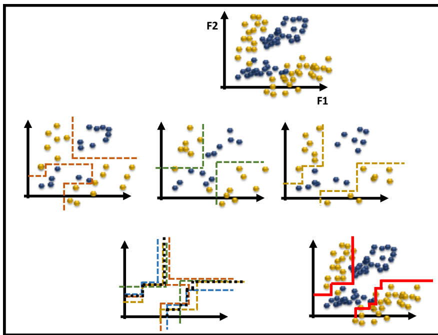
39

Random forest

1. Do **Bagging** (Bootstrap Aggregating) by selecting repeatedly subsets of data using drawing with replacement
2. If dataset consists of many features, sample them also (let different subsets use also subsets of features!)
3. Train simple decision trees based on these subsets (each tree uses different subset)
4. Average responses from many trees

The more **diverse** and **accurate** the learners are, the better

40



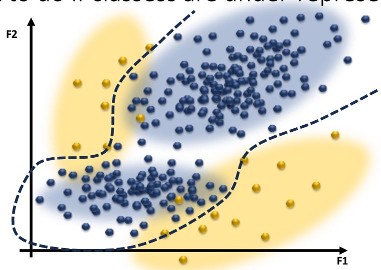
41

Ensembles of learners

In general: Ensembles provide **increased reliability** at the cost of
- **lack of direct control over classification proces**,
- **almost no structural optimization possibilities,** and
- **a lot of required memory and time**

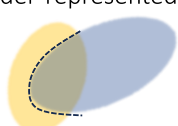
42

What to do if classess are under-represented?



43

What to do if classess are under-represented?



However, we are rarely interested in raw average accuracy (which would be minimized in such a scenario). Usually, in-class accuracy can be just as important

Imagine a cancer-detection system. Assuming that just because cancer is rarely seen it is not important to detect it - would be wrong

44

What to do if classess are under-represented?

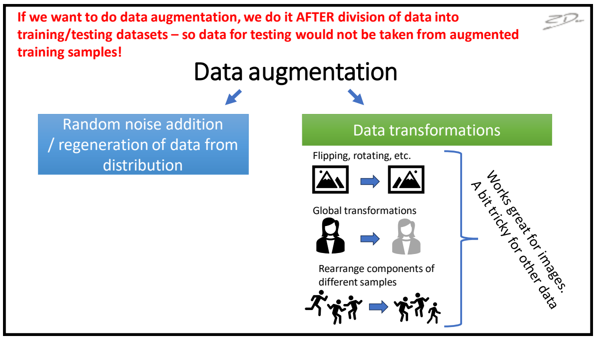
Use classifier that ignores class density – e.g. SVM

Augment dataset: Generate artificial data for low-density classes

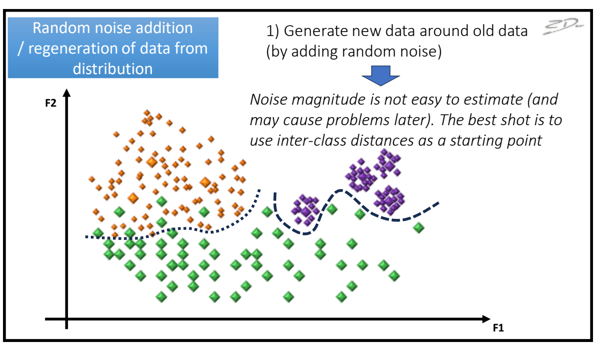
Assign weights to data, low-density classes get more important

This is often used also if ALL the classes are under-represented (if we just don't have enough data)

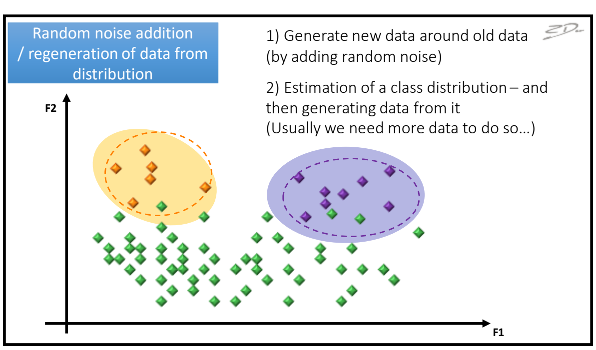
45


If we want to do data augmentation, we do it AFTER division of data into training/testing datasets – so data for testing would not be taken from augmented training samples!
Data augmentation
Random noise addition / regeneration of data from distribution
Data transformations
Flipping, rotating, etc.
Global transformations
Rearrange components of different samples
Works great for images.
A bit tricky for other data

