Machine Learning Exams Collection
University of Padua

by Antonio Tangaro

Last update: February 8, 2025

Contents

Introduction

This document is a collection of Machine Learning exam solutions from the course held between 2018 and 2022. The material is organized chronologically, with each chapter representing a different exam date.

Each exam chapter follows a consistent structure:

- The chapter title indicates the exam date.
- The exam text is presented first, including any figures or tables.
- The solutions are provided in dedicated solution blocks using the `solution` environment.
- Mathematical formulas and equations are formatted using appropriate LaTeX notation.
- Figures and diagrams are included where necessary to support the explanations.

Each solution aims to provide clear explanations of concepts, mathematical derivations where required, and practical implementations when necessary. The document serves as both a reference for exam preparation and a comprehensive review of key machine learning concepts.

Exam 1
Quiz

1.1 Multiple Choice Questions

1. The goal of supervised learning is:

   a) to learn useful features
   b) None of the above
   c) to learn a neural network
   d) to learn a model with low generalization error
   e) to learn a model with low training error

2. If you flip 10 times a coin that has probability 0.25 to give tail, the probability that you obtain 10 heads is:

   a) (1 - 0.25) $\times$ 10
   b) $(1 - 0.25)^{10}$
   c) None of the above
   d) $(0.25)^{10}$
   e) 0.25 $\times$ 10

3. To use the ERM approach means that we learn a model by:

   a) finding the hypothesis with smallest training error
   b) finding the hypothesis with smallest generalization error
   c) finding the hypothesis that minimizes the expected regularization
   d) None of the above
   e) finding the hypothesis with smallest complexity

4. The difference between classification and regression is given by:

   a) the loss function L you can use
   b) the label set Y
   c) None of the above
   d) the type of models you can use (e.g., SVM vs linear models)
   e) the approach used to find the model

5. What does "overfitting" refer to?

   a) Failing to converge during training
   b) None of the above
   c) Learning a model that performs well on training data but poorly on new data
   d) Learning a model that is too simple
   e) Learning a model that has perfect accuracy on all datasets

6. For an hypothesis class H, being PAC learnable with respect to a loss function L, means:

   a) that for some D we can find the best h in H, independently of the amount of data
   b) that for some D we can find the best h in H, with enough data
   c) None of the above
   d) that for all D we can find the best h in H, independently of the amount of data
   e) that for all D we can find the best h in H, with enough data

7. What is the main idea behind the concept of bias-complexity trade-off?

   a) Balancing the trade-off between model simplicity and interpretability
   b) Balancing the trade-off between accuracy and training time
   c) Balancing the trade-off between estimation error and approximation error
   d) Balancing the trade-off between feature selection and feature engineering
   e) None of the above

8. What is the purpose of regularization?

   a) To increase model complexity so to prevent overfitting
   b) None of the above
   c) To reduce model complexity so to prevent overfitting
   d) To eliminate bias in the model
   e) To improve training speed

9. What is the purpose of validation?

    a) To assess the model's performance on the training set
    b) To train the model on multiple datasets
    c) None of the above
    d) To obtain a good estimate of the generalization error
    e) To compare different models on unseen data

10. What is a main difference between SVMs and linear models?

    a) None of the above
    b) SVM can be used to learn models that are polynomial in the features
    c) there is no difference
    d) SVMs can be used only for linearly separable data
    e) SVMs consider the margin of the model, linear models do not

11. The VC dimension is a measure of:

    a) None of the above
    b) the dimension of each point in a dataset
    c) the complexity of an hypothesis class
    d) the number of features in a model
    e) the generalizability of an hypothesis

12. What is the primary goal of clustering?

    a) assigning labels to data points
    b) None of the above
    c) identifying patterns in unlabeled data
    d) predicting a continuous target variable
    e) classifying data into predefined categories

13. Let $X, Y, D, \ell(x, y), \mathcal{H}, h, S$ defined as usual during the course. The definition of training error $L_S(h)$ is:

    a) $L_S(h) = E_{x,y\sim D}[\ell(h(x) - y)^2]$
    b) $L_S(h) = \frac{1}{|S|}\sum_{(x_i,y_i)\in S}(h(x_i) - y_i)^2$
    c) $L_S(h) = E_{x,y\sim D}[\ell(h(x), y)]$
    d) $L_S(h) = \frac{1}{|S|}\sum_{(x_i,y_i)\in S}\ell(h(x_i), y_i)$
    e) none of the above

14. Let $X, Y, D, \ell(x, y), \mathcal{H}, h, S$ defined as usual during the course. The definition of generalization error $L_D(h)$ is:

    a) $L_D(h) = E_{x,y\sim D}[\ell(h(x) - y)^2]$
    b) $L_D(h) = \frac{1}{|S|}\sum_{(x_i,y_i)\in S}(h(x_i) - y_i)^2$
    c) $L_D(h) = E_{x,y\sim D}[\ell(h(x), y)]$
    d) $L_D(h) = \frac{1}{|S|}\sum_{(x_i,y_i)\in S}\ell(h(x_i), y_i)$
    e) none of the above

15. The realizability assumption is defined as:

    a) there exists $h^* \in \mathcal{H}$ with $L_S(h) = 0$
    b) there exists $h^* \in \mathcal{H}$ with $L_D(h) = 0$
    c) ERM finds $h^* \in \mathcal{H}$ such that $L_S(h) = 0$
    d) ERM finds $h^* \in \mathcal{H}$ such that $L_D(h) = 0$
    e) none of the above

1.2 Answer Key

1. (1, d)
2. (2, b)
3. (3, a)
4. (4, b)
5. (5, c)
6. (6, c)
7. (7, c)
8. (8, c)
9. (9, e)
10. (10, e)
11. (11, c)
12. (12, c)
13. (13, d)
14. (14, c)
15. (15, b)

Exam 2
31 January 2023

Exercise 1 [8 points]

Consider the problem of supervised learning.

1. Describe all the components of the formal statistical learning model (domain set, ...) and the general goal of supervised learning.
2. Formally define when a training set is ε-representative.
3. *Provide* and *prove* the upper bound to the generalization error $L_{\mathcal{D}}(h_S)$ of the hypothesis h_S picked by empirical risk minimization when the training set is $\frac{\varepsilon}{2}$-representative (briefly motivating all the steps of the proof).

Exercise 2 [8 points]

1. Describe the stochastic gradient descent (SGD) algorithm (in general).
2. What is the main advantage of SGD with respect to the gradient descent (GD) algorithm?
3. Consider a regression problem where the training data is $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_m, y_m)\}$ with $\mathbf{x}_i = [x_{i,1}, x_{i,2}] \in \mathbb{R}^2$ and $y_i \in \mathbb{R}$ for $i = 1, \dots, m$.

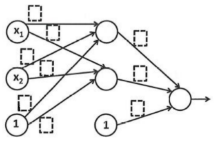
   Assume the hypothesis class $\mathcal{H}$ is given by (simplified) linear models, that is, $\mathcal{H}$ is defined as $\mathcal{H} = \{h_{\mathbf{w}} : h_{\mathbf{w}}(\mathbf{x}) = w_1 x_1 + w_2 x_2, \mathbf{w} = [w_1, w_2] \in \mathbb{R}^2\}$, with $\mathbf{x} = [x_1, x_2]$. Assume that the loss function is $\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = (h_{\mathbf{w}}(\mathbf{x}) - y)^2$, and that SGD is used to learn a model from the training data S.

   Write the SGD update when the hypothesis class $\mathcal{H}$ is as described above.

Exercise 3 [8 points]

Consider neural networks (NNs) for classification with $0-1$ loss.

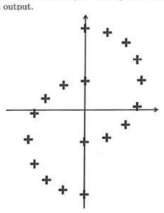
1. Describe what you need to fix to define the hypothesis class of a NN, and what is instead learned from data.
2. Let $\mathbf{x} = [x_1, x_2]$, with $x_1, x_2 \in \{-1, 1\}$. Consider the NN in the figure below, where the activation function for each hidden node and the output node is the *sign* function. Assume that the weights are constrained to be in the set $\{-1, 0, 1\}$. You want your network to represent the function f that is 1 when the input is $[1, 1]$ or $[-1, -1]$ (for all other inputs the function f is -1).
   (a) Find the network's weights so that the network represents the function f described above. *Write the weights in the dashed boxes in the figure.*
   (b) Briefly describe the procedure you used to find the weights.



Exercise 4 [8 points]

Consider the clustering problem.

1. Briefly describe *linkage-based clustering*: what is the input, what is the output, the general algorithm it employs, and a termination condition.
2. Consider *single linkage* clustering. Describe how it is obtained by the general *linkage-based clustering* (no pseudocode needed).
3. Show the output of single linkage clustering when the input is given by the points in $\mathbb{R}^2$ shown as crosses below and the termination condition is given by having the points partitioned in $k = 2$ clusters. (You can draw directly in the figure below.) Briefly describe how the algorithm reaches such output.



2.1 Exercise 1

Consider the problem of supervised learning.

1. Describe all the components of the formal statistical learning model (domain set, ...) and the general goal of supervised learning.

Solution

A supervised learning problem consists of the following elements:

- **Input space** X: The set of possible input instances.
- **Output space** Y: The set of possible labels (discrete for classification, continuous for regression).
- **Probability distribution** D: The unknown distribution over $X \times Y$ that generates data.
- **Hypothesis class** $\mathcal{H}$: A set of functions $h : X \to Y$ representing possible models.
- **Loss function** $\ell(h(x), y)$: A function that quantifies the error of a hypothesis on an individual example.
- **Expected risk** $L_D(h)$: The average loss over the true distribution:

$$L_D(h) = \mathbb{E}_{(x,y)\sim D}[\ell(h(x), y)].$$

- **Empirical risk** $L_S(h)$: The average loss over a training sample $S = \{(x_i, y_i)\}_{i=1}^m$:

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h(x_i), y_i).$$

The goal of supervised learning is to find a hypothesis $h \in \mathcal{H}$ that minimizes the expected risk $L_D(h)$. Since D is unknown, we use empirical risk minimization (ERM) to approximate it.

2. Formally define when a training set is ϵ-representative.

Solution

A training set S is ε-**representative** for a hypothesis class $\mathcal{H}$ if the empirical risk is a good approximation of the expected risk for all hypotheses:

$$\sup_{h \in \mathcal{H}} |L_S(h) - L_D(h)| \leq \varepsilon.$$

This ensures that the empirical risk does not deviate from the expected risk by more than ε, making S a reliable approximation of the true distribution.

3. Provide and prove the upper bound to the generalization error $L_D(h_S)$ of the hypothesis h_S picked by empirical risk minimization when the training set is $\frac{\epsilon}{2}$-representative (briefly motivating all the steps of the proof).

Solution

We prove that if S is $\frac{\varepsilon}{2}$-representative, the generalization error of the empirical risk minimizer h_S satisfies:

$$L_D(h_S) \leq L_D(h^*) + \varepsilon,$$

where $h^* = \arg\min_{h \in \mathcal{H}} L_D(h)$ is the optimal hypothesis.

Proof:

(a) By assumption, S is $\frac{\varepsilon}{2}$-representative, meaning:

$$\sup_{h \in \mathcal{H}} |L_S(h) - L_D(h)| \leq \frac{\varepsilon}{2}.$$

(b) Since h_S is chosen by ERM, it satisfies:

$$L_S(h_S) \leq L_S(h^*).$$

(c) Applying the $\frac{\varepsilon}{2}$-representative property:

$$L_D(h_S) \leq L_S(h_S) + \frac{\varepsilon}{2},$$

$$L_D(h^*) \leq L_S(h^*) + \frac{\varepsilon}{2}.$$

(d) Substituting $L_S(h_S) \leq L_S(h^*)$ into the first inequality:

$$L_D(h_S) \leq L_S(h^*) + \frac{\varepsilon}{2}.$$

(e) Using $L_S(h^*) \leq L_D(h^*) + \frac{\varepsilon}{2}$, we obtain:

$$L_D(h_S) \leq L_D(h^*) + \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = L_D(h^*) + \varepsilon.$$

Thus, the generalization error of h_S is at most ε worse than the best possible hypothesis in $\mathcal{H}$.

2.2 Exercise 2

1. Describe the stochastic gradient descent (SGD) algorithm (in general).

Solution

Stochastic Gradient Descent (SGD) is an optimization algorithm designed to minimize an objective function, typically a loss function, by iteratively updating the model parameters using a single randomly chosen data point per iteration.

The general update rule for SGD is:

$$w^{(t+1)} = w^{(t)} - \eta \nabla_w \ell(w; x_i, y_i)$$

where:

- $w^{(t)}$ represents the model parameters at iteration t.
- η is the **learning rate**, which controls the step size of the update.
- $\nabla_w \ell(w; x_i, y_i)$ is the **gradient** of the loss function with respect to the parameters, computed using only a single example (x_i, y_i).

Unlike **Batch Gradient Descent (GD)**, which computes the gradient using the entire dataset at each iteration, SGD updates the model parameters using only **one sample at a time**, making it more efficient for large-scale problems.

2. What is the main advantage of SGD with respect to the gradient descent (GD) algorithm?

Solution

SGD has several advantages over Gradient Descent (GD):

(a) **Computational efficiency:**

- GD computes the gradient over the entire dataset, making it inefficient for large datasets.
- SGD updates the parameters using only one sample per iteration, significantly reducing computational cost.

(b) **Faster convergence for large datasets:**

- SGD performs more frequent updates, allowing faster convergence compared to GD in large-scale scenarios.
- GD requires processing all samples before each update, slowing down the process.

(c) **Better for non-convex optimization:**

- The randomness in SGD updates introduces noise, preventing the risk of getting stuck in poor-quality local minima.
- GD, being deterministic, is more likely to converge to a local minimum rather than a global one.

(d) **Online learning capability:**

- SGD is suitable for **online learning**, where data arrives sequentially and must be processed immediately.
- GD requires the entire dataset to be available before starting training, making it less suitable for real-time applications.

Consequently, **SGD is preferred for large datasets and online learning due to its computational efficiency and ability to escape poor local minima**.

3. Consider a regression problem where the training data is $S = \{(x_i, y_i), (x_2, y_2), \ldots, (x_m, y_m)\}$ with $x_i = [x_{i,1}, x_{i,2}] \in \mathbb{R}^2$ and $y_i \in \mathbb{R}$ for $i = 1, \ldots, m$. Assume the hypothesis class $\mathcal{H}$ is given by (simplified) linear models, that is, $\mathcal{H}$ is defined as $\mathcal{H} = \{h_w(x) : h_w(x) = w_1x_1 + w_2x_2, w = [w_1, w_2] \in \mathbb{R}^2\}$,

with $x = [x_1, x_2]$. Assume that the loss function is $\ell(h_w(x), y) = (h_w(x) - y)^2$, and that SGD is used to learn a model from the training data S.
Write the SGD update when the hypothesis class $\mathcal{H}$ is as described above.

Solution

Consider a **linear regression problem** with a training dataset:

$$S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$$

where:

- Each input x_i is a two-dimensional vector $x_i = [x_{i,1}, x_{i,2}] \in \mathbb{R}^2$.
- Each output y_i is a real number $y_i \in \mathbb{R}$.

The hypothesis class consists of linear models:

$$h_w(x) = w_1 x_1 + w_2 x_2$$

where the parameter vector is $w = [w_1, w_2] \in \mathbb{R}^2$.
The loss function used is the **squared loss**:

$$\ell(h_w(x), y) = (h_w(x) - y)^2.$$

To compute the SGD update, we first derive the gradient of the loss function with respect to w:

$$\nabla_w \ell(h_w(x_i), y_i) = 2(h_w(x_i) - y_i)\nabla_w h_w(x_i).$$

Since $h_w(x_i) = w_1 x_{i,1} + w_2 x_{i,2}$, the partial derivatives are:

$$\frac{\partial h_w}{\partial w_1} = x_{i,1}, \quad \frac{\partial h_w}{\partial w_2} = x_{i,2}.$$

Thus, the gradient is:

$$\nabla_w \ell(h_w(x_i), y_i) = 2(h_w(x_i) - y_i)[x_{i,1}, x_{i,2}].$$

Applying the SGD update rule:

$$w_j^{(t+1)} = w_j^{(t)} - \eta \frac{\partial \ell}{\partial w_j}, \quad \text{for } j = 1, 2.$$

Substituting the gradient:

$$w_1^{(t+1)} = w_1^{(t)} - \eta \cdot 2(h_w(x_i) - y_i)x_{i,1},$$

$$w_2^{(t+1)} = w_2^{(t)} - \eta \cdot 2(h_w(x_i) - y_i)x_{i,2}.$$

Therefore, for each randomly selected training point (x_i, y_i), the weight update follows:

$$w^{(t+1)} = w^{(t)} - 2\eta(h_w(x_i) - y_i)x_i.$$

where x_i is the feature vector $[x_{i,1}, x_{i,2}]$.

2.3 Exercise 3

Consider neural networks (NNs) for classification with 0-1 loss.

1. Describe what you need to fix to define the hypothesis class of a NN, and what is instead learned from data.

Solution

To define the **hypothesis class** of a neural network, we must fix:

- The number of **hidden layers** and **neurons per layer**.
- The **activation function** for each layer, in this case, the **sign function**.
- The **weight constraints**, here limited to $\{-1, 0, 1\}$.
- The **connectivity structure**, which defines how neurons are linked.

Instead, what is **learned from data** during training includes:

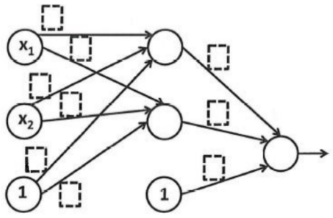
- The **weights** connecting neurons.
- The **bias terms**, which allow neurons to shift activation thresholds.
- A function that approximates the **decision boundary** by minimizing a loss function.

Once these elements are set, the neural network learns an optimal function by adjusting its weights through optimization algorithms.

2. Let $x = [x_1, x_2]$, with $x_1, x_2 \in \{-1, 1\}$. Consider the NN in the figure below, where the activation function for each hidden node and the output node is the *sign* function. Assume that the weights are constrained to be in the set $\{-1, 0, 1\}$. You want your network to represent the function f that is 1 when the input is $[1, 1]$ or $[-1, -1]$ (for all other inputs the function is -1).

   (a) Find the network's weights so that the network represents the function f described above. Write the weights in the dashed boxes in the figure.

   (b) Briefly describe the procedure you used to find the weights.



Solution

The goal is to design a neural network that implements the function:

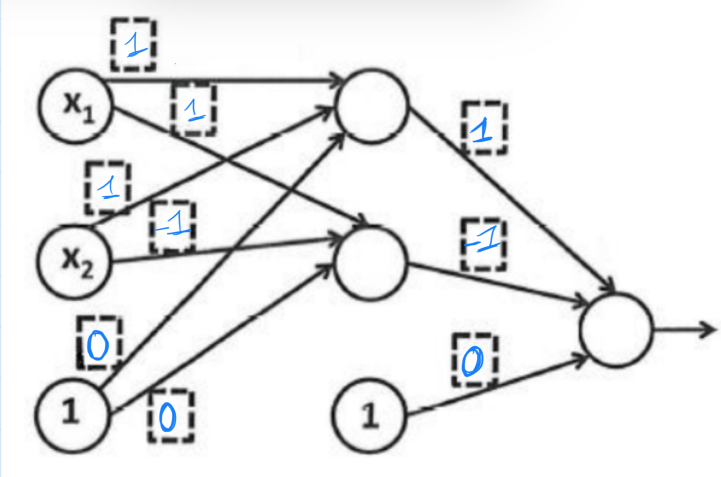
$$f(x_1, x_2) = 1 \quad \text{if} \quad (x_1, x_2) = (1, 1) \text{ or } (-1, -1), \quad \text{else} \quad f(x) = -1.$$

Since the activation function is the **sign function**, we need to construct a **two-layer network** that correctly classifies these inputs.

(a) Choosing the Weights

The required weights are assigned as follows:

- **Hidden Layer Weights:**
  - One neuron detects when $x_1 = x_2$ by computing $x_1 + x_2$.
  - Another neuron detects when $x_1 \neq x_2$ using $x_1 - x_2$.
- **Output Layer Weights:**
  - The output neuron applies the **sign function** to distinguish the correct class.



This weight assignment ensures that:

- When $x_1 = x_2$, the first hidden neuron is active.
- When $x_1 \neq x_2$, the second hidden neuron is active.
- The output neuron sums these signals and applies the **sign function**, correctly classifying the input.

(b) Procedure for Finding the Weights

The method follows a logical breakdown:

(a) **Analyze the function**: The network must recognize cases where $x_1 = x_2$.

(b) **Design the hidden neurons**: One neuron detects equality, another detects inequality.

(c) **Assign weights**: Set values to enforce correct neuron activation.

(d) **Validate output behavior**: Ensure the final neuron applies the correct decision boundary.

This systematic approach guarantees correct classification while respecting the constraints on weights.

2.4 Exercise 4

Consider the clustering problem.

1. Briefly describe linkage-based clustering: what is the input, what is the output, the general algorithm it employs, and a termination condition.

Solution

Linkage-based clustering is a hierarchical clustering method used to partition a dataset into clusters based on a **distance function**.

Input:

- A set X of objects.
- A **distance function** $d : X \times X \to \mathbb{R}^+$, which must satisfy properties like symmetry, identity of indiscernibles, and the triangle inequality.

Output: A **dendrogram**, representing how clusters merge at different distance thresholds.

Algorithm:

(a) **Initialization:** Each data point starts as its own cluster.

(b) **Iterative Merging:** The two closest clusters are merged at each step based on a chosen **linkage criterion**.

(c) **Termination Condition:** The process stops when a predefined stopping criterion is met.

Common termination conditions:

- **Fixed number of clusters:** The process stops when k clusters remain.
- **Distance threshold:** Clusters are not merged if the minimum distance between them exceeds a threshold r.
- **Complete merging:** The process continues until all points belong to a single cluster, producing a dendrogram.

2. Consider single linkage clustering. Describe how it is obtained by the general linkage-based clustering (no pseudocode needed).

Solution

Single linkage clustering is a specific type of linkage-based clustering where the distance between two clusters A and B is defined as:

$$D(A, B) = \min\{d(x, x') \mid x \in A, x' \in B\}.$$

This means that at each step, the algorithm merges the two clusters whose **closest points** have the smallest distance.

Characteristics:

- **Chaining effect:** The algorithm tends to create elongated clusters rather than compact ones.
- **Hierarchical approach:** The process follows the general linkage-based clustering structure but differs in how it measures distances between clusters.

At each iteration, the algorithm identifies and merges the two clusters with the smallest minimum inter-cluster distance. This process continues until a stopping condition is met.

3. Show the output of single linkage clustering when the input is given by the points in $\mathbb{R}^2$ shown as crosses below and the termination condition is given by having the points partitioned in $k = 2$ clusters. (You can draw directly in the figure below.) Briefly describe how the algorithm reaches such output.

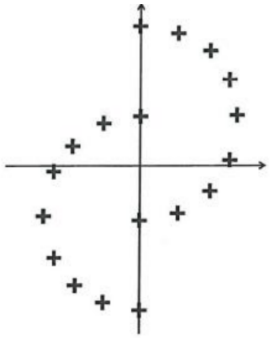
Solution

The dataset consists of points in $\mathbb{R}^2$ arranged in a curved pattern. The task is to apply **single linkage clustering** and stop when **two clusters remain** ($k = 2$).

Algorithm Execution:

- Each point starts as its own cluster.
- The algorithm finds and merges the two closest clusters at each iteration.
- Due to the **chaining effect**, clusters gradually extend along the data distribution.
- The process stops when only **two clusters remain**.

Final Clustering Output: The final partition consists of **two clusters**, corresponding to the **upper and lower sets of points** in the figure. These clusters reflect the chaining effect of single linkage, where points progressively merge based on their closest neighbor.



Exam 3
16 February 2022

Exercise 1 [8 points]

Consider the task of binary classification with 0-1 loss, domain set $\mathcal{X} = \mathbb{R}^d$, and hypothesis class $\mathcal{H} = \{h_{\mathbf{w}}(\mathbf{x}) : h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(< \mathbf{w}, \mathbf{x} >)\}$.

1. Clearly describe the perceptron algorithm, and provide a brief intuition for its update rule.
2. With respect to the context above, what is is a tight value ℓ such $VCdim(\mathcal{H}) \geq \ell$? Prove that $VCdim(\mathcal{H}) \geq \ell$ (for your choice of ℓ).

Exercise 2 [8 points]

1. Describe the stochastic gradient descent (SGD) algorithm (in general). What is the main advantage of SGD with respect to the gradient descent algorithm?
2. Consider the regression problem, with training data $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_m, y_m)\}$ with $\mathbf{x}_i = [x_{i,1}, x_{i,2}] \in \mathbb{R}^2$ and $y_i \in \mathbb{R}$ for $i = 1, \ldots, m$.

   Assume the hypothesis class $\mathcal{H}$ is given by the simple neural network with the architecture below, where the edge weights are w_1 and w_2.

   Assume the activation function for the output node is $\sigma(z) = z$ (the final prediction is the output of the output node), and the loss function is $\ell(h, (\mathbf{x}, y)) = (h(\mathbf{x}) - y)^2$.

   Write the SGD update when the hypothesis class $\mathcal{H}$ is as specified above.

Exercise 3 [8 points]

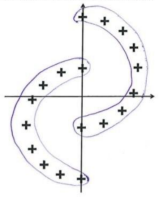
Consider the regression problem with squared loss.

1. Formally define the label set $\mathcal{Y}$, the loss function, the training error, and the generalization error.
2. Consider the setting above with the addition of Tikhonov regularization. Describe the framework of regularized loss minimization (with Tikhonov regularization) and the relation of the parameter λ with the decomposition of the generalization error in approximation error and estimation error.
3. Assume the hypothesis class is given by linear models and Tikhonov regularization is used. *Derive* the formula for the best model in this case.

Exercise 4 [8 points]

Consider the clustering problem.

1. Briefly describe *linkage-based clustering* (what is the input, what is the output, the general algorithm it employs, and a termination condition).
2. Consider *single linkage* clustering. Describe how it is obtained by the general *linkage-based clustering* (no pseudocode needed).
3. Show the output of single linkage clustering when the input is given by the points in $\mathbb{R}^2$ shown as crosses below and the termination condition is given by having the points partitioned in $k = 2$ clusters. (You can draw directly in the figure below.) Briefly describe how the algorithm reaches such output.



3.1 Exercise 1

Consider the task of binary classification with 0-1 loss, domain set $\mathcal{X} = \mathbb{R}^d$, and hypothesis class $\mathcal{H} = \{h_w(x) : h_w(x) = sign(< w, x >)\}$.

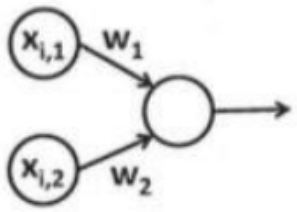
1. Clearly describe the perceptron algorithm, and provide a brief intuition for its update rule.

2. With respect to the context above, what is is a tight value ℓ such $VCdim(\mathcal{H}) \geq \ell$? Prove that $VCdim(\mathcal{H}) \geq \ell$ (for your choice of ℓ).

3.2 Exercise 2

1. Describe the stochastic gradient descent (SGD) algorithm (in general). What is the main advantage of SGD with respect to the gradient descent algorithm?

2. Consider the regression problem with training data $S = \{(x_i, y_i), (x_2, y_2), \ldots, (x_m, y_m)\}$ with $x_i = [x_{i,1}, x_{i,2}] \in \mathbb{R}^2$ and $y_i \in \mathbb{R}$ for $i = 1, \ldots, m$. Assume the hypothesis class $\mathcal{H}$ is given by the simple neural network with the architecture below, where the edge weights are w_1 and w_2.



   Assume the activation function for the output node is $\sigma(z) = z$ (the final prediction is the output of the output node), and the loss function is $\ell(h(x), y) = (h(x) - y)^2$. Write the SGD update when the hypothesis class $\mathcal{H}$ is as specified above.

3.3 Exercise 3

Consider a regression problem with squared loss.

1. Formally define the label set Y, the loss function, the training error, and the generalization error.

2. Consider the setting above with the addition of Tikhonov regularization. Describe the framework of regularized loss minimization (with Tikhonov regularization) and the relation of the parameter λ with the

decomposition of the generalization error in approximation error and estimation error.

3. Assume the hypothesis class is given by linear models and Tikhonov regularization is used. Derive the formula for the best model in this case.

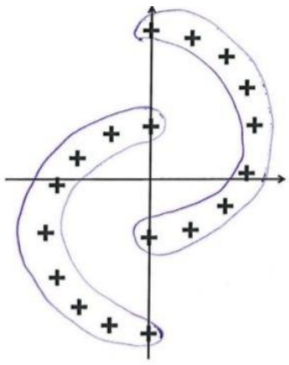
3.4 Exercise 4

Consider a clustering problem

1. Briefly describe linkage-based clustering (what is the input, what is the output, the general algorithm it employs, and a termination condition).

2. Consider single linkage clustering. Describe how it is obtained by the general linkage-based clustering (no pseudocode needed).

3. Show the output of single linkage clustering when the input is given by the points in $\mathbb{R}^2$ as shown in the figure below and the termination condition is given by having the points partitioned in $k = 2$ clusters. (You can draw directly in the figure below.) Briefly describe how the algorithm reaches such output.



Exam 4
1 February 2022

Exercise 1 [8 points; max 2 pages.]

Consider the problem of supervised learning.

1. Describe all the components of the (more general) learning model and the goal of supervised learning.
2. Provide the definition of a training set being ε-representative.
3. *Provide* and *prove* the upper bound to the generalization error of the hypothesis picked by empirical risk minimization when the training set is $\frac{\varepsilon}{2}$-representative (briefly motivating all the steps of the proof).

Exercise 2 [8 points; max 2 pages.]

