From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification

André F. T. Martins[†‡] ANDRE.MARTINS@UNBABEL.COM
Ramón F. Astudillo[†◇] RAMON@UNBABEL.COM
[†]Unbabel Lda, Rua Visconde de Santarém, 67-B, 1000-286 Lisboa, Portugal
[‡]Instituto de Telecomunicações (IT), Instituto Superior Técnico, Av. Rovisco Pais, 1, 1049-001 Lisboa, Portugal
[◇]Instituto de Engenharia de Sistemas e Computadores (INESC-ID), Rua Alves Redol, 9, 1000-029 Lisboa, Portugal

Abstract

We propose sparsemax, a new activation function similar to the traditional softmax, but able to output sparse probabilities. After deriving its properties, we show how its Jacobian can be efficiently computed, enabling its use in a network trained with backpropagation. Then, we propose a new smooth and convex loss function which is the sparsemax analogue of the logistic loss. We reveal an unexpected connection between this new loss and the Huber classification loss. We obtain promising empirical results in multi-label classification problems and in attention-based neural networks for natural language inference. For the latter, we achieve a similar performance as the traditional softmax, but with a selective, more compact, attention focus.

1. Introduction

The softmax transformation is a key component of several statistical learning models, encompassing multinomial logistic regression (McCullagh & Nelder, 1989), action selection in reinforcement learning (Sutton & Barto, 1998), and neural networks for multi-class classification (Bridle, 1990; Goodfellow et al., 2016). Recently, it has also been used to design attention mechanisms in neural networks, with important achievements in machine translation (Bahdanau et al., 2015), image caption generation (Xu et al., 2015), speech recognition (Chorowski et al., 2015), natural language understanding (Hermann et al., 2015; Sukhbaatar et al., 2015; Rocktäschel et al., 2015), and computation learning (Graves et al., 2014; Grefenstette et al., 2015).

There are a number of reasons why the softmax transformation is so appealing: it is simple to evaluate and differentiate, and it can be turned into a (concave) log-likelihood function by taking the logarithm of its output. Alternatives proposed in the literature, such as the Bradley-Terry model (Bradley & Terry, 1952; Zadrozny, 2001), the multinomial probit (Albert & Chib, 1993), the spherical softmax (Ollivier, 2013; de Brébisson & Vincent, 2015), or softmax approximations (Bouchard, 2007), while advantageous in certain scenarios, lack some of these convenient properties.

In this paper, we propose the **sparsemax transformation**. Sparsemax has the distinctive feature that it can return sparse posterior distributions, assigning zero probability to some output variables. This property makes it appealing for filtering large output spaces, predicting multiple labels, or as a component to identify which of a group of variables are relevant for a decision, making the model more interpretable. Crucially, this is done while preserving most of the attractive properties of softmax: we show that sparsemax is also simple to evaluate, it is even cheaper to differentiate, and that it can be turned into a convex loss function.

To sum up, our contributions are as follows:

- We formalize the new sparsemax transformation, derive its properties, and show how it can be efficiently computed (§2.1–2.3). We show that in the binary case sparsemax reduces to a hard sigmoid (§2.4).
- We derive the Jacobian of sparsemax, comparing it to the softmax case, and show that it can lead to faster gradient backpropagation (§2.5).
- We propose the **sparsemax loss**, a new loss function that is the sparsemax analogue of logistic regression (§3). We show that it is convex, everywhere differentiable, and can be regarded as a multi-class generalization of the Huber classification loss (Huber, 1964; Zhang, 2004).
- We apply the sparsemax loss to train multi-label linear classifiers (which predict a *set* of labels instead of a single label) on benchmark datasets (§4.1–4.2).

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

- Finally, we devise a neural selective attention mechanism using the sparsemax transformation, applying to a natural language inference problem (§4.3).

2. The Sparsemax Transformation

2.1. Definition

Let $\Delta^{K-1} := \{\boldsymbol{p} \in \mathbb{R}^K \mid \mathbf{1}^\top \boldsymbol{p} = 1,\ \boldsymbol{p} \geq \mathbf{0}\}$ be the $(K-1)$-dimensional simplex. We are interested in maps from $\mathbb{R}^K$ to Δ^{K-1}. Such maps are useful to convert a vector of real weights (*e.g.*, label scores) to a probability distribution (*e.g.* posterior probabilities of labels). The classical example is the **softmax**, defined componentwise as:

$$\mathrm{softmax}_i(\boldsymbol{z}) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}. \tag{1}$$

A limitation of the softmax transformation is that the resulting probability distribution always has full support, *i.e.*, $\mathrm{softmax}_i(\boldsymbol{z}) \neq 0$ for every $\boldsymbol{z}$ and i. This is a disadvantage in applications where a sparse probability distribution is desired, in which case it is common to define a threshold below which small probability values are truncated to zero.

In this paper, we propose as an alternative the following transformation, which we call **sparsemax**:

$$\mathrm{sparsemax}(\boldsymbol{z}) := \operatorname*{argmin}_{\boldsymbol{p} \in \Delta^{K-1}} \|\boldsymbol{p} - \boldsymbol{z}\|^2. \tag{2}$$

In words, sparsemax returns the Euclidean projection of the input vector $\boldsymbol{z}$ onto the probability simplex. This projection is likely to hit the boundary of the simplex, in which case $\mathrm{sparsemax}(\boldsymbol{z})$ becomes sparse. We will see that sparsemax retains most of the important properties of softmax, having in addition the ability of producing sparse distributions.

2.2. Closed-Form Solution

Many algorithms have been proposed to project onto the simplex (Michelot, 1986; Pardalos & Kovoor, 1990; Duchi et al., 2008). We start by recalling the result that such projections correspond to a soft-thresholding operation. Below, we denote $[K] := \{1, \ldots, K\}$ and $[t]_+ := \max\{0, t\}$.

Proposition 1 *The solution of Eq. 2 is of the form:*

$$\mathrm{sparsemax}_i(\boldsymbol{z}) = [z_i - \tau(\boldsymbol{z})]_+, \tag{3}$$

where $\tau : \mathbb{R}^K \to \mathbb{R}$ is the (unique) function that satisfies $\sum_j [z_j - \tau(\boldsymbol{z})]_+ = 1$ for every $\boldsymbol{z}$. Furthermore, τ can be expressed as follows. Let $z_{(1)} \geq z_{(2)} \geq \ldots \geq z_{(K)}$ be the sorted coordinates of $\boldsymbol{z}$, and define $k(\boldsymbol{z}) := \max\left\{k \in [K] \mid 1 + k z_{(k)} > \sum_{j \leq k} z_{(j)}\right\}$. Then,

$$\tau(\boldsymbol{z}) = \frac{\left(\sum_{j \leq k(\boldsymbol{z})} z_{(j)}\right) - 1}{k(\boldsymbol{z})} = \frac{\left(\sum_{j \in S(\boldsymbol{z})} z_j\right) - 1}{|S(\boldsymbol{z})|}, \tag{4}$$

Algorithm 1 Sparsemax Evaluation

Input: $\boldsymbol{z}$
Sort $\boldsymbol{z}$ as $z_{(1)} \geq \ldots \geq z_{(K)}$
Find $k(\boldsymbol{z}) := \max\left\{k \in [K] \mid 1 + k z_{(k)} > \sum_{j \leq k} z_{(j)}\right\}$
Define $\tau(\boldsymbol{z}) = \frac{\left(\sum_{j \leq k(\boldsymbol{z})} z_{(j)}\right) - 1}{k(\boldsymbol{z})}$
Output: $\boldsymbol{p}$ s.t. $p_i = [z_i - \tau(\boldsymbol{z})]_+$.

where $S(\boldsymbol{z}) := \{j \in [K] \mid \mathrm{sparsemax}_j(\boldsymbol{z}) > 0\}$ is the support of $\mathrm{sparsemax}(\boldsymbol{z})$.

