Support-vector machines and application in connectomics

BSc Thesis in Applied Mathematics

Keresztes László

Supervisor: Dr. Vince Grolmusz
Department of Computer Science

Eötvös Loránd University
Faculty of Science
2020

Contents

Acknowledgements

I would like to thank Dr. Vince Grolmusz for his help and guidance in writing this thesis. Thank him for all the inspiration I got over the last year. Thank him for the trust and the opportunity to deal with what I was really interested in under his direction.

I want to thank the members of the PIT research group for their professional help: Balázs Szalkai, Bálint Varga, Kristóf Takács, and Máté Fellner. Special thanks to Evelin Szögi for a lot of work and help, constant support, and enthusiasm.

Many thanks to my family and friends for the endless support and love I received from them. Thank my family for providing me a calm and pleasant environment to write my thesis.

Thank my teachers and ELTE for the high quality, attentive, and interesting education I received for three years.

1 Introduction

As computational technology reached the level where extracting knowledge from large data sets is possible, a new field of mathematics emerged, called data science. It is an application-driven science; the theories and methods are created to solve real problems in many areas of life. Data science became an exciting discipline both from the theoretical and practical points of view, and it also included machine learning and statistics.

One of the many fields that benefit from the evolving data science methods is biology. In different parts of biology, large data sets collected, now the modern techniques enable us to extract interesting and useful knowledge from these data. One of these parts is connectomics, the study of connectomes or brain graphs. The brain graph is a macroscopic model of the connection network of the human brain.

In my thesis, I would like to introduce some machine learning techniques and their application in connectomics. Firstly I present a commonly used, practical, and successful classification method called support-vector machines, and then I present a bunch of practices called feature selection methods, which are capable of choosing the most important features of a data set for a predefined property. As an application, I present a solution for the sex classification of the individual from his/her brain graph edges. Then I select a few edges that already specify the gender of the individual.

I use several concepts and conventional methods from machine learning and statistical learning [1, 2].

2 Support-vector machines

Support-vector machines were first suggested by Cortes and Vapnik (1995) [3]. The general idea behind the method is solving a two-class classification problem with linear boundary by finding the hyperplane between the two class, that maximizes the distance between the two class. The method was firstly introduced for linearly separable data points; in this case, the method is called the maximum-margin classifier (Vapnik, Chervonenkis 1963). Then it was developed to solve the linear separation when the data points are not linearly separable by allowing some points to be on the wrong side of the hyperplane; in this case, the method is called the soft-margin classifier (Vapnik, Cortes 1995). Usually, both the maximum-margin classifier and the soft-margin classifier solve the Lagrangian dual of a convex optimization problem. In the Lagrangian dual of SVM, not the vectors of the data points matter, but only the dot products of these vectors. This observation led to the kernel trick (Vapnik, Guyon, Boser 1992), where the data points are transformed into a higher-dimensional space, but using only the kernel products of the data points in the optimization. Using only these pairwise kernel products, the difficulty of solving the optimization does not change. This method was initially called the support-vector machine, but it is also the overall name of these methods. The SVM methods have been extended to multi-class classification and regression. The main advantages of SVM methods:

- they are really fast (even in high dimensional spaces)
- according to experience, they generalize well, but they are still a simple linear model (not the kernel-SVM)
- they seem useful even when the number of dimensions is bigger than the number of samples
- the interpretability of an SVM classifier (with a few features) is high

from the feature side, but it is also high from the data side, considering that an SVM method uses only the support vectors in the classifier construction, which can be considered as hard-to-classify data points

In the support-vector machines section, I would like to give a detailed introduction to the creation of the maximum-margin classifier in the linearly separable case and the soft-margin classifier in the linearly non-separable case. Finally, I mention some further extensions of these SVM methods.

2.1 Linearly separable case

In the linearly separable case, there are always infinitely many hyperplanes that separate the data points. The support-vector machine tries to find one of these, namely the one whose minimal distance from the data points is maximal. When our data points are linearly separable, we call the method the maximum-margin classifier. I want to give a detailed, geometric deduction of the standard form of the SVM (primal) problem, and then use the theory of convex optimization to deduce another form of the SVM problem, called the dual problem. The SVM dual problem has some interesting properties, and solving that is computationally more efficient.

In the linearly separable case, we consider N points in $\mathbb{R}^n$: $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N} \in \mathbb{R}^n$. Each of these points have a known label: $y_i \in \{-1, 1\}$ for $i = 1, \ldots, N$. We can refer to these data points as positive (where $y_i = 1$) and negative (where y_i = -1) examples.

In the SVM methods, we are always looking for a hyperplane that separates the data points, thus we should define these concepts.

Definition 2.1.1. *H is a hyperplane in $\mathbb{R}^n$ if it is a $(n-1)$-dimensional affin subspace in $\mathbb{R}^n$ or equivalently if*

$$H = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{w}\mathbf{x} + b = 0\}$$

for any $\mathbf{w} \in \mathbb{R}^n$ *vector and* $b \in \mathbb{R}$. *We call* $\mathbf{w}$ *the hyperplane's normal vector,* b *the hyperplane's intercept, and we call* $(\mathbf{w}, b)$ *a representation of* H. *One can easily see, that a hyperplane cuts* $\mathbb{R}^n$ *to two open and connected half-spaces:*

$$A_1 = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{w}\mathbf{x} + b > 0\}$$

$$A_2 = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{w}\mathbf{x} + b < 0\}$$

and $A_1 \cup A_2$ *is not connected, meaning that the hyperplane has actually two sides.*

Definition 2.1.2. *Let* $(\mathbf{x_i}, y_i) \in \mathbb{R}^n \times \{-1, 1\}$ *for* $i = 1, \ldots, N$ *and* H *is a hyperplane in* $\mathbb{R}^n$ *with* $\mathbf{w}$ *normal vector and* b *intercept. We say that* H *separates the positive and negative examples (or alternatively separates the data points) if:*

$$\mathbf{w}\mathbf{x_i} + b > 0 \text{ if } y_i = 1$$

$$\mathbf{w}\mathbf{x_i} + b < 0 \text{ if } y_i = -1$$

for $i = 1, \ldots, N$, *or simply:*

$$y_i(\mathbf{w}\mathbf{x_i} + b) > 0 \text{ for } i = 1, \ldots, N$$

The general purpose of every classification method is the creation of a decision function, which will be accurate on previously not seen data. We can define a decision function for every hyperplane:

Definition 2.1.3. *Let* H *is a hyperplane, that separates the data points. Then we can create a decision function from* H *for classifying new observations:*

$$f_H(\mathbf{x}) = sgn(\mathbf{x}\mathbf{w} + b)$$

This is a linear classifier, meaning that in $\mathbb{R}^n$ *the boundary of the two possible decision-area is linear, this boundary now is exactly* H.

We would like to classify new data points as accurately as we can, but f only depends on H; hence we should select a H hyperplane for accurate predictions.

From Definition 2.1.1, one can see there are infinitely many representations of a hyperplane with a normal vector and an intercept, because if $(\mathbf{w}, b)$ is a representation of H and $\lambda \in \mathbb{R}$, then $(\lambda\mathbf{w}, \lambda b)$ is also a representation of H. Thus, we can restrict the representations by requiring that $\|\mathbf{w}\| = 1$, where $\|.\|$ is the Euclidean norm in $\mathbb{R}^n$. This requirement has a powerful property.

Statement 2.1.1. *Let H is a hyperplane in $\mathbb{R}^n$ with unit length $\mathbf{w}$ normal vector and b intercept. Then for every $\mathbf{x} \in \mathbb{R}^n$ $\mathbf{wx} + b$ is the signed distance of $\mathbf{x}$ from the H hyperplane. (The sign is only decides on which side of H is $\mathbf{x}$.)*

Proof. Let $\mathbf{x}$ is an arbitrary point in $\mathbb{R}^n$ that is not in H. Let $\mathbf{x_0}$ is an arbitrary point of H, thus $\mathbf{wx_0} = -b$. Either $\mathbf{w} + \mathbf{x_0}$ or $-\mathbf{w} + \mathbf{x_0}$ is on the same side of H as $\mathbf{x}$. Let ε assign this direction. Then for the real distance, using orthogonal decomposition: $d(\mathbf{x}, H) = (\mathbf{x} - \mathbf{x_0})\varepsilon\mathbf{w} = \varepsilon(\mathbf{wx} + b)$, which proves the statement. □

It is clear, that if $(\mathbf{w}, b)$ is a representation of H with unit length normal vector, the representation is still not unique, because $(-\mathbf{w}, -b)$ is also a representation of H with unit length normal vector. But if H separates the data points (Definition 2.1.2), then the representation of H with unit length normal vector is unique.

As mentioned earlier, our purpose is to find a hyperplane, whose minimal distance from the data points is maximal. We call this the optimal separating hyperplane.

