TEL-AVIV UNIVERSITY
RAYMOND AND BEVERLY SACKLER
FACULTY OF EXACT SCIENCES
THE BLAVATNIK SCHOOL OF COMPUTER SCIENCE

Deep Representation Learning For Cluster Analysis

Thesis submitted in partial fulfillment of the requirements
for the M.Sc. degree of Tel-Aviv University
by
Daniel Carmon

The research work for this thesis has been carried out at
Tel-Aviv University under the supervision of

Prof. Amir Globerson

July 2018

Acknowledgments

First, I would like to thank my advisor, Prof. Amir Globerson, for his devoted advising and many hours of discussions. He was always open-minded and encouraged both self-thinking and thorough brainstroming.

I would also like to thank Dr. Yoav Goldberg for great insights and thoughtful suggestions.

To my officemates, for many discussions, fun times and being ever-willing to help.

Finally, to my loving family, thanks for your endless support and patience.

Abstract

Cluster Analysis is the important and well-studied problem of finding grouping patterns in data, and has found vast applications across many fields of science. Most existing clustering algorithms rely on data to be represented as points in $\mathbb{R}^n$, and their output is only of interest if geometric distance between these points sufficiently corresponds to the similarity notion of interest. For most real-world data, and in particular for images, such a correspondence generally does not hold. There is thus interest in algorithms for learning a data representation such that high-level similarity notion of interest between data points can be encoded as geometric distances, and thus can be easily read-out.

We present an algorithm to achieve this goal, by composing a differentiable formulations of the K-Means++ algorithm on top of a Deep Neural Network which operates as an embedding module. By end-to-end differentiation of this pipeline, we can train the network's parameters to learn an embedding such that relevant data semantics are easily read-out, and can thus be employed by downstream clustering procedures.

We empirically demonstrate that this straightforward method reaches state-of-the-art/competitive performance on several real-world image analysis benchmarks, in which the representation learned from the set of examples generalizes successfully to find meaningful clusters in data from different image categories.

Contents

Chapter 1

Introduction

There are many machine learning tasks in which the crux of the problem is in acquiring a representation for data such that meaningful features or relationships are easily read-out and exploited. E.g., the ubiquitous ML task of classification can be solved by looking at the nearest labeled neighbor, given that one had the proper metric in the first place. Another example is the task of information retrieval, which given the proper metric can be reduced as well to nearest neighbor queries.

In tasks related to machine understanding of high-level semantics, the ability to learn a meaningful data representation is especially important since data is usually given in ”low-level” terms (e.g. pixel intensities in digital images, audiograms for sound recordings, etc.) from which high-level concepts cannot be naively decoded, since individual features do not correlate with labels, and complex combinations of them need to be considered.
E.g. while for data such as the MNIST handwritten-digit classification dataset, fitting a naive logistic regression model yields a classifier with a test accuracy of 92%, while attempting to do so for the real-world image dataset CIFAR10 results in a test accuracy of only 33.5%. Similarly, fitting a Gaussian kernel regression model on MNIST yields test accuracy, while doing the same for CIFAR10 results only in a test accuracy of 54%. Both these models predict a class label by relying on direct weighting of features, which in this case were simply the low-level pixel

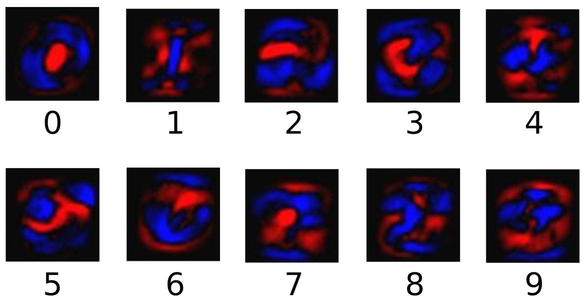

Figure 1.1: Weights learned by a fitting a logistic regression model on the MNIST dataset. Red represents negative weights, while blue represents positive weights. The Logistic Regression model learns correlation between individual features and labels.

intensities.

Model \ Dataset	MNIST	CIFAR10
Logistic Regression	92%	33.5%
Gaussian Kernel Regression	98.7%	54%

Table 1.1: Test accuracy for some common models using naïve features

Arguably, natural cognitive agents in the wild also do not represent image or sound observations merely as arrays of floating point numbers.

As a private case, in the task of cluster analysis, clustering algorithms search for clusters with regard to some similarity notion. The meaningfulness of these algorithms' outputs resides on whether the underlying similarity notion was meaningful in the first place.

A common approach to represent a similarity notion, which we also take in this work, is to

————————¿ HERE What metric would be applicable for real world images? In real-world image semantics, it is crucial to have a meaningful, high-level similarity notion, since E.g. applying standard metrics (such as the Euclidean metric, Minkowski metrics, etc.) will fail to capture semantic similarity on real-world data.

To formalize ”meaningful”, an ill-defined term, we focus in this work on transfer learning setups, where we attempt to learn a similarity notion under which clustering patterns conform best to a ground-truth desired clustering. The learned similarity notion will then be tested over data from different distibutions, which share the same underlying semantic similarity.

More formaly, we represent a clustering for n data points $\{x_i\}_{i=1}^{n}$ from a data domain $\mathcal{D}$ as a boolean matrix $Y \in \{0, 1\}^{n,n}$, where the ij'th entry is the indicator for whether x_i and x_j belong to the same cluster.