Proof: See App. A.1 in the supplemental material. ■

In essence, Prop. 1 states that all we need for evaluating the sparsemax is to compute the threshold $\tau(\boldsymbol{z})$; all coordinates above this threshold (the ones in the set $S(\boldsymbol{z})$) will be shifted by this amount, and the others will be truncated to zero. We call τ in Eq. 4 the **threshold function**: this piecewise linear function will play an important role in the sequel. Alg. 1 illustrates a naïve $O(K \log K)$ algorithm that uses Prop. 1 for evaluating the sparsemax. More elaborate $O(K)$ algorithms exist based on partitioning and median-finding (Blum et al., 1973; Pardalos & Kovoor, 1990).[1]

2.3. Basic Properties

We now highlight some properties that are common to softmax and sparsemax. Let $z_{(1)} := \max_k z_k$, and denote by $A(\boldsymbol{z}) := \{k \in [K] \mid z_k = z_{(1)}\}$ the set of maximal components of $\boldsymbol{z}$. We define the indicator vector $\mathbb{1}_{A(\boldsymbol{z})}$, whose kth component is 1 if $k \in A(\boldsymbol{z})$, and 0 otherwise. We further denote by $\gamma(\boldsymbol{z}) := z_{(1)} - \max_{k \notin A(\boldsymbol{z})} z_k$ the gap between the maximal components of $\boldsymbol{z}$ and the second largest. We let $\mathbf{0}$ and $\mathbf{1}$ be vectors of zeros and ones, respectively.

Proposition 2 *The following properties hold for $\rho \in \{\mathrm{softmax}, \mathrm{sparsemax}\}$.*

1. *$\rho(\mathbf{0}) = \mathbf{1}/K$ and $\lim_{\epsilon \to 0^+} \rho(\epsilon^{-1}\boldsymbol{z}) = \mathbb{1}_{A(\boldsymbol{z})}/|A(\boldsymbol{z})|$ (uniform distribution, and distribution peaked on the maximal components of $\boldsymbol{z}$, respectively). For sparsemax, the last equality holds for any $\epsilon \leq \gamma(\boldsymbol{z}) \cdot |A(\boldsymbol{z})|$.*
2. *$\rho(\boldsymbol{z}) = \rho(\boldsymbol{z} + c\mathbf{1})$, for any $c \in \mathbb{R}$ (*i.e.*, ρ is invariant to adding the same constant to each coordinate).*
3. *$\rho(\mathbf{P}\boldsymbol{z}) = \mathbf{P}\rho(\boldsymbol{z})$ for any permutation matrix $\mathbf{P}$ (*i.e.*, ρ commutes with permutations).*
4. *If $z_i \leq z_j$, then $0 \leq \rho_j(\boldsymbol{z}) - \rho_i(\boldsymbol{z}) \leq \eta(z_j - z_i)$, where $\eta = \frac{1}{2}$ for softmax, and $\eta = 1$ for sparsemax.*

Proof: See App. A.2 in the supplemental material. ■

[1] In practice, there are *observed* linear time algorithms which are often a better choice. See Condat (2014, Table 1), who provide an empirical comparison among several of these algorithms.

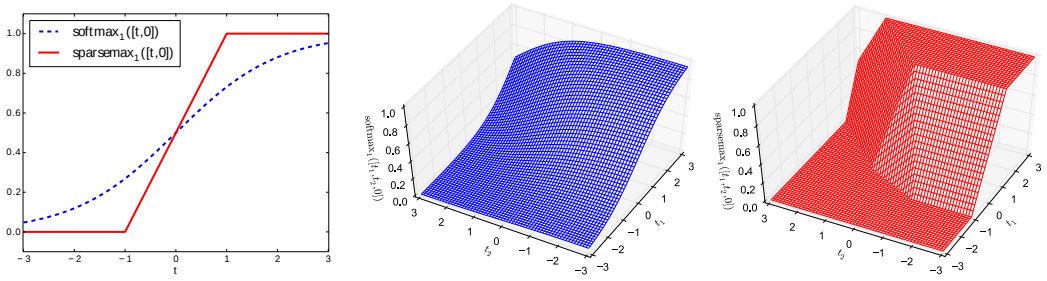

Figure 1. Comparison of softmax and sparsemax in 2D (left) and 3D (two righmost plots).

Interpreting ϵ as a "temperature parameter," the first clause of Prop. 2 shows that the sparsemax has the same "zero-temperature limit" behaviour as the softmax, but without the need of making the temperature arbitrarily small. We will use this fact in the experiments in §4.2 to control the degree of sparsity achieved with the sparsemax activation.

Prop. 2 is reassuring, since it shows that the sparsemax transformation, despite being defined very differently from the softmax, has a similar behaviour and preserves the same invariances. Note that some of these properties are not satisfied by other proposed replacements of the softmax: for example, the spherical softmax (Ollivier, 2013), defined as $\rho_i(\boldsymbol{z}) := z_i^2 / \sum_j z_j^2$, does not satisfy properties 2 and 4.

2.4. Two and Three-Dimensional Cases

For the two-class case, it is well known that the softmax activation becomes the logistic (sigmoid) function. More precisely, if $\boldsymbol{z} = (t, 0)$, then $\text{softmax}_1(\boldsymbol{z}) = \sigma(t) := (1 + \exp(-t))^{-1}$. We next show that the analogous in sparsemax is the "hard" version of the sigmoid. In fact, using Prop. 1, Eq. 4, we have that, for $\boldsymbol{z} = (t, 0)$,

$$\tau(\boldsymbol{z}) = \begin{cases} t - 1, & \text{if } t > 1 \\ (t-1)/2, & \text{if } -1 \leq t \leq 1 \\ -1, & \text{if } t < -1, \end{cases} \tag{5}$$

and therefore

$$\text{sparsemax}_1(\boldsymbol{z}) = \begin{cases} 1, & \text{if } t > 1 \\ (t+1)/2, & \text{if } -1 \leq t \leq 1 \\ 0, & \text{if } t < -1. \end{cases} \tag{6}$$

Fig. 1 provides an illustration for the two and three-dimensional cases. For the latter, we parameterize $\boldsymbol{z} = (t_1, t_2, 0)$ and plot $\text{softmax}_1(\boldsymbol{z})$ and $\text{sparsemax}_1(\boldsymbol{z})$ as a function of t_1 and t_2. We can see that sparsemax is piecewise linear, but asymptotically similar to the softmax.

