LETTER Communicated by Taeyoung Hahn

An EM Algorithm for Capsule Regression

Lawrence K. Saul
saul@cs.ucsd.edu
Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093-0404

We investigate a latent variable model for multinomial classification inspired by recent capsule architectures for visual object recognition (Sabour, Frosst, & Hinton, 2017). Capsule architectures use vectors of hidden unit activities to encode the pose of visual objects in an image, and they use the lengths of these vectors to encode the probabilities that objects are present. Probabilities from different capsules can also be propagated through deep multilayer networks to model the part-whole relationships of more complex objects. Notwithstanding the promise of these networks, there still remains much to understand about capsules as primitive computing elements in their own right. In this letter, we study the problem of capsule regression—a higher-dimensional analog of logistic, probit, and softmax regression in which class probabilities are derived from vectors of competing magnitude. To start, we propose a simple capsule architecture for multinomial classification: the architecture has one capsule per class, and each capsule uses a weight matrix to compute the vector of hidden unit activities for patterns it seeks to recognize. Next, we show how to model these hidden unit activities as latent variables, and we use a squashing nonlinearity to convert their magnitudes as vectors into normalized probabilities for multinomial classification. When different capsules compete to recognize the same pattern, the squashing nonlinearity induces nongaussian terms in the posterior distribution over their latent variables. Nevertheless, we show that exact inference remains tractable and use an expectation-maximization procedure to derive least-squares updates for each capsule's weight matrix. We also present experimental results to demonstrate how these ideas work in practice.

1 Introduction

Recently Sabour, Frosst, and Hinton (2017) introduced a novel capsule-based architecture for visual object recognition. A capsule is a group of hidden units that responds maximally to the presence of a particular object (or object part) in an image. But most important, the capsule responds to this presence in a specific way: its hidden units encode a pose vector

Neural Computation **33, 194–226** (2021)
https://doi.org/10.1162/neco_a_01336

for the object—a vector that varies (for instance) with the object's position and orientation—while the length of this vector encodes the probability that the object is present. Capsules were conceived by Hinton, Krizhevsky, and Wang (2011) to address a shortcoming of convolutional neural nets, whose hidden representations of objects in deeper layers are designed to be invariant to changes in pose. With such representations, it is difficult to model the spatial relationships between different objects in the same image. By contrast, the pose vectors in capsules learn equivariant representations of visual objects: these vectors do change with the pose, but in such a way that the object is still recognized with high probability. These ideas have led to a surge of interest in multilayer capsule networks for increasingly difficult problems in computer vision (Duarte, Rawat, & Shah, 2018; Kosiorek, Sabour, Teh, & Hinton, 2019; Qin et al., 2020).

Much of this work has focused on the message passing between capsules in different layers, as is needed to model the part-whole relationships of complex objects or entire scenes (Wang & Liu, 2018; Bahadori, 2018; Hinton, Sabour, & Frosst, 2018; Jeong, Lee, & Kim, 2019; Hahn, Pyeon, & Kim, 2019; Ahmed & Torresani, 2019; Tsai, Srivastava, Goh, & Salakhutdinov, 2020; Venkataraman, Balasubramanian, & Sarma, 2020). But capsules also introduced a novel paradigm for subspace learning that deserves to be explored—and elucidated—in its own right (Zhang, Edraki, & Qi, 2018). Consider, for instance, an individual capsule: its vector of hidden unit activities already represents a powerful generalization of the scalar activity computed by a simple neural element. In this letter, we shall discover an additional source of richness by modeling these hidden unit activities as latent variables in a probabilistic graphical model. The models we study are not exactly a special case of previous work, but they are directly motivated by it. As Sabour et al. (2017) wrote, "There are many possible ways to implement the general idea of capsules. . . . We want the length of the output vector of a capsule to represent the probability that the entity represented by the capsule is present in the current input." This general idea is also the starting point for our work.