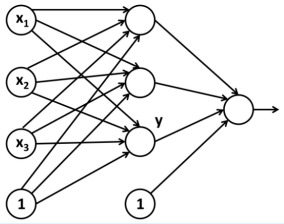
Consider the regression problem with linear models and squared loss where an instance $\mathbf{x} = [x_1, x_2, x_3]$ contains 3 real features (i.e., $\mathcal{X} = \mathbb{R}^3$).

1. Derive the ERM hypothesis.
2. Assume that, given the application domain, you know that $(x_1)^4$, $(x_2)^2$, and $(x_3)^5$ would be the best features for a polynomial model. Describe how you can learn a polynomial model built on such features.

Exercise 3 [8 points; max 2 pages.]

Consider neural networks (NNs) for classification with $0-1$ loss.

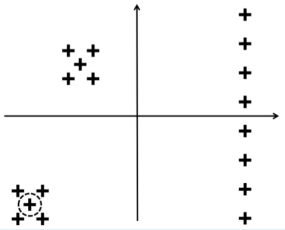
1. Describe what you need to fix to define the hypothesis class of a NN.
2. Assume that you use a NN for a problem, and at the end you obtain a model with very low training error but very high test error. You cannot obtain more data. How would you proceed to try to learn a better model?
3. Let $\mathbf{x} = [x_1, x_2, x_3] \in \{-1, 1\}$. Consider the NN in the figure below, where the activation function for each hidden node and the output node is the *sign* function. Assume that the weights are constrained to be in the set $\{-2, -1, 0, 1, 2\}$. You want your network to represent the function that takes value 1 when the input is $[1, -1, 1]$ or $[-1, 1, 1]$ or $[1, 1, 1]$ (for all other inputs the function takes value -1).
   (a) Find the network's weights so that the training error is 0.
   (b) Briefly describe how you have chosen the weights.



Exercise 4 [8 points; max 2 pages.]

Consider the k-means clustering problem.

1. Define its cost function.
2. Describe Lloyd's algorithm.
3. Consider the points in $\mathbb{R}^2$ shown as crosses below. Assume that the `k-means++` algorithm is used to initialize the centers for Lloyd's algorithm with $k = 4$, and that the first center chosen is the point circled in the figure. Show the *most likely other centers* chosen by `k-means++`, the order in which they are chosen (providing a short motivation for your answer) and the output of Lloyd's algorithm with such initialization of the centers.



4.1 Exercise 1

Consider the problem of supervised learning.

1. Describe all the components of the (more general) learning model and the goal of supervised learning.

Solution

In supervised learning we have:

- An instance set X
- Label set Y
- Given a set $S = \{(\vec{x}_1, y_1), ..., (\vec{x}_m, y_m)\}$ of m labeled samples

Our goal is we have to choose an hypothesis class H of functions $h : X \to Y$ and a loss function $l : H \times Z \to \mathbb{R}^+$, where $Z = X \times Y$, that tells us how much we lose by predicting $h(x)$ instead of the correct label y_i.

The goal is to find an hypothesis $h \in H$ of low generalization error $L_D(h) = E_{z\sim D}[l(h, z)]$ where $z = (x, y)$ is sampled by an unknown distrib. D.

2. Provide the definition of a training set being ϵ-representative.

Solution

S is ε-repr. if $\forall h \in H : |L_D(h) - L_S(h)| \leq \varepsilon$

3. Provide and prove the upper bound to the generalization error of the hypothesis picked by empirical risk minimization when the training set is $\frac{\epsilon}{2}$-representative (briefly motivating all the steps of the proof).

Solution

If S is $\varepsilon/2$-repr. then $L_D(h_S) \leq \min_{h\in H} L_D(h) + \varepsilon$

Proof:

$$\begin{aligned} \forall h \in H \quad L_D(h_S) &\leq L_S(h_S) + \varepsilon/2 \quad [\varepsilon/2\text{-repr.}] \\ &\leq L_S(h) + \varepsilon/2 \quad [\text{ERM } (L_S(h_S) \leq L_S(h))] \\ &\leq L_D(h) + \varepsilon/2 + \varepsilon/2 \quad [\varepsilon/2\text{-repr.}] \\ &\leq L_D(h) + \varepsilon \end{aligned}$$

This is also true for $\arg\min_{h \in H} L_D(h)$
So $L_D(h_S) \leq \min_{h \in H} L_D(h) + \varepsilon$

4.2 Exercise 2

Consider the regression problem with linear models and squared loss where an instance $\mathbf{x} = [x_1, x_2, x_3]$ contains 3 real features (i.e., $\mathbf{X} = \mathbb{R}^3$).

1. Derive the ERM hypothesis.

Solution

$l(h, (x_i, y_i)) = (y_i - h(x_i))^2 = (y_i - \langle \vec{w}, \vec{x}_i \rangle)^2$
$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \langle \vec{w}, \vec{x}_i \rangle)^2 = (y_i - X\vec{w})^T (y_i - X\vec{w})$
We have to minimize $L_S(h)$
We can do it by picking the derivative = 0:

$$\begin{aligned} \frac{\partial L_S(h)}{\partial \vec{w}} = 0 &\Leftrightarrow -X^T(y_i - X\vec{w}) - X^T(y_i - X\vec{w}) = 0 \\ &\Leftrightarrow -2X^T(y_i - X\vec{w}) = 0 \\ &\Leftrightarrow X^T y_i - X^T X\vec{w} = 0 \\ &\Leftrightarrow \vec{w} = (X^T X)^{-1} X^T y_i \quad \text{[it's invertible]} \end{aligned}$$

2. Assume that, given the application domain, you know that $(x_1)^2$, $(x_2)^2$, and $(x_3)^2$ would be the best features for a polynomial model. Describe how you can learn a polynomial model built on such features.

Solution

We can use feature expansion: we apply a transformation to each instance of the dataset: $x = [1, x_1, ..., x_1^2, ..., x_2, ..., x_3, ..., x_n]$. Then learn a linear model on this transformed dataset.

4.3 Exercise 3

Consider neural networks (NNs) for classification with 0 - 1 loss.

1. Describe what you need to fix to define the hypothesis class of a NN.

Solution

We need to define:

- The number of hidden layers
- How many neurons we have in each layer
- The activation function we want to use: $\sigma = \{\text{sigmoid}, \tanh, ...\}$

The NN hypothesis class is $H_{(V,E,\sigma)}$

2. Assume that you use a NN for a problem, and at the end you obtain a model with very low training error but very high test error. You cannot obtain more data. How would you proceed to try to learn a better model?

Solution

Low training error and high test error means we are overfitting.

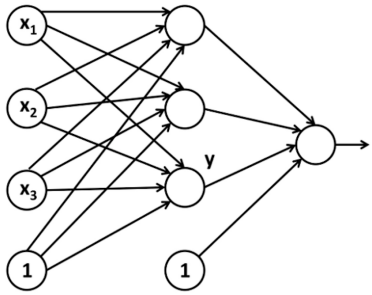
We can use several approaches:

- Probably the model is too complex so we can reduce the number of layers or number of neurons.
- We can use regularization that adds to the loss a term $\lambda\|\vec{w}\|^2$ that takes into account the model complexity and controls it with hyper parameter λ
- We can use dropout that at each iteration of the SGD ignores some neurons.
- Since we have a small amount of data, we can use cross-fold validation that allows us to train on several sub-sets of S and test with 1 fold.

3. Let $x = [x_1, x_2, x_3] \in -1, 1$. Consider the NN in the figure below, where the activation function for each hidden node and the output node is the sign function. Assume that the weights are constrained to be in the set $-2, -1, 0, 1, 2$. You want your network to represent the function that takes value 1 when the input is $[1, -1, 1]$ or $[-1, 1, 1]$ or $[1, 1, 1]$ (for all other inputs the function takes value -1).

   (a) Find the network's weights so that the training error is 0.

   (b) Briefly describe how you have chosen the weights.



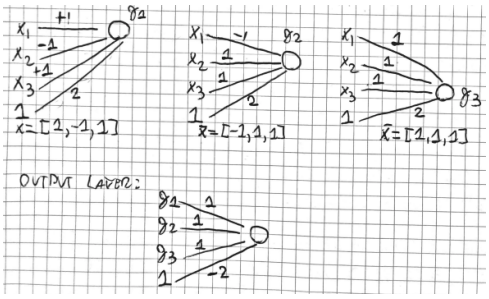
Solution

In class we have seen that a function $f : \{-1, 1\}^d \to \{-1, 1\}$ can be implemented by the network in the figure.

- The weights between the last 2 layers are all 1 except for the bias that is $1 - k = -2$
- Each neuron in the hidden layer implements a function:

$$y_i = sgn(\langle w, x \rangle)$$

  For each input that has as output 1 for a total of $k = 3$ hidden neurons. Where its weights are the value of such instance except for the bias that is equal to: $d - 1 = 2$



4.4 Exercise 4

Consider the k-means clustering problem.

1. Define its cost function.

Solution

k-means cost function:

$$\sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^2$$

where μ_i with $i \in [1, ..., k]$ are centroids

2. Describe Lloyd's algorithm.

Solution

Lloyd algo.: $O(tkdm)$ time complexity
where $k =$ num. clusters, $m =$ num. samples
$d =$ space dimension, $t =$ num. iterations

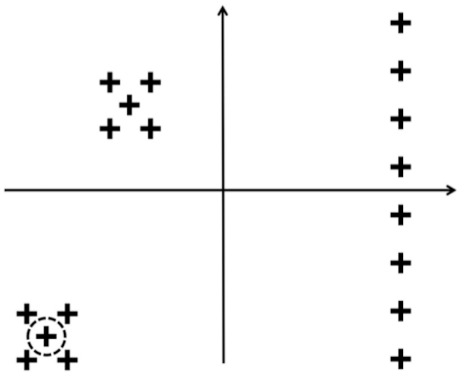
Input: $X = (x_1, ..., x_m)$ with $x_i \in \mathbb{R}^d$, $k \in \mathbb{N}^+$
Output: $\{\vec{\mu}_1, ..., \vec{\mu}_k\}$ centroids, $C = \{C_1, ..., C_k\}$ clustering of X

1: Initialize $\vec{\mu}_1^{(0)}, ..., \vec{\mu}_k^{(0)}$ at random ▷ Assign instances
2: For $t = 0, ...$ till convergence: ▷ to the nearest cluster
3: **for** $i = 1, ..., k$ **do**
4: $C_i \leftarrow \{x \in X : i = \arg\min_j d(x, \mu_j)\}$
5: **end for**
6: **for** $i = 1, ..., k$ **do**
7: $\vec{\mu}_i^{(t+1)} \leftarrow \frac{1}{|C_i|} \sum_{x \in C_i} x$ ▷ Update centroids
8: **end for**
9: **if** convergence reached **then**
10: Return $\vec{\mu}_1^{(t+1)}, ..., \vec{\mu}_k^{(t+1)}$ and $C = \{C_1, ..., C_k\}$
11: **end if**

Common convergence conditions:

- $\sum_{i=1}^{k} d(\vec{\mu}_i^{(t)}, \vec{\mu}_i^{(t+1)}) \leq \varepsilon$
- $\max_{i \in [1,...,k]} \{d(\vec{\mu}_i^{(t)}, \vec{\mu}_i^{(t+1)})\} \leq \varepsilon$
- Cost function on current iteration is not lower than the previous iteration

3. Consider the points in $\mathbb{R}^2$ shown as crosses below. Assume that the $k-means++$ algorithm is used to initialize the centers for Lloyd's algorithm with k = 4, and that the first center chosen is the point circled in the figure. Show the most likely other centers chosen by $k-means++$ (providing a short motivation for your answer) and the output of Lloyd's algorithm with such initialization of the centers.



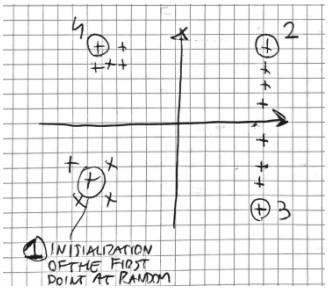
Solution

k-means++:

1: $\mu_1 \leftarrow$ random $x \in X$
2: $F = \{\mu_1\}$
3: **for** $i = 2, ..., k$ **do**
4: $\quad \mu_i \leftarrow \arg\max_{x \in X \setminus F} d(x, F)^2$ ▷ Distance from x to each point in F
5: $\quad F \leftarrow F \cup \{\mu_i\}$
6: **end for**
7: Return F

At each iteration picks the point that is farthest from the already chosen centroids (quadratic).

So the order in which k-means++ chose the centroids is:



Exam 5
7 September 2021

Exercise 1 [8 points; max 2 pages.]

Consider the *binary* classification problem with 0-1 loss.

1. Provide a formal definition of the problem, describing the data, the learner's input, the learner's output, the loss function, the assumed generative model for the data, the learner's goal, and what choices the learner has to make when trying to solve the problem.
2. Assume the hypothesis class is $\mathcal{H}$, the data generative distribution is $\mathcal{D}$, and the training data is S. Provide the definition of the training error $L_S(h)$ and generalization error $L_{\mathcal{D}}(h)$, $h \in \mathcal{H}$, and prove that the expectation (over the distribution of the training set) of $L_S(h)$ is equal to $L_{\mathcal{D}}(h)$.
3. Use the result above to argue that the ERM procedure can be appropriate when the training data is large enough.

Exercise 2 [8 points; max 2 pages.]

Consider the *regression* problem with squared loss.

1. Provide a formal definition of the problem, describing the data, the learner's input, the learner's output, the loss function, the assumed generative model for the data, the learner's goal, and what choices the learner has to make when trying to solve the problem.
2. Within the context above, define the coefficient of determination R^2, provide an intuition for what it is capturing, and describe why it makes sense to look at R^2 instead of considering the average loss.
3. Consider the problem of feature selection in the framework described above. Provide a brief motivation for feature selection, and describe one procedure to select a model with at most k features, where $k << d$.

Exercise 3 [8 points; max 2 pages.]

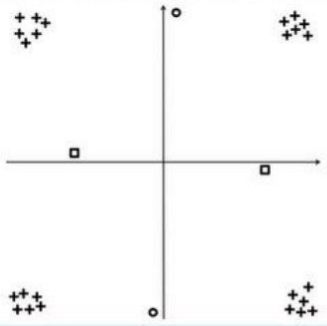
Consider a binary classification problem with 0-1 loss.

1. Provide the definition of VC dimension $VCdim(\mathcal{H})$ of a hypothesis set $\mathcal{H}$. What is the relation between the empirical error and the true risk in terms of the VC dimension of $\mathcal{H}$?
2. Assume that you have n hypothesis sets, denoted by $\mathcal{H}_i, i = 1, 2, \ldots, n$. Describe one strategy to choose a good hypothesis set $\mathcal{H}_i$ and a good model $h_i \in \mathcal{H}_i$.

Exercise 4 [8 points; max 2 pages.]

1. Briefly introduce the clustering problem.
2. Define and explain the cost function used in K-means clustering.
3. Consider the data in the figure below where each point $\mathbf{x} \in \mathbb{R}^2$ is represented by a cross. Show the results (i.e., draw approximately the final centroid locations and the final assignment of the points to the clusters) of clustering into $k = 2$ clusters with K-means when
   (a) the initial centers for the algorithm are the circles;
   (b) the initial centers for the algorithm are the squares.

   Is one solution *significantly* better than the other one? Briefly motivate your answer.



5.1 Exercise 1

Consider the binary classification problem with 0-1 loss.

1. Provide a formal definition of the problem, describing the data, the learner's input, the learner's output, the loss function, the assumed generative model for the data, the learner's goal, and what choices the learner has to make when trying to solve the problem.

Solution

The binary classification problem with 0-1 loss is a supervised learning problem with:

- Domain set X which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Label set $Y = \{-1, +1\}$, that defines the set of all possible labels in this case the labels are only 2

Given a training set $S = ((\mathbf{x}_1, y_1)...(\mathbf{x}_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ we need to choose an hypothesis class H, which defines the possible models or classification rules, from which we can pick our model to make predictions, and a loss function $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y. In particular in the case of 0-1 loss the function is exactly the following:

$$l(h, (\vec{x}, y)) = \begin{cases} 1 \text{ if } h(\vec{x}) \neq y \\ 0 \text{ if } h(\vec{x}) = y \end{cases}$$

The goal is to find an hypothesis $\hat{h} \in H$ with low generalization error: $L_d(\hat{h}) = E_{z \sim D}[l(\hat{h}, z)]$ where:

- D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples).

2. Assume the hypothesis class is $\mathcal{H}$, the data generative distribution is $\mathcal{D}$, and the training data is $\mathcal{S}$. Provide the definition of the training error $L_S(h)$ and generalization error $L_{\mathcal{D}}(h)$, $h \in \mathcal{H}$, and prove that the expectation (over the distribution of the training set) of $L_S(h)$ is equal to $L_{\mathcal{D}}(h)$.

Solution

Let:

- X be the domain set, which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Y be the label set that defines the set of all possible labels
- $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ with $\vec{x}_i \in X, y_i \in Y \ \forall i = 1, .., m$ be the training set
- $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ be the loss function namely a function that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the
- D be the unknown distribution
- H be the hypothesis class

We define the training error as:
$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, (\vec{x}_i, y_i))$.
We define the generalization error as:
$L_d(h) = E_{z \sim D}[l(h, z)]$ where $z = (\vec{x}, y)$

$$
\begin{aligned}
E[L_s(h)] &= E\left[\frac{1}{m}\sum_{i=1}^{m} l(h, (\vec{x}_i, y_i))\right] \\
&= \frac{1}{m}\sum_{i=1}^{m} E[l(h, (\vec{x}_i, y_i))] \\
&= \frac{1}{m}\sum_{i=1}^{m} L_d(h) \\
&= L_d(h)
\end{aligned}
$$

3. Use the result above to argue that the ERM procedure can be appropriate when the training data is large enough.

Solution

$L_s(h)$ is the probability that a pair $(\vec{x}, y)$ taken uniformly at random from S the event $h(\vec{x}) \neq y$. That means, if we compute $E[L_s(h)]$, then it is equal to $L_d(h)$ if we have enough data because S is taken from D. Therefore $L_s(h) \approx L_d(h)$ because with enough data $E[L_s(h)] = L_d(h)$.

5.2 Exercise 2

Consider the regression problem with squared loss.

1. Provide a formal definition of the problem, describing the data, the learner's input, the learner's output, the loss function, the assumed generative model for the data, the learner's goal, and what choices the learner has to make when trying to solve the problem.

Solution

Regression problem with squared loss is a supervised learning task with:

- Domain set X which is the set of all possible objects to make predictions about, where a domain point $x \in X$ is called instance and is usually represented by a vector of features, usually is equal to $\mathbb{R}^d$
- Label set $Y = \mathbb{R}$

Given a training set $S = ((x_1, y_1)...(x_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ we need to choose an hypothesis class H, which defines the possible models or classification rules, from which we can pick our model to make predictions, and a loss function $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y. In particular in the case of squared loss such function is exactly the following: $l(h, (\vec{x}, y)) = (h(x) - y)^2$.

The goal is to find an hypothesis $\hat{h} \in H$ with low generalization error: $L_d(\hat{h}) = E_{z \sim D}[l(\hat{h}, z)]$ where:

- D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples).

2. Within the context above, define the coefficient of determination R^2, provide an intuition for what it is capturing, and describe why it makes sense to look at R^2 instead of considering the average loss.

Solution

The coefficient of determination in the context of a regression problem with squared loss is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$$

where $\hat{y}_i$ is the value predicted, y_i is the true label, $\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$ and $S = ((x_1, y_1)...(x_m, y_m))$ is the training set.

This coefficient measures how good it performs our model compare to a "dumb" model that predicts always the average value of the labels. In particular the values of R^2 can be:

- Equal to 1 it means that everything is correctly classified
- Equal to 0 it means that our model has the same power as the "dumb" one
- If it is less than 0 the performance of the model are very poor

Such value are used instead of the average loss because it is easier to interpret in certain situations.

3. Consider the problem of feature selection in the framework described above. Provide a brief motivation for feature selection, and describe one procedure to select a model with at most k features, where $k \ll d$.

Solution

The feature selection problem is the problem of selecting which features we use to learn. The goal is to find the number $k \ll d$ of features that:

- Prevents overfitting (because less features = less complexity)
- Improve performance (less feature are faster to process)

This problem can be solved through the use of the ERM procedure, therefore mathematically it can be stated as follows:

$$\min_{\vec{w}} L_S(\vec{w}) \text{ subject to: } \|\vec{w}\|_0 \leq k$$

where $\|\vec{w}\|_0$ is the 0-norm of the vector $\vec{w}$ i.e. $\|\vec{w}\|_0 = |\{i : w_i \neq 0\}|$

Let

$$I = \{1, ..., m\}$$
$$p = \{i_1, ..., i_k\}$$
$$H_p =$$
hypothesis or models where only $w_{i_1}, ..., w_{i_k}$ are used as features

To solve this problem we can use the following algorithms:

- Subset Selection algorithm: it select the set composed by k features that minimizes the training error. The pseudocode is the following:

  1: $P^{(k)} = \{J \subseteq I : |J| = k\}$
  2: **for all** $p \in P^{(k)}$ **do**
  3: $\quad h_p = \arg\min_{h \in H_p} L_S(h)$
  4: **end for**
  5: **return** $h^{(k)} = \arg\min_{h_p \in P^{(k)}} L_S(h_p)$

  Time complexity is: $O(d^k)$

- Forward selection algorithm: It is a greedy algorithm. Iteratively it add to the current set of features a feature not yet added to such set, such that it minimizes the training error. (It is useful to apply such algorithm when k is small and d is quite huge) The pseudocode is the following:

  $sol = \emptyset$
  while $|sol| < k$ **do**
      for all $i \in (I \setminus sol)$ **do**
          $p = sol \cup \{i\}$
          $h_p = \arg\min_{h \in H_p} L_S(h)$
      end for
      $sol = sol \cup \arg\min_{i \in I \setminus sol} L_S(h_{sol \cup \{i\}})$
  end while
  return sol

- Backward selection algorithm: It is a greedy algorithm. Iteratively it remove to the current set of features a feature not yet removed to such set that maximizes the training error. (It is useful to apply such algorithm when k is small and d is quite huge) The pseudocode is the following:

  $sol = I$
  while $|sol| > k$ **do**
      for all $i \in sol$ **do**
          $p = sol \setminus \{i\}$
          $h_p = \arg\min_{h \in H_p} L_S(h)$
      end for
      $sol = sol \setminus \arg\max_{i \in sol} L_S(h_{sol \setminus \{i\}})$
  end while
  return sol

5.3 Exercise 3

Consider a binary classification problem with 0-1 loss.

1. Provide the definition of VC dimension $VCdim(\mathcal{H})$ of a hypothesis set $\mathcal{H}$. What is the relation between the empirical error and the true risk in terms of the VC dimension of $\mathcal{H}$?

Solution

Let $h \in H$ be such that $h : X \to \{0, 1\}$. Let $C = \{c_1, ..., c_m\}$ with $C \subset X$. The restriction H_C of H to C is:

$$H_C = \{[h(c_1), ..., h(c_m)] : h \in H\}.$$

We say that H shatters C if $|H_C| = 2^m$, that is H_C contains all $2^m = 2^{|C|}$ functions from C to $\{0, 1\}$.

The Vc-dimension $VCdim(H)$ of H is the maximal size of set $C \subset X$ that can be shattered by H; if H can shatter sets of arbitrary large size then $VCdim(H) = +\infty$.

The relationship between $L_S(h)$ and $L_D(h)$ is the following: let H be an hypothesis class with $VCdim(H) = d < +\infty$ then, with probability $\geq 1 - \delta$ we have:

$$\forall h \in H \quad L_D(h) \leq L_S(h) + C\sqrt{\frac{VCdim(H) + \log(\frac{1}{\delta})}{2m}}$$

where C is the universal constant.

2. Assume that you have n hypothesis sets, denoted by $\mathcal{H}_i$, $i = 1, 2, \ldots, n$. Describe one strategy to choose a good hypothesis set $\mathcal{H}_i$ and a good model $h \in \mathcal{H}_i$.

Solution

One strategy is to use cross-validation to choose H_i and then, once H_i is chosen, we use all the data to learn the $\hat{h}_i \in H_i$. Therefore recalling i, ϑ, we can apply the cross validation method.

Assuming that we have enough data another strategy is to use validation to choose H_i and then, once H_i is chosen, we use all the data to learn the $\hat{h}_i \in H_i$.

CROSS VALIDATION

To be more precise, cross validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have.

To do so, we split the dataset into k different folds of size m/k (this quantity is supposed to be integer). Then for each value of the parameter ϑ we find the best model for all the possible k-1 folds that we can select. In addition to that, we compute the average error of each parameter as the average of the errors on the folds left out, and when we have repeated such procedure for all the values of the parameters, then we select the optimal one by choosing the one that minimizes the error computed for each parameter. To conclude, we train our model on all the dataset with the parameter found.

The pseudo code is the following:

Input: $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$; set of parameters Θ; integer k; learning algorithm A

Split S into $S_1, ..., S_k$
for all $\vartheta \in \Theta$ **do**
    for i = 1 ... k **do**
        $h_{i,\vartheta} = A(S \setminus S_i; \vartheta)$
        $error(\vartheta) = \frac{1}{k} \sum_{i=1}^{k} L_{S_i}(h_{i,\vartheta})$
    end for
end for
Output: $\vartheta^* = \arg\min_{\vartheta}(error(\vartheta))$
        $h_{\vartheta^*} = A(S; \vartheta^*)$

VALIDATION

To be more precise validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that

such parameter can assume with the data we have.

To do so we split out dataset into 2 parts training set and validation set. Then, for each value of the parameter that we have, we find the best model for every possible values of the parameter on the training set. Then among such models that correspond to a value of the parameter we select the one that minimizes the validation error. The model obtain then is trained on the entire dataset.

Notice that training error is computed as:
$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, (\vec{x}_i, y_i))$
where S is the training set, namely, $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$, $l : H \times Z \to \mathbb{R}^+$ is the loss function where $Z = X \times Y$, that, given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

The validation error instead is computed as:
$L_V(h) = \frac{1}{mv} \sum_{i=1}^{mv} l(h, (\vec{x}_i, y_i))$
where V is the validation set, namely $V = ((\vec{x}_1, y_1)...(\vec{x}_{mv}, y_{mv}))$, $l : H \times Z \to \mathbb{R}^+$ is the loss function where $Z = X \times Y$, that, given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

5.4 Exercise 4

1. Briefly introduce the clustering problem.

Solution

Clustering is an unsupervised learning problem, namely a type of problem in which the training set is $(x_1, ..., x_m)$ so, there are no target values. The goal is to find interesting structure in the data or organize it in some meaningful way. To be precise: clustering is the task of grouping a set of objects such that similar objects end up in the same group and dissimilar objects are separated into different groups.

More formally the problem has the following input and output:

- Input: set X of objects and a distance function $d : X \times X \to \mathbb{R}^+$ that is a function that:
  (a) Is symmetric: $d(x, x') = d(x', x)$ for all $x, x' \in X$
  (b) $d(x, x) = 0$ for all $x \in X$
  (c) d satisfies the triangular inequality, namely: $d(x, x') \leq d(x, z) + d(z, x')$
- Output: a partition of X into clusters, that is $C = (C_1, .., C_k)$ such that:
  (a) $\bigcup_{i=1}^{k} C_i = X$
  (b) For all $i \neq j : C_i \neq C_j$

Sometimes the input includes the number of clusters to produce. In addition the output depends on the definitions of the problem for instance, it could be a dendogram, or, the clusters produced depends from a cost function. Also, we can also have that instead of using a distance function as input we use a similarity function $s : X \times X \to \mathbb{R}^+$ that is a function that:

- Is symmetric: $s(x, x') = s(x', x)$ for all $x, x' \in X$
- $s(x, x) = 1$ for all $x \in X$

2. Define and explain the cost function used in K-means clustering.

Solution

Let the input data points be $\vec{x}_1, ..., \vec{x}_m$ with $\vec{x}_i \in \mathbb{R}^d$ for i=1...m. The K-means cost function is:

$$\sum_{i=1}^{k} \sum_{\vec{x} \in C_i} d(\vec{x}, \vec{\mu}_i)^2$$

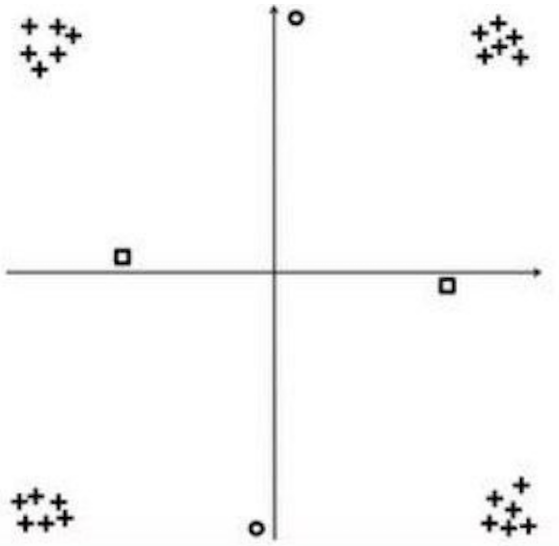
where $C_1, .., C_k$ are the clusters, $d(.)$ is the Euclidian distance in $\mathbb{R}^d$ and $\vec{\mu}_i$ is the center of cluster C_i for i=1...m. Therefore the cost of a clustering for k-means is defined as the sum of the squares of the distances of each point to the center of the cluster it belongs to.

3. Consider the data in the figure below where each point $\mathbf{x} \in \mathbb{R}^2$ is represented by a cross. Show the results (i.e., draw approximately the final centers locations and the final assignment of the points to the clusters) of clustering into $k = 2$ clusters with K-means when:

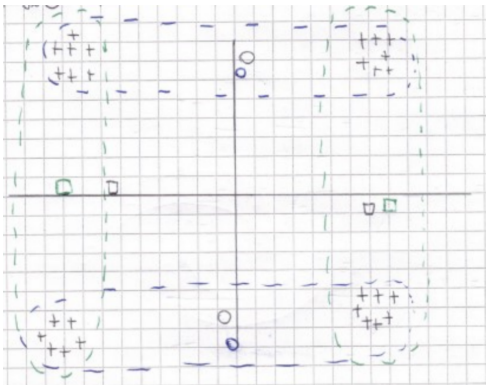
   (a) the initial centers for the algorithm are the circles;

   (b) the initial centers for the algorithm are the squares.