2.5. Jacobian of Sparsemax

The Jacobian matrix of a transformation ρ, $J_\rho(\boldsymbol{z}) := [\partial \rho_i(\boldsymbol{z}) / \partial z_j]_{i,j}$, is of key importance to train models with gradient-based optimization. We next derive the Jacobian of the sparsemax activation, but before doing so, let us recall how the Jacobian of the softmax looks like. We have

$$\begin{aligned} \frac{\partial \text{softmax}_i(\boldsymbol{z})}{\partial z_j} &= (\delta_{ij} e^{z_i} \textstyle\sum_k e^{z_k} - e^{z_i} e^{z_j}) / (\sum_k e^{z_k})^2 \\ &= \text{softmax}_i(\boldsymbol{z})(\delta_{ij} - \text{softmax}_j(\boldsymbol{z})), \end{aligned} \tag{7}$$

where δ_{ij} is the Kronecker delta, which evaluates to 1 if $i = j$ and 0 otherwise. Letting $\boldsymbol{p} = \text{softmax}(\boldsymbol{z})$, the full Jacobian can be written in matrix notation as

$$\mathbf{J}_{\text{softmax}}(\boldsymbol{z}) = \mathbf{Diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top, \tag{8}$$

where $\mathbf{Diag}(\boldsymbol{p})$ is a matrix with $\boldsymbol{p}$ in the main diagonal.

Let us now turn to the sparsemax case. The sparsemax is differentiable everywhere except at splitting points $\boldsymbol{z}$ where the support set $S(\boldsymbol{z})$ changes, *i.e.*, where $S(\boldsymbol{z}) \neq S(\boldsymbol{z} + \epsilon \boldsymbol{d})$ for some $\boldsymbol{d}$ and infinitesimal ϵ.[2] From Eq. 3, we have that:

$$\frac{\partial \text{sparsemax}_i(\boldsymbol{z})}{\partial z_j} = \begin{cases} \delta_{ij} - \frac{\partial \tau(\boldsymbol{z})}{\partial z_j}, & \text{if } z_i > \tau(\boldsymbol{z}), \\ 0, & \text{if } z_i \leq \tau(\boldsymbol{z}). \end{cases} \tag{9}$$

From Eq. 4, the gradient of the threshold function τ is:

$$\frac{\partial \tau(\boldsymbol{z})}{\partial z_j} = \begin{cases} \frac{1}{|S(\boldsymbol{z})|}, & \text{if } j \in S(\boldsymbol{z}), \\ 0, & \text{if } j \notin S(\boldsymbol{z}). \end{cases} \tag{10}$$

Note that $j \in S(\boldsymbol{z}) \Leftrightarrow z_j > \tau(\boldsymbol{z})$. Therefore we obtain:

$$\frac{\partial \text{sparsemax}_i(\boldsymbol{z})}{\partial z_j} = \begin{cases} \delta_{ij} - \frac{1}{|S(\boldsymbol{z})|}, & \text{if } i, j \in S(\boldsymbol{z}), \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Let $\boldsymbol{s}$ be an indicator vector whose ith entry is 1 if $i \in S(\boldsymbol{z})$, and 0 otherwise. We can write the Jacobian matrix as

$$\mathbf{J}_{\text{sparsemax}}(\boldsymbol{z}) = \mathbf{Diag}(\boldsymbol{s}) - \boldsymbol{s}\boldsymbol{s}^\top / |S(\boldsymbol{z})|. \tag{12}$$

It is instructive to compare Eqs. 8 and 12. We may regard the Jacobian of sparsemax as the Laplacian of a graph whose elements of $S(\boldsymbol{z})$ are fully connected. To compute

[2] For those points, we can take an arbitrary matrix in the set of generalized Clarke's Jacobians (Clarke, 1983), the convex hull of all points of the form $\lim_{t \to \infty} \mathbf{J}_{\text{sparsemax}}(\boldsymbol{z}_t)$, where $\boldsymbol{z}_t \to \boldsymbol{z}$.

it, we only need $S(\boldsymbol{z})$, which can be obtained in $O(K)$ time with the same algorithm that evaluates the sparsemax.

Often, *e.g.*, in the gradient backpropagation algorithm, it is not necessary to compute the full Jacobian matrix, but only the product between the Jacobian and a given vector $\boldsymbol{v}$. In the softmax case, from Eq. 8, we have:

$$\mathbf{J}_{\text{softmax}}(\boldsymbol{z}) \cdot \boldsymbol{v} = \boldsymbol{p} \odot (\boldsymbol{v} - \bar{v}\mathbf{1}), \text{ with } \bar{v} := \textstyle\sum_j p_j v_j, \quad (13)$$

where $\odot$ denotes the Hadamard product; this requires a linear-time computation. For the sparsemax case, we have:

$$\mathbf{J}_{\text{sparsemax}}(\boldsymbol{z}) \cdot \boldsymbol{v} = \boldsymbol{s} \odot (\boldsymbol{v} - \hat{v}\mathbf{1}), \text{ with } \hat{v} := \frac{\sum_{j \in S(\boldsymbol{z})} v_j}{|S(\boldsymbol{z})|}. \quad (14)$$

Interestingly, if $\text{sparsemax}(\boldsymbol{z})$ has already been evaluated (*i.e.*, in the forward step), then so has $S(\boldsymbol{z})$, hence the nonzeros of $\mathbf{J}_{\text{sparsemax}}(\boldsymbol{z}) \cdot \boldsymbol{v}$ can be computed in only $O(|S(\boldsymbol{z})|)$ time, which can be sublinear. This can be an important advantage of sparsemax over softmax if K is large.

Computational efficiency. In sum, the computational needs of softmax and sparsemax are compared as follows:

- At training time, sparsemax can backpropagate gradients faster due to the sparsity (as stated above).
- At inference time, the softmax forward pass is faster, but asymptotically both are linear time. In our experiments (§4), even with the $O(K \log K)$ Algorithm 1, similar runtimes were achieved with softmax and sparsemax.

3. A Loss Function for Sparsemax

Now that we have defined and analyzed the sparsemax transformation, we will use it to design a new loss function that resembles the logistic loss, but can yield sparse posterior distributions. Later (in §4.1–4.2), we apply this loss to label proportion estimation and multi-label classification.

3.1. Logistic Loss

Let $\mathcal{D} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ be a dataset, where each $\boldsymbol{x}_i \in \mathbb{R}^D$ is an input vector and $y_i \in \{1, \dots, K\}$ is a target label. We consider empirical risk minimization problems of the form

$$\begin{aligned} \text{minimize} \quad & \frac{\lambda}{2}\|\mathbf{W}\|_F^2 + \frac{1}{N}\sum_{i=1}^N L(\mathbf{W}\boldsymbol{x}_i + \boldsymbol{b}; y_i), \\ \text{w.r.t.} \quad & \mathbf{W} \in \mathbb{R}^{K \times D},\ \boldsymbol{b} \in \mathbb{R}^K, \end{aligned} \quad (15)$$

where L is a loss function, $\mathbf{W}$ is a matrix of weights, and $\boldsymbol{b}$ is a bias vector. The loss function associated with the softmax is the logistic loss (or negative log-likelihood):

$$\begin{aligned} L_{\text{softmax}}(\boldsymbol{z}; k) &= -\log \text{softmax}_k(\boldsymbol{z}) \\ &= -z_k + \log \sum_j \exp(z_j), \end{aligned} \quad (16)$$

where $\boldsymbol{z} = \mathbf{W}\boldsymbol{x}_i + \boldsymbol{b}$, and $k = y_i$ is the "gold" label. The gradient of this loss is, invoking Eq. 7,

$$\nabla_{\boldsymbol{z}} L_{\text{softmax}}(\boldsymbol{z}; k) = -\boldsymbol{\delta}_k + \text{softmax}(\boldsymbol{z}), \quad (17)$$

where $\boldsymbol{\delta}_k$ denotes the delta distribution on k, $[\delta_k]_j = 1$ if $j = k$, and 0 otherwise. This is a well-known result; when plugged into a gradient-based optimizer, it leads to updates that move probability mass from the distribution predicted by the current model (*i.e.*, $\text{softmax}_k(\boldsymbol{z})$) to the gold label (via $\boldsymbol{\delta}_k$). Can we have something similar for sparsemax?