E.g most image understanding tasks are usually tackled today by first feeding the image to a neural network which ex For example in classification, if data can be represented as linearily seperated vector sets, then

E.g image classification was trivial if the class identity Another example is in machine learning applications for games such as Chess or Go, where the learning system doesn’t lea

This is most relevant in domains where discrepency between low-level and high-level description is stark. Computer vision is an example . low-level details need to be integrated

In the task of cluster analysis for example, (it is done w.r.t representation)

There are many scenarios in which one needs to learn a representation for it’s input We focus on the case of clustering natural to represent data as points in a metric space we want to learn an embedding Learning a joint embedding (deepsets)

Chapter 2

End-To-End Representation Learning

2.1 Domain Adaptation and Zero-Shot Learning

Since the train and test data in our experiments come from different distributions, our experimental setup fits under the term of Domain Adaptation [?]. This term describes the setting where one trains a learner to do a certain task (e.g classification) by labeled examples from a source distribution, and then applies the learned model on inputs from a different distribution. One reason that interest in such a setting exists is that in practice, when applying learned models in the real world, the input distribution may differ from the original dataset used for training, and may include an unlimited number of novel signals and scenarios. Thus for a learner to have a certain degree of robustness to such changes is of practical importance.

In the specific case of classification, if the target distribution is made of classes which were not seen at the train set, the problem is called Zero-Shot Learning (ZSL). Our experimental setting falls under this term as well, since although we don't annotate classes with specific distinctive labels, the labels that do exist (i.e the clustering matrices) come from an underlying distinctive classification of data points, and indeed the classes which underlie the partitions in the train set are

mostly different than those in the test set (to the extent that in the first two datasets we experiment on, they are completely disjoint). Inductive transfer problems in general, and ZSL problems in particular, are challenging since a successful solution for such a problem cannot simply rely on memorizing a specific, fixed signal from the source distribution (since it may not appear later at the target distribution). Instead, a successful solution would have learned a representation for it's input, such that the novel signals encountered at test time are still represented "meaningfully", i.e in a way which is supportive for a correct decision. Let us illustrate this with an example.
For an example, suppose that at train time the learner is given images of birds from two classes, and it needs to map them to a metric space such that a K-means based clusterer will cluster them correctly (i.e, as in our setting). Suppose now that these classes only differ by that birds of class A have black wings, and birds of class B have white wings. Obviously, if we removed background pixels, a single neuron or "feature detector" would suffice for this task. Yet if

Lampert 09 does something similar, by transfering hand-crafted attributes.

Chapter 3

Related Work

3.1 Structured Prediction with Constraints

Many real world tasks involve outputting multiple interdependent variables. *Structured prediction* aims at learning models for such problems. Due to the exponential size of the output space of such problems, it is often the case that some structures are more probable than others. Specifically, there are cases where some of the outputs are not valid. Such restrictions are considered *hard constraints*. We can also consider *soft constraints*, for cases in which a family of structures is rare. As discussed in Chapter **??**, a first approach for leveraging these constraints, is considering these constraints at inference time. For hard constraints, a reasonable approach would be outputting only valid structures, namely presenting inference as a constrained combinatorial optimization problem. This is done successfully for dependency parsing, where structures are enforced to be directed trees (e.g. in [MPRH05]). Soft constraints can be tackled in a similar constrained fashion, e.g. in projective dependency parsing [ZCL16] (where not necessarily all trees are indeed projective). Instead of constrained optimization, inference can be seen as a tradeoff between local potentials and global ones (corresponding to the structural constraints). This is common for cardinality constraints [GDS07, TSZ+12].

CCM [CRRR08] is a generic framework for injecting structural knowledge at inference time. CoDL [CRR07] extends CCMs by incorporating the constraints in the learning process with a bootstrapping-like process, such that unlabeled data is given labels by the (soft or hard) constrained inference.

The method which is most related to ours is Posterior Regularization [GGT10]. The idea of PR is to regularize the posterior distributions so that they are consistent with some prior knowledge. Model parameters θ and auxiliary label distributions $\mathbf{q}$ are learned jointly using EM approach, to both fit the labeled data and minimize the KL divergence between $\mathbf{q}$ and $p_\theta(\mathbf{y}|\mathbf{x})$. Each step in their learning algorithm then involves solving an inner optimization problem[1]. Note that in our method, such auxiliary distributions are not necessary.

High-Order Regularization [LZ14] is a max-margin framework for constraints injection. They introduce a regularizer function R. However, contrary to our approach, it is applied on discrete structures. Their learning process then involves both finding structures with low R value and minimizing the difference between the score of the maximal-scored structures to theirs. This is done by alternating methods.

Note that all above methods require solving combinatorial optimization problems (for finding structures or label distributions), whereas our method directly minimizes a continuous function over the model's scores.

Recently, [RSS$^+$] proposed extracting automatically structural constraints from labeled data using an adversary, learning to discriminate true labels from predicted ones.

3.2 Semi-Supervised Methods for Parsing

We now give a brief overview of semi-supervised methods designed specifically for parsing tasks.

The most familiar algorithm for semi-supervised learning (SSL) is *self-training*.

[1]over either θ or $\mathbf{q}$

The key idea is to start with a small seed of labeled examples, and augment it with new predicted ones. Once the dataset is augmented, a new model is learned from scratch on the concatenated dataset. This direction was exhaustively researched for SSL of dependency parsing. [MCJ06a, MCJ06b] demonstrated that self-training can boost (constituent) parsing performance for in-domain and out-of-domain scenarios, when a reranker is incorporated. [YEB15, YB15] are able to improve dependency parsers for domain adaptation and multilingual scenarios through confidence-based methods.

Co-training [BM98] is another way to train models using both labeled and unlabeled data. It requires multiple learners, each with a different view of the data. When one learner is confident of a prediction, other learners' training data is augmented with the prediction. [SR10] offered a similar method, stacking 3 dependency parsers and augmenting training data when two of them agree on a prediction for an unlabeled example.

Until recently, dependency parsers were often linear models over hand-crafted features. Engineering these features drew a lot of research attention. Some works considered SSL of such models by incorporating features based on unlabeled data. [KCC08] offered features that are based on word clusters. [KG15] suggested using lexical statistics derived from unlabeled data, parsed by a different baseline model. Additional features were presented by [SICC09, CZZ13]. Note that these methods have become less relevant due to deep learning methods, which learn features of the data automatically[2].

Active Learning was also considered for parsing [LZZ$^+$16]. At each iteration, they select a few most uncertain words and manually annotate their heads. These partial annotations are then added to the training data. The intuition behind is clear: adding quality data where the parser has most difficulty.