   Is one solution significantly better than the other one? Briefly motivate your answer.



Solution



There is no solution that is significantly better than the other one since the data consists of 4 groups of points that are essentially symmetric with respect to the origin, and the two solutions are just two different ways to group them into two groups. Also the coefficient silhouette of the two solutions is probably similar.

Exam 6
6 July 2021

Exercise 1 [8 points; max 2 pages.]

1. Describe the general framework of regression.
2. Write the optimization problem for linear regression with squared loss, and describe a solution for the situation in which $\mathbf{X}^T\mathbf{X}$ is not invertible (where $\mathbf{X}$ is the matrix whose i-th row is given by the features of the i-th training sample).

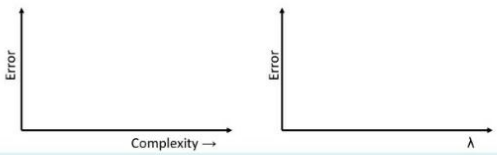
Exercise 2 [8 points; max 2 pages.]

1. Introduce the concept of separating hyperplane and describe how it leads to the formulation of hard SVM.
2. In most practical situations data are not linearly separable. How can hard SVM be adapted to soft SVM, so to deal with this situation while still using a linear separation boundary? Formulate this adaptation in the form of an optimization problem, discussing the role and effect of the parameter λ that trades the two components of the cost function.

Exercise 3 [8 points; max 2 pages.]

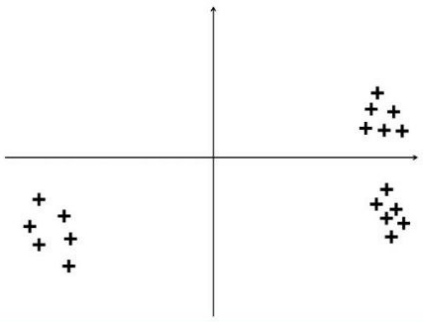
With reference to the binary classification problem:

1. Describe how the true (generalization) error $L_{\mathcal{D}}(h_s)$ of a model h_s can be decomposed into approximation ϵ_{app} and estimation ϵ_{est} error. Discuss how this decomposition is related to the complexity of the selected hypothesis class and draw a plot with the expected behaviour of the approximation, estimation, and true error, as functions of the complexity of the hypothesis class (the axes for such a plot are shown below, on the left).
2. Formally introduce the Tikhonov regularization model writing down the function to be optimized. In relation with the previous answer, discuss how Tikhonov regularization can be used to control the fitting-stability trade-off. In particular discuss the behaviour of the approximation, estimation, and true error, as functions of the regularization parameter λ (the axes for such a plot are shown below, on the right).



Exercise 4 [8 points; max 2 pages.]

1. Introduce the k-means clustering problem and present the standard (i.e., Lloyd's) algorithm for its solution.
2. Assume that you need to cluster the points shown in the figure below into $k = 3$ different clusters using k-means. The final solution depends on the initialization of the centroids: suggest two possible initializations, one leading to the optimal solution and one leading to a sub-optimal solution.



6.1 Exercise 1

1. Describe the general framework of regression.

Solution

Regression task is a supervised learning task with:

- Domain set X which is the set of all possible objects to make predictions about, where a domain point $x \in X$ is called instance and is usually represented by a vector of features, usually is equal to $\mathbb{R}^d$
- Label set $Y = \mathbb{R}$

Given a training set $S = ((x_1, y_1)...(x_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ we need to choose an hypothesis class H, which defines the possible models or classification rules, from which we can pick our model to make predictions, and a loss function $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

The goal is to find an hypothesis $\hat{h} \in H$ with low generalization error: $L_d(\hat{h}) = E_{z \sim D}[l(\hat{h}, z)]$ where:

- D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples).

2. Write the optimization problem for linear regression with squared loss, and describe a solution for the situation in which $X^T X$ is not invertible (where X is the matrix whose i-th row is given by the features of the i-th training sample).

Solution

The optimization problem for linear regression with squared loss can be described as follow:

Given:

- $S = ((x_1, y_1)...(x_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ training set
- $H = L_d = \{h_{\vec{w},b} : \mathbb{R}^d \to \mathbb{R}\}$ where $h_{\vec{w},b}(\vec{x}) = \langle \vec{w}, \vec{x} \rangle + b$ be the hypothesis class
- $l(h, (\vec{x}, y)) = (h(\vec{x}) - y)^2 = (\langle \vec{w}, \vec{x} \rangle + b - y)^2 = (\langle \vec{w}', \vec{x}' \rangle - y)^2$ if $\vec{w}' = [b, w_1, ..., w_m]$ and $x' = [1, x_1, ..., x_m]$ be the loss function

Then the function to be optimized is $\sum_{i=1}^{m} (\langle \vec{w}', \vec{x}_i' \rangle - y_i)^2$ therefore we obtain that $(X^T X)\vec{w} = X^T \vec{y}$

Now if $X^T X$ is not invertible then we need to compute the generalized inverse of that matrix. So, let $A = X^T X$, the generalized inverse of A is the matrix A^+ such that: $AA^+A = A$. To define A^+ we need to apply the following steps:

- Apply the eigenvalue decomposition to A and obtain $A = VDV^T$ where:
  (a) V is the matrix in which columns there are the eigenvectors normalized of A
  (b) D is a diagonal matrix where in the diagonal there are the eigenvalues in the correspondence position of the correspondent eigenvector in V
- Finally define $A^+ = VD^+V^T$ where:
  (a) V is the matrix as above
  (b) D^+ is defined as follows:

$$d_{ij}^+ = \begin{cases} \frac{1}{d_{ij}} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

  where d_{ij}^+ is the element in position (i, j) of D^+ and d_{ij} is the element in position (i, j) of D

6.2 Exercise 2

1. Introduce the concept of separating hyperplane and describe how it leads to the formulation of hard SVM.

Solution

Given a binary classification problem where:

- Domain set X which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Label set $Y = \{-1, +1\}$, that defines the set of all possible labels in this case the labels are only 2
- set $S = ((x_1, y_1)...(x_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ is the so called training set

A separating hyperplane is an hyperplane that separates all the points of the training set S into two parts, namely into two halfspaces which contains all and only the points that have either class -1 or $+1$. In case in which the training set S is linearly separable, then there exists more than one such hyperplane and in particular we can define the optimal as the one that maximizes the margin i.e. the distance between the hyperplane and the closest points in S to him.

The optimal separating hyperplane introduced above is exactly the solution computed by the SVM when data are linearly separable. In particular if we define:

- hyperplane as: $L = \{v : \langle \vec{w}, \vec{v} \rangle + b = 0\}$
- distance between a point x and L: $d(x, L) = \min\{|\vec{x} - \vec{v}| : v \in L\}$

Then the optimal separating hyperplane can be computed as:

$$\underset{(\vec{w},b):\|\vec{w}\|=1}{\arg\max} \ \min_{i \in \{1,...,m\}} |\langle \vec{w}, \vec{x}_i \rangle + b| \text{ subject to: } \forall i \ y_i(\langle \vec{w}, \vec{x}_i \rangle + b)$$

Which corresponds to the formulation of the Hard-SVM, namely the formulation of the SVM when data are linearly separable.

2. In most practical situations data are not linearly separable. How can hard SVM be adapted to soft SVM, so to deal with this situation while still using a linear separation boundary? Formulate this adaptation in the form of an optimization problem, discussing the role and effect of the parameter λ that trades the two components of the cost function.

Solution

Soft-SVM is similar to Hard-SVM, but can be used also when training data is not linearly separable (i.e., the training data cannot be perfectly classified with a linear model). Soft-SVM finds a model of large margin while allowing the training set to be inside the margin or wrongly classified. This is obtained by adding slack variables ξ_i to the constraints of hard SVM. The constraint for soft-SVM are then:

- $\xi_i \geq 0$ for each $i = 1, ..., m$
- $y_i(\langle \vec{w}, \vec{x}_i \rangle + b) \geq 1 - \xi_i$ for each (x_i, y_i) $i = 1, ..., m$ in the training set where $\vec{w}$ and b defines the model.

The interpretation of ξ_i is the following:

- if $\xi_i = 0$ then $\vec{x}_i$ is correctly classified and outside the margin;
- if $0 < \xi_i < 1$ then $\vec{x}_i$ is correctly classified and inside the margin;
- if $\xi_i \geq 1$ then $\vec{x}_i$ is wrongly classified.

The function optimized by soft-SVM consider both the margin and the "violation" of the hard-SVM given by ξ_i.

Therefore the optimization problem for soft-SVM can be described as follows:

Input: $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ and parameter $\lambda > 0$

Goal: $\min_{\vec{w},b,\vec{\xi}} \lambda \|\vec{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i$

Output: $\vec{w}, b, \vec{\xi}$

The parameter lambda controls the tradeoff between an high margin with several points misclassified and a small margin with most of the points correctly classified in particular, if $\lambda \approx 0$ then, soft-SVM will produce a model that tries to be as similar as possible as the one produced by hard-SVM namely, it tries

to classified as many as possible points correctly. On the other hand, if $\lambda \gg 0$ then, soft-SVM cares only about maximizing the margin therefore, it produces a model that has most of the points misclassified but it has an high margin.

6.3 Exercise 3

With reference to the binary classification problem:

1. Describe how the true (generalization) error $L_D(h_s)$ of a model h_s can be decomposed into approximation ϵ_{app} and estimation ϵ_{est} error. Discuss how this decomposition is related to the complexity of the selected hypothesis class and draw a plot with the expected behaviour of the approximation, estimation, and true error, as functions of the complexity of the hypothesis class (the axes for such a plot are shown below, on the left).

Solution

The generalization error $L_D(h_S)$ can be decomposed into $L_D(h_S) = \epsilon_{app} + \epsilon_{est}$ where $\epsilon_{app} = \min_{h \in H} L_D(h)$ and $\epsilon_{est} = L_D(h_S) - \min_{h \in H} L_D(h)$.

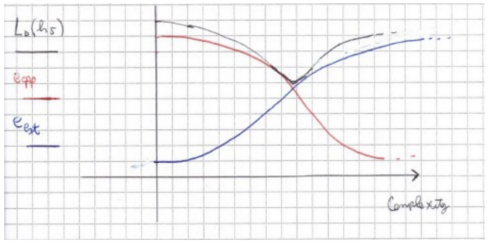
In words:

- ϵ_{app} = approximation error: is the minimum risk achievable by a predictor in the hypothesis class.
- ϵ_{est} = estimation error: is the difference between the approximation error and the error achieved by the ERM predictor.

Therefore if we plot in a graph those two error and the $L_D(h_S)$ as functions of the complexity we obtain that:

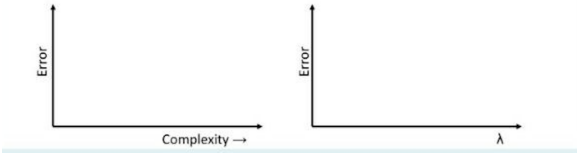
- ϵ_{app}: will decrese if the complexity decreases if complexity increases
- ϵ_{est}: will increase if the complexity increases
- $L_D(h_S)$: has a "bell" behaviour

The graph is the following:



Notice that a low complexity hypothesis class corresponds to a underfitting situation while a high complexity class corresponds to a overfitting situation.

2. Formally introduce the Tikhonov regularization model writing down the function to be optimized. In relation with the previous answer, discuss how Tikhonov regularization can be used to control the fitting-stability trade-off. In particular discuss the behaviour of the approximation, estimation, and true error, as functions of the regularization parameter λ (the axes for such a plot are shown below, on the right).



Solution

Tikhonov regularization is the Regularized loss minimization paradigm with regularization function $R(\vec{w}) = \lambda\|\vec{w}\|^2$ where $\lambda > 0$ and $\|\vec{w}\|^2$ is the l_2 norm of the vector $\vec{w}$.

Therefore the function to be optimized, i.e. minimized is the following: $\arg\min_{\vec{w}}(L_S(\vec{w})+\lambda\|\vec{w}\|^2)$ where $L_S(\vec{w})$ is the training error and S is the training set.

Notice that $\|\vec{w}\|^2$ measures the complexity of the hypothesis $\vec{w}$, while λ is the parameter used to tradeoff the empirical risk and the complexity of the model.

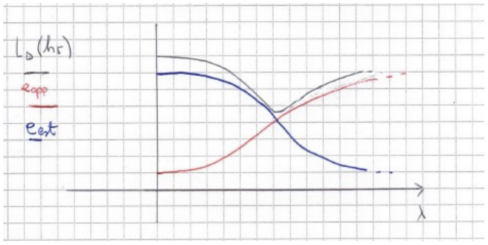
Now, the behavior of the ϵ_{app}, ϵ_{est} and $L_D(h_S)$ in terms of the parameter λ is the following:

- ϵ_{app} will increase if λ increases because with a low λ we have a complex hypothesis class
- ϵ_{est} will decrease if λ increases because with an high λ we

have a low-complexity hypothesis class

- $L_D(h_S)$: has a "bell" behaviour

The graph is the following:



6.4 Exercise 4

1. Introduce the k-means clustering problem and present the standard (i.e., Lloyd's) algorithm for its solution.

Solution

K-means is a cost minimization clustering problem. Let $X \subseteq X'$ be the set of points to be clustered with $X = \{\vec{x}_1, ..., \vec{x}_m\}$ while X' is the space of possible points that we assume to be in $\mathbb{R}^d$ (i.e $X' = \mathbb{R}^d$). Let $k \in \mathbb{N}^+$ be the number of clusters, that is, with X, the input of the problem. Let $d(.)$ be a distance function: $d(\vec{x}, \vec{x'}) = \|\vec{x} - \vec{x'}\|$.

The goal is to find:

- A partition $C = (C_1, .., C_k)$ of X
- Centroids $\vec{\mu}_1, ..., \vec{\mu}_k$ of $C_1, .., C_k$ respectively

That minimizes the k-means cost function: $\sum_{i=1}^{k} \sum_{\vec{x} \in C_i} d(\vec{x}, \vec{\mu}_i)^2$

To find the solution of this problem we can use an heuristic algorithm called Lloyd's Algorithm. Such algorithm works as follows:

Input: data points $X = \{\vec{x}_1, ..., \vec{x}_m\}$ and $k \in \mathbb{N}^+$
Output: $C = (C_1, ..., C_k)$ of X and centroids $\vec{\mu}_1, ..., \vec{\mu}_k$ of $C_1, ..., C_k$ respectively

To produce such things the algorithm works as follows:

- Randomly initialize the centroids $\vec{\mu}_1, ..., \vec{\mu}_k$

- Until the convergence is not reached iteratively repeats the following steps:
  (a) Compute each cluster C_i by associating the points to him that are closer to its centroid compared to others centroids.
  (b) Compute the new centroids for the next iterations.
  (c) If the convergence is reached then the output described above will be returned.

The pseudocode for such algorithm is the following:

Randomly choose $\vec{\mu}_1^0, ..., \vec{\mu}_k^0$
for $t \leftarrow 0, ...$ do **do**
  for $i = 1, ..., k$ **do**
    $C_i \leftarrow \{\vec{x} \in X : i = \arg\min_j d(\vec{x}, \vec{\mu}_j^t)\}$
  end for
  for $i = 1, ..., k$ **do**
    $\vec{\mu}_i^{t+1} = \frac{1}{|C_i|} \sum_{\vec{x} \in C_i} \vec{x}$
  end for
  if convergence reached **then**
    return $C = (C_1, ..., C_k), \vec{\mu}_1^{t+1}, ..., \vec{\mu}_k^{t+1}$
  end if
end for

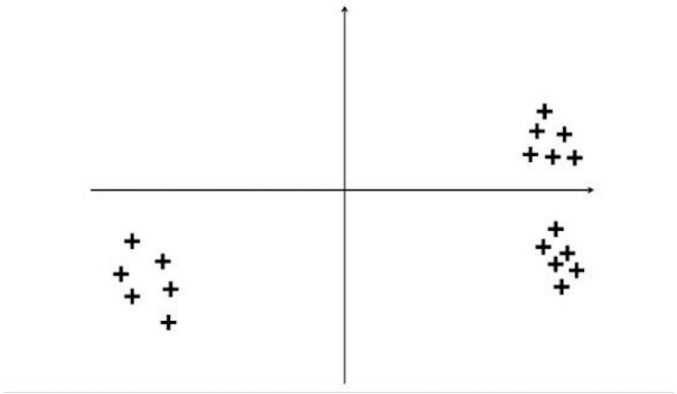
The convergence criteria that can be used for such algorithm are the following:

- The k-means objective function is no lower than the k-means objective function at the next iteration
- $\sum_{i=1}^{k} d(\vec{\mu}_i^{t+1}, \vec{\mu}_i^t) \leq \varepsilon$
- $\max_{1 \leq i \leq k} d(\vec{\mu}_i^{t+1}, \vec{\mu}_i^t) \leq \varepsilon$

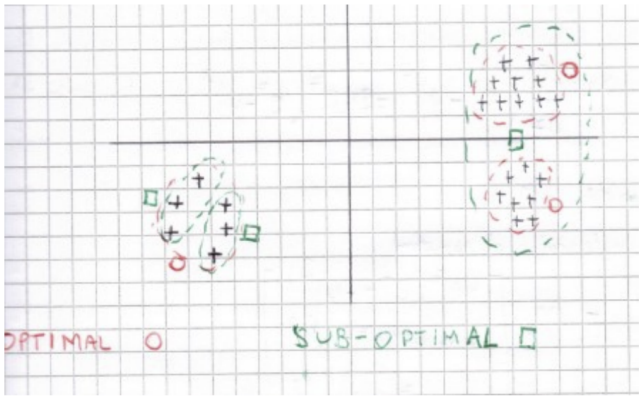
Note that if the first criteria is used then, the algorithm will always converge.

The time complexity is $O(tkmd)$ where $O(kmd)$ is due to the assignments of the points, while $O(md)$ is due to the computation of the centroids. Therefore the complexity depends on t which is the number of iterations of the algorithm.

2. Assume that you need to cluster the points shown in the figure below into $k = 3$ different clusters using k-means. The final solution depends on the initialization of the centroids; suggest two possible initializations, one leading to the optimal solution and one leading to a sub-optimal solution.



Solution



Exam 7
28 January 2021

Time: 2 hours

Exercise 1 [8 points; max 2 pages.]

Consider the *binary* classification problem with 0-1 loss.

1. Provide a formal definition of the problem, describing the data, the learner's input, the learner's output, the loss function, the assumed generative model for the data, the learner's goal, and what choices the learner has to make when trying to solve the problem.
2. Assume the hypothesis class is $\mathcal{H}$, the data generative distribution is $\mathcal{D}$, and the training data is S. Provide the definition of the training error $L_S(h)$ and generalization error $L_{\mathcal{D}}(h)$, $h \in \mathcal{H}$, and prove that the expectation (over the distribution of the training set) of $L_S(h)$ is equal to $L_{\mathcal{D}}(h)$.
3. Use the result above to argue that the ERM procedure can be appropriate when the training data is large enough.
4. Provide the definition of ε-representative sample, and prove that if the training set S is $\frac{\varepsilon}{2}$-representative (w.r.t. domain Z, hypothesis class $\mathcal{H}$, loss function ℓ, and distribution $\mathcal{D}$), then any output h_S of the ERM procedure (using $\mathcal{H}$) satisfies

$$L_{\mathcal{D}}(h_S) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \varepsilon.$$

(Provide a motivation for each step of the proof.)

Exercise 2 [8 points; max 2 pages.]

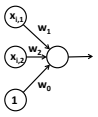
Consider the regression problem with training data $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_m, y_m)\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ for $i = 1, \dots, m$. Assume the hypothesis class is given by linear models, the squared loss is used, and Tikhonov regularization is used.

1. Formally define the hypothesis class, the loss function, and the function that is optimized by ERM with Tikhonov regularization (i.e., the objective function).
2. Provide the formula for the best model (i.e., which optimizes the objective function) learned from data in the context above. (The derivation is not required.)
3. In Tikhonov regularization, the parameter λ regulates the tradeoff between the empirical risk and the complexity of the model. Describe one approach to choose λ.

Exercise 3 [8 points; max 2 pages.]

Consider the regression problem, with training data $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_m, y_m)\}$ with $\mathbf{x}_i = [x_{i,1}, x_{i,2}] \in \mathbb{R}^2$ and $y_i \in \mathbb{R}$ for $i = 1, \dots, m$.

Assume the hypothesis class $\mathcal{H}$ is given by the simple neural network with the architecture described in the figure below, where w_0, w_1, w_2 are the edges' weights.



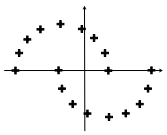
Assume the activation function for the output node is $\sigma(z) = e^z$ (the final prediction is the output of the output node), and the loss function is $\ell(h, (\mathbf{x}, y)) = (h(\mathbf{x}) - y)^4$.

1. Derive a closed-form expression for the hypotheses in $\mathcal{H}$.
2. Describe the stochastic gradient descent (SGD) algorithm (in general). What is the main advantage of SGD with respect to the gradient descent algorithm?
3. Write the SGD update for learning a model from the hypothesis class $\mathcal{H}$ above.

Exercise 4 [8 points; max 2 pages.]

Consider the clustering problem.

1. Briefly describe *linkage-based clustering*, what is the input, what is the output, the general algorithm it employs, and three common termination conditions.
2. Consider *single linkage* clustering. Describe how it is obtained by the general *linkage-based clustering* (no pseudocode needed).
3. Show the output of single linkage clustering when the input is given by the points in $\mathbb{R}^2$ shown as crosses below and the termination condition is given by having the points partitioned in $k = 2$ clusters. Briefly describe how the algorithm reaches such output.



7.1 Exercise 1

Consider the binary classification problem with 0-1 loss.

1. Provide a formal definition of the problem, describing the data, the learner's input, the learner's output, the loss function, the assumed generative model for the data, the learner's goal, and what choices the learner has to make when trying to solve the problem.

Solution

The binary classification problem with 0-1 loss is a supervised learning problem with:

- Domain set X which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Label set $Y = \{-1, +1\}$, that defines the set of all possible labels in this case the labels are only 2

Given a training set $S = ((x_1, y_1)...(x_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ we need to choose an hypothesis class H, which defines the possible models or classification rules, from which we can pick our model to make predictions, and a loss function $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

In particular in the case of 0-1 loss the function is exactly the following:

$$l(h, (\vec{x}, y)) = \begin{cases} 1 \text{ if } h(\vec{x}) \neq y \\ 0 \text{ if } h(\vec{x}) = y \end{cases}$$

The goal is to find an hypothesis $\hat{h} \in H$ with low generalization error: $L_d(\hat{h}) = E_{z \sim D}[l(\hat{h}, z)]$ where:

- D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples).

2. Assume the hypothesis class is $\mathcal{H}$, the data generative distribution is $\mathcal{D}$, and the training data is S. Provide the definition of the training error $L_S(h)$ and generalization error $L_D(h)$, $h \in \mathcal{H}$, and prove that the

expectation (over the distribution of the training set) of $L_S(h)$ is equal to $L_D(h)$.

Solution

Let:

- X be the domain set, which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Y be the label set that defines the set of all possible labels
- $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ be the training set
- $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ be the loss function namely a function that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the
- D be the unknown distribution
- H be the hypothesis class

We define the training error as: $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, (\vec{x}_i, y_i))$.

We define the generalization error as: $L_d(h) = E_{z \sim D}[l(h, z)]$ where $z = (\vec{x}, y)$

$$\begin{aligned} E[L_s(h)] &= E\left[\frac{1}{m} \sum_{i=1}^{m} l(h, (\vec{x}_i, y_i))\right] \\ &= \frac{1}{m} \sum_{i=1}^{m} E[l(h, (\vec{x}_i, y_i))] \\ &= \frac{1}{m} \sum_{i=1}^{m} L_d(h) \\ &= L_d(h) \end{aligned}$$

3. Use the result above to argue that the ERM procedure can be appropriate when the training data is large enough.

Solution

$L_s(h)$ is the probability that a pair $(\vec{x}_i, y_i)$ taken uniformly at random from S the event $h(\vec{x}_i) \neq y_i$. That means, if we compute $E[L_s(h)]$, then it is equal to $L_d(h)$ if we have enough data because S is taken from D. Therefore $L_s(h) \approx L_d(h)$ because with enough data $E[L_s(h)] = L_d(h)$.

$$L_D(h_S) \leq \min_{h \in \mathcal{H}} L_D(h) + \varepsilon.$$

4. Provide the definition of ε-representative sample, and prove that if the training set S is $\frac{\varepsilon}{2}$-representative (w.r.t. domain Z, hypothesis class $\mathcal{H}$, loss function ℓ, and distribution $\mathcal{D}$), then any output h_S of the ERM procedure (using $\mathcal{H}$) satisfies (Provide a motivation for each step of the proof.)

Solution

A set S is ε-representative (with respect to domain Z, hypothesis class H, loss function l and distribution D if:

$$\forall h \in H \quad |L_S(h) - L_d(h)| \leq \varepsilon$$

7.2 Exercise 2

Consider the regression problem with training data $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_m, y_m)\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ for $i = 1, \ldots, m$. Assume the hypothesis class is given by linear models, the squared loss is used, and Tikhonov regularization is used.

1. Formally define the hypothesis class, the loss function, and the function that is optimized by ERM with Tikhonov regularization (i.e., the objective function).

Solution

With the assumption made we have that:

- H: the hypothesis class is exactly equal to $L_d = \{h_{\vec{w},b} : \mathbb{R}^d \to \mathbb{R}\}$ where $h_{\vec{w},b}(\vec{x}) = \langle \vec{w}, \vec{x} \rangle + b$
- l: the loss function is exactly equal to $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ such that $l(h, (\vec{x}, y)) = (h(\vec{x}) - y)^2 = (\langle \vec{w}, \vec{x} \rangle + b - y)^2 = (\langle \vec{w}', \vec{x}' \rangle - y)^2$ if $\vec{w}' = [b, w_1, ..., w_m]$ and $\vec{x}' = [1, x_1, ..., x_m]$

- The function to be optimized becomes: $\lambda\|\vec{w}\|^2+\sum_{i=1}^{m}(\langle\vec{w}', \vec{x}_i'\rangle - y_i)^2$

2. Provide the formula for the best model (i.e., which optimizes the objective function) learned from data in the context above. (The derivation is not required.)

Solution

The formula to compute the best model with the previous assumption is $\vec{w} = (\lambda I + X^T X)^{-1} X^T \vec{y}$ where:

$$X = \begin{bmatrix} \vec{x}_1' \\ \vdots \\ \vdots \\ \vec{x}_m' \end{bmatrix} \text{ and } \vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ \vdots \\ y_m \end{bmatrix}$$

3. In Tikhonov regularization, the parameter λ regulates the tradeoff between the empirical risk and the complexity of the model. Describe one approach to choose λ.

Solution

A possible strategy to estimate the parameter λ is to use cross-validation. Therefore recalling λ, ϑ, we can apply the cross validation method.

To be more precise, cross validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have. To do so, we split the dataset into k different folds of size m/k (this quantity is supposed to be integer). Then for each value of the parameter ϑ we find the best model for all the possible k-1 folds that we can select. In addition to that, we compute the average error of each parameter as the average of the errors on the folds left out, and when we have repeated such procedure for all the values of the parameters, then we select the optimal one by choosing the one that minimizes the error computed for each parameter. To conclude, we train our model on all the dataset with the parameter found.

The pseudo code is the following:

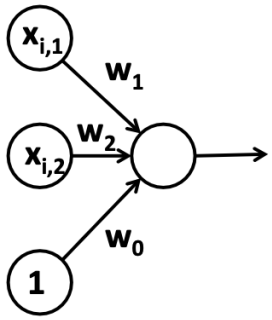
1: Input: $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$; set of parameters Θ; integer k; learning algorithm A

Split S into $S_1, ..., S_k$
for $\vartheta \in \Theta$ **do**
    for i = 1 ... k **do**
        $h_{i,\vartheta} = A(S \setminus S_i; \vartheta)$
    end for
    $error(\vartheta) = \frac{1}{k} \sum_{i=1}^{k} L_{S_i}(h_{i,\vartheta})$
end for
Output: $\vartheta^* = \arg\min_{\vartheta}(error(\vartheta))$
        $h_{\vartheta^*} = A(S; \vartheta^*)$

7.3 Exercise 3

Consider the regression problem, with training data $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_m, y_m)\}$ with $\mathbf{x}_i = [x_{i,1}, x_{i,2}] \in \mathbb{R}^2$ and $y_i \in \mathbb{R}$ for $i = 1, \dots, m$.

Assume the hypothesis class $\mathcal{H}$ is given by the simple neural network with the architecture described in the figure below, where w_0, w_1, w_2 are the edges' weights.



Assume the activation function for the output node is $\sigma(z) = e^z$ (the final prediction is the output of the output node), and the loss function is $\ell(h, (\mathbf{x}, y)) = (h(\mathbf{x}) - y)^4$.

1. Derive a closed-form expression for the hypotheses in $\mathcal{H}$.

Solution

The closed-form expression for hypotheses in $\mathcal{H}$ is:

$H_{V,E,e^z,w} = \{h_{V,E,e^z,w} : \mathbb{R}^2 \to \mathbb{R}$ such that w is a mapping from E to $\mathbb{R}\}$

where:

$$h_{V,E,e^z,w}(\vec{x}) = e^{(w_0 + w_1 x_1 + w_2 x_2)}$$

2. Describe the stochastic gradient descent (SGD) algorithm (in general).

What is the main advantage of SGD with respect to the gradient descent algorithm?

Solution

The Stochastic Gradient Descent (SGD for short) algorithm is a general approach to minimize a differentiable convex function $f(\vec{w})$ where $f(\vec{w}) : \mathbb{R}^d \to \mathbb{R}$.