In this letter, we study the problem of capsule regression, a problem in which multiple capsules must learn in tandem how to map inputs to pose vectors with competing magnitudes. We have written this letter with two readers in mind. The first is a practitioner of deep learning. She will view our models as a kind of evolutionary precursor to existing capsule networks; certainly, she will recognize at once how pose vectors are computed from inputs and converted via squashing nonlinearities into normalized probabilities. The second reader we have in mind is the working data scientist. Where the letter succeeds, she will view our models as a natural generalization of logistic and softmax regression—essentially, a higher-dimensional analog of these workhorses in which vectors compete in magnitude to determine how inputs should be classified. We hope that

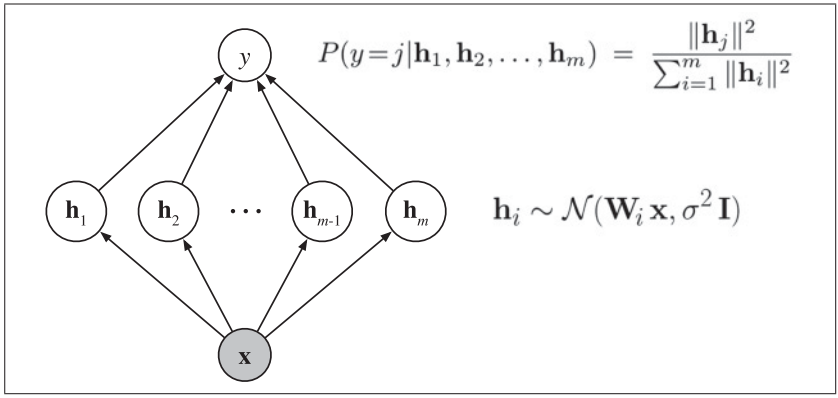

Figure 1: Capsule regression as a belief network for multinomial classification. The model's latent variables $\mathbf{h}_i \in \mathbb{R}^d$ are conditionally independent given the input $\mathbf{x}$, and the class label $y \in \{1, 2, \ldots, m\}$ is conditionally independent of the input $\mathbf{x}$ given the latent variables $\{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m\}$.

both types of readers see the novel possibilities for learning that these models afford.

The organization of this letter is as follows. In section 2, we formulate our latent variable model for capsule regression. Despite the model's squashing nonlinearity, we show that exact inference remains tractable and use an expectation-maximization (EM) procedure (Dempster, Laird, & Rubin, 1977) to derive least-squares updates for each capsule's weight matrix. In section 3, we present experimental results on images of handwritten digits and fashion items. Our results highlight how capsules use their internal distributed representations to learn more accurate classifiers. In section 4, we discuss issues that are deserving of further study, such as regularization, scaling, and model building. Finally, in the appendixes, we fill in the technical details and design choices that were omitted from the main development.

2 Model

Our model can be visualized as the belief network with latent variables shown in Figure 1, and our mathematical development is based closely on this representation. Section 2.1 gives an overview of the model and its EM algorithm for parameter estimation; here, we cover what is necessary to understand the model at a high level, though not all of what is required to implement it in practice. The later sections fill in these gaps. Section 2.2 focuses on the problem of inference; we show how to calculate likelihoods and statistics of the posterior distribution of our model. Section 2.3

focuses on the problem of learning; we show how the EM algorithm uses the model's posterior statistics to iteratively reestimate its parameters. Finally, section 2.4 describes a simple heuristic for initializing the model parameters, based on singular value decomposition, that seems to work well in practice.

2.1 Overview. We use the model in Figure 1 for multinomial classification—that is, to parameterize the conditional probability $P(y|\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^p$ is a vector-valued input and $y \in \{1, 2, \ldots, m\}$ is a class label. The model predicts the class label y based on the magnitudes of the m vector-valued latent variables $\{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m\}$; in particular, for each input $\mathbf{x}$, the most likely label y is determined by whichever vector $\mathbf{h}_i$ has the largest magnitude. The model's prediction depends essentially on three constructions: how the latent variables $\mathbf{h}_i$ depend on the input $\mathbf{x}$; how the class label y depends on the magnitudes $\|\mathbf{h}_i\|$ of these latent variables; and how the prediction $P(y|\mathbf{x})$ depends on the distribution over these magnitudes. We now describe each of these in turn.