3.2. Sparsemax Loss

A nice aspect of the log-likelihood (Eq. 16) is that adding up loss terms for several examples, assumed i.i.d, we obtain the log-probability of the full training data. Unfortunately, this idea cannot be carried out to sparsemax: now, some labels may have *exactly* probability zero, so any model that assigns zero probability to a gold label would zero out the probability of the entire training sample. This is of course highly undesirable. One possible workaround is to define

$$L^{\epsilon}_{\text{sparsemax}}(\boldsymbol{z}; k) = -\log \frac{\epsilon + \text{sparsemax}_k(\boldsymbol{z})}{1 + K\epsilon}, \quad (18)$$

where ϵ is a small constant, and $\frac{\epsilon + \text{sparsemax}_k(\boldsymbol{z})}{1 + K\epsilon}$ is a "perturbed" sparsemax. However, this loss is non-convex, unlike the one in Eq. 16.

Another possibility, which we explore here, is to construct an alternative loss function whose *gradient* resembles the one in Eq. 17. Note that the gradient is particularly important, since it is directly involved in the model updates for typical optimization algorithms. Formally, we want $L_{\text{sparsemax}}$ to be a differentiable function such that

$$\nabla_{\boldsymbol{z}} L_{\text{sparsemax}}(\boldsymbol{z}; k) = -\boldsymbol{\delta}_k + \text{sparsemax}(\boldsymbol{z}). \quad (19)$$

We show below that this property is fulfilled by the following function, henceforth called the **sparsemax loss**:

$$L_{\text{sparsemax}}(\boldsymbol{z}; k) = -z_k + \frac{1}{2}\sum_{j \in S(\boldsymbol{z})} (z_j^2 - \tau^2(\boldsymbol{z})) + \frac{1}{2}, \quad (20)$$

where τ^2 is the square of the threshold function in Eq. 4. This loss, which has never been considered in the literature to the best of our knowledge, has a number of interesting properties, stated in the next proposition.

Proposition 3 *The following holds:*

1. $L_{\text{sparsemax}}$ *is differentiable everywhere, and its gradient is given by the expression in Eq. 19.*
2. $L_{\text{sparsemax}}$ *is convex.*
3. $L_{\text{sparsemax}}(\boldsymbol{z} + c\mathbf{1}; k) = L_{\text{sparsemax}}(\boldsymbol{z}; k)$, $\forall c \in \mathbb{R}$.

4. $L_{\text{sparsemax}}(\boldsymbol{z}; k) \geq 0$, *for all* $\boldsymbol{z}$ *and* k.
5. *The following statements are all equivalent: (i)* $L_{\text{sparsemax}}(\boldsymbol{z}; k) = 0$*; (ii)* $\text{sparsemax}(\boldsymbol{z}) = \boldsymbol{\delta}_k$*; (iii) margin separation holds,* $z_k \geq 1 + \max_{j \neq k} z_j$.

Proof: See App. A.3 in the supplemental material. ■

Note that the first four properties in Prop. 3 are also satisfied by the logistic loss, except that the gradient is given by Eq. 17. The fifth property is particularly interesting, since it is satisfied by the hinge loss of support vector machines. However, unlike the hinge loss, $L_{\text{sparsemax}}$ is everywhere differentiable, hence amenable to smooth optimization methods such as L-BFGS or accelerated gradient descent (Liu & Nocedal, 1989; Nesterov, 1983).

3.3. Relation to the Huber Loss

Coincidentally, as we next show, the sparsemax loss in the binary case reduces to the Huber classification loss, an important loss function in robust statistics (Huber, 1964).

Let us note first that, from Eq. 20, we have, if $|S(\boldsymbol{z})| = 1$,

$$L_{\text{sparsemax}}(\boldsymbol{z}; k) \quad = \quad -z_k + z_{(1)}, \tag{21}$$

and, if $|S(\boldsymbol{z})| = 2$, $L_{\text{sparsemax}}(\boldsymbol{z}; k) =$

$$-z_k + \frac{1 + (z_{(1)} - z_{(2)})^2}{4} + \frac{z_{(1)} + z_{(2)}}{2}, \tag{22}$$

where $z_{(1)} \geq z_{(2)} \geq \ldots$ are the sorted components of $\boldsymbol{z}$. Note that the second expression, when $z_{(1)} - z_{(2)} = 1$, equals the first one, which asserts the continuity of the loss even though $|S(\boldsymbol{z})|$ is non-continuous on $\boldsymbol{z}$.

In the two-class case, we have $|S(\boldsymbol{z})| = 1$ if $z_{(1)} \geq 1 + z_{(2)}$ (unit margin separation), and $|S(\boldsymbol{z})| = 2$ otherwise. Assume without loss of generality that the correct label is $k = 1$, and define $t = z_1 - z_2$. From Eqs. 21–22, we have

$$L_{\text{sparsemax}}(t) = \begin{cases} 0 & \text{if } t \geq 1 \\ -t & \text{if } t \leq -1 \\ \frac{(t-1)^2}{4} & \text{if } -1 < t < 1, \end{cases} \tag{23}$$

whose graph is shown in Fig. 2. This loss is a variant of the Huber loss adapted for classification, and has been called "modified Huber loss" by Zhang (2004); Zou et al. (2006).

3.4. Generalization to Multi-Label Classification

We end this section by showing a generalization of the loss functions in Eqs. 16 and 20 to multi-label classification, *i.e.*, problems in which the target is a non-empty set of labels $\mathcal{Y} \in 2^{[K]} \setminus \{\varnothing\}$ rather than a single label.[3] Such problems have attracted recent interest (Zhang & Zhou, 2014).

[3]Not to be confused with "multi-class classification," which denotes problems with a target variable $y \in [K]$ where $K > 2$.

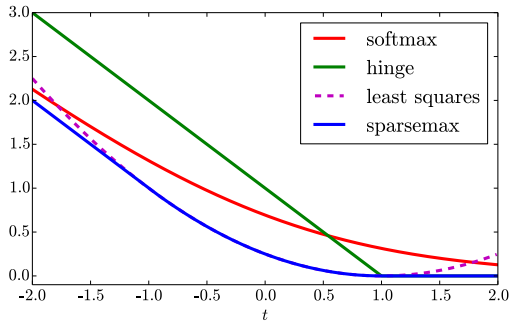

Figure 2. Comparison between the sparsemax loss and other commonly used losses for binary classification.