To do so, we update the current solution vector $\vec{w}^{(t)}$ as the following formula:

- Choose a vector $\vec{v}^{(t)}$ at random such that: $E[\vec{v}^{(t)}|\vec{w}^{(t)}] \in \nabla f(\vec{w})$
- $\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta\vec{v}^{(t)}$

Which can be translated, if $f(\vec{w}) = L_S(\vec{w})$ to:

- Pick a random point $(\vec{x}_i, y_i)$ uniformly at random from the training set $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$
- Compute $\vec{v}^{(t)} = \nabla l(h, (\vec{x}_i, y_i))$
- $\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta\vec{v}^{(t)}$

Therefore the main advantage is that with stochastic gradient descent we compute the gradient of a much less complicated function, (i.e. the gradient of the loss function), compare to the gradient descent algorithm because, in such algorithm we need to update $\vec{w}^{(t)}$ with the following rule: $\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta\nabla f(\vec{w})$. In addition it can be used also when $f(\vec{w}) = L_S(\vec{w}) + R(\vec{w})$.

3. Write the SGD update for learning a model from the hypothesis class $\mathcal{H}$ above.

Solution

In general, for stochastic gradient descent (SGD) let $\vec{w}^{(t)}$ be the weights optimize the model at iteration t. The update rule is given by:

- Pick $(\vec{x}_i, y_i) \in S$ uniformly at random
- $\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta\nabla l(h, (\vec{x}_i, y_i))$

We need to compute the gradient for a specific model class and

loss. In our case each model in our model class is a function:

$$h(\vec{x}) = e^{w_0+w_1x_1+w_2x_2}$$

Therefore:

$$\nabla l(h,(\vec{x}_i,y)) = \begin{bmatrix} \frac{\partial l}{\partial w_0} \\ \frac{\partial l}{\partial w_1} \\ \frac{\partial l}{\partial w_2} \end{bmatrix} \text{ where } z = w_0 + w_1x_1 + w_2x_2$$

$$\frac{\partial l}{\partial w_0} = \frac{\partial l}{\partial z} \cdot \frac{\partial z}{\partial w_0} = 4(e^z - y)^3 \cdot e^z$$

$$\frac{\partial l}{\partial w_1} = \frac{\partial l}{\partial z} \cdot \frac{\partial z}{\partial w_1} = 4x_1e^z(e^z - y)^3$$

$$\frac{\partial l}{\partial w_2} = \frac{\partial l}{\partial z} \cdot \frac{\partial z}{\partial w_2} = 4x_2e^z(e^z - y)^3$$

Therefore the SGD update rule is:

- Pick $(\vec{x}_i, y_i) \in S$ uniformly at random
- $\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta \begin{bmatrix} 4e^z(e^z - y)^3 \\ 4x_1e^z(e^z - y)^3 \\ 4x_2e^z(e^z - y)^3 \end{bmatrix}$

7.4 Exercise 4

Consider the clustering problem.

1. Briefly describe linkage-based clustering, what is the input, what is the output, the general algorithm it employs, and three common termination conditions.

Solution

Linkage-based clustering is a class of algorithms that solves the general problem of clustering (i.e. grouping together similar element and separate into different groups dissimilar objects).

Such algorithms can be described as follows:

Input: set X of objects and a distance function $d : X \times X \to \mathbb{R}^+$ that is a function that:

- Is symmetric: $d(x, x') = d(x', x)$ for all $x, x' \in X$
- $d(x, x) = 0$ for all $x \in X$
- d satisfies the triangular inequality, namely: $d(x, x') \leq d(x, z) + d(z, x')$

Output: most of the time a dendogram, namely a graph that represent in which cluster each element are putted.

Notice that, sometimes, the output is directly the partition of the element into k different cluster, and this is mostly likely if as input we give also the value k of how many clusters must be produced.

The solution can be found through the following algorithm:

- Start with a trivial cluster: each point represent a cluster
- Until "termination condition" repeatedly merge the closest cluster in the previous clustering

The most common termination condition used are:

(a) Data points are partitioned into k clusters

(b) The minimum distance between pairs of clusters is greater than a value r

(c) All the points are in a cluster that means, the output is a dendogram.

2. Consider single linkage clustering. Describe how it is obtained by the general linkage-based clustering (no pseudocode needed).

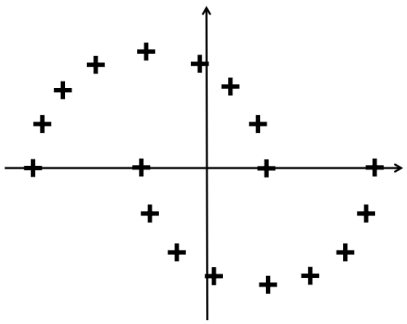
Solution

As we have said at the beginning Linkage-based clustering is a class of algorithm and each of these algorithm works exactly as explained above, except on how to compute the distance between clusters. In particular one of the way to compute such distance is to compute it as the minimum distance between the points inside of those two different clusters. Mathematically speaking the formula is the following:

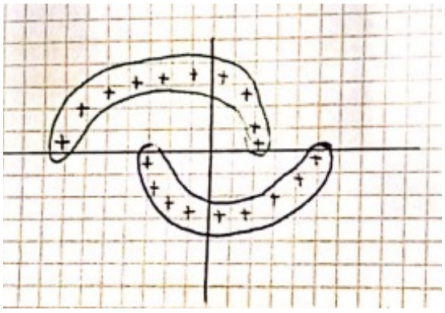
$$D(A,B) = \frac{1}{|A||B|} \min\{d(x,x') : x \in A; x' \in B\}$$

So, if we use such formula (to compute the distance between two clusters i.e. between cluster A and cluster B) then we are applying the so called single linkage clustering method.

3. Show the output of single linkage clustering when the input is given by the points in $\mathbb{R}^2$ shown as crosses below and the termination condition is given by having the points partitioned in $k = 2$ clusters. Briefly describe how the algorithm reaches such output.



Solution



Exam 8
29 June 2020

Exercise 1 [8 points]

1. Describe the general framework of binary classification.
2. Discuss the use of the logistic regression model for binary classification and describe how it can be trained using stochastic gradient descent.

Exercise 2 [8 points]

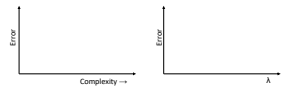
1. Introduce the concept of separating hyperplane and describe how it leads to the formulation of hard SVM.
2. In most practical situations data are not linearly separable. How can hard SVM be adapted to cope with this situation while still using a linear separation boundary? Formulate this adaptation in the form of an optimization problem, discussing the role and effect of the parameter λ that trades the two components of the cost function.

[Solution: Exercise 2]

Exercise 3 [8 points]

With reference to the binary classification problem:

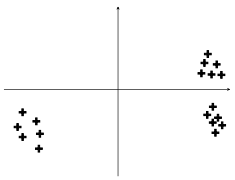
1. Describe how the true error $L_{\mathcal{D}}(h_s)$ of a predictor h_s can be decomposed into approximation ϵ_{app} and estimation ϵ_{est} error. Discuss how this decomposition is related to the complexity of the selected hypothesis class and draw a plot with the expected behaviour of the approximation, estimation, and true error, as functions of the complexity of the hypothesis class.
2. Formally introduce the Tikhonov regularization model writing down the function to be optimized. In relation with the previous answer, discuss how Tikhonov regularization can be used to control the fitting-stability trade-off. In particular discuss the behaviour of the approximation, estimation, and true error, as functions of the regularization parameter λ.



[Solution: Exercise 3]

Exercise 4 [8 points]

1. Introduce the k-means clustering problem and present the standard (i.e., Lloyd's) algorithm for its solution.
2. Assume that you need to cluster the points shown in the figure below into $k = 3$ different clusters using k-means. The final solution depends on the initialization of the centroids: suggest two possible initializations, one leading to the optimal solution and one leading to a sub-optimal solution.



[Solution: Exercise 4]

8.1 Exercise 1

1. Describe the general framework of binary classification.

Solution

The binary classification problem is a supervised learning problem with:

- Domain set X which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Label set $Y = \{-1, +1\}$, that defines the set of all possible labels in this case the labels are only 2

Given a training set $S = ((x_1, y_1)...(x_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ we need to choose an hypothesis class H, which defines the possible models or classification rules, from which we can pick our model to make predictions, and a loss function $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

The goal is to find an hypothesis $\hat{h} \in H$ with low generalization error: $L_d(\hat{h}) = E_{z \sim D}[l(\hat{h}, z)]$ where:

- D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples).

2. Discuss the use of the logistic regression model for binary classification and describe how it can be trained using stochastic gradient descent.

8.2 Exercise 2

1. Introduce the concept of separating hyperplane and describe how it leads to the formulation of hard SVM.

Solution

Given a binary classification problem where:

- Domain set X which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Label set $Y = \{-1, +1\}$, that defines the set of all possible labels in this case the labels are only 2
- set $S = ((x_1, y_1)...(x_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ is the so called training set

A separating hyperplane is an hyperplane that separates all the points of the training set S into two parts, namely into two halfspaces which contains all and only the points that have either class -1 or $+1$. In case in which the training set S is linearly separable, then there exists more than one such hyperplane and in particular we can define the optimal as the one that maximizes the margin i.e. the distance between the hyperplane and the closest points in S to him.

The optimal separating hyperplane introduced above is exactly the solution computed by the SVM when data are linearly separable. In particular if we define:

- hyperplane as: $L = \{v : \langle \vec{w}, \vec{v} \rangle + b = 0\}$
- distance between a point x and L: $d(x, L) = \min\{|\vec{x} - \vec{v}| : v \in L\}$

Then the optimal separating hyperplane can be computed as:

$$\underset{(\vec{w},b):\|\vec{w}\|=1}{\arg\max} \ \min_{i \in \{1,...,m\}} |\langle \vec{w}, \vec{x}_i \rangle + b| \text{ subject to: } \forall i \ y_i(\langle \vec{w}, \vec{x}_i \rangle + b)$$

2. In most practical situations data are not linearly separable. How can hard SVM be adapted to cope with this situation while still using a linear separation boundary? Formulate this adaptation in the form of an optimization problem, discussing the role and effect of the parameter

λ that trades the two components of the cost function.

Solution

Soft-SVM is similar to Hard-SVM, but can be used also when training data is not linearly separable (i.e., the training data cannot be perfectly classified with a linear model). Soft-SVM finds a model of large margin while allowing the training set to be inside the margin or wrongly classified. This is obtained by adding slack variables ξ_i to the constraints of hard SVM. The constraint for soft-SVM are then:

- $\xi_i \geq 0$ for each $i = 1, ..., m$
- $y_i(\langle \vec{w}, \vec{x}_i \rangle + b) \geq 1 - \xi_i$ for each (x_i, y_i) $i = 1, ..., m$ in the training set where $\vec{w}$ and b defines the model.

The interpretation of ξ_i is the following:

- if $\xi_i = 0$ then $\vec{x}_i$ is correctly classified and outside the margin;
- if $0 < \xi_i < 1$ then $\vec{x}_i$ is correctly classified and inside the margin;
- if $\xi_i \geq 1$ then $\vec{x}_i$ is wrongly classified.

The function optimized by soft-SVM consider both the margin and the "violation" of the hard-SVM given by ξ_i.

Therefore the optimization problem for soft-SVM can be described as follows:

Input: $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ and parameter $\lambda > 0$

Goal: $\min_{\vec{w},b,\vec{\xi}} \lambda \|\vec{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i$

Output: $\vec{w}, b, \vec{\xi}$

The parameter lambda controls the tradeoff between an high margin with several points misclassified and a small margin with most of the points correctly classified in particular, if $\lambda \approx 0$ then, soft-SVM will produce a model that tries to be as similar as possible as the one produced by hard-SVM namely, it tries to classified as many as possible points correctly. On the other hand, if $\lambda \gg 0$ then, soft-SVM cares almost to maximize the margin therefore, it produces a model that has most of the points misclassified but it has an high margin.

8.3 Exercise 3

With reference to the binary classification problem:

1. Describe how the true error $L_D(h_s)$ of a predictor h_s can be decomposed into approximation ϵ_{app} and estimation ϵ_{est} error. Discuss how this decomposition is related to the complexity of the selected hypothesis class and draw a plot with the expected behaviour of the approximation, estimation, and true error, as functions of the complexity of the hypothesis class.

Solution

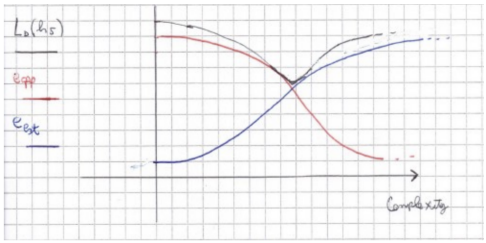
The generalization error $L_D(h_S)$ can be decomposed into $L_D(h_S) = \epsilon_{app} + \epsilon_{est}$ where $\epsilon_{app} = \min_{h \in H} L_D(h)$ and $\epsilon_{est} = L_D(h_S) - \min_{h \in H} L_D(h)$.

In words:

- ϵ_{app} = approximation error: is the minimum risk achievable by a predictor in the hypothesis class
- ϵ_{est} = estimation error: is the difference between the approximation error and the error achieved by the ERM predictor

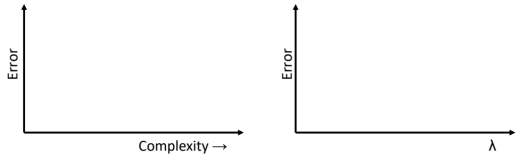
Therefore if we plot in a graph those two error and the $L_D(h_S)$ as functions of the complexity we obtain that:

- ϵ_{app}: will decrease if the complexity decreases if complexity increases
- ϵ_{est}: will increase if the complexity increases
- $L_D(h_S)$: has a "bell" behaviour



Notice that a low complexity hypothesis class corresponds to a underfitting situation while a high complexity class corresponds to a overfitting situation.

2. Formally introduce the Tikhonov regularization model writing down the function to be optimized. In relation with the previous answer, discuss how Tikhonov regularization can be used to control the fitting-stability trade-off. In particular discuss the behaviour of the approximation, estimation, and true error, as functions of the regularization parameter λ.



Solution

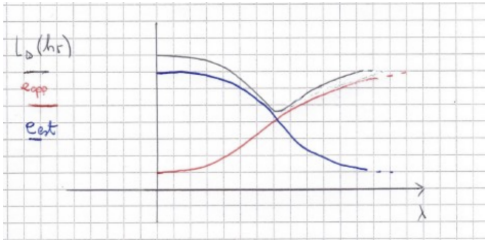
Tikhonov regularization is the Regularized loss minimization paradigm with regularization function $R(\vec{w}) = \lambda\|\vec{w}\|^2$ where $\lambda > 0$ and $\|\vec{w}\|^2$ is the l_2 norm of the vector $\vec{w}$.

Therefore the function to be optimized, i.e. minimized is the following: $\arg\min_{\vec{w}}(L_S(\vec{w})+\lambda\|\vec{w}\|^2)$ where $L_S(\vec{w})$ is the training error and S is the training set.

Notice that $\|\vec{w}\|^2$ measures the complexity of the hypothesis $\vec{w}$, while λ is the parameter used to tradeoff the empirical risk and the complexity of the model.

Now, the behavior of the ϵ_{app}, ϵ_{est} and $L_D(h_S)$ in terms of the parameter λ is the following:

- ϵ_{app} will increase if λ increases because with a low λ we have a complex hypothesis class
- ϵ_{est} will decrease if λ increases because with an high λ we have a low-complexity hypothesis class
- $L_D(h_S)$: has a "bell" behaviour



8.4 Exercise 4

1. Introduce the k-means clustering problem and present the standard (i.e., Lloyd's) algorithm for its solution.

Solution

K-means is a cost minimization clustering problem. Let $X' \subseteq X$ be the set of points to be clustered with $X = \{\vec{x}_1, ..., \vec{x}_m\}$ while X' is the space of possible points that we assume to be in $\mathbb{R}^d$ (i.e $X' = \mathbb{R}^d$). Let $k \in \mathbb{N}^+$ be the number of clusters, that is, with X, the input of the problem. Let $d(.)$ be a distance function: $d(\vec{x}, \vec{x'}) = \|\vec{x} - \vec{x'}\|$.

The goal is to find:

- A partition $C = (C_1, .., C_k)$ of X
- Centroids $\vec{\mu}_1, ..., \vec{\mu}_k$ of $C_1, .., C_k$ respectively

That minimizes the k-means cost function: $\sum_{i=1}^{k} \sum_{\vec{x} \in C_i} d(\vec{x}, \vec{\mu}_i)^2$

To find the solution of this problem we can use an heuristic algorithm called Lloyd's Algorithm. Such algorithm works as follows:

Input: data points $X = \{\vec{x}_1, ..., \vec{x}_m\}$ and $k \in \mathbb{N}^+$
Output: $C = (C_1, ..., C_k)$ of X and centroids $\vec{\mu}_1, ..., \vec{\mu}_k$ of $C_1, ..., C_k$ respectively

To produce such things the algorithm works as follows:

- Randomly initialize the centroids $\vec{\mu}_1, ..., \vec{\mu}_k$
- Until the convergence is not reached iteratively repeats the following steps:
  (a) Compute each cluster C_i by associating the points to him that closest to its centroids compared to others centroids.
  (b) Compute the new centroids for the next iterations.
  (c) If the convergence is reached then the output described above will be returned.

The pseudocode for such algorithm is the following:

1: Randomly choose $\vec{\mu}_1^0, ..., \vec{\mu}_k^0$
2: **for** t $<$ 0,... do **do**

3: **for** i = 1,...,k **do**
4: $C_i \leftarrow \{\vec{x} \in X : i = \arg\min_j d(\vec{x}, \vec{\mu}_j^t)\}$
5: **end for**
6: **for** i = 1,...,k **do**
7: $\vec{\mu}_i^{t+1} = \frac{1}{|C_i|} \sum_{\vec{x} \in C_i} \vec{x}$
8: **end for**
9: **if** convergence reached **then**
10: **return** $C = (C_1, ..., C_k), \vec{\mu}_1^{t+1}, ..., \vec{\mu}_k^{t+1}$
11: **end if**
12: **end for**

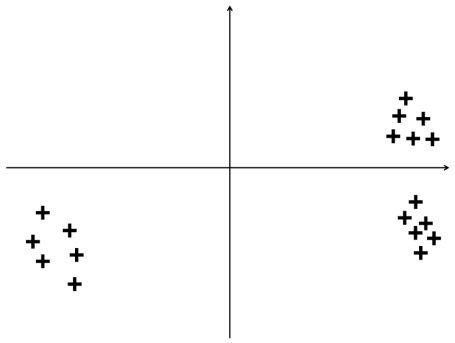
The convergence criteria that can be used for such algorithm are the following:

- The k-means objective function is no lower than the k-means objective function at the next iteration
- $\sum_{i=1}^{k} d(\vec{\mu}_i^{t+1}, \vec{\mu}_i^t) \leq \varepsilon$
- $\max_{1 \leq i \leq k} d(\vec{\mu}_i^{t+1}, \vec{\mu}_i^t) \leq \varepsilon$

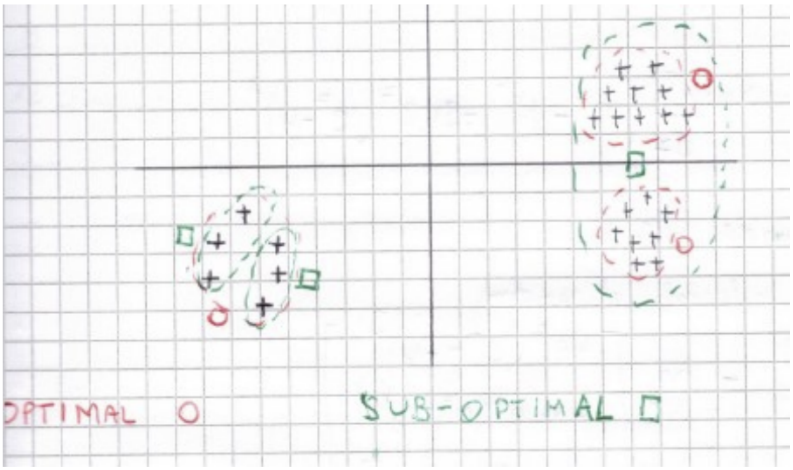
Note that if the first criteria is used then, the algorithm will always converge.

The time complexity is $O(tkmd)$ where $O(kmd)$ is due to the assignments of the points, while $O(md)$ is due to the computation of the centroids. Therefore the complexity depends on t which is the number of iterations of the algorithm.

2. Assume that you need to cluster the points shown in the figure below into $k = 3$ different clusters using k-means. The final solution depends on the initialization of the centroids: suggest two possible initializations, one leading to the optimal solution and one leading to a sub-optimal solution.



Solution



Exam 9
21 February 2020

EXAM 21/02/2020

Exercise 1 [8 points]

1. Provide the formulation of PAC learning (including the definition of loss, risk, training algorithms, etc.).
2. In the context of PAC learning define the concept of realizability and discuss how the formulation above changes when the realizability assumption cannot be made.
3. Provide a (probabilistic) bound on the generalization error for the ERM when working with finite hypothesis classes (proof not needed).

Exercise 2 [8 points]

1. Introduce the neural network model highlighting its main components and pointing out which are the parameters to be learned in the training process. Describe how the output layer of a neural network for binary classification can be designed.
2. Consider a fully connected neural network N with $L = 4$ layers, with $n_1 = 5$ neurons in the input layer, $n_2 = n_3 = 4$ neurons in the two inner layers and a single neuron ($n_4 = 1$) in the last (i.e., output) layer. How many trainable parameters are there in the network? How is this number related to the number of neurons in each layer?
3. Which provisions are used in the Convolutional Neural Network (CNN) model to reduce the number of trainable parameters? Highlight in your answer the differences with respect to the fully connected model.

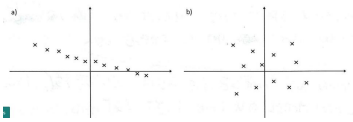
Exercise 3 [8 points]

With reference to the binary classification problem:

1. Describe a framework under which the decision boundary can be an arbitrary polynomial function of degree M.
2. Discuss how this can be related to kernel SVM, possibly highlighting which is the advantage of the "kernel" interpretation.
3. Assuming one has $\{\mathbf{x}_i, y_i\}$, $i \in [m]$ data points with m "small", describe a procedure to perform the selection of M (deciding the "most suitable" degree of the polynomial boundary), $M \in \{2, 3, 4, .., 10\}$.

Exercise 4 [8 points]

1. Introduce the problem of dimensionality reduction, describing what is the input, what is the output, and what is its goal.
2. Consider a linear regression problem with squared loss, where the input feature vectors are $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$. Assume that the (design) matrix $\mathbf{X}$, whose i-th row is $\mathbf{x}_i^T$ is such that $\mathbf{X}^T\mathbf{X}$ is almost singular. (Remember that a square matrix is singular if and only if it is not invertible.) Explain how dimensionality reduction can be used to reduce the number of parameters to be estimated and to learn a model in this situation.
3. Consider the two datasets with points $\mathbf{x} \in \mathbb{R}^2$ shown in Figure (a) and (b) below. For both datasets, describe i) whether it is possible to meaningfully reduce the dimensionality of the data and ii) what is the most appropriate dimension of the data after dimensionality reduction.



9.1 Exercise 1

1. Provide the formulation of PAC learning (including the definition of loss, risk, training algorithms, etc.).

Solution

An hypothesis class H is PAC (Probably Approximately Correct) learnable with respect to Z and a loss function $l : H \times Z \to \mathbb{R}^+$, if there exists a function $m_H : (0,1)^2 \to \mathbb{N}$ (sample complexity) and a learning algorithm such that for every $\delta \in (0,1)$, $\varepsilon \in (0,1)$, for every distribution D over Z and for every true labeling function $f : X \to \{0,1\}$, if the realizability assumption holds with respect to H, D, f; when running the learning algorithm on $m \geq m_H(\varepsilon, \delta)$ iid samples generated by D and labeled by f the algorithm returns an hypothesis h such that, with probability $\geq 1 - \delta$: $L_{D,f}(h) \leq \varepsilon$.

Where:

- $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ is the loss function namely a function that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value.
- A learning algorithm is a type of algorithm that given input data it learns a function that maps the input into output also for data it has not seen yet.
- $L_{D,f}(h) = E_{z \sim D}[l(h, z)]$ is the so called risk or generalization error.

2. In the context of PAC learning define the concept of realizability and discuss how the formulation above changes when the realizability assumption cannot be made.

Solution

The realizability assumption is the assumption that consists of having an hypothesis $h^* \in H$ such that $L_{D,f}(h^*) = 0$. If this assumption doesn't hold, then the formulation of PAC learning changes into agnostic PAC learning.

An hypothesis class H is agnostic PAC learnable with respect to Z and a loss function $l : H \times Z \to \mathbb{R}^+$, if there exists a function $m_H : (0,1)^2 \to \mathbb{N}$ (sample complexity) and a learning algorithm such that for every $\delta \in (0,1)$, $\varepsilon \in (0,1)$, for every distribution D over Z, when running the learning algorithm on $m \geq m_H(\varepsilon, \delta)$ iid samples generated by D the algorithm returns an hypothesis such that, with probability $\geq 1 - \delta$: $L_{D,f}(h) \leq \min_{h' \in H} L_{D,f}(h') + \varepsilon$ where $L_{D,f}(h) = E_{z \sim D}[l(h, z)]$.

3. Provide a (probabilistic) bound on the generalization error for the ERM when working with finite hypothesis classes (proof not needed).

Solution

Let H be a finite hypothesis class. Let $\delta \in (0,1)$, $\varepsilon \in (0,1)$, $m \in \mathbb{N}$ such that: $m \geq \frac{\log(|H|/\delta)}{\varepsilon}$, then for any labeling function f, for any distribution D for which the realizability assumption holds with probability $\geq 1 - \delta$ we have that for every ERM hypothesis h_S it holds that: $L_{D,f}(h_S) \leq \varepsilon$.

9.2 Exercise 2

1. Introduce the neural network model highlighting its main components and pointing out which are the parameters to be learned in the training process. Describe how the output layer of a neural network for binary classification can be designed.

Solution

Neural networks models are the most complex models that we can use. To use them we need to define the architecture of a neural network i.e. the graph specifics or practically speaking the number of edges and neurons of a neural network and the activation function.

Therefore the hypothesis class of neural networks becomes:

$H_{V,E,\sigma} = \{h_{V,E,\sigma,w} : \mathbb{R}^{|V_0|-1} \to \mathbb{R}^{|V_T|}$ where w is a mapping from E to R$\}$

So, the learning process of a neural network is aimed to learn the weights in order to define a function that maps the set of edges to the set of real numbers.

- Layers are not fully connected one's like for MLP. That means given a non fully connected layer of a CNN we have that the neurons of the next layer are connected with not all the neurons of the current one
- Parameters are shared in a layer. In addition to what we have said before, namely that the layers are not fully connected, the weights on such edges are the same for a certain subset of edges therefore, the numbers of parameter to learn decreases

In particular, about the graph specific, one important thing is that the output layer of the neural network must be composed by only two neurons for binary classification one used to highlighting the probability that the label of the input will be +1 while the other to highlighting the probability that the label of the input will be -1. This is due to the fact that, binary classification is a problem that has only two possible output values i.e. the values +1 or -1. For what concerns the other layers and the activation function, a good strategy to choose them, is to copy the architecture of neural networks already used in literature that works well with the problem of binary classification.

One neuron?

2. Consider a fully connected neural network N with $L = 4$ layers, with $n_1 = 5$ neurons in the input layer, $n_2 = n_3 = 4$ neurons in the two inner layers and a single neuron ($n_4 = 1$) in the last (i.e., output) layer. How many trainable parameters are there in the network? How is this number related to the number of neurons in each layer?

Solution

# of trainable parameters: $(n_1 + 1)\,(n_2)+(n_2 + 1)\,n_3+(n_3 + 1)\,n_4$

$$= 6 \cdot 4 + 5 \cdot 4 + 5 \cdot 1$$
$$= 24 + 20 + 5 = 49$$

3. Which provisions are used in the Convolutional Neural Network (CNN) model to reduce the number of trainable parameters? Highlight in your answer the differences with respect to the fully connected model.

Solution

The main provision done by CNN to reduce the number of parameters are the followings:

- Layers are not fully connected one's like for MLP. That means given a non fully connected layer of a CNN we have that the neurons of the next layer are connected with not all the neurons of the current one
- Parameters are shared in a layer. In addition to what we have said before, namely that the layers are not fully connected, the weights on such edges are the same for a certain subset of edges therefore, the numbers of parameter to learn decreases

9.3 Exercise 3

With reference to the binary classification problem:

1. Describe a framework under which the decision boundary can be an arbitrary polynomial function of degree M.

Solution

To have decision boundaries that are polynomial functions of degree M we can use two strategies:

- We can use SVM with polynomial kernels with Q=M
- We can use different representation of the data namely feature expansion/transformation where we map a vector $\vec{x}$ to a polynomial function of degree M in the features $x_i's$ of $\vec{x}$ and then use linear models with the new feature space.

  Ex $\vec{x} \in R \rightarrow \varphi(\vec{x}) = \begin{bmatrix} x_1 \\ . \\ . \\ x_M \end{bmatrix} \rightarrow$ linear model for the transoformed $\varphi(\vec{x}) = w_0 + w_1 x_1 + \cdots + w_M x_M$ and so for binary classification $sign(w_0 + w_1 x_1 + \cdots + w_M x_M)$

2. Discuss how this can be related to kernel SVM, possibly highlighting which is the advantage of the "kernel" interpretation.

Solution

What we have introduced in the previous point is related to Kernels SVM because to have polynomial functions as boundaries we use polynomial kernels with Q=M. In addition to that the main advantage is that we do not have to apply any transformation of the input because everything is done by the kernel.