The model assumes that the latent variables $\{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m\}$ are conditionally independent given the input $\mathbf{x}$. Thus, we may write

$$P(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m|\mathbf{x}) = \prod_{i=1}^{m} P(\mathbf{h}_i|\mathbf{x}), \tag{2.1}$$

where the conditional independence is represented in the network of Figure 1 by the absence of edges between nodes in its intermediate layer. In addition, the model assumes that each latent variable $\mathbf{h}_i \in \mathbb{R}^d$ is normally distributed as

$$P(\mathbf{h}_i|\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{h}_i - \mathbf{W}_i\mathbf{x}\right\|^2\right), \tag{2.2}$$

where the weight matrix $\mathbf{W}_i \in \mathbb{R}^{d\times p}$ is a model parameter that must be learned from examples. Each weight matrix $\mathbf{W}_i$ specifies a linear transformation from the input space to the latent space, and from equation 2.2, we see that it determines the conditional mean $\mathrm{E}[\mathbf{h}_i|\mathbf{x}] = \mathbf{W}_i\mathbf{x}$ of the latent variable $\mathbf{h}_i$. We have not added an explicit offset (or bias term) to this linear transformation, but if desired, the same effect can be accomplished by appending the input vector with an extra element equal to unity.

There is one additional model parameter in equation 2.2, namely, the variance σ^2, which determines the sharpness of the gaussian distribution and which (unlike the weight matrix) we assume is common to all the distributions $P(\mathbf{h}_1|\mathbf{x}), \ldots, P(\mathbf{h}_m|\mathbf{x})$ that appear in equation 2.1. We note that the model has an especially simple behavior in the limits $\sigma^2 \to 0$ and $\sigma^2 \to \infty$: in the former, the latent variables are deterministically specified by

$\mathbf{h}_i = \mathbf{W}_i\mathbf{x}$, while in the latter, they are completely delocalized. Though these limits are trivial, we shall see in section 2.2 that many nontrivial aspects of the model can be exactly calculated by a simple interpolation between these two regimes.

Next, we describe how the class label y depends on the model's latent variables. Inspired by previous capsule architectures (Sabour et al., 2017), the model uses a squashing nonlinearity to convert the magnitudes $\|\mathbf{h}_i\|$ into normalized probabilities $P(y|\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m)$ for multinomial classification. In particular, the model assumes that

$$P(y = j|\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m) = \frac{\|\mathbf{h}_j\|^2}{\sum_{i=1}^m \|\mathbf{h}_i\|^2}. \tag{2.3}$$

We note that equation 2.3 is used to compute probabilities for $m \geq 2$ outcomes, while the squashing nonlinearity in Sabour et al. (2017) was used to compute probabilities of binary outcomes. This difference is more or less analogous to the progression from logistic to softmax regression in the simplest linear models of classification.

It is obvious that the model in Figure 1 is far more primitive than the multilayer capsule networks that have been explored for difficult problems in visual object recognition (Sabour et al., 2017; Hinton et al., 2018; Ahmed & Torresani, 2019; Hahn et al., 2019; Jeong et al., 2019; Venkataraman et al., 2020; Tsai et al., 2020). Nevertheless, this model does provide what is arguably the simplest expression of the raison d'etre for capsules—namely, the idea that the length of a vector of hidden unit activities can encode the probability that some pattern is present in the input. The model can also be viewed as a higher-dimensional analog of logistic/probit regression (for $m = 2$ classes) or softmax regression (for $m \geq 3$ classes) in which each class is modeled by a vector of hidden unit activities as opposed to a single scalar dot product.

In developing the model further, it becomes needlessly cumbersome to list the m latent variables $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m$ wherever they are collectively employed. In what follows, therefore, we denote the collection of these m latent variables by $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m)$. In this way, we may simply write the factorization in equation 2.1 as $P(\mathbf{h}|\mathbf{x}) = \prod_i P(\mathbf{h}_i|\mathbf{x})$ and the squashing nonlinearity in equation 2.3 as $P(y = j|\mathbf{h}) = \|\mathbf{h}_j\|^2/\|\mathbf{h}\|^2$. As a similar shorthand, we also denote the collection of m weight matrices in the model by $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_m)$.