More generally, we consider the problem of estimating **sparse label proportions**, where the target is a probability distribution $\boldsymbol{q} \in \Delta^{K-1}$, such that $\mathcal{Y} = \{k \mid q_k > 0\}$. We assume a training dataset $\mathcal{D} := \{(\boldsymbol{x}_i, \boldsymbol{q}_i)\}_{i=1}^N$, where each $\boldsymbol{x}_i \in \mathbb{R}^D$ is an input vector and each $\boldsymbol{q}_i \in \Delta^{K-1}$ is a target distribution over outputs, assumed sparse.[4] This subsumes single-label classification, where all $\boldsymbol{q}_i$ are delta distributions concentrated on a single class. The generalization of the multinomial logistic loss to this setting is

$$\begin{aligned} L_{\text{softmax}}(\boldsymbol{z}; \boldsymbol{q}) \quad &= \quad \mathbf{KL}(\boldsymbol{q} \parallel \text{softmax}(\boldsymbol{z})) \\ &= \quad -\mathbf{H}(\boldsymbol{q}) - \boldsymbol{q}^\top \boldsymbol{z} + \log \textstyle\sum_j \exp(z_j), \end{aligned} \tag{24}$$

where $\mathbf{KL}(.\|.)$ and $\mathbf{H}(.)$ denote the Kullback-Leibler divergence and the Shannon entropy, respectively. Note that, up to a constant, this loss is equivalent to standard logistic regression with soft labels. The gradient of this loss is

$$\nabla_{\boldsymbol{z}} L_{\text{softmax}}(\boldsymbol{z}; \boldsymbol{q}) = -\boldsymbol{q} + \text{softmax}(\boldsymbol{z}). \tag{25}$$

The corresponding generalization in the sparsemax case is:

$$L_{\text{sparsemax}}(\boldsymbol{z}; \boldsymbol{q}) = -\boldsymbol{q}^\top \boldsymbol{z} + \frac{1}{2} \sum_{j \in S(\boldsymbol{z})} (z_j^2 - \tau^2(\boldsymbol{z})) + \frac{1}{2} \|\boldsymbol{q}\|^2, \tag{26}$$

which satisfies the properties in Prop. 3 and has gradient

$$\nabla_{\boldsymbol{z}} L_{\text{sparsemax}}(\boldsymbol{z}; \boldsymbol{q}) = -\boldsymbol{q} + \text{sparsemax}(\boldsymbol{z}). \tag{27}$$

We make use of these losses in our experiments (§4.1–4.2).

4. Experiments

We next evaluate empirically the ability of sparsemax for addressing two classes of problems:

1. Label proportion estimation and multi-label classification, via the sparsemax loss in Eq. 26 (§4.1–4.2).
2. Attention-based neural networks, via the sparsemax transformation of Eq. 2 (§4.3).

[4]This scenario is also relevant for "learning with a probabilistic teacher" (Agrawala, 1970) and semi-supervised learning (Chapelle et al., 2006), as it can model label uncertainty.

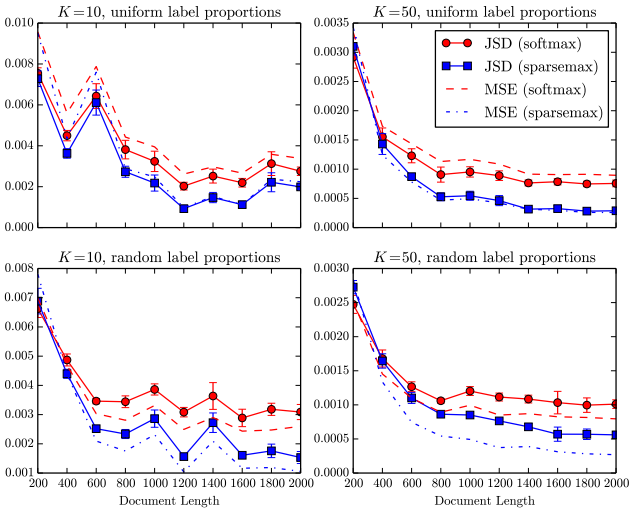

Figure 3. Simulation results for label proportion estimation with uniform (top) and random mixtures (bottom). Shown are the mean squared error and Jensen-Shannon divergence as a function of the document length, for the logistic and the sparsemax estimators.

4.1. Label Proportion Estimation

We show simulation results for sparse label proportion estimation on synthetic data. Since sparsemax can predict sparse distributions, we expect its superiority in this task. We generated datasets with 1,200 training and 1,000 test examples. Each example emulates a "multi-labeled document": a variable-length sequence of word symbols, assigned to multiple topics (labels). We pick the number of labels $N \in \{1, \dots, K\}$ by sampling from a Poisson distribution with rejection sampling, and draw the N labels from a multinomial. Then, we pick a document length from a Poisson, and repeatedly sample its words from the mixture of the N label-specific multinomials. We experimented with two settings: **uniform mixtures** ($q_{k_n} = 1/N$ for the N active labels $k_1, \dots, k_N$) and **random mixtures** (whose label proportions q_{k_n} were drawn from a flat Dirichlet).[5] We set the vocabulary size to be equal to the number of labels $K \in \{10, 50\}$, and varied the average document length between 200 and 2,000 words. We trained models by optimizing Eq. 15 with $L \in \{L_{\text{softmax}}, L_{\text{sparsemax}}\}$ (Eqs. 24 and 26). We picked the regularization constant $\lambda \in \{10^j\}_{j=-9}^{0}$ with 5-fold cross-validation.

Results are shown in Fig. 3. We report the mean squared error (average of $\|\boldsymbol{q} - \boldsymbol{p}\|^2$ on the test set, where $\boldsymbol{q}$ and $\boldsymbol{p}$ are respectively the target and predicted label posteriors) and the Jensen-Shannon divergence (average of $\mathbf{JS}(\boldsymbol{q}, \boldsymbol{p}) := \frac{1}{2}\mathbf{KL}(\boldsymbol{q}\|\frac{\boldsymbol{p}+\boldsymbol{q}}{2}) + \frac{1}{2}\mathbf{KL}(\boldsymbol{p}\|\frac{\boldsymbol{p}+\boldsymbol{q}}{2})$).[6] We observe that the two losses perform similarly for small document lengths (where the signal is weaker), but as the average document length exceeds 400, the sparsemax loss starts outperforming the logistic loss consistently. This is because with a stronger signal the sparsemax estimator manages to identify correctly the support of the label proportions $\boldsymbol{q}$, contributing to reduce both the mean squared error and the JS divergence. This occurs both for uniform and random mixtures.

[5] Note that, with uniform mixtures, the problem becomes essentially multi-label classification.

[6] Note that the KL divergence is not an appropriate metric here, since the sparsity of $\boldsymbol{q}$ and $\boldsymbol{p}$ could lead to $-\infty$ values.

Table 1. Statistics for the 5 multi-label classification datasets.