3. Assuming one has $(x_i, y_i), i \in [m]$ data points with m "small", describe a procedure to perform the selection of M (deciding the "most suitable" degree of the polynomial boundary), $M \in \{2, 3, 4, .., 10\}$.

Solution

The strategy to select the best value for parameter M is to use cross-validation and once M is chosen we use all the data to learn the best model with such parameter.

To be more precise, cross validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have. To do so, we split the dataset into k different folds of size m/k (this quantity is supposed to be integer). Then for each value of the parameter ϑ we find the best model for all the possible k-1 folds that we can select. In addition to that, we compute the average error of each parameter as the average of the errors on the folds left out, and when we have repeated such procedure for all the values of the parameters, then we select the optimal one by choosing the one that minimizes the error computed for each parameter. To conclude, we train our model on all the dataset with the parameter found.

The pseudo code is the following:

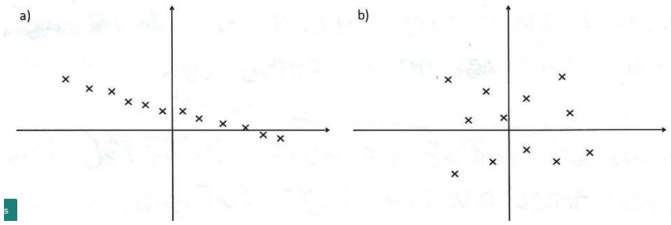
Input: $S = ((\bar{x}_1, y_1) \dots (\bar{x}_m, y_m))$; set of parameters θ; integer k; learning algorithm A
Split S into $S_1, \dots, S_k$
for $\theta \in \Theta$ **do**
    for $i = 1 \dots k$ **do**
        $h_{i,\theta} = A(S \setminus S_i; \theta)$
        $error(\theta) = \frac{1}{k} \sum_{i=1}^{k} L_{S_i}(h_{i,\theta})$
    end for
end for
Output: $\theta^* = \arg\min_\theta(error(\theta))$
$h_{\theta^*} = A(S; \theta^*)$

9.4 Exercise 4

1. Introduce the problem of dimensionality reduction, describing what is the input, what is the output, and what is its goal.

2. Consider a linear regression problem with squared loss, where the input feature vectors are $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$. Assume that the (design) matrix $\mathbf{X}$, whose i-th row is $\mathbf{x}_i^T$ is such that $\mathbf{X}^T\mathbf{X}$ is almost singular. (Remember that a square matrix is singular if and only if it is not invertible.) Explain how dimensionality reduction can be used to reduce the number of parameters to be estimated and to learn a model in this situation.

3. Consider the two datasets with points $\mathbf{x} \in \mathbb{R}^2$ shown in Figure (a) and (b) below. For both datasets, describe i) whether it is possible to meaningfully reduce the dimensionality of the data and ii) what is the most appropriate dimension of the data after dimensionality reduction.



Exam 10
7 February 2020

Exercise 1 [8 points]

Consider the regression problem when the training data is $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_m, y_m)\}$ with $\mathbf{x}_i = [x_{i,1}, x_{i,2}] \in \mathbb{R}^2$ and $y_i \in \mathbb{R}$ for $i = 1, \dots, m$.

1. Formally define the problem when the hypothesis class is $\mathcal{H} = \{h(\mathbf{x}) \text{ s.t. } h(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2; w_0, w_1, w_2 \in \mathbb{R}\}$ and the squared loss is used. In particular, describe what the goal is.
2. Describe ℓ_2 regularization within the context described above. Given an hypothesis $h \in \mathcal{H}$, let $L_S(h)$ be its training error (i.e., the average loss on the training set), $J(h) = L_S(h) + \lambda R(h)$ be the regularized training error, where $R(h)$ is the ℓ_2 regularization function. Formally define the regularization function for ℓ_2 regularization and *derive* the hypothesis that minimizes the ℓ_2 regularized training error.
3. Given an hypothesis h, let $\mathcal{L}(h)$ be the true (or generalization) error of h. Let h_S be the hypothesis that, given data S, minimizes the regularized training error $J(h)$. Plot the typical behavior of $L_S(h_S)$ and $\mathcal{L}(h_S)$ as a function of $\lambda \geq 0$, and describe how this is linked to overfitting.

Exercise 2 [8 points]

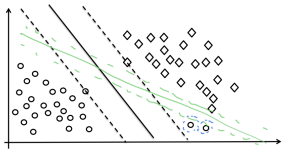
Consider the classification problem in machine learning.

1. Provide a formal definition, describing data, loss functions, classification rules etc.
2. Assuming inputs (or features) $x \in \mathbb{R}$, and consider the model class, which is a modified version of logistic regression, defined as the set of models obtained composing the sigmoid function
$$\frac{1}{1+e^{-z}}$$
with the function
$$z = h_{\mathbf{w}}(x) = w_1 + w_2 x^2$$
where the parameters are $\mathbf{w} = [w_1,\ w_2]^\top \in \mathbb{R}^2$. Assume the loss is $\ell(h, (x_i, y_i)) = (y_i - h(x_i))$ if $y_i = 1$, and $\ell(h, (x_i, y_i)) = h(x_i)$ if $y_i = 0$. Write the stoochastic gradient descent update for learning this model from data (x_i, y_i), $i = 1, .., m$.

Exercise 3 [8 points]

The Soft-SVM classifier aims at minimizing the following function: $\lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_i \xi_i$.

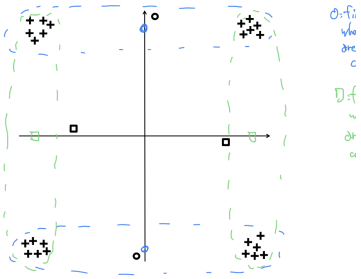
1. Briefly explain how the Soft-SVM classification method works and which are the constraints under which the function has to be minimized.
2. The figure shows the results of a binary classification performed using a Soft-SVM model with parameter $\lambda = 1$. The training samples are the circles and diamonds and the two shapes correspond to the two classes to which the samples belong. The solid line is the computed separating hyperplane, while the dotted lines represent the margins. For which points ξ_i is different from 0?
3. Does the margin increase or decrease when λ decreases? Guess how the solution changes when a very small value for the λ parameter (i.e., $\lambda \approx 0$) is used, and draw an estimate of the separating hyperplane that could be obtained in this case.



Exercise 4 [8 points]

1. Briefly introduce the clustering problem.
2. Define and explain the cost function used in K-means clustering.
3. Consider the data in the figure below where each point $\mathbf{x} \in \mathbb{R}^2$ is represented by a cross. Show the results (i.e., draw approximately the final centroid locations and the final assignment of the points to the clusters) of clustering into $k = 2$ clusters with K-means when
   (a) the initial centers for the algorithm are the circles;
   (b) the initial centers for the algorithm are the squares.

   Is one solution *significantly* better than the other one? Briefly motivate your answer.



10.1 Exercise 1

Consider the regression problem when the training data is $S = \{(x_1, y_1), (x_2, y_2), \dots (x_m, y_m)\}$ with $x_i = [x_{i,1}, x_{i,2}] \in \mathbb{R}^2$ and $y_i \in \mathbb{R}$ for $i = 1, \dots, m$.

1. Formally define the problem when the hypothesis class is $\mathcal{H} = \{h(x) \text{ s.t. } h(x) = w_0 + w_1 x_1 + w_2 x_2; w_0, w_1, w_2 \in \mathbb{R}\}$ and the squared loss is used. In particular, describe what the goal is.

Solution

Given the training data $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$, with $x_i \in \mathbb{R}^2$, $y_i \in \mathbb{R}$ for $i = 1, \dots, m$, our goal is to find a hypothesis $\hat{h}$ that minimizes the generalization error. That is, we want to find $\hat{h}$ such that

$$E_{(x,y)\sim D}[\ell(\hat{h}, (x, y))]$$

is minimized, where

$$\ell(\hat{h}, (x, y)) = (\hat{h}(x) - y)^2$$

and D is the (unknown) probability distribution from which $(x_i, y_i) \in S$, $i = 1, \dots, m$ have been drawn independently.

2. Describe ℓ_2 regularization within the context described above. Given an hypothesis $h \in \mathcal{H}$, let $L_S(h)$ be its training error (i.e., the average loss on the training set), $J(h) = L_S(h) + \lambda R(h)$ be the regularized training error, where $R(h)$ is the ℓ_2 regularization function. Formally define the regularization function for ℓ_2 regularization and *derive* the hypothesis that minimizes the ℓ_2 regularized training error.

Solution

l_1 regularization in the context above corresponds to ridge regression. In particular, the design matrix is

$$X = \begin{bmatrix} -\bar{X}_1^1 \\ -\bar{X}_2^1 \\ -\bar{X}_3^1 \end{bmatrix}, \text{ where } \bar{X}_i^1 = [1, x_{i_1}, x_{i_2}] \text{ and } \bar{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}.$$

Let $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$. Since an hypothesis block is defined by the coefficient vector $\vec{w}$, we have that the training error is

$$L_s(\hat{h}) = L_s(\vec{w}) = \frac{1}{m} \left(\vec{Y} - X\vec{w}\right)^T \left(\vec{Y} - X\vec{w}\right),$$

and minimizing $L_s(\hat{h})$ (or $L_s(\vec{w})$) corresponds to minimizing

$$\left(\vec{Y} - X\vec{w}\right)^T \left(\vec{Y} - X\vec{w}\right),$$

In ridge regression, the regularization function $R(\hat{h}) = \|\vec{w}\|^2$, therefore, the hypothesis that minimizes the regularized training error $L_s(\hat{h}) + \lambda R(\hat{h})$ is

$$\arg\min_{\vec{w}} \left(\left(\vec{Y} - X\vec{w}\right)^T \left(\vec{Y} - X\vec{w}\right) + \lambda \vec{w}^T \vec{w} \right).$$

To find $\vec{w}$, we compute the gradient and set it to zero:

$$\frac{\partial(L_s(\vec{w}) + \lambda R(\vec{w}))}{\partial \vec{w}} = 2\lambda\vec{w} - 2X^T(\vec{y} - X\vec{w}) = 0.$$

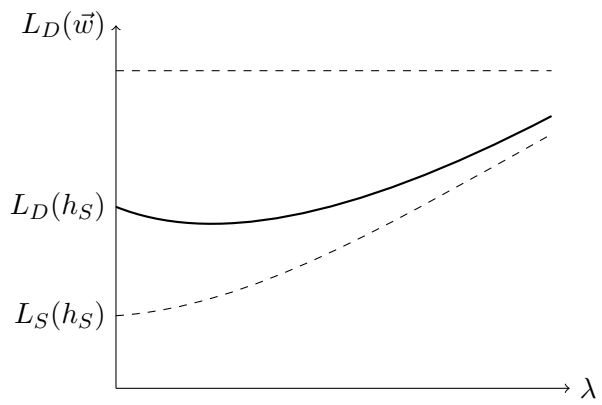
This leads to the equation:

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

Note that $\lambda I + X^T X$ is positive definite, thus invertible.

3. Given an hypothesis h, let $\mathcal{L}(h)$ be the true (or generalization) error of h. Let h_s be the hypothesis that, given data S, minimizes the regularized training error $J(h)$. Plot the typical behavior of $L_S(h_s)$ and $\mathcal{L}(h_s)$ as a function of $\lambda \geq 0$, and describe how this is linked to overfitting.

Solution



For low values of λ we are ignoring the complexity of h_S and only care about minimizing the training error, thus we may find h_S with low training error $L_S(h_S)$ but high generalization error $L_D(h_S)$, which corresponds to overfitting. Increasing λ we take the complexity of h_S more and more into account, and for some value of λ we may have a good balance between training error and complexity, thus providing an hypothesis h_S with lower generalization error.

10.2 Exercise 2

Consider the classification problem in machine learning.

1. Provide a formal definition, describing data, loss functions, classification rules etc.

Solution

The classification problem is a supervised learning problem, with:

- domain set X, that is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance, and is usually represented by a vector of features
- label set Y, that defines the set of possible labels. Y is a discrete set in classification. For example, in binary classification Y = {-1,1} or similar

Given training data $S = ((\vec{x}_1, y_1), ..., (\vec{x}_m, y_m))$, with $\vec{x}_i \in X$, $y_i \in Y \ \forall i = 1, ..., m$, we need to choose an hypothesis class H, which defines the possible models or classification rules we can pick to make prediction, and a loss function $l : H \times Z \to \mathbb{R}^+$, with $Z = X \times Y$, that given an hypothesis provides a measure of how much we lose by predicting the label $h(\vec{x})$ for $\vec{x}$ instead of the (correct) label y. The goal is then to find an hypothesis $h \in H$ with low generalization error, defined as:

$L_D(h) = E_{z \sim D}[l(h, z)]$

where D is the unknown probability distribution over Z from which $(\vec{x}_i, y_i) \in S$, $i = 1, ..., m$, have been drawn (as independent samples).

2. Assuming inputs (or features) $x \in \mathbb{R}$, and consider the model class, which is a modified version of logistic regression, defined as the set of models obtained composing the sigmoid function

$$\frac{1}{1 + e^{-z}}$$

with the function

$$z = h_w(x) = w_1 + w_2 x^2$$

where the parameters are $\mathbf{w} = [w_1, w_2]^\top \in \mathbb{R}^2$. Assume the loss is $\ell(h, (x_i, y_i)) = (y_i - h(x_i))$ if $y_i = 1$, and $\ell(h, (x_i, y_i)) = h(x_i)$ if $y_i = 0$. Write the stochastic gradient descent update for learning this model from data (x_i, y_i), $i = 1, ..., m$.

Solution

In general, for stochastic gradient descent (SGD), let $\vec{w}^{(t)}$ be the weights defining the model in iteration t. Then the update rule is given by:

- pick $(\vec{x}_i, y_i) \in S$ uniformly at random
- $\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta \nabla l(\vec{w}^{(t)}, (\vec{x}_i, y_i))$

We need to compute $\nabla l(\vec{w}^{(t)}, (\vec{x}_i, y_i))$ for specific model class and loss we are considering. Each model h in our model class is a function of the form:

$$h(x) = \frac{1}{1 + e^{-(w_1 + w_2 x^2)}}$$

Since the loss depends on the value of y_i, the gradient will depend on y_i as well.

Let's consider the two cases:

i) $y_i = 1 \Rightarrow \nabla l(h(x), (x, y)) = \left[\frac{\partial l}{\partial w_1}, \frac{\partial l}{\partial w_2}\right]^T$, and:

$\frac{\partial l}{\partial w_1} = \frac{\partial z}{\partial w_1} \cdot \frac{\partial l}{\partial z} = 1 \cdot \frac{\partial}{\partial z}(1 - \frac{1}{1+e^z}) = -\frac{e^z}{(1+e^z)^2}$, with $z = w_1 + w_2 x^2$

$\frac{\partial l}{\partial w_2} = \frac{\partial z}{\partial w_2} \cdot \frac{\partial l}{\partial z} = x^2 \cdot (-\frac{e^z}{(1+e^z)^2}) = -x^2 \frac{e^z}{(1+e^z)^2}$

ii) $y_i = 0 \Rightarrow \nabla l(h(x), (x, y)) = \left[\frac{\partial l}{\partial w_1}, \frac{\partial l}{\partial w_2}\right]^T$, and:

$\frac{\partial l}{\partial w_1} = \frac{\partial z}{\partial w_1} \cdot \frac{\partial l}{\partial z} = 1 \cdot \frac{\partial}{\partial z}(\frac{1}{1+e^z}) = \frac{e^z}{(1+e^z)^2}$

$\frac{\partial l}{\partial w_2} = \frac{\partial z}{\partial w_2} \cdot \frac{\partial l}{\partial z} = x^2 \frac{e^z}{(1+e^z)^2}$

Therefore the SGD update rule is:

- pick $(\vec{x}_i, y_i) \in S$ uniformly at random;
- $z \leftarrow w_1^{(t)} + w_2^{(t)} x_i^2$
- if $y_i = 1$ then $\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} + \eta \begin{bmatrix} e^z/(1+e^z)^2 \\ x^2 e^z/(1+e^z)^2 \end{bmatrix}$
- else $\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta \begin{bmatrix} e^z/(1+e^z)^2 \\ x^2 e^z/(1+e^z)^2 \end{bmatrix}$

10.3 Exercise 3

The Soft-SVM classifier aims at minimizing the following function:

$$\lambda||\mathbf{w}||^2 + \frac{1}{m}\sum_i \xi_i.$$

1. Briefly explain how the Soft-SVM classification method works and which are the constraints under which the function has to be minimized.

Solution

Soft-SVM is similar to hard-SVM, but can be used also when the training data is not linearly separable, (i.e., the training data cannot be perfectly classified with a linear model). Soft-SVM finds a model of large margin while allowing some points of the training set to be inside the margin or wrongly classified. This is obtained by adding slack variables ξ_i to the constraints of hard-SVM. The constraints for soft-SVM are then:

- $\xi_i \geq 0$ for each $i = 1, ..., m$
- $y_i(\langle \vec{w}, \vec{x}_i \rangle + b) \geq 1 - \xi_i$ for each point $(\vec{x}_i, y_i)$, $i = 1, ..., m$ in the training set, where $\vec{w}$ and b define the model.

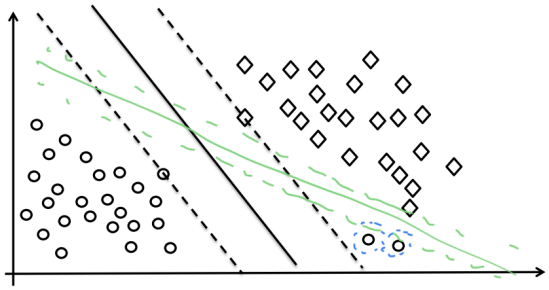
The interpretation of ξ_i is the following: if $\xi_i = 0$, then $\vec{x}_i$ is correctly classified and outside the margin; if $0 < \xi_i < 1$, then $\vec{x}_i$ is correctly classified but inside the margin. If $\xi_i \geq 1$ then $\vec{x}_i$ is incorrectly classified. The function optimized by soft-SVM considers both the margin and the "violation" of the hard-SVM given by ξ_i.

2. The figure shows the results of a binary classification performed using a Soft-SVM model with parameter $\lambda = 1$. The training samples are the circles and diamonds and the two shapes correspond to the two classes to which the samples belong. The solid line is the computed separating hyperplane, while the dotted lines represent the margins. For which points ξ_i is different from 0?

Solution

See the points circled in blue in the figure. Since they are not correctly classified, the corresponding $\xi_i's$ are > 0.

3. Does the margin increase or decrease when λ decreases? Guess how the solution changes when a very small value for the λ parameter (i.e., $\lambda \approx 0$) is used, and draw an estimate of the separating hyperplane that could be obtained in this case.



Solution

When λ decreases, the term $\frac{1}{m}\sum_{i=1}^{m}\xi_i$ becomes more important in the function minimized by soft-SVM, therefore a model that reduces the values of $\xi_i's$ is sought, even if the margin is smaller. In particular, for the dataset shown in the figure, when $\lambda \approx 0$ the margin is very small, since there is a linear model (shown in green in the figure) of small margin but that correctly classifies all points in the training set, so that $\xi_i = 0\ \forall i = 1, ..., m$.

10.4 Exercise 4

1. Briefly introduce the clustering problem.

Solution

Clustering is the problem of grouping a set of objects such that similar objects end up in the same group and dissimilar objects are separated into different groups.

More formally, the problem has the following input and output:

Input: set X of objects and a distance function $d : X \times X \to \mathbb{R}^+$

Output: a partition of X into clusters, that is $C = (C_1, C_2, ..., C_k)$ such that:

- $\bigcup_{i=1}^{k} C_i = X$
- for all $i \neq j : C_i \cap C_j = \emptyset$

(Sometimes the input includes the number k of clusters.)

The partition to be produced in output depends on the specific definition of the problem, and is sometimes captured by defining a cost for a given clustering.

2. Define and explain the cost function used in K-means clustering.

Solution

Let the input data points be $\vec{x}_1, \vec{x}_2, ..., \vec{x}_m$ with $\vec{x}_i \in \mathbb{R}^d$ for $i = 1, ..., m$. The k-means cost function is:

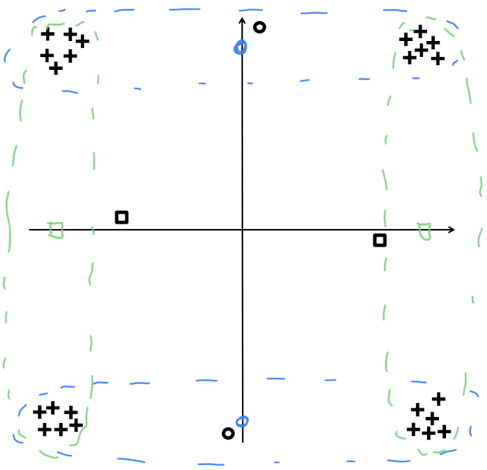
$$\sum_{i=1}^{k} \sum_{\vec{x} \in C_i} d(\vec{x}, \vec{\mu}_i)^2$$

where $C_1, ..., C_k$ are the clusters, $d(\cdot, \cdot)$ is the Euclidean distance in $\mathbb{R}^d$, and $\vec{\mu}_i$ is the center of cluster C_i for $i = 1, ..., k$ (the centers are part of the output). Therefore, the cost of a clustering for k-means is defined as the sum of the squares of the distances of each point to the center of the cluster it belongs to.

3. Consider the data in the figure below where each point $x \in \mathbb{R}^2$ is represented by a cross. Show the results (i.e., draw approximately the final centroid locations and the final assignment of the points to the clusters) of clustering into $k = 2$ clusters with K-means when

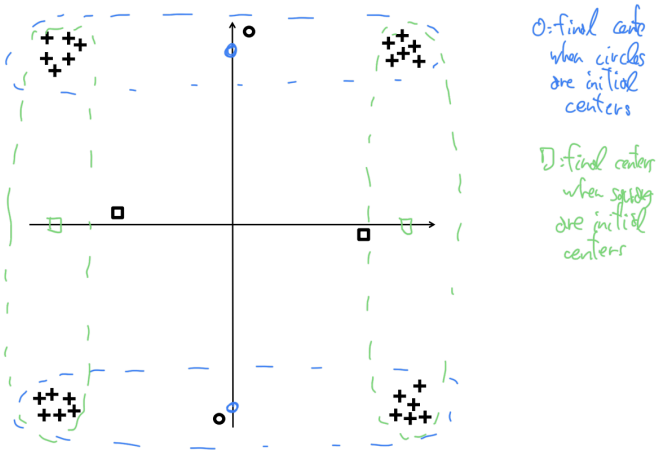
   - (a) the initial centers for the algorithm are the circles;
   - (b) the initial centers for the algorithm are the squares.

   Is one solution *significantly* better than the other one? Briefly motivate your answer.



Solution

The solutions are shown in the figure. There is no solution that is significantly better than the other one, since the data consists of 4 groups of points that are essentially symmetric with respect to the origin, and the 2 solutions are just 2 different ways to group them into 2 groups. Also, the silhouette coefficient of the 2 solutions is probably very similar.



Exam 11
4 September 2019

Exercise 1 [8 points]

1. With reference to the binary classification problem, introduce the concepts of model class, loss function, empirical risk and (expected) risk.
2. In the context above, provide the formulation of PAC learning.
3. Discuss the role that the model class complexity plays in determining sample complexity.

[Solution: Exercise 1]

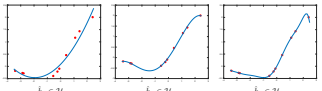
Exercise 2 [8 points]

With reference to the regression problem:

1. Formulate the problem of estimating a function $h(x) : \mathbb{R}^d \to \mathbb{R}$ under the squared loss
2. Assume you have to choose among the three model classes $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$; let

$$\dot{h}_i := \arg\min_{h \in \mathcal{H}_i} L_S(h)$$

be as in the figure below. Which of the three model classes would you immediately discard and how would you choose between the other two?



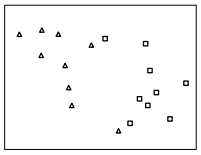
$\dot{h}_1 \in \mathcal{H}_1$ $\dot{h}_2 \in \mathcal{H}_2$ $\dot{h}_3 \in \mathcal{H}_3$

[Solution: Exercise 2]

Exercise 3 [8 points]

1. Describe hard SVM for binary classification, highlighting how it is different from "standard" linear methods (e.g., the perceptron).
2. Describe the difference between hard SVM and soft SVM, using the objective function of soft SVM for the comparison.
3. The following figure shows an input for soft SVM for binary classification on the data points (in $\mathbb{R}^2$) in the figure, where the class of each point is represented by its shape (triangle or square). Let λ be the (regularization) parameter for the objective function of soft SVM. Draw in the figure below (approximate) solutions for the soft SVM when:
   (a) the value of λ is ≈ 0;
   (b) the value of λ is high.

   Explain the reasoning you followed to derive the solution.

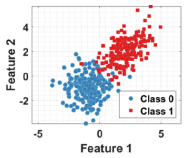


[Solution: Exercise 3]

Exercise 4 [8 points]

Reducing the dimensionality of data can help machine learning tools to achieve better performances. Various techniques for this task have been proposed.

- Present a dimensionality reduction technique a briefly explain how it works.
- Assume you need to classify the data in the Figure using a machine learning algorithm. Firstly describe what happens when you apply your dimensionality reduction technique to the data in the Figure in order to convert them from the bi-dimensional representation in the Figure to a one dimensional representation (you can use a drawing to show the transformation).
- Does your dimensionality reduction approach allow to use a simpler classifier in order to recognize the two classes? Explain how could you classify the data before and after applying your dimensionality reduction tool.



[Solution: Exercise 4]

11.1 Exercise 1

Consider the binary classification problem:

1. With reference to the binary classification problem, introduce the concepts of model class, loss function, empirical risk and (expected) risk.

Solution

The binary classification problem is a supervised learning problem with:

- Domain set X which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Label set $Y = \{-1, +1\}$, that defines the set of all possible labels in this case the labels are only 2
- Model class H: is the set of functions that represents a mapping from X to Y
- Loss function: is a function $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ that, given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y
- Generalization error: $L_d(h) = E_{z \sim D}[l(h, z)]$ where D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples)
- Training error: $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, (x_i, y_i))$ where $S = ((x_1, y_1)...(x_m, y_m))$ is the training set

2. In the context above, provide the formulation of PAC learning.

Solution

An hypothesis class H is PAC learnable with respect to Z and a loss function $l : H \times Z \to \mathbb{R}^+$, if there exists a function $m_H : (0,1)^2 \to \mathbb{N}$ (sample complexity) and a learning algorithm such that for every $\delta \in (0,1)$, $\varepsilon \in (0,1)$, for every distribution D over Z and for every true labeling function $f : X \to \{0,1\}$, if the realizability assumption holds with respect to H, D, f; when running the learning algorithm on $m \geq m_H(\varepsilon, \delta)$ iid samples generated by D and labeled by f the algorithm returns an hypothesis h such that, with probability $\geq 1 - \delta$: $L_{D,f}(h) \leq \varepsilon$.

3. Discuss the role that the model class complexity plays in determining sample complexity.

Solution

For instance, let H be an hypothesis class from domain X to 0,1 and consider the 0-1 loss. Assume $VCdim(H) = d < +\infty$. Then there are two absolute constants C_1, C_2 such that:

$$C_1 \frac{(d + \log(\frac{1}{\delta}))}{\varepsilon^2} \leq m_H(\varepsilon, \delta) \leq C_2 \frac{(d + \log(\frac{1}{\delta}))}{\varepsilon^2}$$

This shows that the sample complexity $m_H(\varepsilon, \delta)$ grows:

- Linearly with the VC-dimension d of the hypothesis class
- Logarithmically with $\frac{1}{\delta}$
- Quadratically with $\frac{1}{\varepsilon}$

11.2 Exercise 2

With reference to the regression problem:

1. Formulate the problem of estimating a function $h(x) : \mathbb{R}^d \to \mathbb{R}$ under the squared loss.

Solution

Given:

- Domain set X which is the set of all possible objects to make predictions about, where a domain point $x \in X$ is called instance and in this case is equal to $\mathbb{R}^d$
- Label set $Y = \mathbb{R}$

Given a training set $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ with $\vec{x}_i \in X, y_i \in Y \ \forall i = 1, .., m$ we need to choose an hypothesis class H, which defines the possible models or classification rules, from which we can pick our model to make predictions, and a loss function $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

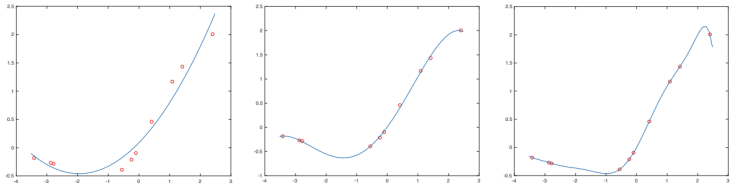
The goal is to find an hypothesis $\hat{h} \in H$ with low generalization error: $L_d(\hat{h}) = E_{z \sim D}[l(\hat{h}, z)]$ where:

- D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples)
- $l(h, (\vec{x}, y) = (h(\vec{x}) - y)^2$ is the squared loss function

2. Assume you have to choose among the three model classes $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$; let

$$\hat{h}_i := \arg\min_{h \in \mathcal{H}_i} L_S(h)$$

be as in the figure below. Which of the three model classes would you immediately discard and how would you choose between the other two?



$\hat{h}_1 \in \mathcal{H}_1$ $\hat{h}_2 \in \mathcal{H}_2$ $\hat{h}_3 \in \mathcal{H}_3$

Solution

I will immediately discard $\hat{h}_1$ because it is the one with highest error. Between $\hat{h}_2$ and $\hat{h}_3$ I will choose $\hat{h}_2$ because it is not overfitting the data points.

11.3 Exercise 3

1. Describe hard SVM for binary classification, highlighting how it is different from "standard" linear methods (e.g., the perceptron).