Definition 2.1.4. *(Optimal separating hyperplane)*
A H^* hyperplane with unit length normal vector $\mathbf{w}^*$ and intercept b^* is

optimal, if $\mathbf{w}^*, b^*$ *is the optimum of the following problem:*

$$\begin{aligned} &\max_{\mathbf{w},b} && \min_{i=1,\dots,N} |\mathbf{w}\mathbf{x_i} + b| \\ &\textit{subject to} && y_i(\mathbf{w}\mathbf{x_i} + b) > 0 \textit{ for } i = 1, \dots, N \end{aligned} \tag{1}$$

Note that the condition is only that H *separates the datapoints.*

If we imagine the H hyperplane that separates the data points, represented by $(\mathbf{w}, b)$, then there is at least one negative example, which is the closest to H of the negatives, and also at least a positive example, which is the closest to H of the positives. Let $\mathbf{w}$ fixed, and try to change b to maximizing the minimal distance in (1). Then if we are changing b, we are only shifting the hyperplane between the closest negative and closest positive points. For a fixed $\mathbf{w}$, we maximize the minimal distance to the negative closest and positive closest, if we set b that H would be halfway between the closest negative and closest positive points. Thus, in searching for an optimal separating hyperplane, we can restrict the search by requiring an equal distance to the two classes. The previous idea proves the following statement.

Statement 2.1.2. *A* H^* *hyperplane with unit length normal vector* $\mathbf{w}^*$ *and* b^* *intercept is an optimal separating hyperplane if and only if* $(\mathbf{w}^*, b^*)$ *is the optimum of the following problem:*

$$\begin{aligned} &\max_{\mathbf{w},b} && r \\ &\textit{subject to} && y_i(\mathbf{w}\mathbf{x_i} + b) \geq r \textit{ for } i = 1, \dots, N \end{aligned} \tag{2}$$

From the previous idea follows that in (2), there is at least one positive and at least one negative example, where equality stands. We call these examples the support vectors because these points are supporting the hyperplane from the two sides. There are two hyperplanes parallel to the separating

hyperplane and containing the support vectors. These are called the support hyperplanes.

Definition 2.1.5. *Let H an arbitrary hyperplane, that separates the data points. Let r_+ and r_- the unsigned distance of the closest positive and closest negative example to the hyperplane. We define H's margin as $m_H = r_+ + r_-$. The margin is the distance of the two support hyperplanes.*

Now we are able to see, why this method is called maximum-margin classifier. From Statement 2.1.2, for an optimal hyperplane $r_+ = r_-$, thus in (2) we can also maximize $m_H = 2r$, and seeking for the maximal margin hyperplane. We would like to create the decision function from the maximal margin hyperplane, because we assume that future points belonging to either class would have less chance for misclassification by falling away from the same class points.

In (2), because we presume $\|\mathbf{w}\| = 1$, it is easy to read that every datapoint's distance from the hyperplane is at least r. We are trying to get bigger r, hence removing the data points away from the hyperplane, and with this, remove the positives away from the negatives.

We can get rid of the restriction $\|\mathbf{w}\| = 1$, because for every $(\mathbf{w}, b)$ representation of H, $(\frac{\mathbf{w}}{\|\mathbf{w}\|}, \frac{b}{\|\mathbf{w}\|})$ is also a representation of H with unit length normal vector. Using this idea, we can deduce the standard from of maximum-margin optimization:

Statement 2.1.3. *Let H^* a hyperplane represented by $(\mathbf{w}^*, b^*)$. H^* is an optimal separating hyperplane if and only if $(\mathbf{w}^*, b^*)$ is the optimum of the following problem:*

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \textit{subject to} \quad & y_i(\mathbf{w}\mathbf{x_i} + b) \geq 1 \textit{ for } i = 1, \ldots, N \end{aligned} \tag{3}$$

Proof. From Statement 2.1.2: H is an optimal hyperplane represented by $(\mathbf{w}, b)$ (now $\mathbf{w}$ is not necessarily unit length) if and only if $(\mathbf{w}, b)$ maximizes $2r$ (the margin) subject to $y_i(\mathbf{w}\mathbf{x_i} + b) \geq r\|\mathbf{w}\|$ for $i = 1, \ldots, N$. Because for every $\lambda > 0$ $(\lambda\mathbf{w}, \lambda b)$ also satisfies the conditions, we can assume $r\|\mathbf{w}\| = 1$. We got the following equivalent problem for the optimal separating hyperplane: Satisfying the conditions $y_i(\mathbf{w}\mathbf{x_i} + b) \geq 1$ for $i = 1, \ldots, N$, search $\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|} \iff \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\| \iff \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$. □

The formula (3) is the standard form of the SVM optimization problem, called the primal problem. Now, we are diving into (not very detailed) the solving of the optimization problem. We will need the major concepts and theorems of convex optimization [4, 5].

Definition 2.1.6. *Convex optimization problem:*

$$\begin{aligned} \min_x \quad & f(x) \\ \textit{subject to} \quad & x \in C \\ & g_i(x) \leq 0 \textit{ for } i = 1, \ldots, m \end{aligned}$$

where $C \subseteq \mathbb{R}^n$ *convex and* $f, g_i (i = 1, \ldots, m)$ *are convex on* C *and continuously differentiable on an open set which contains* C*.*

It is worth to derive the definition for the $C = \mathbb{R}^n$ case, because we can define every restriction with convex (linear) functions for our purposes.

Definition 2.1.7. *Convex optimization problem (on* $\mathbb{R}^n$*):*

$$\begin{aligned} \min_x \quad & f(x) \\ \textit{subject to} \quad & g_i(x) \leq 0 \textit{ for } i = 1, \ldots, m \end{aligned} \tag{4}$$

where $f, g_i (i = 1, \ldots, m)$ *are convex and continuously differentiable functions*

With this last formula, we see that the SVM primal problem falls into that problem set; it is a convex optimization problem. Because we know the linear separability of our data points, the constraints of the convex optimization problem could be satisfied altogether.

Without proving, consider the following great properties of convex optimization:

Statement 2.1.4. *For a convex optimization problem:*

1. *Every local minimum is a global minimum*
2. *The optimal set is convex*
3. *If the objective function is strictly convex, then the problem has at most one optimal point*

Applying this set of statements to the SVM primal problem:

Statement 2.1.5. *If the data points are linearly separable, then the SVM primal problem always has an optimal solution, and the optimal solution is unique. Furthermore, one can search the optimal solution in the space defined by the constraints with appropriate gradient descent algorithms (local minimum searching algorithms).*

Because SVM primal problem is also a quadratic minimization problem, many methods are available for solving the SVM problem. However, the fastest way of solving SVM is to solve the SVM dual problem. For this, we will need further concepts from convex optimization.

Statement 2.1.6. *(Karush-Kuhn-Tucker theorem):*
Let assume that in the (4) convex optimization problem, the set defined by

the constraints is nonempty. Then x^* is an optimal solution if and only if

$$\begin{aligned} \exists \mu_i \geq 0 \quad & (i = 1, \ldots, m) : \\ & \mu_i = 0 \text{ if } g_i(x^*) < 0 \\ & \nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) = 0 \end{aligned} \tag{5}$$

This (5) formula is called the KKT (Karush-Kuhn-Tucker) conditions.

Applying the KKT Theorem for the SVM primal problem, $(\mathbf{w}^*, b^*)$ is the optimal solution of the SVM primal problem if and only if:

$$\begin{aligned} \exists \mu_i \geq 0 \quad & (i = 1, \ldots, N) : \\ & \mu_i = 0 \text{ if } y_i(\mathbf{w}^* \mathbf{x_i} + b^*) > 1 \\ & \mathbf{w}^* = \sum_{i=1}^{N} \mu_i y_i \mathbf{x_i} \\ & 0 = \sum_{i=1}^{N} \mu_i y_i \end{aligned} \tag{6}$$

Here, the last line is the derivation with respect to b. The KKT conditions tell that only the support vectors affect the optimal solution ($\mu_i > 0$ only possible for support vectors). If we could find out the values of μ_i, we could easily determine $\mathbf{w}^*$ and also b^*. This appropriate μ_i values could be found with the Lagrangian dual problem.

Definition 2.1.8. *(Lagrange function)*
For a convex optimization problem (4) the Lagrange function:

$$L(x, \mu) = f(x) + \sum_{i=1}^{m} \mu_i g_i(x) \text{ for } x \in \mathbb{R}^n, \mu \in \mathbb{R}^m_+$$

For a fixed $\mu \in \mathbb{R}^m_+$ we define $L(\mu)$ as the following:

$$L(\mu) = min\{L(x, \mu) : x \in \mathbb{R}^n\}$$

Definition 2.1.9. *(Lagrangian dual problem)*
For a convex optimization problem (4) the Lagrangian dual problem is:

$$\max_{\mu \in \mathbb{R}_+^m} L(\mu)$$

This is also a convex optimization problem.