Finally, we describe how the model predicts class labels from inputs. We obtain the conditional probability $P(y = j|\mathbf{x})$ by marginalizing the model's latent variables:

$$P(y = j|\mathbf{x}) = \int_{\mathbf{h}\in\mathbb{R}^{dm}} P(y = j|\mathbf{h})\, P(\mathbf{h}|\mathbf{x}). \tag{2.4}$$

Note that the multidimensional integral in equation 2.4 incorporates the squashing nonlinearity in $P(y = j|\mathbf{h})$ through equation 2.3; in particular, equation 2.4 is not a purely gaussian integral. Nevertheless, we show in section 2.2 that this integral can be evaluated analytically. We shall see, in fact, that the result is also simple enough to permit the further calculations required for inference.

We leave these more detailed results for later. For now, though, it behooves us to examine how the integral in equation 2.4 simplifies in the opposing limits that the model's variance σ^2 either vanishes or diverges. In these limits, we have respectively that

$$P(y = j|\mathbf{x}) \;\to\; \begin{cases} \dfrac{\|\mathbf{W}_j\mathbf{x}\|^2}{\sum_i \|\mathbf{W}_i\mathbf{x}\|^2} & \text{as } \sigma^2 \to 0, \\ \dfrac{1}{m} & \text{as } \sigma^2 \to \infty. \end{cases} \tag{2.5}$$

These limits are simple to understand: as $\sigma^2 \to 0$, the model's latent variables are effectively determined by the inputs, which renders the integral in equation 2.4 trivial, whereas as $\sigma^2 \to \infty$, the model becomes maximally uncertain, predicting each class with equal probability regardless of the input. Once again, we note that while these limits are trivial, we shall see that many nontrivial aspects of the model can be exactly calculated by a simple interpolation between these two regimes.

Having demonstrated how the model makes predictions, we turn briefly to the problem of learning—that is, of estimating parameters $\mathbf{W}$ and σ that yield a useful model. We note that in the course of learning, the predictions of our model typically pass from a regime of high uncertainty (i.e., larger σ^2) to low uncertainty (i.e., smaller σ^2), and therefore it is exactly the intermediate regime between the two limits in equation 2.5 where we expect the bulk of learning to occur.

We consider the problem of supervised learning, where the goal is to estimate the model parameters $\mathbf{W}$ and σ^2 from a data set of n labeled examples $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$. In our model, the simplest form of learning is to maximize the log-conditional likelihood given by

$$\mathcal{L}(\mathbf{W}, \sigma^2) = \sum_{\ell=1}^{n} \log P(y_\ell|\mathbf{x}_\ell). \tag{2.6}$$

As mentioned in section 1, the objective here is the same as maximum likelihood (ML) estimation in more traditional models of logistic, probit, or softmax regression. However, our capsule architecture replaces the scalar activities of these more traditional models, derived from dot products, with

the richer distributed representations derived from matrix-vector multiplications, and it replaces the sigmoidal and softmax nonlinearities of these models with the squashing nonlinearity in equation 2.3.

To maximize the likelihood in equation 2.6, we can avail ourselves of the EM procedure (Dempster et al., 1977) for latent variable models. As is usual, the EM algorithm for our model alternates between two steps—an E-step, which computes posterior statistics of the model's latent variables, and an M-step, which uses these statistics to reestimate the model's parameters and increase the log-conditional likelihood in equation 2.6. For the ℓth labeled example, the posterior distribution over the latent variables is given by Bayes' rule:

$$P(\mathbf{h}|y_\ell, \mathbf{x}_\ell) = \frac{P(y_\ell|\mathbf{h})\, P(\mathbf{h}|\mathbf{x}_\ell)}{P(y_\ell|\mathbf{x}_\ell)}. \tag{2.7}$$

For our purposes, the most important statistics of this distribution are the posterior means $\mathrm{E}[\mathbf{h}_i|y_\ell, \mathbf{x}_\ell]$ for each latent variable $\mathbf{h}_i$. The E-step of the EM algorithm computes these expected values by averaging over the posterior distribution:

$$\mathrm{E}[\mathbf{h}_i|y_\ell, \mathbf{x}_\ell] = \int_{\mathbf{h}\in\mathbb{R}^{dm}} P(\mathbf{h}|y_\ell, \mathbf{x}_\ell)\, \mathbf{h}_i. \tag{2.8}$$