Solution

SVM is a supervised learning method that is used to find a solution for the binary classification problem. Hard-SVM is the formulation of SVM that works when data are linearly separable, namely, with reference to binary classification, when $\forall i = 1, ..., m \; y_i(\langle \vec{w}, \vec{x}_i \rangle + b) > 0$ or equivalently when there exists an halfspace $(\vec{w}, \vec{x})$ such that $y_i = sign(\langle \vec{w}, \vec{x}_i \rangle + b) \; \forall i = 1, ..., m$ where $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ is the training set used to train the SVM.

The objective of the SVM is not only to find a separating hyperplane that perfectly classifies the data as done for other types of methods like perceptron, but it also finds the one that maximizes the margin i.e. the distance between the hyperplane and the closest sample to it in the training set.

Therefore the problem solved by Hard-SVM can be stated as follows:

Input: $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ linearly separable dataset

Goal: $\arg\max_{(\vec{w},b)} \min_{i \in \{1,...,m\}} |\langle \vec{w}, \vec{x}_i \rangle + b|$ subject to $\forall i = 1, ..., m \; y_i(\langle \vec{w}, \vec{x}_i \rangle + b) \geq 1 \; \|\vec{w}\| = 1$

Output: $\frac{\vec{w}}{\|\vec{w}\|}, \frac{b}{\|\vec{w}\|}$

2. Describe the difference between hard SVM and soft SVM, using the objective function of soft SVM for the comparison.

Solution

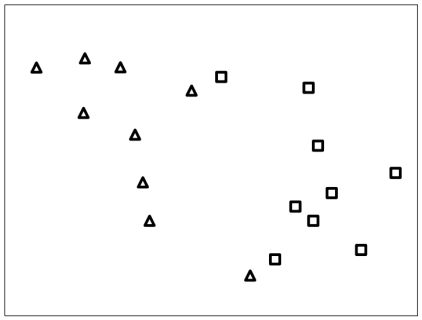
As we can see the main difference between the Soft-SVM formulation and Hard-SVM formulation can be found in the constraint and mainly in the objective function. In fact, the objective function of Soft-SVM is: $\lambda\|\vec{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i$ where ξ_i are slack variables. As we can see, the soft-SVM formulation has two terms in the objective function instead of one. This thing is due to the fact that, hard-SVM is only able to treat linearly separable dataset while the soft formulation is also able to find a solution for non linearly separable one by allowing some violation in the constraints with the slack variables. Violations that can be described through the following constraints: $\forall i = 1, ..., m\ y_i(\langle\vec{w}, \vec{x}_i\rangle + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$.

3. The following figure shows an input for soft SVM for binary classification on the data points (in $\mathbb{R}^2$) in the figure, where the class of each point is represented by its shape (triangle or square). Let λ be the (regularization) parameter for the objective function of soft SVM. Draw in the figure below (approximate) solutions for the soft SVM when:

   (a) the value of λ is ≈ 0;

   (b) the value of λ is high.

   Explain the reasoning you followed to derive the solution.

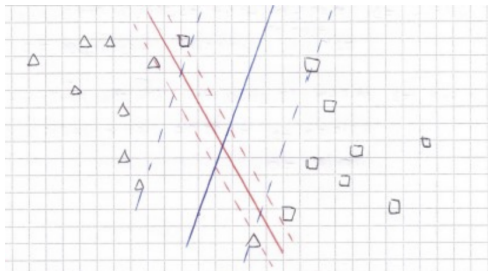


Solution

The reasoning behind the solution is the following:

When $\lambda \approx 0$ the soft-SVM works likely the hard SVM so it tries to classify correctly all the points without caring so very less about the maximization of the margin.

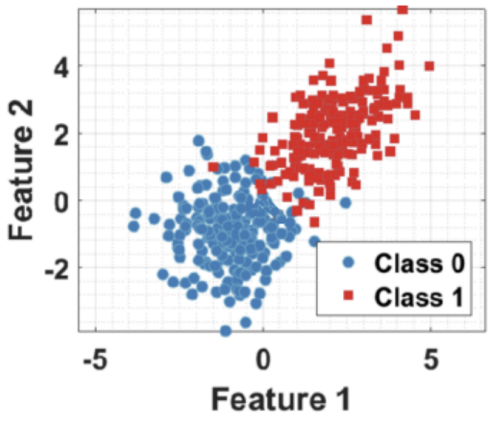
When $\lambda \gg 0$ then, the soft-SVM emphasizes more a bigger margin than all the points correctly classified (i.e. having errors).



11.4 Exercise 4

Reducing the dimensionality of data can help machine learning tools to achieve better performances. Various techniques for this task have been proposed.

- Present a dimensionality reduction technique a briefly explain how it works.
- Assume you need to classify the data in the Figure using a machine learning algorithm. Firstly describe what happens when you apply your dimensionality reduction technique to the data in the Figure in order to convert them from the bi-dimensional representation in the Figure to a one dimensional representation (you can use a drawing to show the transformation).
- Does your dimensionality reduction approach allow to use a simpler classifier in order to recognize the two classes? Explain how could you classify the data before and after applying your dimensionality reduction tool.



Exam 12
1 July 2019

Exercise 1 [8 points]

1. Formulate the supervised learning problem, highlighting its main objective in terms of Generalization Error (= Expected Risk) and Training Error (= Empirical Risk).
2. Discuss one approach to pursue this objective for a given finite data set (x_i, y_i), $i = 1, ..., n$.
3. Denoting with $\hat{\theta}_n$ the estimated model with n training samples, and with θ^* the best model in the given class, draw in Fig. 1 the typical behaviour of the Generalization Error of θ^*, the Generalization Error of $\hat{\theta}_n$ and the Training Error of $\hat{\theta}_n$ as a function of the number of samples n.

[Solution: Exercise 1]

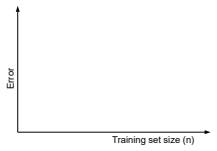


Figure 1

Exercise 2 [8 points]

You want to cluster the points in the figure below using k-means, with $k = 3$.

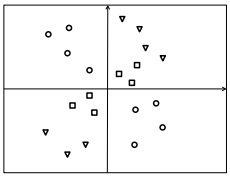
1. Is there a way to cluster the points using k-means so that in the solution the three clusters corresponds to the three sets with different marks (triangles, squares, circles)? Given a short explanation for your answer.
2. Before clustering, you can apply a transformation to the dataset. Describe a transformation such that the application of k-means with $k = 3$ to the transformed dataset results in three clusters corresponding to the three sets with different marks, and plot the transformed dataset.
3. Assume that instead of a clustering that minimizes the k-means objective (cost), you are interested in a clustering minimizing the following objective:

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \boldsymbol{\mu}_i)^3$$

where the distance $d(\mathbf{x}, \boldsymbol{\mu}), \mathbf{x} \in \mathbb{R}^d, \boldsymbol{\mu} \in \mathbb{R}^d$ is defined as:

$$d(\mathbf{x}, \boldsymbol{\mu}) = \left(\sum_{j=1}^{d} (x_j - \mu_j)^2 \right)^{1/3}.$$

What is a good rule to choose (or update) the cluster centers for a given assignment of points to clusters (i.e., once the assignment of points to clusters is fixed)? Briefly motivate your answer.



Exercise 3 [8 points]

1. Consider the neural network in the figure and assume that a Rectified Linear Unit (ReLU) activation function

$$\sigma(x) = \max(0, x)$$

is used for all neurons. Compute the value of the output y when the input $\mathbf{z}$ is the vector $[z_1 \ z_2] = [1 \ 3]$.

2. Describe how a neural network can be trained using the backpropagation algorithm (only the main structure of the algorithm; details of the derivation are not required).
3. The Rectified Linear Unit (ReLU) is typically preferred to hyperbolic tangent or sigmoid activation functions. With reference to the gradient descent algorithm used for training the neural network, which are the main advantages of the ReLU with respect to the other two activation functions ?

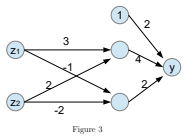


Figure 3

Exercise 4 [8 points]

Three classifiers C_1, C_2 and C_3 are trained iteratively using gradient descent minimizing the training error. Let us denote with k the iteration index.

1. Fig. 4a shows the final models for the different binary classifiers on the training set (the circles and crosses correspond to the 2 different classes). In Fig. 4b the behaviour of the training error as a function of the iteration step k is shown. Associate each curve to the corresponding classifier and justify your answer.
2. Assume that you are given a validation dataset (not used to train C_1, C_2, C_3). Fig. 4c shows the behaviour of the validation error as a function of the iteration step k. Associate each curve in Fig. 4c to the corresponding classifier and justify your answer, pointing out which classifier is overfitting or underfitting?
3. Which is a possible technique to reduce the overfitting issue?

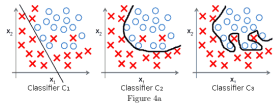


Figure 4a

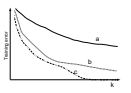


Figure 4b

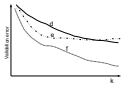


Figure 4c

12.1 Exercise 1

1. Formulate the supervised learning problem, highlighting its main objective in terms of Generalization Error (= Expected Risk) and Training Error (= Empirical Risk).

Solution

The supervised learning problem is the problem to learn a function $h : X \to Y$ where:

- X be the domain set, which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Y be the label set that defines the set of all possible labels
- H be the hypothesis class

The function $\hat{h}$ that we need to pick from H must be the one with the lowest generalization error i.e. $L_d(\hat{h}) = E_{z\sim D}[l(\hat{h}, z)]$ where:

- D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples)
- $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ be the loss function namely a function that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the

However because we do not know D, under certain hypothesizes, a good estimate of $L_d(h)$ is given by the training error namely: $L_S(h) = \frac{1}{m}\sum_{i=1}^{m} l(h, (\vec{x}_i, y_i))$ where $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ is the training set.

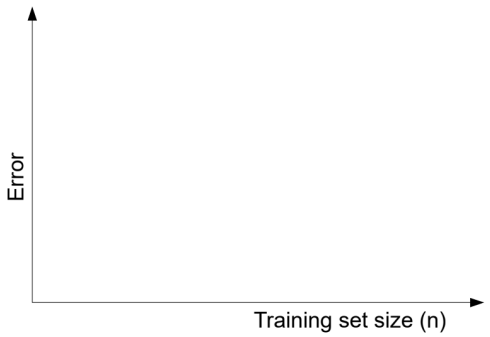
2. Discuss one approach to pursue this objective for a given finite data set (x_i, y_i), $i = 1, ..., n$.

Solution

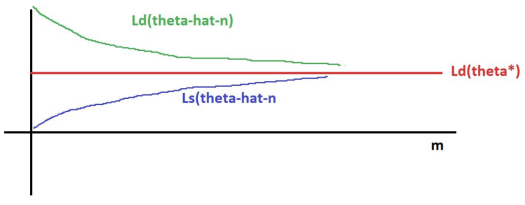
One way to pursue the objective explained in point 1, is, as already introduce in point 1 as well, to use the so called Empirical Risk Minimization (ERM for short). So given a training set $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ with $\vec{x}_i \in X, y_i \in Y \ \forall i = 1, .., m$ to find $\hat{h}$ that has the lowest generalization error we can find the hypothesis that minimizes the training error namely: $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, (\vec{x}_i, y_i))$ where $l : H \times Z \to \mathbb{R}^+$ is the loss function.

Notice that this paradigm works only under certain hypothesizes regarding the hypothesis class H.

3. Denoting with $\hat{\theta}_n$ the estimated model with n training samples, and with θ^* the best model in the given class, draw in Fig. 1 the typical behaviour of the Generalization Error of θ^*, the Generalization Error of $\hat{\theta}_n$ and the Training Error of $\hat{\theta}_n$ as a function of the number of samples n.



Solution



12.2 Exercise 2

You want to cluster the points in the figure below using k-means, with k = 3.

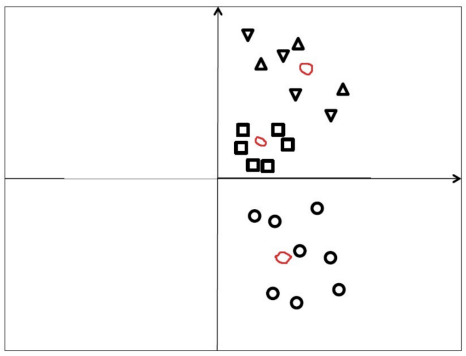
1. Is there a way to cluster the points using k-means so that in the solution the three clusters corresponds to the three sets with different marks (triangles, squares, circles)? Given a short explanation for your answer.

Solution

No there is no way to cluster the △, ◯, □ to obtain a solution where all the △, ◯, □ are separated because with k-means objective function we minimize the distance between points and centers.

2. Before clustering, you can apply a transformation to the dataset. Describe a transformation such that the application of k-means with k = 3 to the transformed dataset results in three clusters corresponding to the three sets with different marks, and plot the transformed dataset.

Solution



Rule to obtain such clustering:

Let $\vec{x} \in X$ then:

- If ($x_1 < 0$ and $x_2 > 0$) then $\vec{x}' = [|x_1|, -|x_2|]$
- else if ($x_1 < 0$ and $x_2 < 0$) then $\vec{x}' = [|x_1|, |x_2|]$
- else then leave as it is

thearpage

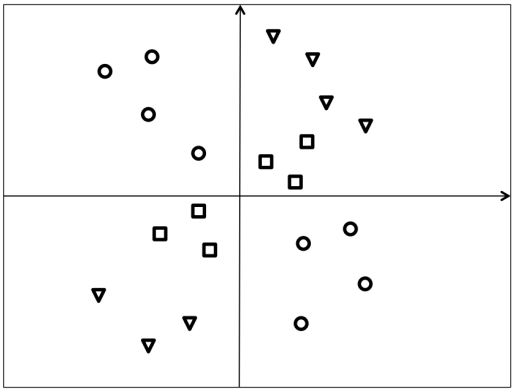
3. Assume that instead of a clustering that minimizes the k-means objective (cost), you are interested in a clustering minimizing the following objective:

$$\sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^3$$

where the distance $d(x, \mu), x \in \mathbb{R}^d, \mu \in \mathbb{R}^d$ is defined as:

$$d(x, \mu) = \left(\sum_{j=1}^{d} (x_j - \mu_j)^2 \right)^{1/3}$$

What is a good rule to choose (or update) the cluster centers for a given assignment of points to clusters (i.e., once the assignment of points to clusters is fixed)? Briefly motivate your answer.



Solution

To identify a good rule for such objective function we must compute the derivative of it with respect to $\vec{\mu}_j$ and then put it equal to 0.

So:

$$\begin{aligned}
\frac{\partial}{\partial \vec{\mu}_j}\left(\sum_{i=1}^{k}\sum_{\vec{x}\in C_i} d(\vec{x},\vec{\mu}_i)^3\right) &= \sum_{i=1}^{k}\frac{\partial}{\partial \vec{\mu}_j}\sum_{\vec{x}\in C_i} d(\vec{x},\vec{\mu}_i)^3 \\
&= \sum_{\vec{x}\in C_j}\frac{\partial}{\partial \vec{\mu}_j}(d(\vec{x},\vec{\mu}_j)^3) \\
&= \sum_{\vec{x}\in C_j}\frac{\partial}{\partial \vec{\mu}_j}\left((\sum_{l=1}^{d}(x_l-\mu_{j,l})^2)^{\frac{1}{3}}\right)^3 \\
&= \sum_{\vec{x}\in C_j}\frac{\partial}{\partial \vec{\mu}_j} d(\vec{x}-\vec{\mu}_j)^2 \\
&= 2|C_j|\vec{\mu}_j - 2\sum_{\vec{x}\in C_j}\vec{x}
\end{aligned}$$

At the optimum:

$$2|C_j|\vec{\mu}_j - 2\sum_{\vec{x}\in C_j}\vec{x} = 0$$

Therefore:

$$\vec{\mu}_j = \frac{1}{|C_j|}\sum_{\vec{x}\in C_j}\vec{x}$$

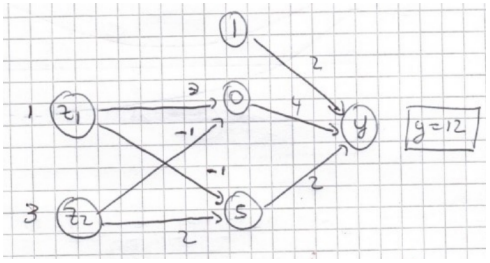
12.3 Exercise 3

1. Consider the neural network in the figure and assume that a Rectified Linear Unit (ReLU) activation function

$$\sigma(x) = \max(0, x)$$

is used for all neurons. Compute the value of the output y when the input z is the vector $[z_1\ z_2] = [1\ 3]$.

Solution



Given $\sigma(z) = \max(0, z)$ and $\vec{z} = [1, 3]$

Let's compute step by step the output y according to the network:

First layer output (hidden nodes):

- Top node: $3 \cdot 1 + (-1) \cdot 3 = 0$ after ReLU: $\max(0, 0) = 0$
- Bottom node: $2 \cdot 1 + (-2) \cdot 3 = 2 - 6 = -4$ after ReLU: $\max(0, -4) = 0$

Final output:

- $y = 4 \cdot 0 + 2 \cdot 0 + 2 \cdot 1 = 2$

Therefore the output is y = 12.

2. Describe how a neural network can be trained using the backpropagation algorithm (only the main structure of the algorithm; details of the derivation are not required).

Solution

The back propagation algorithm is the algorithm to train a neural network based on stochastic gradient descent. The main structure of the algorithm is the following:

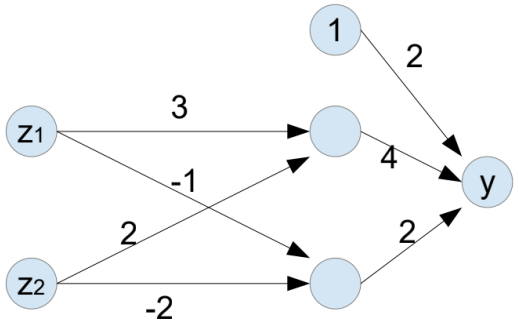
Input: Training data $(\vec{x}_1, y_1)...(\vec{x}_m, y_m)$ and a neural network with no weights

Output: a neural network with weights $w_{i,j}^{(t)}$ $\forall i, j, t$

And it works as follows:

- Randomly initialize the weights $w_{i,j}^{(t)}$ $\forall i, j, t$
- Until convergence is reached then repeat the following steps: 1. Pick a point $\vec{x}_k, y_k$ randomly from the training data 2. Apply the forward propagation algorithm for such data and so compute $v_{t,j}$ $\forall j, t$ 3. Compute the sensitivity vectors to make the updates of the weights namely: $\delta_j^{(t)}$ $\forall j, t$ 4. Update the weights with the following rule: $w_{i,j}^{(t+1)} = w_{i,j}^{(t)} - \eta v_{t-1,j} \delta_j^{(t)}$ $\forall i, j, t$
- If convergence is reached then return all the weights $w_{i,j}^{(t)}$ $\forall i, j, t$

3. The Rectified Linear Unit (ReLU) is typically preferred to hyperbolic tangent or sigmoid activation functions. With reference to the gradient descent algorithm used for training the neural network, which are the main advantages of the ReLU with respect to the other two activation functions?



Solution

The main reasons why the ReLU function is used are the followings:

- It is simpler to compute
- It is as powerful as others
- It does not suffer of the vanishing gradient problem which is common problem in deep learning when using activation functions like the sigmoid or the hyperbolic tangent. Such problem consists of having a roughly 0 value as value of the delta terms to update the weights, on deep levels when applying the back propagation algorithm, and therefore such value will propagates to the shallow levels and so weights are not updated.

12.4 Exercise 4

Three classifiers C_1, C_2 and C_3 are trained iteratively using gradient descent minimizing the training error. Let us denote with k the iteration index.

1. Fig. 4a shows the final models for the different binary classifiers on the training set (the circles and crosses correspond to the 2 different classes). In Fig. 4b the behaviour of the training error as a function of the iteration step k is shown. Associate each curve to the corresponding classifier and justify your answer.

   Solution

   - Classifier C_1 is associated to CURVE 'a' because C_1 is the exact opposite of a classifier that suffers of underfitting, namely, the model produce is too vague and has an high training error.
   - Classifier C_3 is associated to CURVE 'c' because C_3 is the exact reproduction of a classifier that suffers of overfitting, namely it is learning every point of the training set exactly and therefore has a 0 training error at the end of the training iterations.
   - Classifier C_2 is associated to CURVE 'b' because it is the perfect balance between a classifier that predicts anything (like TS) correctly and one that is too vague.

2. Assume that you are given a validation dataset (not used to train C_1, C_2, C_3). Fig. 4c shows the behaviour of the validation error as a function of the iteration step k. Associate each curve in Fig. 4c to the corresponding classifier and justify your answer, pointing out which classifier is overfitting or underfitting?

   Solution

   - Classifier C_1 is associated to curve 'd', namely it is UNDERFITTING both validation and training error are high.
   - Classifier C_2 is associated to curve 'f', namely it has a quite reasonable training error and also validation error.
   - Classifier C_3 is associated to curve 'e', namely it is OVERFITTING because it has an high validation error but a small training error.

3. Which is a possible technique to reduce the overfitting issue?

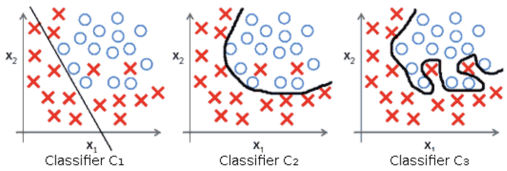


Figure 4a

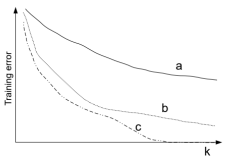


Figure 4b

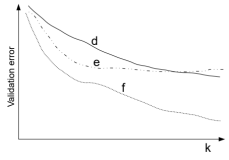


Figure 4c

Solution

One strategy to reduce overfitting issue is to use cross-validation to choose the classifier C_i and then, once C_i is chosen we train C_i over all the data. Therefore recalling i, ϑ, we can apply the cross validation method.

To be more precise, cross validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have. To do so, we split the dataset into k different folds of size m/k (this quantity is supposed to be integer). Then for each value of the parameter ϑ we find the best model for all the possible k-1 folds that we can select. In addition to that, we compute the average error of each parameter as the average of the errors on the folds left out, and when we have repeated such procedure for all the values of the parameters, then we select the optimal one by choosing the one that minimizes the error computed for each parameter. To conclude, we train our model on all the dataset with the parameter found.

The pseudo code is the following:

Input: $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$; set of parameters Θ; integer k; learning algorithm A

Split S into $S_1, ..., S_k$
for all $\vartheta \in \Theta$ **do**
    for i = 1 ... k **do**
        $h_{i,\vartheta} = A(S \setminus S_i; \vartheta)$
        $error(\vartheta) = \frac{1}{k} \sum_{i=1}^{k} L_{S_i}(h_{i,\vartheta})$
    end for
end for
Output: $\vartheta^* = \arg\min_{\vartheta}(error(\vartheta))$
        $h_{\vartheta^*} = A(S; \vartheta^*)$

Assuming that we have enough data another strategy is to use validation to choose C_i and then, once C_i is chosen, we use all the data to train C_i.

To be more precise validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have. To do so we split out dataset into 2 parts training set and validation set. Then, for each value of the parameter that we have, we find the best model for every possible values of the parameter on the training set. Then among such models that correspond to a value of the parameter we select the one that minimizes the validation error. The model obtain then is trained on the entire dataset.

Notice that training error is computed as:
$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, (\vec{x}_i, y_i))$
where S is the training set, namely, $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$, $l : H \times Z \to \mathbb{R}^+$ is the loss function where $Z = X \times Y$, that, given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

The validation error instead is computed as:
$L_V(h) = \frac{1}{mv} \sum_{i=1}^{mv} l(h, (\vec{x}_i, y_i))$
where V is the validation set, namely $V = ((\vec{x}_1, y_1)...(\vec{x}_{mv}, y_{mv}))$, $l : H \times Z \to \mathbb{R}^+$ is the loss function where $Z = X \times Y$, that, given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

Exam 13
31 January 2019

Exercise 1 [8 points]

In the context of supervised learning:

1. provide the definition of the regression task
2. consider the following model class that is linear in the parameter:

$$h(x) := \mathbf{w}^\top \Psi(x) \quad \Psi(x) = [\psi_1(x), .., \psi_L(x)]^\top \quad x \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^L$$

where $\Psi(x) = [\psi_1(x), .., \psi_L(x)]^\top$ can be a generic function, e.g., recall the polynomial regression case where $\Psi(x) = [1, x, x^2, ..., x^{L-1}]^\top$. Write the explicit expression of the least squares estimator of $\mathbf{w}$ given data (x_k, y_k), $k = 1, .., m$.

3. Recalling the answer to the previous question, consider the one-hidden-layer neural network

$$h(x) := \sum_{i=1}^{L} w_i \sigma(\alpha_i (x - \beta_i)) \quad x \in \mathbb{R}$$

where α_i, w_i, β_i, $i = 1, .., L$, are the network parameters. Show that for α_i and β_i fixed, the optimal w_i can be found in closed form under the square loss.

[Solution: Exercise 1]

Exercise 2 [8 points]

Consider a generic machine learning problem and assume that a regularized loss function has been used by the selected algorithm A. In the loss function the relevance of the regularization term is controlled by a parameter λ. Let us denote with $h_{A\lambda}$ the solution found by algorithm A and with $L_S(h_{A\lambda})$ its empirical risk while the true risk (generalization error) of $h_{A\lambda}$ is $L_D(h_{A\lambda})$.

1. Which is the impact of the λ parameter on the empirical risk $L_S(h_{A\lambda})$ of the solution found by A?
2. Which is the expected behavior of the true risk $L_D(h_{A\lambda})$ of the found solution as a function of the λ parameter?
3. Describe how the behavior of the empirical risk and of the true risk in the answers to the previous questions are related to the bias-complexity trade-off.

[Solution: Exercise 2]

Exercise 3 [8 points]

Consider a classification problem with 0-1 loss.

1. Provide the definition of VC dimension $VCdim(\mathcal{H})$ of a hypothesis set $\mathcal{H}$, and of empirical error and true risk (generalization error) for an arbitrary hypothesis $h \in \mathcal{H}$. What is the relation between the empirical error and the true risk in terms of the VC dimension of $\mathcal{H}$?
2. Consider the hypothesis set $\mathcal{H}$ defined as: $\mathcal{H} = \{h_{a,b} : a, b \in \mathbb{R}, a < b\}$ where $h_{a,b} : \mathbb{R} \mapsto \{0, 1\}$ is

$$h_{a,b}(x) = \begin{cases} 1 & \text{if } x \leq a \; OR \; x \geq b \\ 0 & \text{otherwise} \end{cases}$$

What's the value of $VCdim(\mathcal{H})$? Provide a proof of your claim.

3. Assume that you have many hypothesis sets, denoted by $\mathcal{H}_i, i = 1, 2, \ldots, n$. Describe one strategy to choose a good hypothesis set $\mathcal{H}_i$ and a good model $h_i \in \mathcal{H}_i$.

[Solution: Exercise 3]

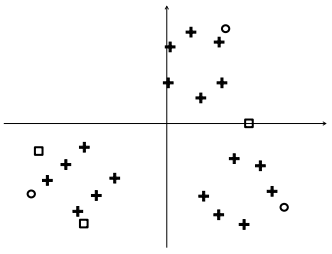
[Solution: Exercise 3]

Exercise 4 [8 points]

Consider the problem of clustering.

1. Introduce the k-means clustering problem, rigorously defining its cost function.
2. Consider Lloyd's algorithm. What is the rule that is used to update the cluster centers after the points are assigned to clusters? Prove that such rule minimizes the k-means cost for the given assignment of points to clusters (i.e., once the assignment of points to clusters is fixed).
3. Consider the data in the figure below where each point $\mathbf{x} \in \mathbb{R}^2$ is represented by a cross. Draw (approximately) the output of Lloyd's algorithm for $k = 3$ when
   (a) the initial centers for the algorithm are the circles;
   (b) the initial centers for the algorithm are the squares.

   Which one of the two resulting clusterings has a lower cost?



13.1 Exercise 1

In the context of supervised learning:

1. provide the definition of the regression task

Solution

Regression task is a supervised learning task with: - domain set $\mathcal{X}$ - label set $\mathcal{Y} = \mathbb{R}$

Given training data $S = ((x_1, y_1), ..., (x_m, y_m))$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ $\forall i = 1, ..., m$, we need to define an hypothesis class $\mathcal{H}$ and a loss function $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$, where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ that given an hypothesis $h \in \mathcal{H}$ provides a measure of how much we lose by predicting the value $h(x)$ for x instead of the value y. The goal is then to find an hypothesis $h \in \mathcal{H}$ with low generalization error $L_{\mathbb{D}}(h) = \mathbb{E}_{z \sim \mathbb{D}}[\ell(h, z)]$ where $\mathbb{D}$ is the (unknown) probability distribution over $\mathcal{Z}$ from which $(x_i, y_i), i = 1, ..., m$ have been drawn (as independent samples).

2. consider the following model class that is linear in the parameter:

$$h(x) := \mathbf{w}^\top \Psi(x) \quad \Psi(x) = [\psi_1(x), .., \psi_L(x)]^\top \quad x \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^L$$

where $\Psi(x) = [\psi_1(x), .., \psi_L(x)]^\top$ can be a generic function, e.g., recall the polynomial regression case where $\Psi(x) = [1, x, x^2, ..., x^{L-1}]^\top$. Write the explicit expression of the least squares estimator of $\mathbf{w}$ given data $(x_k, y_k), k = 1, ..m$.