DATASET	DESCR.	#LABELS	#TRAIN	#TEST
SCENE	IMAGES	6	1211	1196
EMOTIONS	MUSIC	6	393	202
BIRDS	AUDIO	19	323	322
CAL500	MUSIC	174	400	100
REUTERS	TEXT	103	23,149	781,265

Table 2. Micro (left) and macro-averaged (right) F_1 scores for the logistic, softmax, and sparsemax losses on benchmark datasets.

DATASET	LOGISTIC	SOFTMAX	SPARSEMAX
SCENE	70.96 / 72.95	**74.01** / **75.03**	73.45 / 74.57
EMOTIONS	66.75 / **68.56**	**67.34** / 67.51	66.38 / 66.07
BIRDS	45.78 / 33.77	48.67 / 37.06	**49.44** / **39.13**
CAL500	**48.88** / 24.49	47.46 / 23.51	48.47 / **26.20**
REUTERS	**81.19** / 60.02	79.47 / 56.30	80.00 / **61.27**

4.2. Multi-Label Classification on Benchmark Datasets

Next, we ran experiments in five benchmark multi-label classification datasets: the four small-scale datasets used by Koyejo et al. (2015),[7] and the much larger Reuters RCV1 v2 dataset of Lewis et al. (2004).[8] For all datasets, we removed examples without labels (*i.e.* where $\mathcal{Y} = \varnothing$). For all but the Reuters dataset, we normalized the features to have zero mean and unit variance. Statistics for these datasets are presented in Table 1.

Recent work has investigated the consistency of multi-label classifiers for various micro and macro-averaged metrics (Gao & Zhou, 2013; Koyejo et al., 2015), among which a plug-in classifier that trains independent binary logistic regressors on each label, and then tunes a probability threshold $\delta \in [0, 1]$ on validation data. At test time, those labels whose posteriors are above the threshold are predicted to be "on." We used this procedure (called LOGISTIC) as a baseline for comparison. Our second baseline (SOFTMAX) is a multinomial logistic regressor, using the loss function

[7] Obtained from http://mulan.sourceforge.net/datasets-mlc.html.

[8] Obtained from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html.

in Eq. 24, where the target distribution $\boldsymbol{q}$ is set to uniform over the active labels. A similar probability threshold p_0 is used for prediction, above which a label is predicted to be "on." We compare these two systems with the sparsemax loss function of Eq. 26. We found it beneficial to scale the label scores $\boldsymbol{z}$ by a constant $t \geq 1$ at test time, before applying the sparsemax transformation, to make the resulting distribution $\boldsymbol{p} = \text{sparsemax}(t\boldsymbol{z})$ more sparse. We then predict the kth label to be "on" if $p_k \neq 0$.[9]

We optimized the three losses with L-BFGS (for a maximum of 100 epochs), tuning the hyperparameters in a held-out validation set (for the Reuters dataset) and with 5-fold cross-validation (for the other four datasets). The hyperparameters are the regularization constant $\lambda \in \{10^j\}_{j=-8}^{2}$, the probability thresholds $\delta \in \{.05 \times n\}_{n=1}^{10}$ for LOGISTIC and $p_0 \in \{n/K\}_{n=1}^{10}$ for SOFTMAX, and the coefficient $t \in \{0.5 \times n\}_{n=1}^{10}$ for SPARSEMAX.

Results are shown in Table 2. Overall, the performances of the three losses are all very similar, with a slight advantage of SPARSEMAX, which attained the highest results in 4 out of 10 experiments, while LOGISTIC and SOFTMAX won 3 times each. In particular, sparsemax appears better suited for problems with larger numbers of labels.

4.3. Neural Networks with Attention Mechanisms

We now assess the suitability of the sparsemax transformation to construct a "sparse" neural attention mechanism.

We ran experiments on the task of natural language inference, using the recently released SNLI 1.0 corpus (Bowman et al., 2015), a collection of 570,000 human-written English sentence pairs. Each pair consists of a premise and an hypothesis, manually labeled with one the labels ENTAILMENT, CONTRADICTION, or NEUTRAL. We used the provided training, development, and test splits.

The architecture of our system, shown in Fig. 4, is the same as the one proposed by Rocktäschel et al. (2015). We compare the performance of four systems: NOATTENTION, a (gated) RNN-based system similar to Bowman et al. (2015); LOGISTICATTENTION, an attention-based system with independent logistic activations; SOFTATTENTION, a near-reproduction of the Rocktäschel et al. (2015)'s attention-based system; and SPARSEATTENTION, which replaces the latter softmax-activated attention mechanism by a sparsemax activation.

We represent the words in the premise and in the hypothesis with 300-dimensional GloVe vectors (Pennington et al., 2014), not optimized during training, which we linearly project onto a D-dimensional subspace (Astudillo et al., 2015).[10] We denote by $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ and $\boldsymbol{x}_{L+1}, \ldots, \boldsymbol{x}_N$, respectively, the projected premise and hypothesis word vectors. These sequences are then fed into two recurrent networks (one for each). Instead of long short-term memories, as Rocktäschel et al. (2015), we used gated recurrent units (GRUs, Cho et al. 2014), which behave similarly but have fewer parameters. Our premise GRU generates a state sequence $\mathbf{H}_{1:L} := [\boldsymbol{h}_1 \ldots \boldsymbol{h}_L] \in \mathbb{R}^{D \times L}$ as follows:

$$\boldsymbol{z}_t = \sigma(\mathbf{W}^{xz}\boldsymbol{x}_t + \mathbf{W}^{hz}\boldsymbol{h}_{t-1} + \boldsymbol{b}^z) \quad (28)$$
$$\boldsymbol{r}_t = \sigma(\mathbf{W}^{xr}\boldsymbol{x}_t + \mathbf{W}^{hr}\boldsymbol{h}_{t-1} + \boldsymbol{b}^r) \quad (29)$$
$$\bar{\boldsymbol{h}}_t = \tanh(\mathbf{W}^{xh}\boldsymbol{x}_t + \mathbf{W}^{hh}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}^h) \quad (30)$$
$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t)\boldsymbol{h}_{t-1} + \boldsymbol{z}_t\bar{\boldsymbol{h}}_t, \quad (31)$$

with model parameters $\mathbf{W}^{\{xz,xr,xh,hz,hr,hh\}} \in \mathbb{R}^{D \times D}$ and $\boldsymbol{b}^{\{z,r,h\}} \in \mathbb{R}^D$. Likewise, our hypothesis GRU (with distinct parameters) generates a state sequence $[\boldsymbol{h}_{L+1}, \ldots, \boldsymbol{h}_N]$, being initialized with the last state from the premise ($\boldsymbol{h}_L$). The NOATTENTION system then computes the final state $\boldsymbol{u}$ based on the last states from the premise and the hypothesis as follows:

$$\boldsymbol{u} = \tanh(\mathbf{W}^{pu}\boldsymbol{h}_L + \mathbf{W}^{hu}\boldsymbol{h}_N + \boldsymbol{b}^u) \quad (32)$$

where $\mathbf{W}^{pu}, \mathbf{W}^{hu} \in \mathbb{R}^{D \times D}$ and $\boldsymbol{b}^u \in \mathbb{R}^D$. Finally, it predicts a label $\widehat{y}$ from $\boldsymbol{u}$ with a standard softmax layer. The SOFTATTENTION system, instead of using the last premise state $\boldsymbol{h}_L$, computes a weighted average of premise words