[2]Although one may consider combining such features with a deep model

3.3 Structural Properties of Dependency Structures

Structural properties of dependency structures were the focus of many works. The main idea is to design parsers which are on one hand expressive enough, but not too expressive on the other[3].

As discussed previously, *single-head* and *acyclicity* are well-known and considered hard constraints in most scenarios.

Projectivity[4] is also among the most studied restrictions on parse trees. However, frequency of non-projective structures varies between languages, and even between different domains of the same language. Thus, several relaxations to projectivity were examined. These include *planarity*, *well-nestedness*, and *gap-degree*, which are formalized and analyzed in [GRN10, GRWC09, Hav07a, Hav07b, Kuh10].

Many parsers are constrained to output projective trees, both in transition-based [Niv03, Cov01] and graph-based [ZCL16, KG16] parsers. Non-projective parsing is also well-understood for both approaches [Niv09, MPRH05]. Relaxations of projectivity were addressed mainly in TB parsers [NN05, Niv06, GRN10, GRWC09].

[3]In the sense that a structure can be obtained by this model if and only if it is ”valid”

[4]See formal definition in Section **??**

Chapter 4

Deep K-Means++ Learning

4.1 Our Algorithm

We now explain in detail how our algorithm works. As discussed above, our goal is to learn a mapping $F : \mathcal{D} \rightarrow \mathbb{R}^d$ from the data domain $\mathcal{D}$ to an embedding space R^d, for some representation dimensionality d, such that semantically similar objects will be mapped to geometrically closer points than dissimilar objects. This will result in ”naive” geometric algorithms (such as regular K-Means) to be able to cluster data correctly w.r.t a high-level similarity notion of interest. For this goal we compose a differentiable clustering module on top of a parametrized embedding module (which in our experiments we take to be a deep convolutional neural network). What’s left to do then is to learn parameters for the embedding module, such that as a function, it will achieve the goal stated above. These parameters start from some initial value, and are iteratively updated in the following way:

1. Sample labeled data from various different categories.
2. Embed the data with current embedding parameters.
3. Give the embedded data to the clustering module, outputing a soft partition matrix.

4. Use the Frobenius distance between the output and the ground-truth partition matrix as a loss function.

5. Calculate gradient of loss w.r.t embedding parameters.

6. Move embedding parameters in the direction opposite to this gradient.

Let us write this more formally, in the following pseudo-code:

Algorithm 1 Deep K-Means++ Learning

```
1: Input: Set of examples $\{\mathcal{X}_i\}_{i=1}^{M}$ from M different categories, Initial embedding parameters $\Theta_0$, Number of training steps $T$, Number $k$ of categories per train step, Size $n$ of batch per training step, Learning rates $\{\mu_i\}_{i=1}^{T}$.
2: for i = 0 to T-1 do
3:     Sample cluster indices $\{1', .., k'\}$
4:     for j = 0 to k-1 do
5:         $X_j \leftarrow$ Sample $\frac{n}{k}$ elements from $\mathcal{X}_{j'}$
6:     $Y \leftarrow \begin{bmatrix} 1_{\frac{n}{k},\frac{n}{k}} & 0 & \dots & 0 \\ 0 & 1_{\frac{n}{k},\frac{n}{k}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1_{\frac{n}{k},\frac{n}{k}} \end{bmatrix}_{n,n}$    ▷ ground-truth clustering matrix
7:     $X \leftarrow [X_1, X_2, .., X_k]$
8:     $\widetilde{X} \leftarrow$ Embed($X, \Theta_i$)    ▷ data embeddings
9:     $\widetilde{Y} \leftarrow$ Deep K-Means++($\widetilde{X}$,k)    ▷ inferred clustering matrix
10:    $loss \leftarrow \left\| Y - \widetilde{Y} \right\|^2$    ▷ Frobenius norm of matrix differences
11:    $\Theta_{i+1} \leftarrow \Theta_i - \mu_i \frac{\partial loss}{\partial \Theta_i}$    ▷ gradient descent
12: return $\Theta_T$
```

At lines 3-5, we sample a sub-problem, by sampling a small number of clusters and data points therefrom. The ground-truth "label" for this sub-problem is simply the block diagonal matrix shown in the pseudo-code. What's left to describe is the operation of the functions called at lines 8 and 9. Line 8 includes a call to the embedding module $Embed$, which should generally be a domain-dependent function. In our empirical evaluations over images, we take $Embed$ to be a deep convolutional neural network. In our synthetic experiments, we take it to be a simple linear mapping. Line 9 includes the call to the clustering module. As discussed above, this module is a differentiable formulation of the K-Means++ algorithm. Section 4.2 in the current chapter describes this module in detail.

Sub-Problem Gradients

Our training regime works by sampling ”clustering sub-problems”. We sample uniformly $\frac{n}{k}$ data points from k uniformly sampled clusters, and define the loss to be the Frobenius distance between the ground-truth partition matrix and the our clustering module’s output matrix. Performing gradient descent w.r.t a loss defined over a random sample of data is a common practice in contemporary large-scale settings, where the data simply cannot fit altogether inside a single GPU/RAM. In most supervised learning scenarios though, there is a supervision signal associated with each data point individually, and the loss function over the data $L(X, Y; \Theta)$ is defined as an average of pointwise losses $\frac{1}{n}\sum_{i\in[n]} L(x_i, y_i; \Theta)$. In such scenarios, if the ”sub-problems” are sampled uniformly, the training regime is known as Stochastic Gradient Descent, and the gradient of the loss over the sub-problems is, as a random variable, an unbiased estimator for the gradient of the empirical risk in general. In the case of clustering, it’s not immediate to see how an ”empirical clustering risk” should be defined. Let us say we define it as how well was the entire data set jointly clustered. If using the Frobenius metric, this can formalized as:

$$L_A(X, Y; \Theta) = \mathbb{E}_A\left[||A(X; \Theta) - Y||^2\right] \tag{4.1}$$

Where A is the clustering algorithm, and the expectation is taken w.r.t it’s randomness. The output of A is an nxn (possibly soft) partition matrix. Since it is meaningless to cluster a single point, this empirical loss doesn’t decompose to an average pointwise losses. Still, one might ask if the gradients for clustering sub-problems forms an unbiased estimator for the gradient of the empirical risk in general. Our simulations suggest that at least for the Frobenius norm, this is not necessarily the case.