Without proof, the dual problem has the following properties:

Statement 2.1.7. *Let assume that for a convex optimization problem (4) the set defined by the constraints is nonempty. Then*

- *(Weak duality) If for $x^* \in \mathbb{R}^n$, $\mu^* \in \mathbb{R}_+^m$: $L(\mu^*) = f(x^*)$ then x^* is an optimal solution for (4) and μ^* is an optimal solution for the Lagrangian dual problem.*
- *(Strong duality) If x^* is an optimal solution for (4) then $\exists \mu^* \in \mathbb{R}_+^m$ for which $L(\mu^*) = f(x^*)$*
- *(KKT relation) If x^* is an optimal solution for (4) then μ is optimal solution of the Lagrangian dual problem if and only if (x^*, μ) satisfy the (5) KKT conditions.*

For the SVM optimization problem the Lagrange function is the following:

$$L(\mathbf{w}, b, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \mu_i y_i (\mathbf{w}\mathbf{x_i} + b) + \sum_{i=1}^{N} \mu_i$$

And the Lagrangian dual problem is this:

$$\max_{\boldsymbol{\mu} \in \mathbb{R}_+^N} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\mu})$$

Because of Statement 2.1.7 we can restrict the search in $(\mathbf{w}, b)$ for the optimal $(\mathbf{w}^*, b^*)$:

$$\max_{\boldsymbol{\mu} \in \mathbb{R}_+^N} L(\mathbf{w}^*, b^*, \boldsymbol{\mu}) \tag{7}$$

From (6) we know the connection between the $\mathbf{w}^*, b^*$ optimal solution and the $\boldsymbol{\mu}^*$ optimal solution of the Lagrangian, writing these into (7) we get the standard form of the SVM dual problem:

$$\begin{aligned} \max_{\boldsymbol{\mu}} \quad & \sum_{i=1}^{N} \mu_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mu_i \mu_j y_i y_j \mathbf{x_i} \mathbf{x_j} \\ \text{subject to} \quad & \mu_i \geq 0 \quad (i = 1, \ldots, N) \\ & \sum_{i=1}^{N} \mu_i y_i = 0 \end{aligned} \tag{8}$$

By solving the SVM dual problem we get an optimal $\boldsymbol{\mu}^*$. For creating the $\mathbf{w}^*, b^*$ optimal solution of the primal problem we recall the KKT conditions from (6). For $\mathbf{w}^*$:

$$\mathbf{w}^* = \sum_{i=1}^{N} \mu_i y_i \mathbf{x_i} = \sum_{\substack{\mathbf{x_i} \\ \text{support} \\ \text{vector}}} \mu_i y_i \mathbf{x_i}$$

. For b^* we select an $\mathbf{x_i}$ support vector (where μ_i not zero) and solving $y_i(\mathbf{w}^* \mathbf{x_i} + b^*) = 1$ for b^*.

The dual problem is a simpler quadratic programming exercise on the nonnegative orthant. The modern solver approaches are using gradient descent methods. One of the fastest ways to solve the optimization (especially for large data sets) is to perform a coordinate descent algorithm on the dual problem (LIBLINEAR [14] is using this).

In this section, SVM (problem) is referred to as a maximum-margin classifier (problem). In the following sections, I will evoke these as Max Margin problems.

2.2 Linearly non-separable case

Finding the optimal hyperplane when the data points are linearly separable was introduced in 1963 by Vapnik and Chervonenkis. However, the method

was only extended in the 1990s (by Vapnik and Cortes) to propose a similar solution when the data points are not linearly separable. The proposed solution enables for some data points to be on the "wrong side" of the separating hyperplane or the "wrong side" of the support hyperplane. It softens the margin (from the previous chapter's hard margin); therefore, we call this method the soft-margin classifier.

I would like to give a short illustrative deduction of the soft margin primal problem from the Max Margin primal (previous chapter) and then using convex optimization techniques for creating the dual problem.

We have N points in $\mathbb{R}^n$: $\mathbf{x_1}, \ldots, \mathbf{x_N}$, and each $\mathbf{x_i}$ has a label $y_i \in \{-1, 1\}$ for $i = 1, \ldots, N$. Here the data points are not linearly separable, so there is no hyperplane that separates the positive and negative examples.

We would like to create a similar method like in the linearly separable case but allowing some of the points to present on the wrong side of the support hyperplane. We will try to keep the balance between searching a hyperplane with a wider margin and allowing too many points on the wrong side of the support hyperplane.

By introducing slack variables $\xi_i \geq 0$ for $i = 1, \ldots, N$ we will control whether x_i is on the right place. We will also have a C hyperparameter that tries to control the previously mentioned balance. We will use the concepts and statements of convex optimization from the previous chapter.

Recalling (3) we wrote the Max Margin primal problem as the following:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w}\mathbf{x_i} + b) \geq 1 \text{ for } i = 1, \ldots, N \end{aligned} \tag{9}$$

For a condition $y_i(\mathbf{w}\mathbf{x_i} + b) \geq 1$, the inequality means that the point is on the right side of the separating hyperplane, but also on the right side of the support hyperplane. We will let the datapoints to present on the wrong side of the support hyperplane even on the wrong side of the separating

hyperplane. If $1 > y_i(\mathbf{wx_i} + b) > 0$, the datapoint would be on the wrong side of the supporting hyperplane but on the right side of the separating hyperplane. If $y_i(\mathbf{wx_i} + b) < 0$ then the point is also on the wrong side of the separating hyperplane. Let say we want to explicitly control how many datapoints would be misclassified (on the wrong side of the separating hyperplane). For this purpose one could write the following problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{wx_i} + b) \geq 1 - \xi_i \text{ for } i = 1, \ldots, N \\ & \xi_i \geq 0 \text{ for } i = 1, \ldots, N \\ & \sum_{i=1}^{N} \xi_i \leq K \end{aligned} \tag{10}$$

Here the value of K is determined by us. One advantage of this formula is that at most K misclassifications could occur. This is still a convex optimization problem, now for $\mathbf{w}, b, \boldsymbol{\xi}$. We could write the previous problem as a sort of penalized optimization with penalizing the differences from the support hyperplanes:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{wx_i} + b) \geq 1 - \xi_i \text{ for } i = 1, \ldots, N \\ & \xi_i \geq 0 \text{ for } i = 1, \ldots, N \end{aligned} \tag{11}$$

Here the value of C is determined by us and could be called the coefficent of the penalty. (Here the word penalty is used only for getting insight to the idea, the penalty term in statistical learning refers to an other thing.) In theory by setting $C = \infty$ we get the Max Margin primal for the linearly separable case. The formula (10) and the formula (11) are equivalent; for a K hyperparameter in the first exists a C in the second, that the optimal solution would be the same, and vice versa.

The formula (11) is the standard form of the SVM primal problem in the nonseparable case (or Soft Margin primal problem). This is also a convex optimization problem; thus we can apply the statements from the previous section.

Statement 2.2.1. *For a fixed $C < \infty$, the Soft Margin primal problem has an optimal solution, and that solution is unique. Furthermore, one can search the optimal solution in the space defined by the constraints with appropriate gradient descent algorithms (local minimum searching algorithms).*

As in the previous case, we could solve the dual problem instead of the primal for accelerating the solving process. The Lagrange function for the problem is the following:

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\mu_i(-y_i(\mathbf{w}\mathbf{x_i}+b) - \xi_i + 1) + \sum_{i=1}^{N} -\alpha_i\xi_i$$

Applying the KKT theorem for the Soft Margin primal problem, we get that $(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*)$ is the optimal solution if and only if:

$$\begin{aligned} \exists\mu_i \geq 0 \quad & (i = 1, \ldots, N) \\ \alpha_i \geq 0 \quad & (i = 1, \ldots, N) : \\ & \mu_i = 0 \text{ if } y_i(\mathbf{w}^*\mathbf{x_i} + b) > 1 - \xi_i^* \\ & \alpha_i = 0 \text{ if } \xi_i^* > 0 \\ & \mathbf{w}^* = \sum_{i=1}^{N}\mu_i y_i \mathbf{x_i} \\ & 0 = \sum_{i=1}^{N}\mu_i y_i \\ & \alpha_i = C - \mu_i \text{ for } i = 1, \ldots, N \end{aligned} \tag{12}$$

The dual problem could be solved by:

$$\max_{\substack{\boldsymbol{\mu}\in\mathbb{R}_+^N \\ \boldsymbol{\alpha}\in\mathbb{R}_+^N}} L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\mu}, \boldsymbol{\alpha})$$

Writing the KKT conditions into the dual we get the standard form of the Soft Margin dual problem:

$$\begin{aligned} \max_{\boldsymbol{\mu}} \quad & \sum_{i=1}^{N} \mu_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mu_i \mu_j y_i y_j \mathbf{x_i} \mathbf{x_j} \\ \text{subject to} \quad & 0 \leq \mu_i \leq C \quad (i = 1, \ldots, N) \\ & \sum_{i=1}^{N} \mu_i y_i = 0 \end{aligned} \tag{13}$$

This is the same problem as the Max Margin dual problem by setting $C = \infty$. The upper bound on μ_i does not change the set of applicable solver methods. Thus the solving part is the same; only the purpose is different. For real-world data sets, linear separability is not typical. Even if the linear separability occurs, a soft-margin method will usually result a better performing classifier (by handling better the bias-variance tradeoff).