An important feature of our model is that the multidimensional integral in equation 2.8 can also be calculated in closed form. As a result, we can implement the EM algorithm without recourse to approximations such as variational inference (Wainwright & Jordan, 2008) or stochastic simulation (Neal, 1993). Finally, the M-step of the EM algorithm takes a particularly simple form in our model: the weight matrix $\mathbf{W}_i$ is reestimated by solving the least-squares problem,

$$\mathbf{W}_i \leftarrow \operatorname*{arg\,min}_{\mathbf{\Lambda}\in\mathbb{R}^{d\times p}} \left\{ \sum_{\ell=1}^{n} \Big\| \mathrm{E}[\mathbf{h}_i|y_\ell, \mathbf{x}_\ell] - \mathbf{\Lambda}\mathbf{x}_\ell \Big\|^2 \right\} \tag{2.9}$$

where the posterior means $\mathrm{E}[\mathbf{h}_i|y_\ell, \mathbf{x}_\ell]$ that appear in equation 2.9 are precisely those computed by the E-step (in terms of the current weight matrices).

In sum, our model is defined by the multivariate gaussian distributions in equation 2.1 and the squashing nonlinearity in equation 2.3, and its essential parameters (namely, the weight matrices $\mathbf{W}_i$) are reestimated by computing the posterior statistics in equation 2.8 and solving the least-squares problems in equation 2.9. The next sections provide the results that are needed to implement these steps in practice.

2.2 Inference. To perform inference in our model, we must integrate over the model's latent variables. For example, to predict class labels, we must compute the conditional probability $P(y = j|\mathbf{x})$. Substituting equations 2.1 and 2.3 into 2.4, we see that

$$P(y = j|\mathbf{x}) = \int_{\mathbf{h}\in\mathbb{R}^{dm}} \frac{\|\mathbf{h}_j\|^2}{\|\mathbf{h}\|^2} \prod_i \frac{e^{-\frac{1}{2\sigma^2}\|\mathbf{h}_i - \mathbf{W}_i\mathbf{x}\|^2}}{(2\pi\sigma^2)^{d/2}}, \tag{2.10}$$

Similarly, to carry out the EM algorithm, we must compute the posterior means $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$. Substituting equation 2.7 into 2.8, we see that

$$\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j] = \frac{1}{P(y = j|\mathbf{x})} \int_{\mathbf{h}\in\mathbb{R}^{dm}} \mathbf{h}_i \frac{\|\mathbf{h}_j\|^2}{\|\mathbf{h}\|^2} \prod_i \frac{e^{-\frac{1}{2\sigma^2}\|\mathbf{h}_i - \mathbf{W}_i\mathbf{x}\|^2}}{(2\pi\sigma^2)^{d/2}}, \tag{2.11}$$

We show how to evaluate these multidimensional integrals exactly in appendix A. In this section, though, we mainly present the results of these calculations, focusing on what is required to implement the model in practice.

As discussed in the previous section, the integrals in equations 2.10 and 2.11 behave simply in the limits $\sigma^2 \to 0$ and $\sigma^2 \to \infty$. It is more convenient, however, to express their overall behavior in terms of the dimensionless ratio,

$$\beta = \frac{1}{2\sigma^2} \sum_i \|\mathbf{W}_i\mathbf{x}\|^2, \tag{2.12}$$

where $\beta \to 0$ and $\beta \to \infty$ correspond, respectively, to the limits of complete and zero uncertainty. Note that β is defined with respect to a particular input $\mathbf{x}$ and that it also depends on the model parameters $\mathbf{W}$ and σ. As shorthand, though, we will not denote this dependence explicitly.