Solution

Since the model is linear in the parameter, the least square estimator is analogous to the least square estimator for linear models (i.e., $h(\vec{x}) = \vec{w}^\top \vec{x}$), with $\vec{\Psi}(x)$ playing the role of $\vec{x}$. In particular, let:

$$X' = \begin{bmatrix} \vec{\Psi}(x_1)^\top \\ \vec{\Psi}(x_2)^\top \\ \vdots \\ \vec{\Psi}(x_m)^\top \end{bmatrix} \text{ and } \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Then the least square estimator is: $\vec{w} = (X'^\top X')^{-1} X'^\top \vec{y}$ $(*)$

3. Recalling the answer to the previous question, consider the one-hidden-layer neural network

$$h(x) := \sum_{i=1}^{L} w_i \sigma(\alpha_i (x - \beta_i)) \quad x \in \mathbb{R}$$

where $\alpha_i, w_i, \beta_i, i = 1, .., L$, are the network parameters. Show that for α_i and β_i fixed, the optimal w_i can be found in closed form under the square loss.

Solution

Since α_i and β_i are fixed, $\sigma(\alpha_i(x - \beta_i))$ is a generic function $\Psi_i(x) = \sigma(\alpha_i(x-\beta_i))$. Therefore $h(x) = \sum_{i=1}^{L} w_i\sigma(\alpha_i(x-\beta_i)) = \vec{w}^\top \Psi(x)$

where $\vec{w} = [w_1, w_2, ..., w_L]^\top$ and $\vec{\Psi}(x) = [\Psi_1(x), \Psi_2(x), ..., \Psi_L(x)]^\top$

Since the optimal $\vec{w}$ for a linear model under the square loss is the least squares estimator, from 2. the optimal $\vec{w}$ (and therefore the optimal w_i's) can be found in closed form as $(*)$.

13.2 Exercise 2

Consider a generic machine learning problem and assume that a regularized loss function has been used by the selected algorithm A. In the loss function the relevance of the regularization term is controlled by a parameter λ. Let us denote with h_A the solution found by algorithm A and with $L_S(h_A)$ its empirical risk while the true risk (generalization error) of h_A is $L_D(h_A)$.

1. Which is the impact of the λ parameter on the empirical risk $L_S(h_A)$ of the solution found by A?

   Solution

   λ controls the trade-off between the empirical risk $L_S(h_A)$ and the complexity of h_A. If $\lambda = 0$, h_A is the hypothesis of minimum empirical risk, but may suffer from overfitting. As $\lambda \to +\infty$, $L_S(h_A)$ increases till it reaches the empirical risk of the hypothesis h^* of minimum complexity (the complexity depends on the regularization function).

   The following plot shows the relation between λ and $L_S(h_A)$: [Plot showing $L_S(h_A)$ increasing with λ until reaching $L_S(h^*)$]

2. Which is the expected behavior of the true risk $L_D(h_A)$ of the found solution as a function of the λ parameter?

   Solution

   For $\lambda = 0$, we have that h_A is complex and may be subject to overfitting, so $L_{\mathbb{D}}(h_A)$ is fairly high. Then as λ increases, $L_{\mathbb{D}}(h_A)$ decreases until reaching its minimum value, and then increases again due to underfitting, reaching the generalization error of the hypothesis h^* of minimum complexity. For $\lambda \to +\infty$, $h^* = h_A$ is chosen independently of the data therefore $L_S(h^*) = L_{\mathbb{D}}(h^*)$

   The plot below describes the relation between λ and $L_{\mathbb{D}}(h_A)$: [U-shaped plot showing how $L_{\mathbb{D}}(h_A)$ varies with λ]

3. Describe how the behavior of the empirical risk and of the true risk in the answers to the previous questions are related to the bias-complexity trade-off.

   Solution

   λ is controlling the trade-off between the bias and the complexity. $L_{\mathbb{D}}(h_A) = \varepsilon_{app} + \varepsilon_{est}$, where $\varepsilon_{app} = \min_{h \in \mathcal{H}} L_{\mathbb{D}}(h)$ and $\varepsilon_{est} = L_{\mathbb{D}}(h_A) - \varepsilon_{app}$. λ is essentially defining the complexity of

$\mathcal{H}$, so for $\lambda = 0$ we have that ε_{app} is small and ε_{est} is large, due to having a complex $\mathcal{H}$ that results in a small $L_S(h_A)$ but still a large $L_{\mathbb{D}}(h_A)$. For $\lambda \to +\infty$, $\mathcal{H}$ has very low complexity (it consists of the hypothesis of smallest complexity) so ε_{app} is large while ε_{est} is small so $L_{\mathbb{D}}(h_A) = L_S(h_A)$ and $L_{\mathbb{D}}(h_A)$ is large. Starting from $\lambda = 0$, ε_{est} decreases and ε_{app} increases until the best choice of λ where ε_{app} and ε_{est} are somehow balanced (and $L_{\mathbb{D}}(h_A)$ is minimized) then by increasing λ more ε_{est} decreases and ε_{app} increases but $L_{\mathbb{D}}(h_A)$ increases.

13.3 Exercise 3

Consider a classification problem with 0-1 loss.

1. Provide the definition of VC dimension $VCdim(\mathcal{H})$ of a hypothesis set $\mathcal{H}$, and of empirical error and true risk (generalization error) for an arbitrary hypothesis $h \in \mathcal{H}$. What is the relation between the empirical error and the true risk in terms of the VC dimension of $\mathcal{H}$?

Solution

Let $h \in \mathcal{H}$ be such that $h : \mathcal{X} \to \{0, 1\}$. Let $C = \{c_1, ..., c_m\}$ with $C \subset \mathcal{X}$. The restriction $\mathcal{H}_C$ of $\mathcal{H}$ to C is $\mathcal{H}_C = \{[h(c_1), ..., h(c_m)] : h \in \mathcal{H}\}$. We say that $\mathcal{H}$ shatters C if $|\mathcal{H}_C| = 2^m$, that is, $\mathcal{H}_C$ contains all $2^m = 2^{|C|}$ functions from C to $\{0, 1\}$.

The VC-dimension $VCdim(\mathcal{H})$ of $\mathcal{H}$ is the maximal size of a set $C \subset \mathcal{X}$ that can be shattered by $\mathcal{H}$; if $\mathcal{H}$ can shatter sets of arbitrary large size then $VCdim(\mathcal{H}) = +\infty$.

Let the training set S be $S = ((x_1, y_1), ..., (x_m, y_m)), x_i \in \mathcal{X}, y_i \in \{0, 1\} \forall i \in \{1, ..., m\}$. Let $\ell(h, (x, y))$ be the 0-1 loss: $\ell(h, (x, y)) = \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{otherwise} \end{cases}$ Let $\mathbb{D}$ be the unknown probability distribution from which $(x_i, y_i), i \in \{1, ..., m\}$ is drawn independently from the other samples. Given an hypothesis $h \in \mathcal{H}$, the empirical risk is: $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h, (x_i, y_i))$ and the true risk is: $L_{\mathbb{D}}(h) = \mathbb{E}_{(x_i, y_i) \sim \mathbb{D}}[\ell(h, (x_i, y_i))]$.

For any $h \in \mathcal{H}$ we have: $L_{\mathbb{D}}(h) \leq L_S(h) + C\sqrt{\frac{VCdim(\mathcal{H}) + \log(1/\delta)}{2m}}$ where C is a universal constant.

2. Consider the hypothesis set $\mathcal{H}$ defined as: $\mathcal{H} = \{h_{a,b} : a, b \in \mathbb{R}, a < b\}$ where $h_{a,b} : \mathbb{R} \mapsto \{0, 1\}$ is

$$h_{a,b}(x) = \begin{cases} 1 & \text{if } x \leq a \text{ OR } x \geq b \\ 0 & \text{otherwise} \end{cases}$$

What's the value of $VCdim(\mathcal{H})$? Provide a proof of your claim.

Solution

$VCdim(\mathcal{H}) = 2$. Let's prove it:

- $VCdim(\mathcal{H}) \geq 2$: let's take two arbitrary points $x_1, x_2 \in \mathbb{R}$ with $x_1 < x_2$. Then the following shows that $\{x_1, x_2\}$ can be shattered by $\mathcal{H}$: [Diagrams showing different labelings]

- $VCdim(\mathcal{H}) < 3$: let's take 3 arbitrary points $x_1, x_2, x_3 \in \mathbb{R}$ with $x_1 < x_2 < x_3$. The following assignment of labels cannot be achieved: [Diagram showing impossible labeling $x_1 : 0, x_2 : 1, x_3 : 0$]

3. Assume that you have many hypothesis sets, denoted by $\mathcal{H}_i, i = 1, 2, ..., n$. Describe one strategy to choose a good hypothesis set $\mathcal{H}_i$ and a good model $\hat{h}_i \in \mathcal{H}_i$.

Solution

One strategy is to use cross-validation to choose $\mathcal{H}_i$ and then, once $\mathcal{H}_i$ is chosen, to use all the data to learn $\hat{h}_i \in \mathcal{H}_i$.

[Short explanation of how cross-validation works.]

Alternative answer: given enough data, we can split in training & validation, learn best model $h_i^* \in \mathcal{H}_i$ using the training set, and then choose $i = \arg\min_{j \in \{1,...,n\}} L_V(h_j^*)$, where $L_V(h_j^*)$ is the error of h_j^* on the validation set. Once i (and, therefore $\mathcal{H}_i$) is chosen, all the data is used to learn the model $\hat{h}_i \in \mathcal{H}_i$.

13.4 Exercise 4

Consider the problem of clustering.

1. Introduce the k-means clustering problem, rigorously defining its cost function.

Solution

k-means is a cost minimization clustering problem. Let $X \subseteq X'$ be the set of points to be clustered, with $X = \{\vec{x}_1, ..., \vec{x}_m\}$ while X' is the space of possible points that we assume to be $\mathbb{R}^d$ (i.e., $X' = \mathbb{R}^d$). Let $k \in \mathbb{N}^+$ be the number of clusters, that is, with k the input of the problem. Let $d(\cdot)$ be the distance function: $d(\vec{x}, \vec{x}') = \|\vec{x} - \vec{x}'\|$. The goal of the k-means clustering problem is to find:

- a partition $C = (C_1, C_2, ..., C_k)$ of X - centroids $\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_k$ of $C_1, C_2, ..., C_k$ respectively

that minimizes the k-means cost function:

$$\sum_{i=1}^{k} \sum_{\vec{x} \in C_i} d(\vec{x}, \vec{\mu}_i)^2$$

2. Consider Lloyd's algorithm. What is the rule that is used to update the cluster centers after the points are assigned to clusters? Prove that such rule minimizes the k-means cost for the given assignment of points to clusters (i.e., once the assignment of points to clusters is fixed).

Solution

The rule that Lloyd's algorithm uses to update clusters centers after the points assigned to clusters is:

$$\vec{\mu}_i \leftarrow \frac{1}{|C_i|} \sum_{\vec{x} \in C_i} \vec{x}$$

Proof: Let consider the cost function as a function of $\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_k$: $f(\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_k) = \sum_{i=1}^{k} \sum_{\vec{x} \in C_i} d(\vec{x}, \vec{\mu}_i)^2$. At the optimum, the gradient is equal to $\vec{0}$. Let's compute a part of the gradient, in particular $\frac{\partial f}{\partial \vec{\mu}_j}$:

$$\begin{aligned}
\frac{\partial f}{\partial \vec{\mu}_j} &= \frac{\partial}{\partial \vec{\mu}_j}(\sum_{i=1}^{k}\sum_{\vec{x}\in C_i} d(\vec{x},\vec{\mu}_i)^2) = \sum_{i=1}^{k}(\frac{\partial}{\partial \vec{\mu}_j}(\sum_{\vec{x}\in C_i} d(\vec{x},\vec{\mu}_i)^2)) \\
&= \frac{\partial}{\partial \vec{\mu}_j}(\sum_{\vec{x}\in C_j} d(\vec{x},\vec{\mu}_j)^2) = \sum_{\vec{x}\in C_j}\frac{\partial}{\partial \vec{\mu}_j}(d(\vec{x},\vec{\mu}_j)^2) \\
&= \sum_{\vec{x}\in C_j}\frac{\partial}{\partial \vec{\mu}_j}(\vec{x}-\vec{\mu}_j)^T(\vec{x}-\vec{\mu}_j) = \sum_{\vec{x}\in C_j}(-2\vec{x}+2\vec{\mu}_j) \\
&= (-2\sum_{\vec{x}\in C_j}\vec{x}) + (2\sum_{\vec{x}\in C_j}\vec{\mu}_j) = -2\sum_{\vec{x}\in C_j}\vec{x} + 2|C_j|\vec{\mu}_j
\end{aligned}$$

At the optimum: $2|C_j|\vec{\mu}_j - 2\sum_{\vec{x}\in C_j}\vec{x} = \vec{0}$

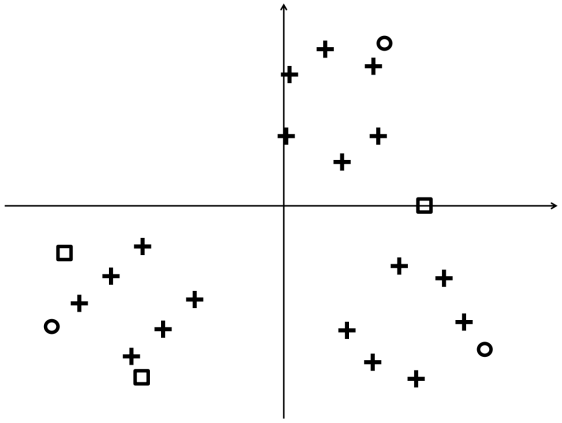
$$\Leftrightarrow |C_j|\vec{\mu}_j = \sum_{\vec{x}\in C_j}\vec{x} \Leftrightarrow \vec{\mu}_j = \frac{1}{|C_j|}\sum_{\vec{x}\in C_j}\vec{x}$$

3. Consider the data in the figure below where each point $\mathbf{x} \in \mathbb{R}^2$ is represented by a cross. Draw (approximately) the output of Lloyd's algorithm for $k = 3$ when

   (a) the initial centers for the algorithm are the circles;

   (b) the initial centers for the algorithm are the squares.

   Which one of the two resulting clusterings has a lower cost?



Solution

Since the cost depends on the square of the distance of points to the centroids, clustering (a) has lower cost.

Exam 14
26 June 2018

Exercise 1 [8 points]

1. Discuss which are the main ingredients of a learning problem, how learning can be formulated as on optimisation problem, and how the objective of learning can be encoded.
2. Define the concept of model class and a way to measure its complexity.
3. Discuss the role of model class complexity on the learning problem. In the context of PAC learning, provide a bound on sample complexity for finite model classes with loss function $\ell : \mathcal{H} \times Z \to [0, 1]$.

[Solution: Exercise 1]

Exercise 2 [8 points]

1. Describe and motivate the regression problem in Machine Learning.
2. Provide an example of linear regression problem where the hypothesis class is

$$Y = X\theta \qquad Y \in \mathbb{R}^n \quad \theta \in \mathbb{R}^d$$

in which it is of interest to perform variable selection and discuss how this can be solved using regularisation, defining explicitly the cost function to be minimised as a function of the usual regularisation parameter λ.

3. Let λ be the regularization parameter in the sparse regression problem discussed above. Draw a typical plot (regularisation path) of how the estimated coefficients $\hat{\theta}_i$ (entries of the parameter vector θ) vary as a function the regularisation parameter λ (one line for each $\hat{\theta}_i$, $i = 1, .., d$, assuming $d = 4$).

[Solution: Exercise 2]

Exercise 3 [8 points]

Consider a neural network with two hidden layers, inputs x, and output y, where the first hidden layer has 5 nodes (say ξ_i, $i = 1, .., 5$) and the second hidden layer 1 node (say z_1) where

$$\xi_i = \mathbf{1}(w_{1,i}^T x + b_i) \quad i = 1, .., 5$$

and

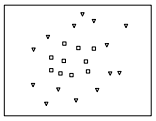
$$z_1 = \mathbf{1}(w_{2,1}^T \xi - 4.5)$$

where $w_{2,1}^T = [1\ 1\ 1\ 1\ 1]$, $\mathbf{1}(a)$ is the indicator function

$$\mathbf{1}(a) = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0 \end{cases}$$

and $y = z_1$.

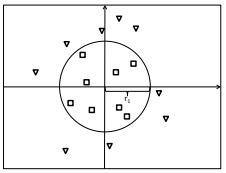
1. Draw a schematic picture of the neural network
2. Assuming the network is trained for the binary classification problem with the data depicted in figure below (the input $x \in \mathbb{R}^2$ are the coordinates of the points while the output y are the labels), say whether there is a combination of weights for which the training error is exactly equal to zero (i.e. the network perfectly classifies the training data). *Note:* you do not need to find the exact weights.
3. Interpret, and illustrate in the picure below, the two hidden layers in the context of linear classification on training data.



Exercise 4 [8 points]

You want to cluster the points in the figure below using k-means. with $k = 2$.

1. Is there a way to cluster the points using k-means so that in the solution the two clusters corresponds to the two sets with different marks (triangles, squares)? Given a short explanation for your answer.
2. Before applying clustering, you can apply a transformation to the dataset. Describe a transformation such that the application of k-means with $k = 2$ to the transformed datasets results in two clusters corresponding to the two sets with different marks, and plot the transformed dataset.
3. Briefly describe the execution of k-means on the transformed dataset (*note:* choose the first centers so that only few iterations are required and that the final clustering corresponds to the two sets with different marks).



[Solution: Exercise 4]

14.1 Exercise 1

1. Discuss which are the main ingredients of a learning problem, how learning can be formulated as an optimisation problem, and how the objective of learning can be encoded.

Solution

A learning problem can be describe through the following things:
- X be the domain set, which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Y be the label set that defines the set of all possible labels
- $S = ((\vec{x}_1, y_1)...(\vec{x}_m, y_m))$ with $\vec{x}_i \in X, y_i \in Y \ \forall i = 1, ..., m$ be the training set or training data
- $h : X \to Y$ the learner's output. Basically a function picked from H that given in input a point in X it outputs the label Y
- D be the unknown distribution from which the points are drawn.
- $f : X \to Y$ the unknown true labeling function with which the points drawn from D are labeled
- H be the hypothesis class which is the set of all predictors
- The error of the classifier or generalization error: $L_d(h) = E_{z \sim D}[l(h, z)]$
- $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ be the loss function namely a function that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the

The goal of a learning problem is to produce a learner $\hat{h} \in H$ with low $L_d(\hat{h})$.

2. Define the concept of model class and a way to measure its complexity.

Solution

A model class is set H composed by functions $h \in H$ where such functions are the candidates to select the one to be used as the model to map instances of X into instances of Y namely $h : X \to Y$. A way to measure the complexity of a model class is to compute the VC-dimension.

Let $h \in H$ be such that $h : X \to \{0, 1\}$. Let $C = \{c_1, ..., c_m\}$ with $C \subset X$. The restriction H_C of H to C is: $H_C = \{[h(c_1), ..., h(c_m)] : h \in H\}$. We say that H shatters C if $|H_C| = 2^m$, that is H_C contains all $2^m = 2^{|C|}$ functions from C to $\{0, 1\}$.

> The Vc-dimension $VCdim(H)$ of H is the maximal size of set $C \subset X$ that can be shattered by H; if H can shatter sets of arbitrary large size then $VCdim(H) = +\infty$.

3. Discuss the role of model class complexity on the learning problem. In the context of PAC learning, provide a bound on sample complexity for finite model classes with loss function $\ell : \mathcal{H} \times \mathcal{Z} \to [0, 1]$.

Solution

The model class complexity determines the behavior of the generalization error of the hypothesis that we have learned. Such complexity cannot be either too large due to the no free lunch theorem or too small due to the underfitting problems.

Let H be a finite hypothesis class, let Z be the domain, and let $l : H \times Z \to [0, 1]$ be a loss function. Then H enjoys uniform convergence property with sample complexity
$m_l^{UC}(\varepsilon, \delta) \leq ceil(\frac{\log(\frac{2|H|}{\delta})}{2\varepsilon^2})$.

Also H is agnostic pac learnable with sample complexity
$m_H(\varepsilon, \delta) \leq ceil(\frac{2\log(\frac{2|H|}{\delta})}{\varepsilon^2})$.

14.2 Exercise 2

1. Describe and motivate the regression problem in Machine Learning.

   Solution

   Regression task is a supervised learning task with: - Domain set X which is the set of all possible objects to make predictions about, where a domain point $x \in X$ is called instance and is usually represented by a vector of features, usually is equal to $\mathbb{R}^d$ - Label set $Y = \mathbb{R}$

   Given a training set $S = ((x_1, y_1)...(x_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, .., m$ we need to choose an hypothesis class H, which defines the possible models or classification rules, from which we can pick our model to make predictions, and a loss function $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

   The goal is to find an hypothesis $\hat{h} \in H$ with low generalization error: $L_d(\hat{h}) = E_{z \sim D}[l(\hat{h}, z)]$ where: - D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples).

2. Provide an example of linear regression problem where the hypothesis class is

   $$Y = X\theta \quad Y \in \mathbb{R}^n \quad \theta \in \mathbb{R}^d$$

   in which it is of interest to perform variable selection and discuss how this can be solved using regularisation, defining explicitly the cost function to be minimised as a function of the usual regularisation parameter λ.

3. Let λ be the regularization parameter in the sparse regression problem discussed above. Draw a typical plot (regularisation path) of how the estimated coefficients $\hat{\theta}_i$ (entries of the parameter vector θ) vary as a function the regularisation parameter λ (one line for each $\hat{\theta}_i, i = 1, .., d$, assuming $d = 4$).

14.3 Exercise 3

Consider a neural network with two hidden layers, inputs x, and output y, where the first hidden layer has 5 nodes (say $\xi_i, i = 1, .., 5$) and the second hidden layer 1 node (say z_1) where

$$\xi_i = \mathbf{1}(w_{1,i}^\top x + b_i) \quad i = 1, .., 5$$

and

$$z_1 = \mathbf{1}(w_{2,1}^\top \xi - 4.5)$$

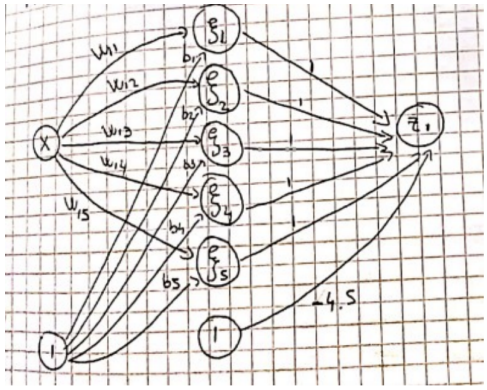
where $w_{2,1}^\top = [1\ 1\ 1\ 1\ 1]$, $\mathbf{1}(a)$ is the indicator function

$$\mathbf{1}(a) = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0 \end{cases}$$

and $y = z_1$.

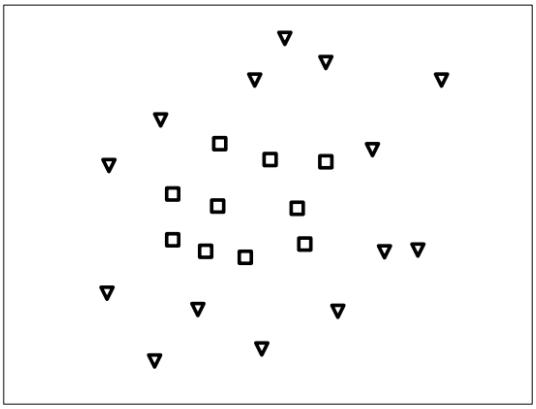
1. Draw a schematic picture of the neural network

Solution



2. Assuming the network is trained for the binary classification problem with the data depicted in figure below (the input $x \in \mathbb{R}^2$ are the coordinates of the points while the output y are the labels), say whether there is a combination of weights for which the training error is exactly equal to zero (i.e. the network perfectly classifies the training data). Note: you do not need to find the exact weights.

3. Interpret, and illustrate in the picure below, the two hidden layers in the context of linear classification on training data.



14.4 Exercise 4

You want to cluster the points in the figure below using k-means with $k = 2$.

1. Is there a way to cluster the points using k-means so that in the solution the two clusters corresponds to the two sets with different marks (triangles, squares)? Given a short explanation for your answer.

   Solution

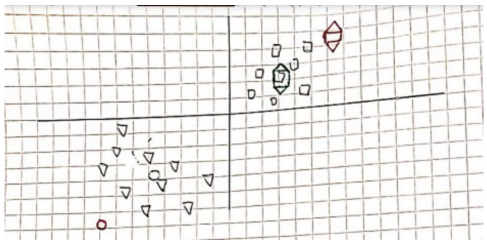
   There is no way to cluster the points like in the solution because with k-means objective function we minimize the distance between points and centers.

2. Before applying clustering, you can apply a transformation to the dataset. Describe a transformation such that the application of k-means with $k = 2$ to the transformed datasets results in two clusters corresponding to the two sets with different marks, and plot the transformed dataset.

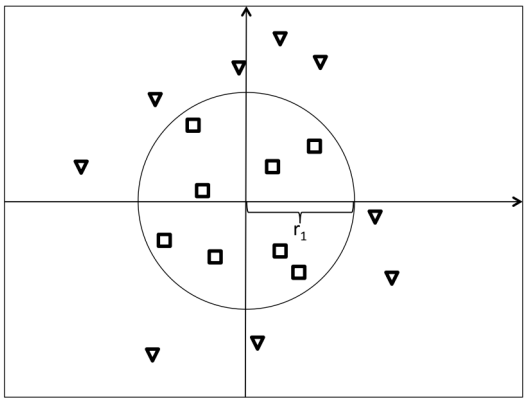
   Solution

   For $k = 2$, transformation to apply: $\psi(\vec{x}) = \begin{cases} [-\|x_1\|, -\|x_2\|]^\top & \text{if } \|\vec{x}\| \leq r_1 \\ [-x_1, -\|x_2\|]^\top & \text{if } \|\vec{x}\| > r_1 \end{cases}$

   Also the distance from (0,0) transformation $\sqrt{x_1^2 + x_2^2}$

3. Briefly describe the execution of k-means on the transformed dataset (note: choose the first centers so that only few iterations are required and that the final clustering corresponds to the two sets with different marks).



Solution

If we run the algorithm to find a solution of k-means problem with centers ◯ and □ then after a few iteration (i.e., the centers that are computed according to the following formula:

$\mu_i^{(t+1)} \leftarrow \frac{1}{|C_i|} \sum_{\vec{x} \in C_i} \vec{x}$

Exam 15
12 February 2018

Exercise 1 [10 points]

1. Provide the formulation of Agnostic PAC learning in terms of sample ($L_{\mathcal{S}}(h)$) and generalisation ($L_{\mathcal{D}}(h)$) errors.
2. Assuming that $x_i \in [a, b]$ (a and b finite but unknown), $i = 1, .., m$ are i.i.d. with mean μ and variance σ^2 and consider the problem of estimating μ solving

$$\hat{\mu} = \arg\min_{\theta} L_{\mathcal{S}}(\theta)$$

where

$$L_{\mathcal{S}}(\theta) = \frac{1}{m}\sum_{i=1}^{m}(x_i - \theta)^2$$

Prove that

$$L_{\mathcal{D}}(\theta) = \sigma^2 + (\theta - \mu)^2$$

and that, $\forall \epsilon > 0$

$$\lim_{m\to\infty}\mathbb{P}[|L_{\mathcal{S}}(\theta) - (\sigma^2 + (\theta - \mu)^2)| > \epsilon] = 0$$

3. Discuss which tools can be used, in general, to prove that

$$\mathbb{P}[|L_{\mathcal{S}}(h) - L_{\mathcal{D}}(h)| > \epsilon]$$

is small. Under the hypothesis made in the course, how does this probability behave as the sample size m goes to infinity, and which role does it play in the agnostic PAC learning problem?

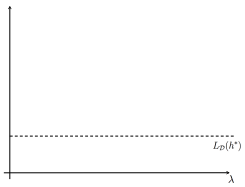
[Solution: Exercise 1]

Exercise 2 [8 points]

1. Describe and motivate the ridge regression problem and derive its solution (showing all the steps of the derivation).
2. Let λ be the regularization parameter in ridge regression. Let $\mathcal{S}$ be a training set containing m i.i.d. examples, $h_{\mathcal{S}}$ be the hypothesis that minimises the empirical risk on $\mathcal{S}$, $h_{\mathcal{S},R}$ be the hypothesis that minimises the ridge regression problem defined above on $\mathcal{S}$, and

$$h^* \in \arg\min_{h\in\mathcal{H}} L_{\mathcal{D}}(h).$$

Assume the number of samples m of $\mathcal{S}$ is fixed and that the relation between m and the dimensionality d of the instance space (e.g., $\mathcal{X} = \mathbb{R}^d$) is $m << d$. Plot below the typical behaviour of $L_{\mathcal{S}}(h_{\mathcal{S}})$, $L_{\mathcal{S}}(h_{\mathcal{S},R})$, $L_{\mathcal{D}}(h_{\mathcal{S}})$, $L_{\mathcal{D}}(h_{\mathcal{S},R})$ as a function of the regularization parameter λ.

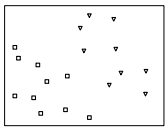


3. Describe the use of cross-validation to estimate the best value of the regularization parameter λ. When is cross-validation a preferable choice compared to the training-validation-test split?

[Solution: Exercise 2]

Exercise 3 [7 points]

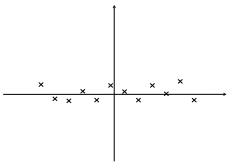
1. Describe linear SVM for binary classification in the case of linearly separable data.
2. The following figure shows an instance for binary classification with data points in $\mathbb{R}^2$, where the class of each point is represented by its shape (triangle or square). Draw (approximately) the separating hyperplane that would result from running (hard) SVM on the instance and mark the support vectors and the margin. Draw them directly in the figure.



[Solution: Exercise 3]

Exercise 4 [7 points]

1. Introduce PCA in the context of unsupervised learning.
2. Describe how to obtain the first r principal components for a data matrix $\mathbf{D}$.
3. With respect to the dataset shown in the figure below: draw approximately the first and second right principal components (directly in the figure). In a separate plot, show (approximately) the projection of the dataset on the first principal component.



[Solution: Exercise 4]

15.1 Exercise 1

1. Provide the formulation of Agnostic PAC learning in terms of sample ($L_S(h)$) and generalisation ($L_{\mathcal{D}}(h)$) errors.