2.3 Other SVM methods, further applications

It turned out, that the SVM methods make useful and accurate classifiers, partly because of the construction of the classifier does not depend on every data point, rather only on the support vectors, which could be considered as hard-to-classify points (they are the closest points to the separating hyperplane). At the same time, most of the linear classifiers built by other methods take into account every data point in some way.

Another great benefit of the method is that in the dual problem neither the construction nor the classifier obligated to use explicitly the vector (coordinates) of the data points, but could use only the dot products of these vectors. This led to the kernel trick. Saying it simple if $\phi : \mathbb{R}^n \to \mathbb{R}^k (n < k)$ a mapping with the property that $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n : K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})\phi(\mathbf{y})$ and K (the kernel) is nice (see [1] for precise characterisation), then constructing

a linear classifier (with a separating hyperplane) in the mapped space, one could only solve:

$$\begin{aligned} \max_{\boldsymbol{\mu}} \quad & \sum_{i=1}^{N} \mu_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mu_i \mu_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}) \\ \text{subject to} \quad & 0 \leq \mu_i \leq C \quad (i = 1, \ldots, N) \\ & \sum_{i=1}^{N} \mu_i y_i = 0 \end{aligned} \tag{14}$$

This problem is not harder than (13) if K is nice. For constructing the classifier it is sufficient again to use only kernel products:

$$\begin{aligned} f(\mathbf{x}) &= sgn(\mathbf{w}\phi(\mathbf{x}) + b) \\ &= sgn(\sum_{i=1}^{N} \mu_i y_i \phi(\mathbf{x_i})\phi(\mathbf{x}) + b) \\ &= sgn(\sum_{i=1}^{N} \mu_i y_i K(\mathbf{x_i}, \mathbf{x}) + b) \end{aligned} \tag{15}$$

The most widely used kernels are:

- RBF kernel: $K(\mathbf{x}, \mathbf{y}) = exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$
- Polynomial kernel: $K(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x}\mathbf{y} + r)^d$
- Sigmoid kernel: $K(\mathbf{x}, \mathbf{y}) = tanh(\gamma \mathbf{x}\mathbf{y} + r)$

These kernels (and other, specialised kernels as well) achieved great results in many classification tasks.

The SVM methods were further extended to solve other machine learning problems: multi-class classification (using pairwise two-class classifications in one vs. rest or one vs. one scheme and summarize the scores as a tournament), regression, and clustering.

Finally, the resulting classifier from the linear separation problem (not the kernelized version) could be easier interpreted than other classifiers if the $\mathbf{w}$ normal vector has many 0 coordinates. In the next section, I will go

around the topic of how we can get a classifier that is still accurate but using only a few features of the input.

3 Feature selection methods

The feature selection methods [6] emerged among the firsts when in a machine learning task (mostly in classification and regression tasks), the number of features (the dimension of input space) exceeded several hundred or even several thousand. A natural need came into being. The researchers wanted to explore how a learning machine makes decisions. For a classification task, when a classifier is accurate but uses multiple thousand features (a trait of the data point), it is hard to determine why the classifier decided this or that or what the main reasons were. The idea of feature selection is selecting only a relatively small subset of the available features while maintaining (or sometimes improving) the accuracy with the same type of classifier. The key assumption of feature selection is that there are irrelevant and redundant features among all the available features. By removing these features, we could expect that the accuracy does not decrease, and the resulting subset contains only the most relevant features. There are several reasons why one would do a feature selection:

- to increase interpretation power by using only a small number of features
- to shorten training time
- to increase generalization ability by reducing overfitting
- to explore the most relevant features of the current classification problem (e.g., for a new dataset [8])

The easiest way would be to test every possible feature subset, but it is computationally infeasible; hence the feature selection techniques must restrict their search space by applying heuristics or greedy algorithms. The main feature selection methods could be categorized as filters, wrappers, and embedded methods. Each category has it's own pros and cons. I would like to give a short introduction to these categories and their methods. I will

present mostly the SVM-related ones.

Feature selection could be more informative for a classification problem than individual feature assessment methods (e.g., statistical tests), because the relevant features may seem irrelevant alone but exactly related to the output altogether. The classification also uses this interdependency between the features. From an accurate classifier using a few features, the strong relation follows. The feature selection could be viewed as the assessment of feature subsets.

Usually, a feature selection method's evaluation is the following: the feature selection method gives a ranking on features (sometimes as nested subsets of features) or has a hyperparameter that controls the number of features. From the many possibilities ($\leq$ number of all features), one could select the best cutting point in the ranking (or best hyperparameter) by evaluating the feature subsets on a validation set, and selecting the "best" one. The best mostly means the highest accuracy, but it depends on the desired goal. For example, one could determine the maximal number of features or the minimal desired accuracy. Another option is penalizing the accuracy someway with the number of features. See Akaike Information Criterion or Bayesian Information Criterion, for example.

For the next sections, using the machine learning terminology let assume we have training data and test data: X_{train} and X_{test}. X_{train} contains the training examples, the size of the matrix is $n_{\text{training examples}} \times n_{\text{features}}$. Similarly X_{test} contains the test examples, and the size is $n_{\text{test examples}} \times n_{\text{features}}$. Let y_{train} and y_{test} the labels of training and test examples with size $n_{\text{training examples}}$ and $n_{\text{test examples}}$. We denote the ith training example with $X_{\text{train}}[i, :]$ and the jth feature vector (in the training examples) $X_{\text{train}}[:, j]$. If $S \subseteq [n]$, we denote with $X[:, S]$ the restriction of the X matrix for the features in S.

3.1 Filters

A filter method ranks (or selects) features only once before the learning. Examples:

- variance ranking; the variance of each feature is measured (in training and test altogether), the ranking is in descending order
- univariate statistics ranking; an univariate statistics is computed from each feature $X_{\text{train}}[:, j]$ and the labels y_{train}. Then we rank the features according to the statistics or the p-values. Some univariate statistics: χ^2, mutual information, correlation. (Selection could happen automatically by requiring an α upper bound on e.g., family-wise error rate or false discovery rate or false positive rate).
- prior knowledge selection; from previous experience the presumably most related features selected

The computation of the rankings (or selections) is fast, although filters do not capture the possible interdependencies between the features and rank highly correlated features similarly; that is why filters are not the best feature selection techniques. Filters could be used to reduce the number of features slightly (e.g., they may halve the features if there are several thousand).

3.2 Wrappers

A wrapper method performs an iterative learning-evaluating procedure, where the learning is usually the same as used in the final model construction. The most often used wrappers using forward or backward stepwise selection for constructing nested feature subsets, which gives a ranking on features. A simple overall of these wrappers:

- Forward Stepwise Selection; starting from the empty set, at each step, a new feature is added to the current set creating a chain of nested subsets. At a step, if S is the current feature set, one could test the

$S_j = S \cup \{j\}$ sets for $j \notin S$. The test could be learning and evaluating on $X_{\text{train}}[:, S_j]$, and the j that gives the S_j with the highest score is selected for the new subset. Finally, the nested subsets are compared on a validation set.

- Backward Stepwise Selection; starting from the set of all features, one could perform a similar method as in the Forward Stepwise Selection by evaluating the $S_j = S - j$ sets for $j \in S$ at a step. Although, more efficient assessment is available (for SVM) by observing only the lack of feature j on the classifier learned on S. Finally, the nested subsets are compared on a validation set.
- Recursive Feature Elimination (RFE); this wrapper is mostly used with SVM, but other linear classifiers are also acceptable. It is a Backward Selection, starting from the set of all available features. At a step, we have the current set S, and we use the linear classifier f trained on $X_{\text{train}}[:, S]$. The linear f gives a weight w_j for each feature $j \in S$. The least important feature is the one with the lowest w_j^2. This feature is removed, at the next step follows. One could accelerate the process in the removing stage by dropping multiple features with the lowest w_j^2 weights. Finally, the nested subsets are compared on a validation set.

The computation of the rankings in usually slow (compared to other feature selection types), but the wrappers perform well according to experience. The RFE with SVM is widely used, see, e.g. [8].