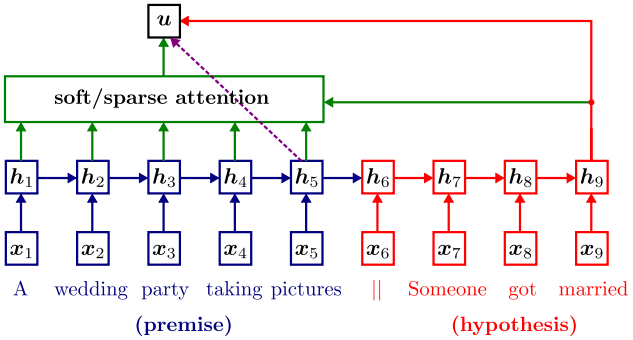

Figure 4. Network diagram for the NL inference problem. The premise and hypothesis are both fed into (gated) RNNs. The NOATTENTION system replaces the attention part (in green) by a direct connection from the last premise state to the output (dashed violet line). The LOGISTICATTENTION, SOFTATTENTION and SPARSEATTENTION systems have respectively independent logistics, a softmax, and a sparsemax-activated attention mechanism. In this example, $L = 5$ and $N = 9$.

[9] The ability of using this "inverse temperature" t to control the sparsity is enabled by Prop. 2, item 1. We may regard this as an hyperparameter analagous to the thresholds δ and p_0 used in the LOGISTIC and SOFTMAX baselines, respectively.

[10] We used GloVe-840B embeddings trained on Common Crawl (http://nlp.stanford.edu/projects/glove/).

Table 3. Accuracies for the natural language inference task. Shown are our implementations of a system without attention, and with logistic, soft, and sparse attentions.

	DEV ACC.	TEST ACC.
NOATTENTION	81.84	80.99
LOGISTICATTENTION	82.11	80.84
SOFTATTENTION	82.86	82.08
SPARSEATTENTION	82.52	**82.20**

Table 4. Examples of sparse attention for the natural language inference task. Nonzero attention coefficients are marked **in bold**. Our system classified all four examples correctly. The examples were picked from Rocktäschel et al. (2015).

A boy **rides on** a **camel** in a crowded area while talking on his cellphone. Hypothesis: *A boy is riding an animal.* [entailment]
A young girl wearing **a pink coat** plays with a **yellow** toy golf club. Hypothesis: *A girl is wearing a blue jacket.* [contradiction]
Two black dogs are **frolicking** around the **grass together**. Hypothesis: *Two dogs swim in the lake.* [contradiction]
A man wearing a yellow striped shirt **laughs** while **seated next** to another **man** who is wearing a light blue shirt and **clasping** his hands together. Hypothesis: *Two mimes sit in complete silence.* [contradiction]

with an attention mechanism, replacing Eq. 32 by

$$z_t = \boldsymbol{v}^\top \tanh(\mathbf{W}^{pm}\boldsymbol{h}_t + \mathbf{W}^{hm}\boldsymbol{h}_N + \boldsymbol{b}^m) \quad (33)$$

$$\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{z}), \text{ where } \boldsymbol{z} := (z_1, \ldots, z_L) \quad (34)$$

$$\boldsymbol{r} = \mathbf{H}_{1:L}\boldsymbol{p} \quad (35)$$

$$\boldsymbol{u} = \tanh(\mathbf{W}^{pu}\boldsymbol{r} + \mathbf{W}^{hu}\boldsymbol{h}_N + \boldsymbol{b}^u), \quad (36)$$

where $\mathbf{W}^{pm}, \mathbf{W}^{hm} \in \mathbb{R}^{D\times D}$ and $\boldsymbol{b}^m, \boldsymbol{v} \in \mathbb{R}^D$. The LOGISTICATTENTION system, instead of Eq. 34, computes $\boldsymbol{p} = (\sigma(z_1), \ldots, \sigma(z_L))$. Finally, the SPARSEATTENTION system replaces Eq. 34 by $\boldsymbol{p} = \mathrm{sparsemax}(\boldsymbol{z})$.

We optimized all the systems with Adam (Kingma & Ba, 2014), using the default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, and setting the learning rate to 3×10^{-4}. We tuned a ℓ_2-regularization coefficient in $\{0, 10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$ and, as Rocktäschel et al. (2015), a dropout probability of 0.1 in the inputs and outputs of the network.

The results are shown in Table 3. We observe that the soft and sparse-activated attention systems perform similarly, the latter being slightly more accurate on the test set, and that both outperform the NOATTENTION and LOGISTICATTENTION systems.[11]

Table 4 shows examples of sentence pairs, highlighting the premise words selected by the SPARSEATTENTION mechanism. We can see that, for all examples, only a small number of words are selected, which are key to making the final decision.[12] Compared to a softmax-activated mechanism, which provides a dense distribution over all the words, the sparsemax activation yields a compact and more interpretable selection, which can be particularly useful in long sentences such as the one in the bottom row.

5. Conclusions

We introduced the sparsemax transformation, which has similar properties to the traditional softmax, but is able to output sparse probability distributions. We derived a closed-form expression for its Jacobian, needed for the backpropagation algorithm, and we proposed a novel "sparsemax loss" function, a sparse analogue of the logistic loss, which is smooth and convex. Empirical results in multi-label classification and in attention networks for natural language inference attest the validity of our approach.

The connection between sparse modeling and interpretability is key in signal processing (Hastie et al., 2015). Our approach is distinctive: it is not the model that is assumed sparse, but the label posteriors that the model parametrizes. Sparsity is also a desirable (and biologically plausible) property in neural networks, present in rectified units (Glorot et al., 2011) and maxout nets (Goodfellow et al., 2013).

There are several avenues for future research. The ability of sparsemax-activated attention to select only a few variables to attend makes it potentially relevant to neural architectures with random access memory (Graves et al., 2014; Grefenstette et al., 2015; Sukhbaatar et al., 2015), since it offers a compromise between soft and hard operations, maintaining differentiability. Sparsemax is also appealing for hierarchical attention: if we define a top-down product of distributions along the hierarchy, the sparse distributions produced by sparsemax will automatically prune the hierarchy, leading to computational savings. A possible disadvantage of sparsemax over softmax is that it seems less GPU-friendly, since it requires sort operations or median-finding algorithms. There is, however, recent work providing efficient implementations of these algorithms on GPUs (Alabi et al., 2012).

[11] Rocktäschel et al. (2015) report scores slightly above ours: they reached a test accuracy of 82.3% for their implementation of SOFTATTENTION, and 83.5% with their best system, a more elaborate word-by-word attention model. Differences in the former case may be due to distinct word vectors and the use of LSTMs instead of GRUs.

[12] On the development set, we observed that only 24.6% of the premise words were selected.

Acknowledgements

We would like to thank the three anonymous reviewers, and also Tim Rocktäschel for answering various implementation questions, and Mário Figueiredo and Chris Dyer for helpful comments. This work was partially supported by Fundação para a Ciência e Tecnologia (FCT), through contracts UID/EEA/50008/2013 and UID/CEC/50021/2013, the LearnBig project (PTDC/EEI-SII/7092/2014), and the GoLocal project (grant CMUPERI/TIC/0046/2014).