Solution

An hypothesis class H is agnostic PAC learnable with respect to Z and a loss function $l : H \times Z \to \mathbb{R}^+$, if there exists a function $m_H : (0,1)^2 \to \mathbb{N}$ (sample complexity) and a learning algorithm such that for every $\delta \in (0,1)$, $\epsilon \in (0,1)$, for every distribution D over Z, when running the learning algorithm on $m \geq m_H(\epsilon, \delta)$ iid samples generated by D the algorithm returns an hypothesis such that, with probability $\geq 1 - \delta$: $L_{D,f}(h) \leq \min_{h' \in H} L_{D,f}(h') + \epsilon$ where $L_{D,f}(h) = E_{z \sim D}[l(h, z)]$.

2. Assuming that $x_i \in [a, b]$ (a and b finite but unknown), $i = 1, .., m$ are i.i.d. with mean μ and variance σ^2 and consider the problem of estimating μ solving

$$\hat{\mu} = \arg\min_{\theta} L_S(\theta)$$

where

$$L_S(\theta) = \frac{1}{m} \sum_{i=1}^{m} (x_i - \theta)^2$$

Prove that

$$L_{\mathcal{D}}(\theta) = \sigma^2 + (\theta - \mu)^2$$

and that, $\forall \epsilon > 0$

$$\lim_{m \to \infty} \mathbb{P}[|L_S(\theta) - (\sigma^2 + (\theta - \mu)^2)| > \epsilon] = 0$$

3. Discuss which tools can be used, in general, to prove that

$$\mathbb{P}[|L_S(h) - L_{\mathcal{D}}(h)| > \epsilon]$$

is small. Under the hypothesis made in the course, how does this probability behave as the sample size m goes to infinity, and which role does it play in the agnostic PAC learning problem?

Solution

With the use of Holdeffing's inequality let us call $\vartheta_i = l(h, z_i)$ thus $P[|L_S(h) - L_D(h)| > \epsilon] = P[|\frac{1}{m} \sum_{i=1}^{m} \vartheta_i - \mu| > \epsilon] \leq 2e^{-2m\epsilon^2}$. Therefore if $m \to +\infty$ then $2e^{-2m\epsilon^2} \to 0$ and so $P[|L_S(h) - L_D(h)| \leq \epsilon] \to 1$ so, every finite hypothesis class H with 0-1 loss are agnostic pac learnable is if $m \to +\infty$.

15.2 Exercise 2

1. Describe and motivate the ridge regression problem and derive its solution (showing all the steps of the derivation).

Solution

Ridge regression is the application of Tikhonov regularization to linear regression with squared loss. This means that: - Domain set X which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is equal to $\mathbb{R}^d$ - Label set $Y = \mathbb{R}^d$ - H: the hypothesis class is exactly equal to $L_d = \{h_{w,b} : \mathbb{R}^d \to \mathbb{R}\}$ where $h_{w,b}(\vec{x}) =< \vec{w}, \vec{x} > +b$ - l: the loss function is exactly equal to $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ such that $l(h, (\vec{x}, y)) = (h(\vec{x}) - y)^2 = (< \vec{w}, \vec{x} > +b - y)^2 = (< \vec{w'}, \vec{x'} > -y)^2$ if $\vec{w'} = [b, w_1, ..., w_m]$ and $\vec{x'} = [1, x_1, ..., x_m]$ - The function to be minimized is: $\lambda\|\vec{w}\|^2 + L_S(\vec{w}) = \lambda\|\vec{w}\|^2 + \sum_{i=1}^m (< \vec{w}, \vec{x_i} > -y_i)^2$ where $S = ((\vec{x_1}, y_1)...(\vec{x_m}, y_m))$ is the training set.

The solution for such problem can be found in the following way:

- Let: $X = \begin{bmatrix} \vec{x_1}^T \\ \vdots \\ \vec{x_m}^T \end{bmatrix}$; $\vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$ and $\vec{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}$

- We need to compute the gradient and evaluate it to 0. So: $\frac{\delta}{\delta(\vec{w})}(\lambda\|\vec{w}\|^2 + \sum_{i=1}^m (< \vec{w}, \vec{x_i} > -y_i)^2) = \frac{\delta}{\delta(\vec{w})}(\lambda\|\vec{w}\|^2 + (\vec{y} - X\vec{w})^T(\vec{y} - X\vec{w})) = 2\lambda\vec{w} - 2X^T(\vec{y} - X\vec{w})$

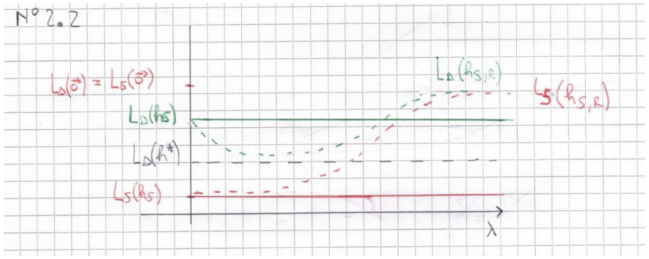
Evaluating to 0 we obtain: $\vec{w} = (\lambda I + X^T X)^{-1} X^T \vec{y}$. Notice that $\lambda I + X^T X$ is a positive definite matrix so the inverse exists.

2. Let λ be the regularization parameter in ridge regression. Let $\mathcal{S}$ be a training set containing m i.i.d. examples, h_S be the hypothesis that minimises the empirical risk on $\mathcal{S}$, $h_{S,R}$ be the hypothesis that minimises the ridge regression problem defined above on $\mathcal{S}$, and

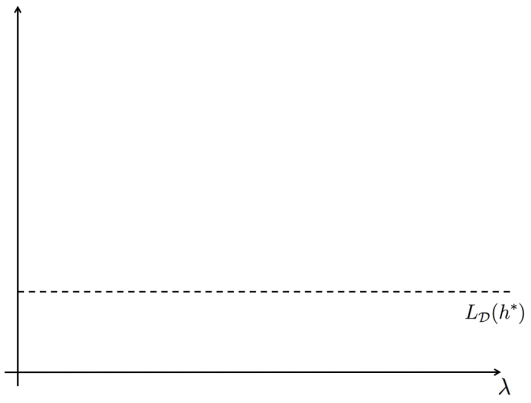
$$h^* \in \arg\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h).$$

Assume the number of samples m of $\mathcal{S}$ is fixed and that the relation between m and the dimensionality d of the instance space (e.g., $\mathcal{X} = \mathbb{R}^d$) is $m \ll d$. Plot below the typical behaviour of $L_S(h_S)$, $L_S(h_{S,R})$, $L_{\mathcal{D}}(h_S)$, $L_{\mathcal{D}}(h_{S,R})$ as a function of the regularization parameter λ.

Solution



3. Describe the use of cross-validation to estimate the best value of the regularization parameter λ. When is cross-validation a preferable choice compared to the training-validation-test split?



Solution

A possible strategy to estimate the parameter λ is to use cross-validation. Therefore recalling λ, ϑ, we can apply the cross validation method.

To be more precise, cross validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have. To do so, we split the dataset into k different folds of size m/k (this quantity is supposed to be integer). Then for each value of the parameter ϑ we find the best model for all the possible k-1 folds that we can select. In addition to that, we compute the average error of each parameter as the average of the errors on the folds left out, and when we have repeated such procedure for all the values of the parameters, then we select the optimal one by choosing the one that minimizes the error computed for each parameter. To conclude, we train our model on all the dataset with the parameter found.

The pseudo code is the following:

Input: $S = ((\vec{x_1}, y_1)...(\vec{x_m}, y_m))$; set of parameters Θ; integer k; learning algorithm A

Split S into $S_1, ..., S_k$

foreach $\vartheta \in \Theta$ for i = 1 ... k $h_{i,\vartheta} = A(S \setminus S_i; \vartheta)$ $error(\vartheta) = \frac{1}{k}\sum_{i=1}^{k} L_{S_i}(h_{i,\vartheta})$ Output: $\vartheta^* = \arg\min_{\vartheta}(error(\vartheta))$ $h_{\vartheta^*} = A(S; \vartheta^*)$

Cross validation is preferable to use instead of validation when-

> ever we have very small dataset. This is due to the fact that for validation we need to split the dataset into two parts, validation set and training set. Therefore if we do not have enough data, then such split will produce very small set with the consequence that, having a small training set and so we haven't enough samples to learn, or a small validation set and so we haven't enough samples to be able to compare the errors.

15.3 Exercise 3

1. Describe linear SVM for binary classification in the case of linearly separable data.

Solution

SVM is a supervised learning method that is used to find a solution for a binary classification problem. Hard-SVM is the formulation of SVM that works when data are linearly separable, namely, with reference to binary classification, when $\forall i = 1, ..., m \; y_i(< \vec{w}, \vec{x_i} > +b) > 0$ or equivalently when there exists an halfspace $(\vec{w}, \vec{x})$ such that $y_i = sign(< \vec{w}, \vec{x_i} > +b)$ $\forall i = 1, ..., m$ where $S = ((\vec{x_1}, y_1)...(\vec{x_m}, y_m))$ is the training set used to train the SVM.

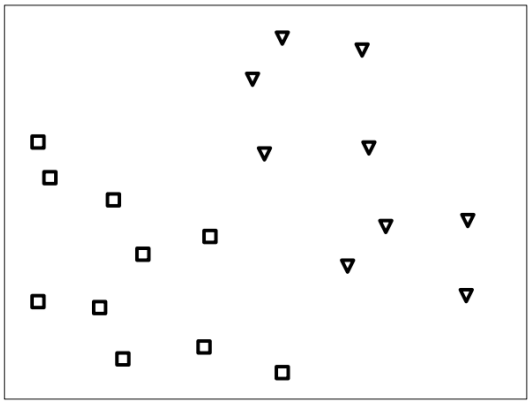
Therefore the problem solved by Hard-SVM can be stated as follows:

Input: $S = ((\vec{x_1}, y_1)...(\vec{x_m}, y_m))$ linearly separable dataset

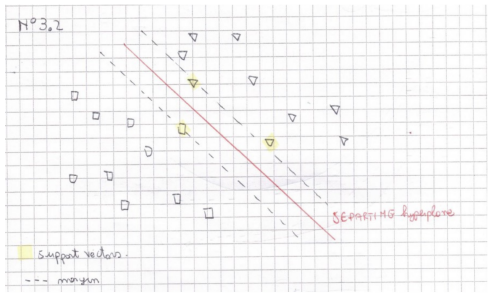
Goal: $\underset{(\vec{w},b)}{\text{argmax}} \min_{i \in \{1,...,m\}} \frac{|<\vec{w},\vec{x_i}>+b|}{\|\vec{w}\|}$ subject to $\forall i = 1, ..., m$ $y_i(< \vec{w}, \vec{x_i} > +b) \geq 1$

Output: $\frac{\vec{w}}{\|\vec{w}\|}; \frac{b}{\|\vec{w}\|}$

2. The following figure shows an instance for binary classification with data points in $\mathbb{R}^2$, where the class of each point is represented by its shape (triangle or square). Draw (approximately) the separating hyperplane that would result from running (hard) SVM on the instance and mark the support vectors and the margin. Draw them directly in the figure.



Solution

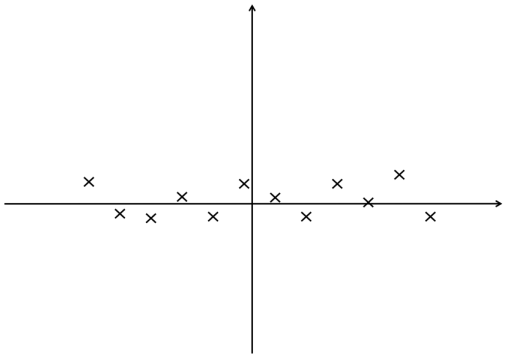


15.4 Exercise 4

1. Introduce PCA in the context of unsupervised learning.

2. Describe how to obtain the first r principal components for a data matrix $\mathbf{D}$.

3. With respect to the dataset shown in the figure below: draw approximately the first and second right principal components (directly in the figure). In a separate plot, show (approximately) the projection of the dataset on the first principal component.



Exam 16
29 January 2018

Exercise 1 [9 points]

1. Describe the classification problem in machine learning and one approach for its solution.
2. Give the definition of empirical and generalisation errors for an arbitrary hypothesis $h \in \mathcal{H}$, and give the definition of ϵ-representative sample $\mathcal{S}$.
3. Prove the following Lemma (motivating each step): If $\mathcal{S}$ is an $\epsilon/2$-representative sample (as defined above), then any empirical risk minimiser $h_{\mathcal{S}}$ satisfies the inequality

$$L_{\mathcal{D}}(h_{\mathcal{S}}) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon.$$

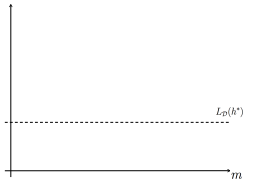
[Solution: Exercise 1]

Exercise 2 [8 points]

1. Introduce and motivate the use of regularisation in regression problems.
2. Let $\mathcal{S}$ be a training set containing m i.i.d. examples, $h_{\mathcal{S}}$ be the hypothesis that minimises the empirical risk on $\mathcal{S}$, $h_{\mathcal{S},R}$ be the hypothesis that minimises the regularised problem defined above on $\mathcal{S}$, and

$$h^* \in \arg\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h).$$

Assume the regularisation parameter (say λ) is fixed (and does not dependent on the sample size m - e.g., $\lambda = 10$). Plot below the typical behaviour of $L_{\mathcal{S}}(h_{\mathcal{S}})$, $L_{\mathcal{S}}(h_{\mathcal{S},R})$, $L_{\mathcal{D}}(h_{\mathcal{S}})$, $L_{\mathcal{D}}(h_{\mathcal{S},R})$ as a function of the training set size m.

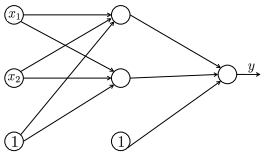


3. Describe an approach to estimate a reasonable value for the regularisation parameter λ.

[Solution: Exercise 2]

Exercise 3 [7 points]

1. Consider the Neural Network depicted in the figure below.



Let the input variables x_1 and x_2 be binary ($x_i \in \{-1, 1\}$) and the activation function be the sign function: $x(z) = sign(z)$. Assume the network weights are all constrained to take value in the discrete set $\{-1, 1\}$. Consider the training set described by the table below:

x_1	x_2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

Find network weights so to that the training error is zero. (You can put the weights directly in the figure above.)

2. Using the example above motivate the fact that neural network architectures are richer than linear models.

[Solution: Exercise 3]

Exercise 4 [8 points]

1. Introduce the clustering problem in the context of unsupervised learning.
2. Describe the k-means optimisation problem and an algorithm to solve this problem; in particular discuss whether the algorithm finds the optimal solution.
3. Introduce the Gaussian Mixture Model (GMM) for clustering points $x_i \in \mathbb{R}^d$, $i = 1.., m$ in k classes. Argue why this can be considered a *soft* version of the *hard assignment* of points to clusters provided by k-means.

[Solution: Exercise 4]

16.1 Exercise 1

1. Describe the classification problem in machine learning and one approach for its solution.

Solution

The classification problem is a supervised learning problem with: - Domain set X which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features - Label set Y, that defines the set of all possible labels Y is a discrete set in classification and often finite. For instance in binary classification $Y = \{-1, +1\}$

Given a training set $S = ((x_1, y_1)...(x_m, y_m))$ with $x_i \in X, y_i \in Y \ \forall i = 1, ..., m$ we need to choose an hypothesis class H, which defines the possible models or classification rules, from which we can pick our model to make predictions, and a loss function $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y. The goal is to find an hypothesis $\hat{h} \in H$ with low generalization error: $L_d(\hat{h}) = E_{z \sim D}[l(\hat{h}, z)]$ where: - D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples).

One approach to solve the problem of classification is to find the $\hat{h} \in H$ by using the ERM procedure, namely we find out what is the hypothesis that minimizes the training error $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, z_i)$.

2. Give the definition of empirical and generalisation errors for an arbitrary hypothesis $h \in \mathcal{H}$, and give the definition of ϵ-representative sample $\mathcal{S}$.

Solution

Let: - X be the domain set, which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features - Y be the label set that defines the set of all possible labels - $S = ((\vec{x_1}, y_1)...(\vec{x_m}, y_m))$ with $\vec{x_i} \in X, y_i \in Y \ \forall i = 1, ..., m$ be the training set - $l : H \times Z \to \mathbb{R}^+$ where $Z = X \times Y$ be the loss function namely a function that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the - D be the unknown distribution - H be the hypothesis class

We define the training error as: $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, (\vec{x_i}, y_i))$. We define the generalization error as: $L_d(h) = E_{z \sim D}[l(h, z)]$ where $z = (\vec{x}, y)$

A set S is ϵ-representative (with respect to domain Z, hypothesis class H, loss function l and distribution D) if: $\forall h \in H \ |L_S(h) - L_d(h)| \leq \epsilon$

3. Prove the following Lemma (motivating each step): If $\mathcal{S}$ is an $\epsilon/2$-representative sample (as defined above), then any empirical risk minimiser h_S satisfies the inequality

$$L_{\mathcal{D}}(h_S) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon.$$

Solution

Proof: $L_D(h_S) \leq L_S(h_S) + \frac{\epsilon}{2}$ [$\frac{\epsilon}{2}$-representative] $\leq L_S(h) + \frac{\epsilon}{2}$ [ERM procedure] $\leq L_D(h) + \frac{\epsilon}{2} + \frac{\epsilon}{2}$ [$\frac{\epsilon}{2}$-representative] $= L_D(h) + \epsilon$

16.2 Exercise 2

1. Introduce and motivate the use of regularisation in regression problems.

Solution

Regularization or Regularized Loss minimization (RLM for short) is a paradigm like Empirical Risk Minimization (ERM) in which we jointly minimize the empirical risk and a regularization function that measure the complexity of a model.

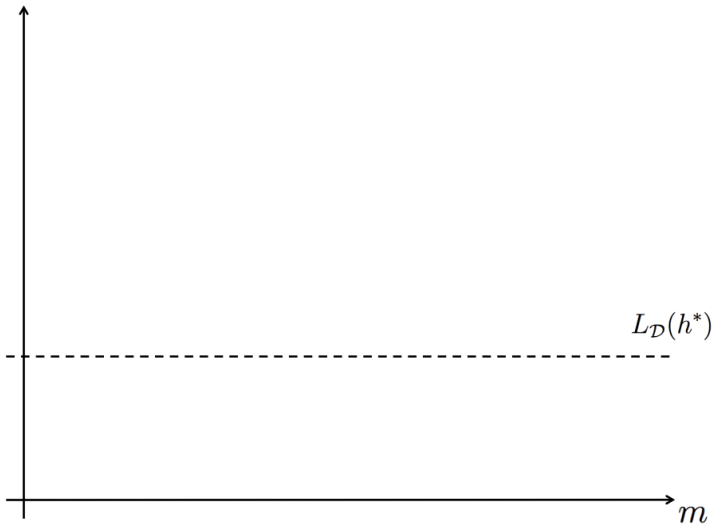
In regression terms this means that we need to minimize the following objective function: $L_S(\vec{w})+R(\vec{w})$ where $\vec{w}$ is the model to learn and $R(\vec{w})$ is the regularization function that measure the complexity of the model.

The most commonly used regularization functions are: - $R(\vec{w}) = \lambda\|\vec{w}\|_1$ where $\|\vec{w}\|_1$ is the L-1 norm of the vector $\vec{w}$ i.e. $\sum_{i=1}^{d}|w_i|$ (if used with linear models with squared loss for regression then we commonly talk about LASSO) - $R(\vec{w}) = \lambda\|\vec{w}\|^2$ (if used with linear models with squared loss for regression then we commonly talk about RIDGE Regression)

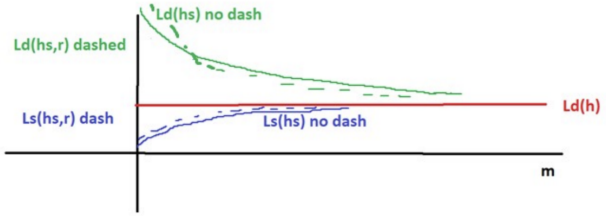
2. Let $\mathcal{S}$ be a training set containing m i.i.d. examples, h_S be the hypothesis that minimises the empirical risk on $\mathcal{S}$, $h_{S,R}$ be the hypothesis that minimises the regularised problem defined above on $\mathcal{S}$, and

$$h^* \in \arg\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h).$$

Assume the regularisation parameter (say λ) is fixed (and does not dependent on the sample size m - e.g., $\lambda = 10$). Plot below the typical behaviour of $L_S(h_S)$, $L_S(h_{S,R})$, $L_{\mathcal{D}}(h_S)$, $L_{\mathcal{D}}(h_{S,R})$ as a function of the training set size m.



Solution



3. Describe an approach to estimate a reasonable value for the regularisation parameter λ.

Solution

A possible strategy to estimate the parameter λ is to use cross-validation. Therefore recalling λ, ϑ, we can apply the cross validation method.

To be more precise, cross validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have. To do so, we split the dataset into k different folds of size m/k (this quantity is supposed to be integer). Then for each value of the parameter ϑ we find the best model for all the possible k-1 folds that we can select. In addition to that, we compute the average error of each parameter as the average of the errors on the folds left out, and when we have repeated such procedure for all the values of the parameters, then we select the optimal one by choosing the one that minimizes the error computed for each parameter. To conclude, we train our model on all the dataset with the parameter found.

The pseudo code is the following:

Input: $S = ((\vec{x_1}, y_1)...(\vec{x_m}, y_m))$; set of parameters Θ; integer k; learning algorithm A

Split S into $S_1, ..., S_k$

foreach $\vartheta \in \Theta$ for i = 1 ... k $h_{i,\vartheta} = A(S \setminus S_i; \vartheta)$ $error(\vartheta) = \frac{1}{k}\sum_{i=1}^{k} L_{S_i}(h_{i,\vartheta})$ Output: $\vartheta^* = \arg\min_\vartheta(error(\vartheta))$ $h_{\vartheta^*} = A(S; \vartheta^*)$

Assuming that we have enough data another strategy is to use validation to choose λ.

To be more precise validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have. To do so we split out dataset into 2 parts training set and validation set. Then, for each value of the parameter that we have, we find the best model for every possible values of the parameter on the training set. Then among such models that correspond to a value of the parameter we select the one that minimizes the validation error. The model obtain then is trained on the entire dataset.

Notice that training error is computed as: $L_S(h) = \frac{1}{m}\sum_{i=1}^{m} l(h, (\vec{x_i}, y_i))$ where S is the training set, namely, $S = ((\vec{x_1}, y_1)...(\vec{x_m}, y_m))$,

$l : H \times Z \to \mathbb{R}^+$ is the loss function where $Z = X \times Y$,that, given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

The validation error instead is computed as: $L_V(h) = \frac{1}{m''} \sum_{i=1}^{m''} l(h, (\vec{x_i}, y_i))$ where V is the validation set, namely, $V = ((\vec{x_1}, y_1)...(\vec{x_{m''}}, y_{m''}))$,

$l : H \times Z \to \mathbb{R}^+$ is the loss function where $Z = X \times Y$,that, given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.

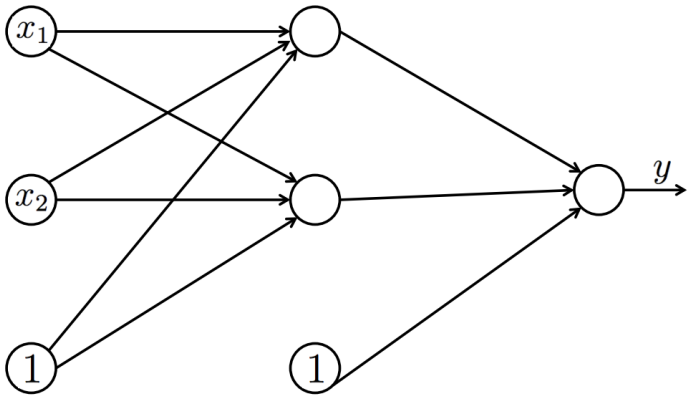
16.3 Exercise 3

Consider the Neural Network depicted in the figure below.

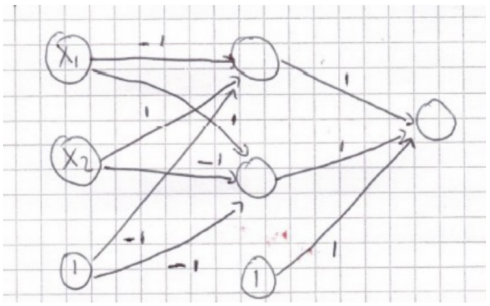
1. Let the input variables x_1 and x_2 be binary ($x_i \in \{-1, 1\}$) and the activation function be the sign function: $\sigma(z) = sign(z)$. Assume the network weights are all constrained to take value in the discrete set $\{-1, 1\}$. Consider the training set described by the table below:

x_1	x_2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

   Find network weights so to that the training error is zero. (You can put the weights directly in the figure above.)



Solution



2. Using the example above motivate the fact that neural network architectures are richer than linear models.

Solution

Neural Networks are richer architecture than linear models because if we try to separate training points with a linear model we find out that there is no linear model that perfectly classifies all the points used. In particular we cannot represent the XOR function with linear models.

16.4 Exercise 4

1. Introduce the clustering problem in the context of unsupervised learning.

Solution

k-means is a cost minimization clustering problem. Let $X \subseteq X'$ be the set of points to be clustered with $X = \{\vec{x_1}, ..., \vec{x_m}\}$ while X' is the space of possible points that we assume to be $\mathbb{R}^d$ (i.e. $X' = \mathbb{R}^d$). Let $k \in \mathbb{N}^+$ be the number of clusters, that is, with k, the input of the problem. Let $d(\cdot)$ be a distance function: $d(\vec{x}, \vec{x'}) = \|\vec{x} - \vec{x'}\|$. The goal is to find: - A partition $C = (C_1, ..., C_k)$ of X - Centroids $\vec{\mu_1}, ..., \vec{\mu_k}$ of $C_1, .., C_k$ respectively

That minimizes the k-means cost function: $\sum_{i=1}^{k} \sum_{x \in C_i} d(\vec{x}, \vec{\mu_i})^2$

To find the solution of this problem we can use an heuristic algorithm called Lloyd's Algorithm. Such algorithm works as follows:

Input: data points $X = \{\vec{x_1}, ..., \vec{x_m}\}$ and $k \in \mathbb{N}^+$ Output: $C = (C_1, ..., C_k)$ of X and centroids $\vec{\mu_1}, ..., \vec{\mu_k}$ of $C_1, ..., C_k$ respectively

To produce such things the algorithm works as follows: - Randomly initialize the centroids $\vec{\mu_1}, ..., \vec{\mu_k}$ - Until the convergence is not reached iteratively repeats the following steps: 1) Compute each cluster C_i by associating the points to him that are closer to its centroid compared to others centroids. 2) Compute the new centroids for the next iterations. 3) If the convergence is reached then the output described above will be returned.

The pseudocode for such algorithm is the following:

Randomly choose $\vec{\mu_1}^0, ..., \vec{\mu_k}^0$ for t < 0,... do for i = 1,...,k do $C_i \leftarrow \{\vec{x} \in X : i = \arg\min_j d(\vec{x}, \vec{\mu_j}^t)\}$ for i = 1,...,k do $\vec{\mu_i}^{t+1} \leftarrow \frac{1}{|C_i|} \sum_{\vec{x} \in C_i} \vec{x}$ if convergence reached then return $C = (C_1, ..., C_k)$ $\vec{\mu_1}^{t+1}, ..., \vec{\mu_k}^{t+1}$

The convergence criteria that can be used for such algorithm are the following: - The k-means objective function is no lower than the k-means objective function at the next iteration - $\sum_{i=1}^{k} d(\vec{\mu_i}^{t+1}, \vec{\mu_i}^t) \leq \epsilon$ - $\max_{1 \leq i \leq k} d(\vec{\mu_i}^{t+1}, \vec{\mu_i}^t) \leq \epsilon$

Note that if the first criteria is used then, the algorithm will always converge.

The time complexity is $O(tkmd)$ where $O(kmd)$ is due to the assignments of the points, while $O(md)$ is due to the computation

of the centroids. Therefore the complexity depends on t which is the number of iterations of the algorithm.

Loyld's algorithm finds a solution almost optimal if the centroids are initialized with the k-means++ algorithm. In particular if we do so then:

let $\Phi_{k-means}(X,k)$ be the cost of the optimal k-means clustering of X and let $\Phi_{k-mean++}(X, F_{k-means++})$ be the cost of the clustering X obtained by using Llyld's algorithm with k-means++ centers and applying only one iteration of Llyld's algorithm then:

$E[\Phi_{k-means}(X, F_{k-mean++})] \leq 8\ln(\ln k + 2)\Phi_{k-means}(X,k)$

2. Describe the k-means optimisation problem and an algorithm to solve this problem; in particular discuss whether the algorithm finds the optimal solution.

Solution

GMM (Gaussian mixture model) is an unsupervised learning algorithm that extends k-means clustering by incorporating a probabilistic approach. Instead of assigning points strictly to one cluster like k-means does (hard assignment), GMM assigns each point a probability of belonging to each cluster (soft assignment).

In GMM, each cluster is modeled as a Gaussian distribution with its own mean and covariance matrix. The model assumes the data points are generated from a mixture of these Gaussian distributions. The probability of a point belonging to a cluster is calculated based on these distributions, allowing for more flexible and nuanced cluster assignments compared to k-means' strict distance-based assignments.

For each point, k-means only considers the distance to cluster centroids for hard assignment (0 or 1 membership), while GMM considers probability distributions for soft assignment (probabilities between 0 and 1 for each cluster membership).

3. Introduce the Gaussian Mixture Model (GMM) for clustering points $x_i \in \mathbb{R}^d$, $i = 1, .., m$ in k classes. Argue why this can be considered a soft version of the hard assignment of points to clusters provided by k-means.