3.3 Embedded methods

An embedded method is paired to the model construction usually by adding a regularization term (or penalty term) to the learning optimization problem which tries to penalize on the number of features. For a linear regression, the ridge regression and the lasso regression are the corresponding embedded

methods. One could construct an embedded method for the SVM learning too. We could rewrite the non-separable linear SVM primal problem (11) as the following:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} |1 - y_i(\mathbf{w}\mathbf{x_i} + b)|_+ \tag{16}$$

In the sum the so called hinge loss could be found. Other types of linear learning methods could be constructed by changing the loss function. However this (16) rewriting of the SVM primal problem can already be considered as a regularized optimization, where we are trying to find a separating hyperplane but regularize on the L2 norm of $\mathbf{w}$ (the size of $\|\mathbf{w}\|$). So the SVM primal problem already contains a penalty term, but this plain type could not be used for feature selection because it does not shrink the w_j coefficients to zero, only erase the completely irrelevant ones (e.g. where $\mathbf{x_{ij}}$ is zero for every i). We want to penalize the number of features would occur in the final linear model, thus we can introduce the $\|.\|_0$ L0 "norm" or zero "norm" (just a name, not norm mathematically) for which $\|\mathbf{w}\|_0 = \sum_{j=1}^{n} \mathbb{I}(w_j \neq 0)$ is exactly the number of used features by $\mathbf{w}$. With a hyperparameter C which controls the importance of feature selection, one could write the following:

$$\min_{\mathbf{w},b} \|\mathbf{w}\|_0 + C\sum_{i=1}^{N} |1 - y_i(\mathbf{w}\mathbf{x_i} + b)|_+ \tag{17}$$

Here the second term could be written again into a linear term in the optimization and some linear constraints. The resulting optimization is a huge integer programming exercise, which is infeasible in practice. Rather, we could decrease the number of features with other functions: $\|\mathbf{w}\|_1 = \sum_{j=1}^{n} |w_j|$ which is the L1 norm of $\mathbf{w}$ or $f(\mathbf{w}) = \sum_{j=1}^{n}(1-\exp(-kw_j^2))$ which is a differentiable approximation of the L0 norm. In practice the L1 norm is used, because with varying the value of C every possible number for the

remaining features is available and the optimization

$$\min_{\mathbf{w},b} \|\mathbf{w}\|_1 + C \sum_{i=1}^{N} |1 - y_i(\mathbf{w}\mathbf{x_i} + b)|_+ \tag{18}$$

is a linear programming exercise, which is solvable by standard techniques (but not as fast as L2 regularized SVM). Other possibility is to use a combination of the two norm: with α hyperparameter $\|\mathbf{w}\|_\alpha = \alpha\|\mathbf{w}\|^2 + (1-\alpha)\|\mathbf{w}\|_1$ which is the same as elastic net penalty for linear regression.

The embedded methods are fast compared to wrappers, and their performance is usually comparable. Still, an embedded method implicitly controls the number of features (wrappers could control explicitly), and it does not give a ranking on the features.

3.4 Notes on the feature selection methods

At the beginning of this section, I mentioned the natural need for being able to reason the decision of a classifier. This need is not always necessary and not always desirable. The black-box behavior of an AI system should remain acceptable while the empirical results are confirming the correctness. Thus this need should depend on the application.

I did not pay too much attention to describing the proper evaluation of a feature selection method. It could depend on the application and the amount of available data. If enough data is available, one could split the data set into training, validation, and test examples. Constructing the classifier from training and validation data (as described above, training data for feature ranking, validation data for choosing the best feature subset) the accuracy of the classifier on the test data is an unbiased estimation of this concrete classifier's performance on future examples (from the same distribution). The result is a concrete classifier that can be used for real classification purposes.

If the amount of data is not sufficient or the variance of the data is great, then the way of splitting the data would strongly affect the classifier's performance. In this case, K-fold cross-validation is suggested for estimating accuracy. The final classifier is computed from all the available data, but the mean accuracy from the cross-validation is not really estimating the final classifier's future performance, rather the method's performance (averaging on training sets) [1]. When using feature selection, one common mistake is selecting the important features from all of the data, restrict every data point to these important features, then performing evaluation (either regular or cv) on the data. Because we are using the test labels in the feature selection (except for some filters), the resulting accuracy would be a poor estimate in both cases.

Feature selection could be used to identify the most important features related to a measured binary output (with this, related to a classification problem). One could get a small feature subset that is strongly connected with the output. Here the importance of a feature subset is the predictive ability from the feature subset, hence the proper evaluation of the method remains necessary.

For a review of feature selection methods, see [6].

4 Experiments

The evolution of machine learning algorithms is mostly directed by their achievements on either virtual or benchmark or real-world data set. Their theory is still important, but their performance on exercises is the leading measure. In this Experiments section, we will consider a real-world data set; 1064 individuals' brain graphs for presenting the previously seen classification and feature selection methods' strength. The brain graph is a relatively new macroscopic model of the human brain based on the axon connections between gray matter areas. Firstly we will see in brief how a brain graph is created; then, we will use SVM for classifying the individuals according to their sex based on their brain graphs. Lastly, we will select a small subset of all the occurring brain graph edges (with feature selection techniques) that still capture the main differences between male and female individuals.

4.1 About brain graphs

In the field of neuroscience, mapping a living creature's neuronal connections is an old desire. The roundworm C.elegans was the first (and the only so far) animal whose neuronal connections are fully mapped; this small animal already has 302 neurons. In the human brain, approximately 86 billion neurons could be found, therefore mapping every brain connection in the neural level is currently impossible. Instead, a macroscopic connection diagram had been considered where the connections between brain areas were measured, not between individual neurons.

The connectome or brain graph of the individual is a spatial graph where the nodes are the corresponding brain areas or ROIs (Regions of Interest), and the edges between two nodes are the axonal fibers running between the two corresponding brain areas. In creating the individual's brain graph, the first step is a Diffusion-Weighted Magnetic Resonance Imaging (DW-MRI)

record. The brain consists of white matter areas and gray matter areas. The axon connections form between two gray matter areas, but the axonal fiber runs mostly in the white matter. During the DW-MRI recording, the flow of water molecules in the white matter measured. The flow of these water molecules is higher along the fibers than across the fibers. The computation of the direction of the flow allows us to determine the direction of the axons. From these diffusion directions, the 3D imaging technique called tractography could compute the paths of the running axons in the white matter. (The axon follows the direction of the eigenvector with the largest eigenvalue of the diffusion tensor at every point, whether a flow has any direction is computed from the fractional anisotropy). Finally, the brain graph is created from the tractogram (tractography created image); namely, there is an edge between two brain areas if, in the tractogram there is a running fiber between the two gray matter areas. Various edge weights could be assigned to the edge: the number of fibers (how many axonal fibers are in the tractogram between the two areas), mean fiber length (the average length of these fibers), mean fractional anisotropy (the average fractional anisotropy along the fibers), and other, derived edge weights, e.g., some sort of electrical conductivity $\frac{\text{number of fibers}}{\text{mean fiber length}}$.

At PIT Bioinformatics Group (ELTE), Varga Bálint and Grolmusz Vince made great efforts to create the best quality brain graphs in larger and larger quantities. So far, they determined 1064 individuals' brain graphs. Five different sizes of parcellation of the brain were considered with 83, 129, 234, 463, and 1015 nodes. The previously mentioned edge weights: number of fibers, mean fiber length, mean fractional anisotropy were also computed at every scale. Now, these brain graphs are publicly available at `https://braingraph.org/`.

In the classification and feature selection tasks, I used all of the 1064 individuals' 83-vertex brain graph.

4.2 Sex classification from brain graph connections

According to previous studies from both neuroscience and connectomics, male and female brains and brain graphs differ in numerous ways. The studies revealed differences in global brain properties (brain size, brain weight, gray/white matter proportion, weight, size, etc.), in local brain properties (size, shape and location, functions of specific brain areas), in graph theoretical parameters of brain graphs [10, 11, 12], but did not observe the difference in the connection level, especially not when observing a bunch of connections altogether. The problem of sex classification from a brain graph structure is rather a theoretical problem than a practical one, for a former solution see [9]. The gender of the subject is an objectively determined binary property. Therefore, if we think that brain graph is a correct macroscopic model of the brain that captures the individual's main traits (firstly the objective ones), and if we think that female and male brains differ, then we should be able to decide whether a brain graph belongs to a male or female individual. This could be translated as a classification problem, where the main goal, classifying whether a subject is male or female, is not practically useful in itself. But by solving the task accurately, we capture the main differences and confirm the correctness of the brain graph model. Classification of sex with feature selection techniques could reveal new (multidimensional) differences of the brain graphs between males and females. This helps to understand how we are functioning, but could also help understand and cure sex-related mental diseases.