References

Agrawala, Ashok K. Learning with a Probabilistic Teacher. *IEEE Transactions on Information Theory*, 16(4):373–379, 1970.

Alabi, Tolu, Blanchard, Jeffrey D, Gordon, Bradley, and Steinbach, Russel. Fast k-Selection Algorithms for Graphics Processing Units. *Journal of Experimental Algorithmics (JEA)*, 17:4–2, 2012.

Albert, James H and Chib, Siddhartha. Bayesian Analysis of Binary and Polychotomous Response Data. *Journal of the American statistical Association*, 88(422):669–679, 1993.

Astudillo, Ramon F, Amir, Silvio, Lin, Wang, Silva, Mário, and Trancoso, Isabel. Learning Word Representations from Scarce and Noisy Data with Embedding Subspaces. In *Proc. of the Association for Computational Linguistics (ACL), Beijing, China*, 2015.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*, 2015.

Blum, Manuel, Floyd, Robert W, Pratt, Vaughan, Rivest, Ronald L, and Tarjan, Robert E. Time Bounds for Selection. *Journal of Computer and System Sciences*, 7(4): 448–461, 1973.

Bouchard, Guillaume. Efficient Bounds for the Softmax Function and Applications to Approximate Inference in Hybrid Models. In *NIPS Workshop for Approximate Bayesian Inference in Continuous/Hybrid Systems*, 2007.

Bowman, Samuel R, Angeli, Gabor, Potts, Christopher, and Manning, Christopher D. A Large Annotated Corpus for Learning Natural Language Inference. In *Proc. of Empirical Methods in Natural Language Processing*, 2015.

Bradley, Ralph Allan and Terry, Milton E. Rank Analysis of Incomplete Block Designs: The Method of Paired Comparisons. *Biometrika*, 39(3-4):324–345, 1952.

Bridle, John S. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. In *Neurocomputing*, pp. 227–236. Springer, 1990.

Chapelle, Olivier, Schölkopf, Bernhard, and Zien, Alexander. *Semi-Supervised Learning*. MIT Press Cambridge, 2006.

Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc. of Empirical Methods in Natural Language Processing*, 2014.

Chorowski, Jan K, Bahdanau, Dzmitry, Serdyuk, Dmitriy, Cho, Kyunghyun, and Bengio, Yoshua. Attention-based Models for Speech Recognition. In *Advances in Neural Information Processing Systems*, pp. 577–585, 2015.

Clarke, Frank H. *Optimization and Nonsmooth Analysis*. New York, Wiley, 1983.

Condat, Laurent. Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, pp. 1–11, 2014.

de Brébisson, Alexandre and Vincent, Pascal. An Exploration of Softmax Alternatives Belonging to the Spherical Loss Family. *arXiv preprint arXiv:1511.05042*, 2015.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient Projections onto the L1-Ball for Learning in High Dimensions. In *Proc. of International Conference of Machine Learning*, 2008.

Gao, Wei and Zhou, Zhi-Hua. On the Consistency of Multi-Label Learning. *Artificial Intelligence*, 199:22–44, 2013.

Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. Deep Learning. Book in preparation for MIT Press, 2016. URL http://goodfeli.github.io/dlbook/.

Goodfellow, Ian J, Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout Networks. In *Proc. of International Conference on Machine Learning*, 2013.

Graves, Alex, Wayne, Greg, and Danihelka, Ivo. Neural Turing Machines. *arXiv preprint arXiv:1410.5401*, 2014.

Grefenstette, Edward, Hermann, Karl Moritz, Suleyman, Mustafa, and Blunsom, Phil. Learning to Transduce with Unbounded Memory. In *Advances in Neural Information Processing Systems*, pp. 1819–1827, 2015.

Hastie, Trevor, Tibshirani, Robert, and Wainwright, Martin. *Statistical Learning with Sparsity: the Lasso and Generalizations*. CRC Press, 2015.

Hermann, Karl Moritz, Kocisky, Tomas, Grefenstette, Edward, Espeholt, Lasse, Kay, Will, Suleyman, Mustafa, and Blunsom, Phil. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems*, pp. 1684–1692, 2015.

Huber, Peter J. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

Kingma, Diederik and Ba, Jimmy. Adam: A Method for Stochastic Optimization. In *Proc. of International Conference on Learning Representations*, 2014.

Koyejo, Sanmi, Natarajan, Nagarajan, Ravikumar, Pradeep K, and Dhillon, Inderjit S. Consistent Multilabel Classification. In *Advances in Neural Information Processing Systems*, pp. 3303–3311, 2015.

Lewis, David D, Yang, Yiming, Rose, Tony G, and Li, Fan. RCV1: A New Benchmark Collection for Text Categorization Research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

Liu, Dong C and Nocedal, Jorge. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

McCullagh, Peter and Nelder, John A. *Generalized Linear Models*, volume 37. CRC press, 1989.

Michelot, C. A Finite Algorithm for Finding the Projection of a Point onto the Canonical Simplex of $\mathbb{R}^n$. *Journal of Optimization Theory and Applications*, 50(1):195–200, 1986.

Nesterov, Y. A Method of Solving a Convex Programming Problem with Convergence Rate $O(1/k^2)$. *Soviet Math. Doklady*, 27:372–376, 1983.

Ollivier, Yann. Riemannian Metrics for Neural Networks. *arXiv preprint arXiv:1303.0818*, 2013.

Pardalos, Panos M. and Kovoor, Naina. An Algorithm for a Singly Constrained Class of Quadratic Programs Subject to Upper and Lower Bounds. *Mathematical Programming*, 46(1):321–328, 1990.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global Vectors for Word Representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543, 2014.

Penot, Jean-Paul. Conciliating Generalized Derivatives. In Demyanov, Vladimir F., Pardalos, Panos M., and Batsyn, Mikhail (eds.), *Constructive Nonsmooth Analysis and Related Topics*, pp. 217–230. Springer, 2014.

Rocktäschel, Tim, Grefenstette, Edward, Hermann, Karl Moritz, Kočiskỳ, Tomáš, and Blunsom, Phil. Reasoning about Entailment with Neural Attention. *arXiv preprint arXiv:1509.06664*, 2015.

Sukhbaatar, Sainbayar, Szlam, Arthur, Weston, Jason, and Fergus, Rob. End-to-End Memory Networks. In *Advances in Neural Information Processing Systems*, pp. 2431–2439, 2015.

Sutton, Richard S and Barto, Andrew G. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Courville, Aaron, Salakhutdinov, Ruslan, Zemel, Richard, and Bengio, Yoshua. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning*, 2015.

Zadrozny, Bianca. Reducing Multiclass to Binary by Coupling Probability Estimates. In *Advances in Neural Information Processing Systems*, pp. 1041–1048, 2001.

Zhang, Min-Ling and Zhou, Zhi-Hua. A Review on Multi-Label Learning Algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 26(8):1819–1837, 2014.

Zhang, Tong. Statistical Behavior and Consistency of Classification Methods Based on Convex Risk Minimization. *Annals of Statistics*, pp. 56–85, 2004.

Zou, Hui, Zhu, Ji, and Hastie, Trevor. The Margin Vector, Admissible Loss and Multi-class Margin-Based Classifiers. Technical report, Stanford University, 2006.