4.2 Differentiable K-Means Clustering

4.2.1 The K-Means Objective

The clustering module searches iteratively for (locally) optimal centroids and beliefs $C \in \mathbb{R}^{k,d}, B \in \mathbb{R}^{n,k}$ w.r.t the relaxed K-Means objective:

$$\begin{aligned} &\underset{C\in\mathbb{R}^{k,d}, B\in\mathbb{R}^{n,k}}{\text{minimize}} && ||C - B\widetilde{X}||^2 \\ &\text{subject to:} && \forall i \in \{1, ..., n\} : B_i \in \Delta_k \text{ (i'th row of B is a distribution)} \end{aligned}$$

Where $\widetilde{X}$ is the embedded data matrix. Intuitively, an optimum for this objective seeks to find a clustering such that data points are close to their cluster's mean. In the integral formulation of this optimization problem, i.e when $B \in \{M \in \{0,1\}^{n,k} | M\vec{1}_k = \vec{1}_n\}$, the optimum corresponds to a clustering which minimizes intra-cluster sum of squares $\sum_{i=1}^{k} \sum_{x\in S_i} ||x - \mu_i||^2$, where S_i is the i'th cluster, and μ_i is the i'th cluster's mean. It is well-known that optimizing the integral version is NP-hard. In fact, it is NP-hard even for $k = 2$ clusters (when dimensionality d is taken as a parameter), and even so when $d = 2$, for a general number of clusters. Using a relaxed formulation is a common approach to derive an approximation algorithm. In chapter 3 we mention several of these approaches in the context of learning a metric for data. Another common approach is to use what is known as Loyd's algorithm (also known as Voronoi iterations), and to "bootstrap" B and C, starting from some initialization. It is much simpler to describe these iterations when switching from the matrix variables B, and using instead the variable $\{S_i\}_{i=1}^{k}$, where the set S_i is the i'th cluster. This switch is possible only since we are dealing with the integral formulation, and the rows of the B matrix in this case are all k-dimensional one-hot vectors, which serve as indicators. I.e, the j'th row indicates to which cluster is the j'th datapoint assigned to. We use C_i to denote the centroid of the i'th. Loyd's algorithm's update steps are thus the following:

$$\begin{aligned} S_i^{(t+1)} &= \{X_p : ||X_p - C_i^{(t)}||^2 \leq ||X_p - C_j^{(t)}||^2 \quad \forall j \in [k]\} \\ C_i^{(t+1)} &= \frac{1}{|S_i^{(t+1)}|} \sum_{X_p \in S_i^{(t+1)}} X_p \end{aligned}$$

Each iteration updates one of the variables S, C in a way which is optimal w.r.t the other variable. Thus the objective is non-increasing as iterations progress. In simple terms, the cluster membership updates corresponds to assigning each point to the cluster whose centroid is nearest, and the centroid updates correspond to having each cluster's centroid to be it's vector mean. In Loyd's original algorithm these bootstraping continues until a fixed-point has been reached. Since each step decreases the objective, this fixed-point will be a local optimum. From NP-hardness of the problem, the initialization scheme (i.e choice of initial centroids) is crucial for performance. Arbitrary initializations can result in convergence to a local optimum arbitrarily worse than the global optimum. Despite this, there are initialization schemes under which there are proven bounds for approximation ratio, i.e how far the result on convergence will be from an optimal one. One of these initialization schemes is K-Means++'s initialization, which will be detailed later on this chapter. Let us first discuss our differentiable formulation of the above update steps.

4.2.2 Differentiable K-Means Iterations

As recalled, the original purpose of our algorithm was to learn parameters for an embedding function, s.t data embedded under it will yield good clustering performance, when clustered via algorithms which seek to minimize the K-Means objective. For this end we need the clustering inference to be a differentiable function, so that gradients could be back-propagated. Let us notice that in Loyd's algorithm, the only part in the update steps which isn't differentiable is the assignment of dat-

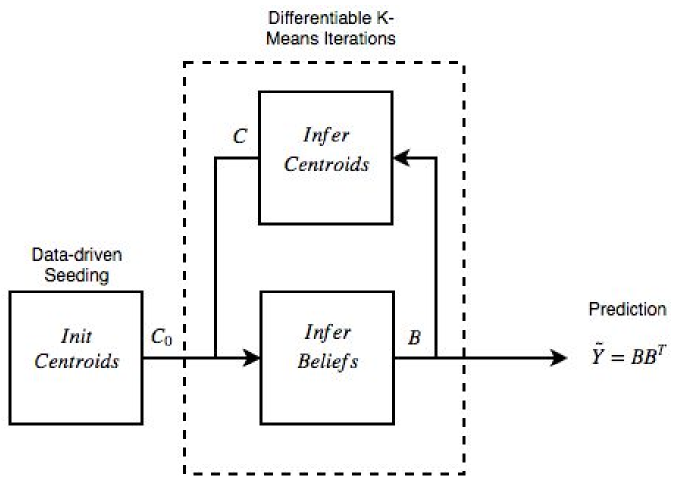

Figure 4.1: Block diagram for clustering module.

apoints to clusters, for which an $argmin$ needs to be computed. This operation is clearly not differentiable, and so instead we propose to use a $softmin$. This operation turns our assignments into soft-decisions, which we denote as beliefs, and represent via a row-stochastic belief matrix $B \in \mathbb{R}^{n,k}$ The equations for the relaxed Voronoi updates are, in matrix notation:

$$B_{t+1} = softmin(dist\_mat(X, C_t))$$

$$C_{t+1} = diag((B_{t+1}^T X)\vec{1})^{-1}(B_{t+1}^T X)$$

Where $dist_mat$ is defined as.. TODO ,and $diag$ is defined by taking a vector as input, and giving a square matrix whose main diagonal is that vector as output. For further clarity, let us view these updates in pseudo-code and describe the meaning of each line:

Algorithm 2 Deep K-Means++

```
1: Input: Embedded Data $\widetilde{X}$, Number of clusters $k$, Unroll depth $l$.
2: $C_0 \leftarrow$ Deep K-Means++ Init($\widetilde{X}$)    ▷ Data-driven centroid initialization
3: for i = 1 to l do
4:     $B_i \leftarrow$ Infer-Beliefs($\widetilde{X}, C_{i-1}, \beta$)
5:     $C_i \leftarrow$ Infer-Centroids($\widetilde{X}, B_{i-1}$)
6: return $B_l B_l^T$
```

Algorithm 3 Get Distance Matrix

```
1: Input: Two matrices $A \in \mathbb{R}_{n,d}, B \in \mathbb{R}_{k,d}$, with same row dimensionality
2: $D \leftarrow (A \odot A)1_{d,k} - 2AB^T + ((B \odot B)1_{d,k})^T$    ▷ Using lin. alg. identity:
   $||u - v||^2 = ||u||^2 - 2 < u, v > +||v||^2$
3: Return $D$ ▷ $ij$'th cell stores squared euclidean distance between $i$'th row of $A$
   and $j$'th row of $B$
```

Algorithm 4 Infer-Beliefs

```
1: Input: Data Matrix $X$, Centroid Matrix $C$
2: $dist\_mat \leftarrow get\_distance\_matrix(X, C)$
3: return $softmin(dist\_mat)$
```

Algorithm 5 Infer-Centroids

```
1: Input: Data Matrix $X$, Belief Matrix $B$
2: $cluster\_sums \leftarrow B^T X$
3: $cluster\_sizes \leftarrow cluster\_sums\dot{1}$
4: $normalizer \leftarrow diag(cluster\_sums)^{-1}$
5: return $normalizer \times cluster\_sums$
```

The clustering module's pseudo-code is described at algorithm 2, and is based on a differentiable formulation of the K-Means++ algorithm. As mentioned above,

This clustering module represents the clustering using a set of underlying cluster centroids, which induce the soft-assignment (which we will denote by the term beliefs) of data points to clusters. The belief vector for each data point is given by applying a softmin operation over the vector of euclidean distances between that data point and the existing centroids. We use a softmin instead of an actual argmin so that our cluster inference will be a differentiable function, and will thus yield to end-to-end gradient descent updates.

4.3 Differentiable Centroid Sampling

for the gradient signal to be helpful, we need the clustering inference to work properly. If the data is embedded correctly, i.e such the the global optimal cluster In order for the learning process not to be derailed by wrong clusterings (i.e, the gradient update will move the embedding parameters in a direction such that the labeled data will be better clustered under a bad local minima in the clustering output space)

Algorithm 6 Deep K-Means++ Init

```
1:  Input: Data matrix $X$, Number of desired clusters $k$.
2:  $C_0 \leftarrow X_0$    ▷ assign first centroid to be arbitrary data point
3:  for i = 1 to k-1 do
4:      $C \leftarrow [C_0, .., C_{i-1}]$
5:      $D \leftarrow get\_dist\_mat(X, C)$   ▷ $ij$'th cell stores squared euclidean distance between $i$'th data point and $j$'th centroid
6:      $\widetilde{D} \leftarrow softmin(D) \odot D$   ▷ apply softmin over every row, and multiply element-wise with $D$
7:      $\pi \leftarrow \frac{\widetilde{D}\vec{1_i}}{\vec{1_n}\widetilde{D}\vec{1_i}}$   ▷ sum over rows and normalize. $\pi$ thus represents a discrete distribution
8:      $\vec{q} \leftarrow gumbel\_softmax\_sampling(\pi)$ ▷ relaxed one-hot vector indicating $i$th centroid picked
9:      $C_i \leftarrow \vec{q}X$
10: $C \leftarrow [C_0, .., C_{k-1}]$
11: return $C$
```

We now describe how we implement *k-means++*'s centroid initialization as a differentiable layer in a neural network.

As in *k-means++* our algorithm proposes centroids iteratively. At the first iteration, we simply propose an arbitrary data point. At the $t + 1$'th iteration, t centroids, denoted as $\{c_i\}_{i=1}^{t}$, have already been proposed. A matrix D of size $[n, t]$ can thus be built, such that $A_{i,j} = ||x_i - c_j||^2$. By taking a soft-min over D's rows and normalizing the result to 1, we have an approximation for the probability vector, whose i'th entry is

4.3.1 Gumbel-Softmax Sampling

Gumbel-Softmax Sampling is a differentiable version of sampling from a discrete probability distribution. This technique allows defining neural networks with layers that act as discrete samplers. TODO: add when this is usefull (gans,etc..) Notice that it is not possible to implement such layers in neural networks without any output smoothing, since sampling from a discrete distribution is inherently non-differentiable.
The sampling layer takes as input a vector of probabilities $\{p_i\}_{i=1}^{n}$, and outputs a distribution over possible

The method uses the following key fact from probability theory:

Theorem 4.3.1. (Gumbel Sampling) Let $p_1, ..., p_n \in \Delta_n$ be n numbers on the n-simplex, and denote by D a random variable over $[n]$, distributed by $P[D = i] = p_i$.
Let Z be a random variable over $\mathbb{R}$ with the following probability density function: $f(z) = e^{-z+e^{-z}}$.
Then, the random variable $argmax_{i\in[n]}\{Z \cdot p_i\}$ has the same distribution as D.