To use our model, we must understand how it behaves for intermediate values of β. To this end, we define the *interpolating* functions $\lambda_0(\beta)$ and $\lambda_1(\beta)$ given by

$$\lambda_0(\beta) = \beta e^{-\beta} \int_0^1 d\rho\, \rho^{\frac{dm}{2}}\, e^{\rho\beta}, \tag{2.13}$$

$$\lambda_1(\beta) = \beta e^{-\beta} \int_0^1 d\rho\, \rho^{\frac{dm}{2}+1}\, e^{\rho\beta}. \tag{2.14}$$

The one-dimensional integrals in these definitions can be evaluated analytically, but the most important properties of $\lambda_0(\beta)$ and $\lambda_1(\beta)$ are in fact more easily derived from these integral representations. We show how to

evaluate these integrals efficiently in section A.1. From these definitions, we also prove that $\lambda_0(\beta)$ and $\lambda_1(\beta)$ are monotonically increasing functions with

$$\lambda_0(\beta),\ \lambda_1(\beta) \to \begin{cases} 0 & \text{as} \quad \beta \to 0, \\ 1 & \text{as} \quad \beta \to \infty. \end{cases} \tag{2.15}$$

Thus, these interpolating functions can be viewed as providing another continuous measure of the model's uncertainty, but unlike the dimensionless ratio β (or the variance σ^2), they are bounded between 0 and 1.

The most important inferences in our model take an especially compact form in terms of these interpolating functions. For example, as shown in section A.2, the conditional probability $P(y = j|\mathbf{x})$ in equation 2.10 is given exactly by

$$P(y = j|\mathbf{x}) = \lambda_0(\beta) \cdot \frac{\|\mathbf{W}_j\mathbf{x}\|^2}{\sum_i \|\mathbf{W}_i\mathbf{x}\|^2} \ + \ \big(1 - \lambda_0(\beta)\big) \cdot \frac{1}{m}. \tag{2.16}$$

Note in this expression that $\lambda_0(\beta)$ appears simply as a coefficient that mixes the model's predictions in the limits of zero and complete uncertainty, as given earlier by equation 2.5. It follows from equation 2.16 that $\arg\max_j P(y = j|\mathbf{x}) = \arg\max_j \|\mathbf{W}_j\mathbf{x}\|^2$, so that the model can be used as a classifier—that is, to predict the most likely label of an input $\mathbf{x}$—without evaluating $P(y = j|\mathbf{x})$ in full. From equation 2.16, it is also trivial to verify that this posterior distribution is properly normalized with $\sum_j P(y = j|\mathbf{x}) = 1$.

Next we consider the posterior mean $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$, given by equation 2.11. Recall that each posterior mean $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ is a d-dimensional vector (where d is the capsule dimensionality) that lives in the same vector space as its corresponding prior mean $\mathrm{E}[\mathbf{h}_i|\mathbf{x}]$. In addition, it follows from the model's symmetries that the vector $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ must be parallel to the vector $\mathrm{E}[\mathbf{h}_i|\mathbf{x}]$, differing in magnitude but not direction. Thus, we may write

$$\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j] \ \propto \ \mathrm{E}[\mathbf{h}_i|\mathbf{x}], \tag{2.17}$$

where the constant of proportionality depends in general on the input $\mathbf{x}$, the class label j, the model parameters $\mathbf{W}$ and σ^2, and the index i of the capsule that hosts the latent variable $\mathbf{h}_i$. From the squashing nonlinearity in equation 2.3, we might intuitively expect the posterior mean $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ to exceed the prior mean $\mathrm{E}[\mathbf{h}_i|\mathbf{x}]$ in magnitude when $i = j$ and vice versa when $i \neq j$. This intuition is borne out by the exact calculation in section A.2, to which we refer readers for further details.

We now complete the result implied by equation 2.17. Like the conditional probability $P(y = j|\mathbf{x})$ in equation 2.16, the posterior mean

$\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ can also be expressed in a highly intelligible form. To this end, we define a new family of distributions, $Q_i(y = j|\mathbf{x})$, given by

$$Q_i(y = j|\mathbf{x}) = \lambda_1(\beta) \cdot \frac{\|\mathbf{W}_j\mathbf{x}\|^2}{\sum_k \|\mathbf{W}_k\mathbf{x}\|^2} + \big(1 - \lambda_1(\beta)\big) \cdot \frac{2\delta_{ij} + d}{2 + dm}, \tag{2.18}$$

where $\lambda_1(\beta)$ is the interpolating function defined in equation 2.14 and δ_{ij} denotes the Kronecker delta function. It is easy to verify that the right-hand side of equation 2.18 defines a distribution, with $\sum_j Q_i(y = j|\mathbf{x}) = 1$ for each value of i. In terms of these distributions, the posterior means are given by