In the sex classification task I used the 83-vertex brain graphs of 1064 individuals. I neglected the graph structure and used the vector of edge weights for learning. Among the 1064 individuals 1950 possible edges occurred, therefore I could consider the X matrix containing the edge weights with size 1064×1950. The ith row of X corresponds to the ith individual's edge weight vector. The following weights was considered: 'un'

- unweighted (1 if there is edge else 0), 'fn' - number of fibers, 'fl' - mean fiber length, 'fa' - mean fractional anisotropy, and some derived weights; 'con1' = $\frac{\text{fn}}{\text{fl}}$, 'con2' = $\frac{\text{fn}\times\text{fa}}{\text{fl}^2}$. The con1 is the original electrical conductivity related quantity, con2 was found experimentally for improving classification performance. The quantity con2 also could be viewed as a sort of electrical conductivity. After a weight is selected, a proper normalization follows: minimum-maximum normalization (minmax) or standardization (std). At each normalization, every feature (edge) normalized individually by the following:

Minimum-maximum normalization:

$$X[:,j] := \frac{X[:,j] - \min_{i=1}^{1064} X[i,j]}{\max_{i=1}^{1064} X[i,j] - \min_{i=1}^{1064} X[i,j]} \quad j = 1,\ldots,1950$$

Standardization:

$$X[:,j] := \frac{X[:,j] - a_j}{s_j} \quad j = 1,\ldots,1950 \quad \text{where:}$$

$$a_j = \frac{1}{1064}\sum_{i=1}^{1064} X[i,j]$$

$$s_j = \sqrt{\frac{1}{1064}\sum_{i=1}^{1064}(X[i,j] - a_j)^2}$$

I used only SVM classifiers (mostly linear ones). The learning process was the following: given a training set and a test set; first, I determined the hyperparameters on a fine grid by 10-fold cross-validation on the training data. I chose the hyperparameters with the highest accuracy on the training set. Then with fixed hyperparameters, a model constructed from the training data and evaluated on the test data. Because of the relatively small number of data points and high variance, I evaluated the learning process with (an outer) 10-fold cross-validation. I used every fold once for the testing, and the others for training. From the 10 runs of evaluation, the final measure was the mean accuracy of these 10 runs' accuracy.

Firstly I built a linear SVMs for the defined weights and measured the accuracy; the deviation is the standard deviation from the 10 cross-validation runs. Table 1 shows the results. The fl^{-1} weight is also showed because it is more natural than fl as if no edge is found, then 0 weight is assigned. The proper normalization found by trying both.

Table 1: Linear SVM classifier on different edge weights

Weight	Normalization	Accuracy
un	none	77.8 ± 5.0
fn	minmax	82.9 ± 3.9
fl	std	84.3 ± 4.4
fl^{-1}	std	83.6 ± 3.5
fa	minmax	82.1 ± 5.1
$\mathrm{con1} = \frac{\mathrm{fn}}{\mathrm{fl}}$	std	83.8 ± 4.0
$\mathrm{con2} = \frac{\mathrm{fn} \times \mathrm{fa}}{\mathrm{fl}^2}$	minmax	85.7 ± 4.0

As we can see, approximately 85% accurate classification of sex from brain graph structure is possible. The choice of normalization did not alter the resulting accuracy that much (maximum 1%). By refining the hyperparameter grid a 0.5% improvement available for some weights, see Table 2 for fn and con2.

Table 2: Linear SVM classifier with finer hyperparameter grid

Weight	Normalization	Accuracy
fn	minmax	83.5 ± 4.8
$\mathrm{con2} = \frac{\mathrm{fn} \times \mathrm{fa}}{\mathrm{fl}^2}$	minmax	86.3 ± 3.5

Neither of the commonly used kernels (polynomial, rbf, sigmoid) could achieve better results than the linear SVM. Table 3 shows the results.

Table 3: SVM classifiers on con2 weight

Weight	Normalization	Kernel	Accuracy
con2 = $\frac{\text{fn}\times\text{fa}}{\text{fl}^2}$	minmax	linear	86.3 ± 3.5
con2 = $\frac{\text{fn}\times\text{fa}}{\text{fl}^2}$	minmax	polynomial(d=4)	85.9 ± 3.8
con2 = $\frac{\text{fn}\times\text{fa}}{\text{fl}^2}$	minmax	rbf	86.4 ± 3.8
con2 = $\frac{\text{fn}\times\text{fa}}{\text{fl}^2}$	minmax	sigmoid	83.7 ± 3.9

As a final result, above 86% accurate sex classification from the brain graph is possible using support vector machines with linear kernel.

However, our main desire is to select a small number of edges that already determine the gender of the individual (applying a proper feature selection method). We should choose the weight and classifier for the feature selection. Among the classifiers, the linear is the best interpreted in terms of individual edge weights; in the linear classifier, the effects of the different edges add up. Among the weights, probably the best choice is the number of fibers (fn). The weight fn is easily interpreted; it describes the graph structure of the brain. At the same time, probably fn is not affected by previously observed properties (as fl by the size of the brain), therefore finding differences less probably reveals another view of an already observed truth. We will apply the feature selection for the linear classifier using the fn weights.

4.3 Sex-related connection subset selection

In the feature selection task, our goal is to propose a method which creates a (relatively) accurate model that uses only a small number of features. If we evaluate the method traditionally (training-validation-test split), we will be able to estimate the performance of a concrete classifier. Again, the relatively small number of data points and high variance in data could result really different performances and feature subsets as the test set selected.

By applying cross-validation evaluation for the method, we can estimate the method's average performance (on variable training and test sets) [1]. Still, on different training sets, the method selects different feature subsets. Various stability measures had been proposed for measuring this difference [7]. This stability related quantities could be computed in the cross-validation evaluation.

The stability of a feature selection method alone is not very informative, but in comparing two feature selections, it could be. Higher stability says that mostly the same features are selected as the training set varies. Paying attention to this is important if we want to explore the most related features because we could include/exclude features "falsely". For many real-world data sets (e.g., Braingraph Data, Gene Expression Data), it is usually impossible for a method (except for filters) to have maximal stability (selecting the same feature subset always), because of the huge number of highly correlated and highly dependent features.

The cross-validation for a method would result an (average) accuracy and a stability quantity. One could select the best method (or it's parameters) based on accuracy, stability, and (average) number of selected features (nsf for short). Finally, one could report the relevant features by considering the resulting feature subset from the full dataset and/or by considering the "common" features of all cross-validation runs. The nature of the feature selection methods (except some filters) indicates that the reported relevant features try to be independent. Remember that the importance is assigned based on the method's accuracy, but constructing a concrete classifier is not our goal now. The stability quantity refers to what we can expect when we compare two selected feature subsets from two independent (or partly independent) data sets using the same method.

Special cross-validation, 2-fold cv might also be useful (besides the classic 5- or 10-fold cv), because the two feature subsets would be independent. I

will use both a 2-fold cv and a 10-fold cv for evaluation.

For the linear classifier using fn weights, the following feature selections would be observed: L1 regularized linear SVM, Recursive Feature Elimination, and correlation filter. I will denote these methods as L1-SVM, RFE-SVM, and CF. (I described each of these selection methods in the Feature selection methods section.)

For comparing these methods' effects I depicted the (average) accuracy as a function of the (average) nsf and the stability as a function of the (average) nsf. These plots came from the 10-fold cv as the key hyperparameter varies. The key hyperparameter in L1-SVM is the C parameter (from formula (18)), in RFE-SVM and CF is the number of selected features (nsf). Remember that L1-SVM implicitly controls the number of selected features, therefore here an average calculated from the 10 runs. I maximized the desired number of features in 50. The stability is measured in average Jaccard index [7] between the resulting feature subsets. If S_k $k = 1, \ldots, 10$ are the resulting feature subsets from the 10 cv runs, then if $i \neq j$ the Jaccard index is:

$$J(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$$

The stability is the average of Jaccard indicies:

$$S = \frac{1}{45} \sum_{i=1}^{9} \sum_{j=i+1}^{10} J(S_i, S_j)$$

Figure 1 shows the plot of the score (or accuracy) as a function of (average) nsf for the three selection methods. Remember, the best mean accuracy for fn weight was 83.5% with the use of (almost) all of the 1950 features. Here, 80% accurate classifiers were constructed using less than 30 features.

Figure 2 shows the plot of stability as a function of (average) nsf for the three selection methods. As we can see, CF outperforms RFE and L1-SVM in