Proof. Let $\{x_i\}_{i=1}^{n}$ be n real numbers s.t $\forall i \in [n] : p_i = \frac{e_i^x}{\sum_{k'}}$ □

Algorithm 7 Gumbel-Softmax Sampling

```
1: **Input:** A discrete probability distribution $\pi = \{p_i\}_{i=1}^n$
2: sample $\vec{z} \in \mathbb{R}^n$, where each coordinate is sampled according to the Gumbel distribution: $p(z) = e^{-(z+e^{-z})}$
3: $\widetilde{\pi} \leftarrow \pi \odot \vec{z}$        ▷ multiplicatively perturbed values
4: Return $softmax(\widetilde{\pi})$
```

Chapter 5

Experiments

5.1 Experiments with synthetic data

5.1.1 Learning a linear projection

In this subsection we describe our experiments with learning the parameters of a simple embedding function, namely a linear projection $f : \mathcal{E}^n \to \mathcal{E}$, whose parameters should be learned.

The performance of an end-to-end clustering algorithm relies on many design choices and factors, and how they relate to the specific problem at hand.

In our experiments we examined the following factors:

- Clustering algorithm to be unrolled.
- Number of iterations which the clustering algorithm performs.
- Various clusterer-specific hyperparameters, such as the bandwidth for the EM-clusterer, and the learn-rate and decrease schedule for the Differentiable-K-Means.

We examined these and related factors on the following synthetic data sets:

I) Co-linear Gaussian Clusters

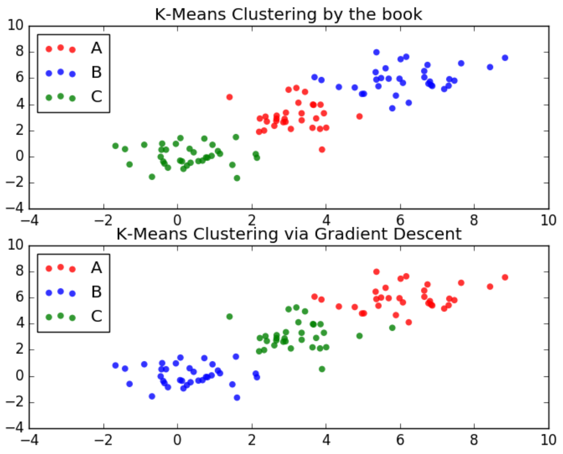

Figure 5.1: Example of a successful K-means clustering via Gradient Descent over the logits of cluster membership probabilities.

In this data set, the data is simply generated by a mixture of Gaussians, whose means are co-linear. The cluster membership for the data points correspond to the Gaussians which they are sampled from.

The Gaussians in this data set have isotopic covariance matrices, and since their means are co-linear, we know that the optimal linear projection parameters should be proportional to the parameters of the line which connects the Gaussians means. An example for $n = 3$ is portrayed at Figure x.

Anecdote: If you learn a linear projection for the ambiguous gaussians (placed at $10 * unit_square$) plus a scaling factor, then using the EM clusterer, the scaling factor is pushed down. If you shrink the data by 0.1, the scale factor is pushed up. It stops changing at about 0.2 times original data. why is that so? II) Ambiguous Gaussians

5.2 Experiments With Real-World Data

5.2.1 Datasets

5.2.2 Experimental Protocol

5.2.3 Implementation Details

5.2.4 Results

5.2.5 General Details

Following the experiments described at zemel17 song16, we test our approach on three known datasets:

- Caltech-UCSD Birds 200 [?]
- Stanford Cars Dataset
- Stanford Products Dataset

Each dataset consists of images which come from some shared context (i.e images of birds, cars or consumer products). The images in each dataset are further categorized into classes (E.g ”Laysan Albatross” for birds, ”2012 Tesla Model S” for cars, etc.) where images in the same class are considered semantically similar. We split the datasets to a train and test set according to the splits described in Zemel17,Song16. At train time, mini-batches of images are sampled, and a clustering matrix over the sampled images is predicted. The L_2 difference between the prediction and the ground-truth matrix is used cost function.
The code for our experiments was written in Tensorflow [?] version 1.8.0. A link to the publicly available git repository is listed below.[1]

[1]Link: https://github.com/DanielCarmon/msc_project

Depth of K-Means unrolling

In our experiments different unroll depths $T \in \{1, 3, 5, 10\}$ are tested. In the case where $T = 1$, we simply output BB^t as the prediction, where $B \in \mathbb{R}^n, k$ is the belief matrix for when the centroids are those initially sampled by the differentiable centroid sampling module described above. In the results section it is seen that deeper doesn't necessarily mean better, and that usually there is a "sweetspot", or optimal unroll depth T for learning through K-Means optimization.

Embedding Function

As an embedding function $f : D \to \mathbb{R}^d$, we use the Inception-v3 [**?**] architecture, with parameters pre-trained on the ImageNet[**?**] dataset. As in Zemel17, song16, we removed the softmax operation at the last layer of the network, and added a linear layer. The output dimentionality d of our linear layer is set to be the number of train classes for the first two datasets, and 512 for the Products dataset, where the number of train classes is larger than 10^4.

Training Details

At train time we use batches of $n = 100$ images per update step, with a ground-truth clustering matrix $Y \in R^{n,n}$. These images are sampled from $k \in \{5, 10, 20\}$ different classes, with each class contributing n/k images to the batch. The k class indices too are sampled at each iteration from a pool of classes whose images constitute the train data. Results for different values of k are shown below.

When training with differentiable clusterers that have intermediate beliefs (i.e when the unroll depth of the clusterer is greater than 1), we use "auxillary gradients" by adding the loss for intermediate beliefs to the loss function we optimize over. This trick appears in previous deep learning works as well. E.g in [**?**], where the original Inception-v3 network was trained on ImageNet, "auxillary classifier" layers are maintained at train time, and their loss is added to the total loss function of the network. At test time, these classifiers are discarded. Using these ”auxillary gradients” in our case favors embeddings who’s clustering under K-Means++ doesn’t just converges to a good local optima, but also does so quickly. Another way in which this trick benefits the training process is that it helps to alleviate the vanishing gradient problem[**?**], which occurs as unroll depth is increased.