46


Random noise addition / regeneration of data from distribution
1) Generate new data around old data (by adding random noise)
Noise magnitude is not easy to estimate (and may cause problems later). The best shot is to use inter-class distances as a starting point
F2
F1

47


Random noise addition / regeneration of data from distribution
1) Generate new data around old data (by adding random noise)
2) Estimation of a class distribution – and then generating data from it
(Usually we need more data to do so…)
F2
F1

48

Random noise addition / regeneration of data from distribution

1) Generate new data around old data (by adding random noise)

2) Estimation of a class distribution – and then generating data from it
(Usually we need more data to do so…)

If you have enough data to correctly derive its distribution – you usually have also enough data to do classification…

So it works mostly if you know additional context (what distribution can be assumed a priori)

49

One-class classification (novelty detection)

50

One-class classification (novelty detection)

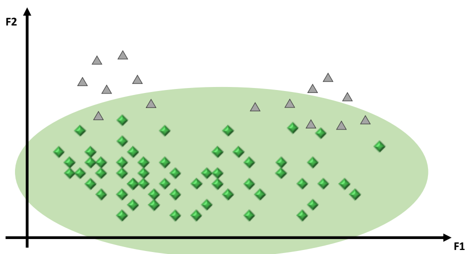
Sometimes we have a dataset with lots of examples belonging to just one class and none or almost none to the other classes:
- „**Healthy** people"
- „**Normal** behavior in subway"
- „**Normal** operation state of an assembly line
- „**Typical** weather conditions in September"

Then, our goal might be to learn this **normal** range of data, to detect any **anomalies (outliers, novelties)**

In Novelty Detection we usually don't know what to expect (there is possibly Infinite set of „norm breaches")

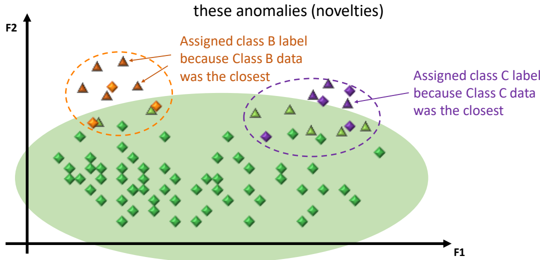
51

1. Learn the normal range
2. Detect what breaches normal range



52

1. Learn the normal range
2. Detect what breaches normal range
3. If you have other class data, use it to classify only these anomalies (novelties)



53

One-class classification (novelty detection)

How to learn „normal range"?

1. Estimate typical distances between samples within training set, detect anything that breaches it

2. Estimate underlying probability density for data – e.g. by fitting multidimensional gaussians into training data

3. Train a regression system to derive some of the features from others – in normal range (for known data) it should work very good, for novel samples it should produce large errors

54

Things to remember*:

1. Graphical interpretation of classification and regression
2. Logistic regression (Idea, graphical interpretation, equation, features)
3. SVM: Principle of opeartion, Soft margin explanation, Kernel idea explanation
4. Graphical interpretation of MLP training (for one neuron)
5. MLP scheme
6. Basic ideas behind standard („vanilla”) gradient descent, ADAM and LM-BFGS
7. Explanation of two regularization methods
8. Configuration of ANNs for multiclass classification
9. Idea behind an ensemble approach for classification and its pros and cons
10. Steps and graphical interpretation of a random forest algorithm
11. Risk of having under-represented classes and three solutions to this problem
12. When can we do data augmentation (in relation to division of data into subsets)?
13. How can we augment data? Explain two methods to this end
14. What is a novelty detection problem? Explain three approaches to solve it

**You don't need to memorize equations if they are not explicitly mentioned in this list!*

55