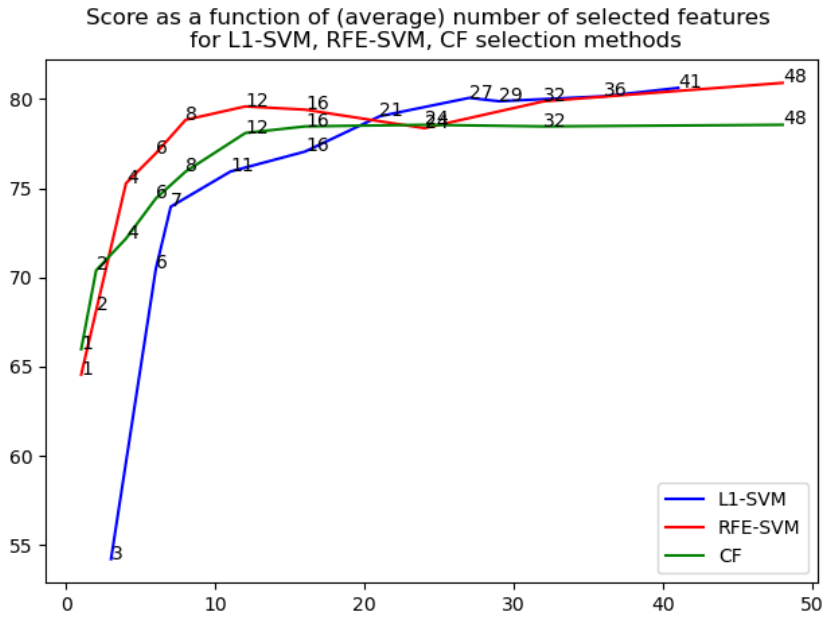

Figure 1: Score as a function of (average) number of selected features

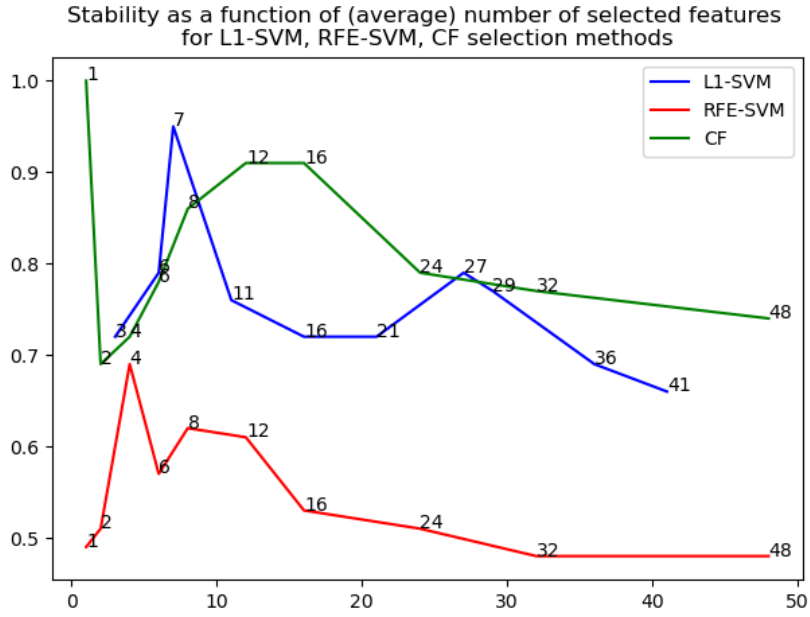

Figure 2: Stability as a function of (average) number of selected features

stability, but in CF, features are more correlated with each other. It depends on the application, whether we want correlated and dependent features as the output.

For us, as the score plot shows, when the features are more correlated, less "information" is stored with the same number of features (CF vs. RFE-SVM). Because we easily find interchangeable, correlated features we interested in the more independent subsets. Therefore we should select the RFE method. The optimal number of features for our purpose is not defined, but looking at these three methods' performances, the 12 feature selector RFE achieves 79.6% on average, which is almost the maximum among these three methods' performances on less than 50 features. We sacrifice roughly 4% accuracy in order to decrease the used number of features to $\frac{12}{1950}\% \approx 0.6\%$.

Finally, the RFE method that selects 12 features performs approximately 79.6% accurately as the training and test sets drawn from the same distribution.

This RFE method is not extremely stable because of the interchangeable correlated features. But it is stable enough that we can conclude what connections differ mostly (and independently) between males and females in terms of the number of fibers.

Applying the RFE 12 feature method to the full dataset, we can report 12 edges, that are very related to the sex on the macroscopic level. Assuming that our dataset is ordinary, the 79.6% estimation also stands for this concrete classifier built from these 12 edges. (This assumption, or rather this estimation is commonly used.) Table 4 shows these 12 edges, and which cross-validation runs they occurred.

Ranking	Vertex1	Vertex2	0	1	2	3	4	5	6	7	8	9
1	lh.superiorparietal	Left-Caudate	✓	✓	✓	✓	✓		✓	✓	✓	✓
2	lh.posteriorcingulate	Left-Putamen	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3	rh.caudalanteriorcingulate	Right-Caudate	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4	rh.precuneus	Right-Putamen	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5	rh.precuneus	Right-Hippocampus	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
6	lh.lingual	Left-Thalamus-Proper	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7	rh.insula	Right-Putamen	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
8	lh.fusiform	Left-Hippocampus	✓	✓	✓	✓	✓		✓	✓		✓
9	lh.rostralmiddlefrontal	lh.insula	✓	✓	✓	✓	✓	✓	✓	✓	✓	
10	Right-Caudate	Right-Putamen				✓	✓	✓				
11	rh.lateraloccipital	Right-Putamen	✓			✓					✓	
12	lh.rostralmiddlefrontal	Left-Caudate		✓	✓				✓			✓

Table 4: RFE - 12 features

Remember that RFE gives a ranking on the features. In Table 4, this ranking gives the order. The first 9 edges seem stable as the training set varies. From the 83 vertices, it seems there are more sex-related brain areas. Of the 24 place, the following areas (aside from hemisphere) occupy the most: Putamen(5), Caudate(4), Hippocampus(2), insula(2), precuneus(2), rostralmiddlefrontal(2). Figure 3 shows the correlations of these 12 features. There are still a few features that are correlated a bit.

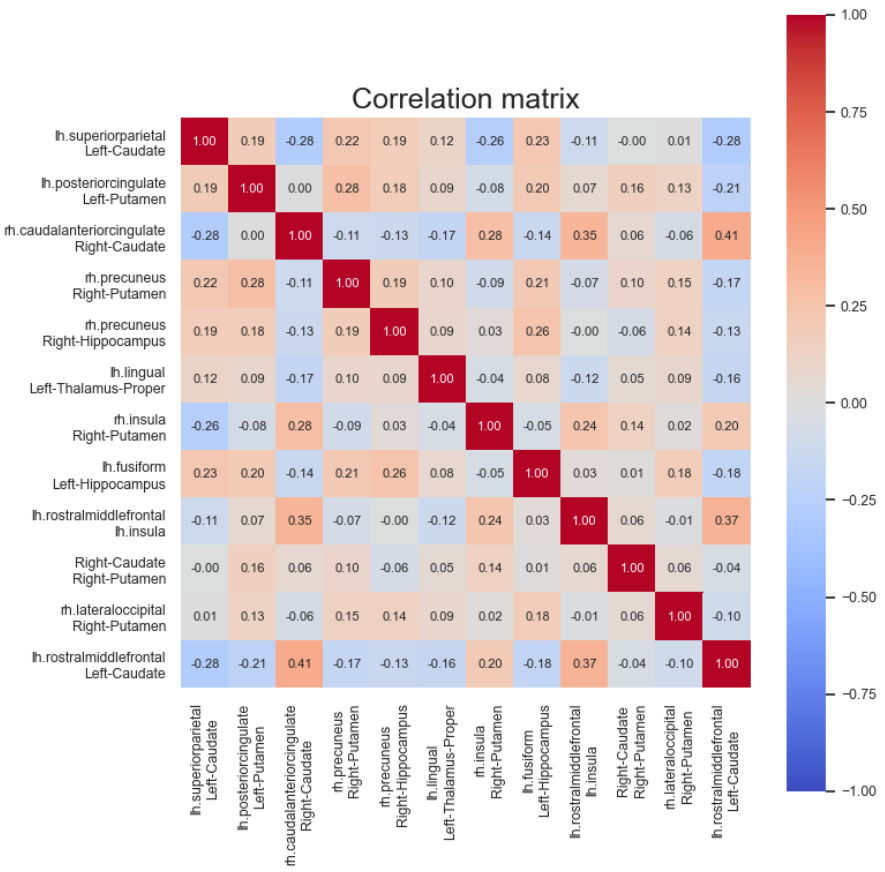

Figure 3: Correlations of the 12 selected features

One more interesting stability test was made. I performed the RFE 12 feature selection independently on the first and second half of the data. Table 5 shows the reported 12 features' ranks in the halves' rankings.

Ranking	Vertex1	Vertex2	1st half	2nd half
1	lh.superiorparietal	Left-Caudate	5	1
2	lh.posteriorcingulate	Left-Putamen	1	5
3	rh.caudalanteriorcingulate	Right-Caudate	2	2
4	rh.precuneus	Right-Putamen	8	6
5	rh.precuneus	Right-Hippocampus	3	4
6	lh.lingual	Left-Thalamus-Proper	24	11
7	rh.insula	Right-Putamen	11	7
8	lh.fusiform	Left-Hippocampus	21	15
9	lh.rostralmiddlefrontal	lh.insula	14	16
10	Right-Caudate	Right-Putamen	52	19
11	rh.lateraloccipital	Right-Putamen	36	68
12	lh.rostralmiddlefrontal	Left-Caudate	23	3

Table 5: The 12 features' ranks from the 1st and 2nd half of data

The top 5 reported edges gained top ranks in both halves, which confirms the importance of these edges again. If we denote with R the set of the reported 12 features and with SH_1 and SH_2 the set of the top 12 features from the 1st and 2nd half of the data, then $|SH_1 \cap SH_2| = 7$, $|R \cap SH_1| = 6$, $|R \cap SH_2| = 8$ and $|R \cap SH_1 \cap SH_2| = 6$. These ratios surely depend on the number of data points, but a simple estimation can be made: half of the RFE-12 features remain stable.