Data pre-processing was done in the following way. Each image was first reshaped to shape $[299, 299, 3]$, in order to fit the Inception network’s fixed input size. Afterwards the pixel intensities of each image were linearly scaled to $[0, 1]$. For data augmentation, we used a single horizontal flip per image.

The following hyperparameters were used in our experiments. The bandwidth for the softmax at the differentiable clusterer’s E-step is $1e - 1$. We used the Adam [] optimizer with a base learn rate of $1e - 6$ to train the deep network’s parameters. We trained our models for $5e + 4$ steps, which on a standard work station with a 3GHz 16-Core PC with 125GB RAM and a GeForce GTX TITAN X GPU with 13GB RAM took about 1 day to complete.

Testing Details

At test time, the embedding function learned during training is used to embed the test data. The original (i.e non-differentiable) K-Means++ algorithm is then used to cluster the embedded test data, and the Normalized Mutual Information (NMI) [**?**] is then used as a difference measure between the K-Means++ clustering over the embedded data and the ground-truth clustering.

In our experiments we used *sklearn*’s [**?**] implementation for K-means++. We tested the K-Means++ clustering every 100 parameter updates, in order to see how the NMI score for the learned embedding changes across training. We present results for the train data and the test data. In the first two datasets, the classes in the train and test data are disjoint, although coming from a shared context (all being images of birds or cars). In the third dataset (Stanford Products), where an average class has about 5.3 images, some classes occur in both data splits and some don’t.

In all cases, the learner must be able to generalize from classes seen during training to previously unseen classes.

5.2.6 CUB Dataset

The CUB Dataset is comprised of 200 classes of different birds. Each class contains about 60 instances belonging to a single species. We sp lit the data to have the first 100 classes (5,864 images) for training, and the last 100 classes (5,924 images) for testing.

Training Details

...

Results

After every 100 parameter updates, we tested how well can the newly represented data be clustered, both for the train data and the test."Goodness of clustering" was measured via normalized mutual information. The following table shows both for train and test data how close with regard to ground truth were the new representations clustered.

TODO: Add image that shows this. When done doing design choices elimina-

Figure 5.2: placeholder for image.

tion, train the best. The following table summarizes the NMI score of previous benchmarks with addition to ours:

Method	NMI
Triplet s.h. (Schroff et al., 2015)	55.38
Lifted struct. (Song et al., 2016)	56.50
Npairs (Sohn, 2016)	57.24
Facility loc.(Song et al., 2017)	59.23
Spectral clust. (Zemel et al., 2017)	59.16
Ours	60+

Table 5.1: NMI evaluation on the Birds (CUB-200- 2011) dataset.

5.2.7 Stanford Cars Dataset

Description of dataset goes here

Training Details

Results

5.2.8 Ablation Studies

We perform the following ablation studies on the Caltech-UCSD Birds-200-2011 dataset. We measure the NMI score of different variations in the architecture we proposed above. The variations we examined are difference in depth of the clustering inference module (i.e, how many belief update iterations occurred) and difference in number of clusters per training batch (i.e, how many bird categories where seen during each training iteration). There are existing trade-offs for each of these design choices. A deeper clustering inference module will give more accurate results, but may dilute the gradient signal such that learning is dampened. For the number of clusters per batch, a high number of seen categories in each batch may help reach a representation which will generalize (see discussion on ZSL in 2) better, yet on the other hand a too high number of categories will emphasize intra-cluster differences, while at the extreme end (when $k = n$) the supervision signal is degenerated to simply be the identity matrix. For both these design choice dimensions it seems from the following ablation studies that there is a ”sweet-spot” somewhere between the extreme values. The variation which reached the highest NMI score over the test data isn’t the model that was suggested by evaluating the different variation’s clustering over the first 100 categories, and thus we did not report it’s score (0.62 NMI) when comparing our approach to other works.

Chapter 6

Discussion

Generally in structured prediction, and specifically in dependency grammars, obtaining annotated data requires expensive human expertise. However, unlabeled data is widely available on the web. Therefore, semi-supervised methods are of high interest in these fields.

In this work, we presented a general semi-supervised approach for injecting well-known structural properties of dependency parsing, *acyclicity* and *projectivity*. This is done by differentiable loss function, punishing for violation of these constraints on unlabeled data. To the best of our knowledge, this is the first work that directly considers the model's continuous scores, rather than discrete outputs. Our method is well-suited for any first-order graph-based parser, and is independent of the underlying scoring mechanism.

We were able to show improvement over a supervised baseline both in prediction performance and structural generalization[1]. In addition, we analyzed a greedy approach for decoding. We also demonstrated how this procedure benefits from the improvement in structural generalization.

We leave two natural extensions of this method for future work. The first considers designing strong discriminators to catch (perhaps unknown) phenomenas occurring in dependency trees. Architectures of such discriminators may vary and

[1]Namely, increase of valid structures during test time

have not been addressed yet. Note that discriminators can also be combined with the inference procedure, by outputting structures with high likelihood[2].

Extending our method to the discrete case is also of interest, since it is not always possible to come up with efficiently computable differentiable loss functions correlated to high-order constraints.

[2]In other words, structures with high scores from the discriminator

Bibliography

[BM98] Avrim Blum and Tom Mitchell. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM.

[Cov01] Michael A. Covington. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, pages 95–102. Citeseer, 2001.

[CRR07] Ming-Wei Chang, Lev Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 280–287, 2007.

[CRRR08] Ming-Wei Chang, Lev-Arie Ratinov, Nicholas Rizzolo, and Dan Roth. Learning and Inference with Constraints. In *AAAI*, pages 1513–1518, 2008.

[CZZ13] Wenliang Chen, Min Zhang, and Yue Zhang. Semi-supervised feature transformation for dependency parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1303–1313, 2013.

[GDS07] Rahul Gupta, Ajit A. Diwan, and Sunita Sarawagi. Efficient inference with cardinality-based clique potentials. In *In ICML*, pages 329–336, 2007.

[GGT10] Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049, 2010.

[GRN10] Carlos Gómez-Rodríguez and Joakim Nivre. A transition-based parser for 2-planar dependency structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1492–1501. Association for Computational Linguistics, 2010.

[GRWC09] Carlos Gómez-Rodríguez, David Weir, and John Carroll. Parsing mildly non-projective dependency structures. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 291–299. Association for Computational Linguistics, 2009.

[Hav07a] Jiri Havelka. Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 608–615, 2007.

[Hav07b] Jirí Havelka. Projectivity in Totally Ordered Rooted Trees: An Alternative Definition of Projectivity and Optimal Algorithms for Detecting Non-Projective Edges and Projectivizing Totally Ordered Rooted Trees. 2007.

[KCC08] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. *Proceedings of ACL-08: HLT*, pages 595–603, 2008.

[KG15] Eliyahu Kiperwasser and Yoav Goldberg. Semi-supervised dependency parsing using bilexical contextual features from auto-parsed data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1348–1353, 2015.

[KG16] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *arXiv preprint arXiv:1603.04351*, 2016.

[Kuh10] Marco Kuhlmann, editor. *Dependency Structures and Lexicalized Grammars: An Algebraic Approach*. Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin Heidelberg, 2010.

[LZ14] Yujia Li and Rich Zemel. High order regularization for semi-supervised learning of structured output problems. In *International Conference on Machine Learning*, pages 1368–1376, 2014.

[LZZ$^+$16] Zhenghua Li, Min Zhang, Yue Zhang, Zhanyi Liu, Wenliang Chen, Hua Wu, and Haifeng Wang. Active learning for dependency parsing with partial annotation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 344–354, 2016.

[MCJ06a] David McClosky, Eugene Charniak, and Mark Johnson. Effective Self-training for Parsing. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 152–159, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[MCJ06b] David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th an-*

nual meeting of the Association for Computational Linguistics, pages 337–344. Association for Computational Linguistics, 2006.

[MPRH05] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005.

[Niv03] Joakim Nivre. An Efficient Algorithm for Projective Dependency Parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*, pages 149–160, 2003.

[Niv06] Joakim Nivre. Constraints on non-projective dependency parsing. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006.

[Niv09] Joakim Nivre. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 351–359. Association for Computational Linguistics, 2009.

[NN05] Joakim Nivre and Jens Nilsson. Pseudo-projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 99–106, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[RSS$^+$] Hongyu Ren, Russell Stewart, Jiaming Song, Volodymyr Kuleshov, and Stefano Ermon. Structured Prediction with Adversarial Constraint Learning. *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.*

[SICC09] Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 551–560. Association for Computational Linguistics, 2009.

[SR10] Anders Søgaard and Christian Rishøj. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1065–1073. Association for Computational Linguistics, 2010.

[TSZ$^+$12] Daniel Tarlow, Kevin Swersky, Richard S. Zemel, Ryan P. Adams, and Brendan J. Frey. Fast Exact Inference for Recursive Cardinality Models. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, pages 825–834, Arlington, Virginia, United States, 2012. AUAI Press.

[YB15] Juntao Yu and Bernd Bohnet. Exploring Confidence-based Self-training for Multilingual Dependency Parsing in an Under-Resourced Language Scenario. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 350–358, 2015.

[YEB15] Juntao Yu, Mohab Elkaref, and Bernd Bohnet. Domain adaptation for dependency parsing via self-training. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 1–10, 2015.

[ZCL16] Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. Dependency Parsing as Head Selection. *arXiv:1606.01280 [cs]*, June 2016. arXiv: 1606.01280.

תקציר

חילוץ תלויות תחביריות (Dependency Parsing) הינה משימה חשובה בתחום עיבוד השפה הטבעית, המשמשת כאבן דרך ביישומי קצה רבים. המטרה המרכזית במשימה זו, הינה מציאת יחסים בין מילים במשפט נתון. יחסים אלה כפופים לאילוצים מבניים, אשר חלקם "קשים" (כלומר מתקיימים לכל משפט) וחלקם "רכים" (מתקיימים ברב המשפטים).

בעבודתנו, אנו מציגים מסגרת כללית לשימוש בידע מבני מוקדם עבור חילוץ תלויות, תוך שימוש במשפטים לא מתתויגים. הרעיון המרכזי הינו להעניש הפרת האילוצים במסגרת פונקציית המטרה. אנו מתרכזים בשני סוגי אילוצים מוכרים על מבנים אלו – עליהם להיות *חסרי מעגלים* ו-*פרוג'קטיביים*. הפונקציות שבעזרתן נבצע זאת, הינן בלתי-תלויות בתיוג הנכון למשפט, ולכן מתאימות למסגרת של למידה מונחית למחצה (Semi Supervised Learning). שיטה זו יכולה להשתלב בכל מערכת חילוץ מבוססת-גרפים מסדר ראשון (first-order graph-based parser) הנלמדת על ידי שיטות גרדיאנט.

אנו מראים אמפירית כי השיטה משפרת את ביצועי המערכת ביחס למערכת הבסיס, בייחוד בעת שימוש במספר קטן של דוגמאות מתויגות. בנוסף, אנו מדגימים כי תדירות המבנים ה"חוקיים" בזמן טסט עולה משמעותית. לסיכום, אנו משתמשים בתהליך היסק חמדן, ומראים שניתן להפחית את זמן הריצה שלו על ידי הגישה המוצגת.

אוניברסיטת תל אביב

הפקולטה למדעים מדויקים

על שם ריימונד ובברלי סאקלר

בית הספר למדעי המחשב

חילוץ תלויות תחביריות על ידי רגולריזציה מבנית

חיבור זה הוגש כחלק מהדרישות לקבלת התואר

"מוסמך אוניברסיטה" – M.Sc.

באוניברסיטת תל אביב

בית הספר למדעי המחשב

על ידי
אורי רם

עבודת המחקר לתזה זו נערכה

באוניברסיטת תל אביב בהנחייתו של

פרופסור אמיר גלוברזון

מרץ 2018