From the 12 reported edges, between the vertex pairs at rank 3, 7, 9, and 12, more fibers occur in males than in females. Between the other vertex pairs from the ranking, more fibers occur in females. Figure 4 shows the location of these 12 edges in the human brain by illustrating the many running fibers with only a single straight line. The brain is shown from the upper view, and the Y-axis indicates the front of the brain at 80-100 (mm) and the back of the brain at 20-40 (mm). The brain is stretched along the X-axis for better

visibility, and only the vertices belong to the 12 vertex pairs are shown. The blue lines indicate edges where males have more fibers. The red lines indicate edges where females have more.

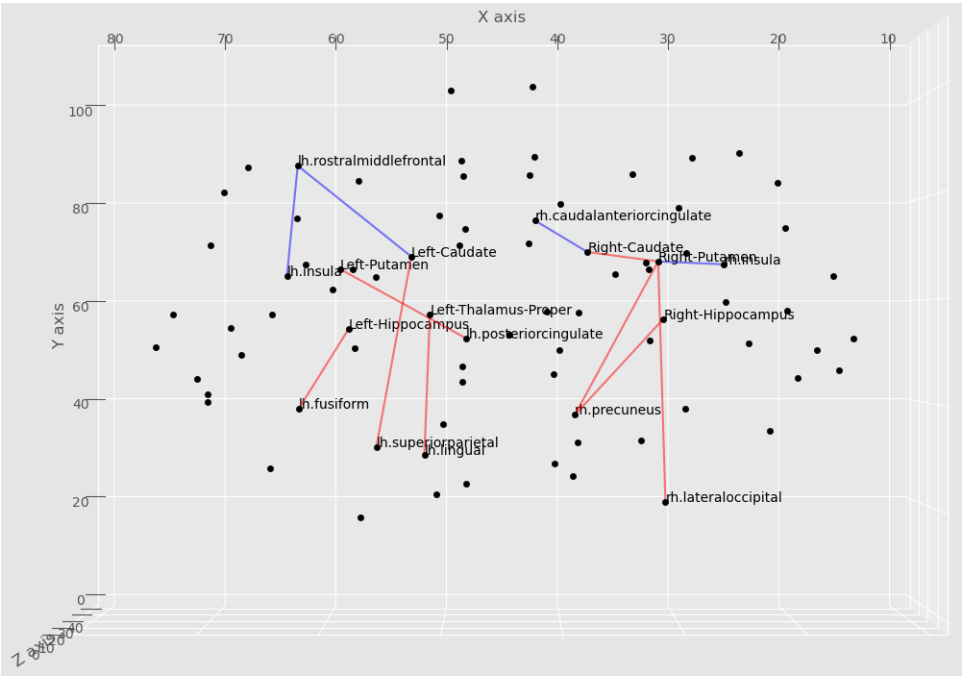

Figure 4: The reported 12 edges in the human brain

With the usage of support vector machines and feature selection, we revealed 12 important edges. The tests for the used method confirmed the importance of these edges; the method is approximately 80% accurate, (presumably) roughly 80% accurate classifier could be built on the reported 12 edges. Stability tests showed that interchangeable edges could occur in the 12 edges, but half of them remain stable.

The location of these edges suggests that male edges are in the front of the brain, and female edges are in the back. I wondered if it is right in general. I selected a p-value ($p = 10^{-8}$), and with the 2-sample Kolmogorov-Smirnov test, I found 158 edges at which the fn weight differs between males and

females (with Pr(Type I Error) $< p$). This collection of edges could be called significant altogether because 1950 tests were made, and the family-wise error rate (any edge was falsely called significant) is less than 2×10^{-5}. Figure 5 shows the location of these edges. Again, blue indicates male and red indicates female edges.

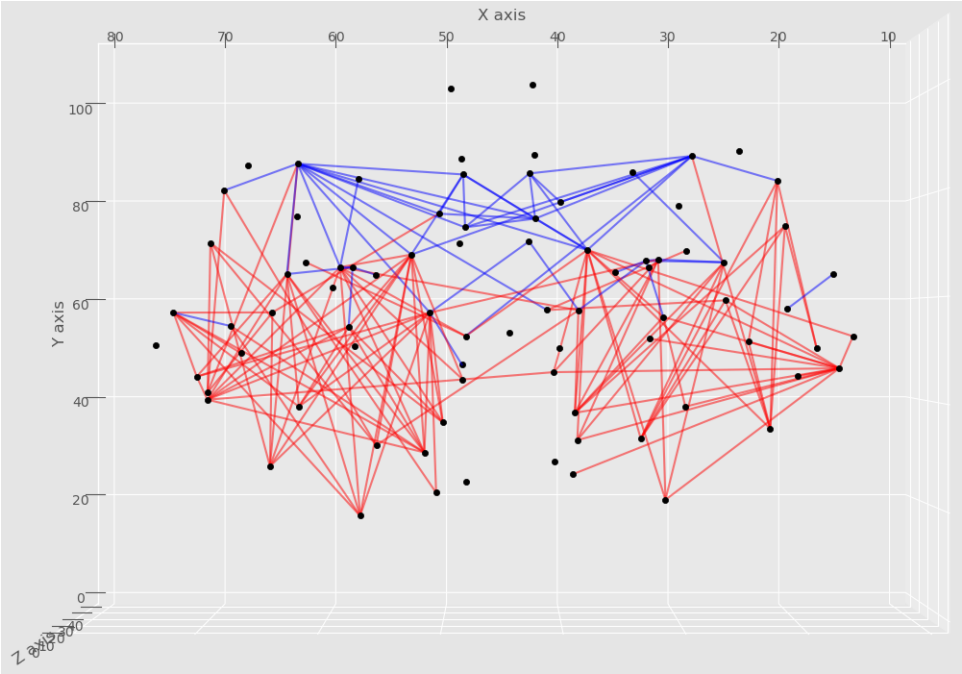

Figure 5: Location of edges that are significantly different between males and females

The location of the selected 12 edges revealed a general truth of the location of sex-related edges.

4.4 Software for experiments

I used Python 2.7 and 3.7 for the experiments. The appropriate packages were used for the tasks; NetworkX for handling graphs, NumPy and Pandas for

handling arrays, Scikit-Learn for SVM classifications and feature selection, Matplotlib and Seaborn for figures, Scipy for statistical tests [13].

References

[1] Hastie, T., Tibshirani, R.,, Friedman, J. (2001). The Elements of Statistical Learning. New York, NY, USA: Springer New York Inc..

[2] Thomas M. Mitchell. 1997. Machine Learning (1st. ed.). McGraw-Hill, Inc., USA.

[3] Cortes, C., Vapnik, V. Support-vector networks. Mach Learn 20, 273–297 (1995). https://doi.org/10.1007/BF00994018

[4] Király Tamás, Papp Olga. Operációkutatás II. jegyzet

[5] Stephen Boyd, Lieven Vandenberghe. Convex optimization

[6] A. Jović, K. Brkić and N. Bogunović, "A review of feature selection methods with applications," 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2015, pp. 1200-1205, doi: 10.1109/MIPRO.2015.7160458.

[7] Utkarsh Mahadeo Khaire, R. Dhanalakshmi, Stability of feature selection algorithm: A review, Journal of King Saud University - Computer and Information Sciences, 2019, , ISSN 1319-1578, (http://www.sciencedirect.com/science/article/pii/S1319157819304379)

[8] Guyon, I., Weston, J., Barnhill, S. et al. Gene Selection for Cancer Classification using Support Vector Machines. Machine Learning 46, 389–422 (2002). https://doi.org/10.1023/A:1012487302797

[9] Joshua T. Vogelstein, William R. Gray, R. Jacob Vogelstein, and Carey E. Priebe. 2013. Graph Classification Using Signal-Subgraphs:

Applications in Statistical Connectomics. IEEE Trans. Pattern Anal. Mach. Intell. 35, 7 (July 2013), 1539–1551. DOI:https://doi.org/10.1109/TPAMI.2012.235

[10] Balázs Szalkai, Bálint Varga, Vince Grolmusz: Graph Theoretical Analysis Reveals: Women's Brains Are Better Connected than Men's. PLoS ONE 10(7): e0130045 (2015) doi: http://dx.doi.org/10.1371/journal.pone.0130045

[11] Balázs Szalkai, Bálint Varga, Vince Grolmusz: Brain Size Bias Compensated Graph-Theoretical Parameters are Also Better in Women's Structural Connectomes, Brain Imaging and Behavior Vol. 12, No. 3, pp. 663-673, (2018) http://dx.doi.org/10.1007/s11682-017-9720-0

[12] Máté Fellner, Bálint Varga, Vince Grolmusz: The Frequent Network Neighborhood Mapping of the Human Hippocampus Shows Much More Frequent Neighbor Sets in Males Than in Females; PLOS ONE 15(1): e0227910 (2020). https://doi.org/10.1371/journal.pone.0227910

[13] Jake VanderPlas. 2016. Python Data Science Handbook: Essential Tools for Working with Data (1st. ed.). O'Reilly Media, Inc.

[14] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. J. Mach. Learn. Res. 9 (6/1/2008), 1871–1874.