$$\mathrm{E}[\mathbf{h}_i|y = j, \mathbf{x}] = \frac{Q_i(y = j|\mathbf{x})}{P(y = j|\mathbf{x})} \cdot \mathrm{E}[\mathbf{h}_i|\mathbf{x}], \tag{2.19}$$

so that we can identify the ratio of the distributions $Q_i(y = j|\mathbf{x})$ and $P(y = j|\mathbf{x})$ with the implied constant of proportionality in equation 2.17. Note that just as the conditional distribution $P(y = j|\mathbf{x})$ must be properly normalized, satisfying $\sum_j P(y = j|\mathbf{x}) = 1$, it must also be true that the prior and posterior means are internally consistent, satisfying $\sum_j P(y = j|\mathbf{x})\mathrm{E}[\mathbf{h}_i|y = j, \mathbf{x}] = \mathrm{E}[\mathbf{h}_i|\mathbf{x}]$. From equation 2.19, it is easy to verify that this identity holds.

In sum, we compute the conditional probability $P(y = j|\mathbf{x})$ from equation 2.16, and we compute the posterior means $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ from equation 2.19. We need both of these results for the EM algorithm—the first to verify that the likelihood of the data in equation 2.6 is increasing and the second to facilitate the update in equation 2.9. In the next section, we examine this update in more detail.

2.3 Learning. We can use the EM procedure to derive updates that reestimate the parameters $(\mathbf{W}, \sigma^2)$ of our model. We begin by observing that the log-conditional likelihood $\mathcal{L}(\mathbf{W}, \sigma^2)$ in equation 2.6 has a symmetry: it is invariant under the rescaling transformation,

$$\mathbf{W} \to \frac{\mathbf{W}}{\sqrt{\sigma^2}}, \tag{2.20}$$

$$\sigma^2 \to 1. \tag{2.21}$$

This property follows at once from our expression for the conditional probability $P(y|\mathbf{x})$ in equation 2.16 and the fact that the dimensionless ratio $\beta = \frac{1}{2\sigma^2}\sum_i \|\mathbf{W}_i\mathbf{x}\|^2$ is also invariant under this transformation. Thus, for any model with $\sigma^2 \neq 1$, we can obtain an equivalent model with $\sigma^2 = 1$ by rescaling the weights in this way. (The transformed model is equivalent in the sense that it predicts labels from inputs with the same probabilities.) Without loss of generality, we can therefore fix $\sigma^2 = 1$ at the outset of learning and only use the EM algorithm to reestimate the model's weight

matrices. In the previous sections of this letter, we used the parameter σ^2 to formulate our model and develop an intuition for how it behaves. At this point, however, we set $\sigma^2 = 1$ and do not bother[1] to reestimate it.

We observed earlier that the update for the weight matrix $\mathbf{W}_i$ takes the form of the least-squares problem in equation 2.9. As Dempster et al. (1977) noted, this result is typical of models with gaussian latent variables, such as factor analysis (Rubin & Thayer, 1982; Ghahramani & Hinton, 1996) and probit regression (Liu, Rubin, & Wu, 1998). In these models, the EM procedure also yields iterative least-squares updates for ML estimation, and the update in equation 2.9 is derived in exactly the same way.

We gain some further insights by writing out the update in equation 2.9 explicitly. Solving the least-squares problem, we see that

$$\mathbf{W}_i \leftarrow \left(\sum_{\ell=1}^{n} \mathrm{E}[\mathbf{h}_i | \mathbf{x}_\ell, y_\ell]\, \mathbf{x}_\ell^\top \right) \left(\sum_{\ell=1}^{n} \mathbf{x}_\ell \mathbf{x}_\ell^\top \right)^{-1}, \tag{2.22}$$

where, as usual, the posterior means $\mathrm{E}[\mathbf{h}_i | \mathbf{x}_\ell, y_\ell]$ on the right-hand side are computed from the current model parameters. Note that the matrix inverse in equation 2.22 can be precomputed from the data at the outset of learning. This step is extremely useful with inputs of moderate to high dimensionality.

With some further work, we can obtain an even more revealing expression for the update in equation 2.22. We begin by introducing another shorthand: for each input $\mathbf{x}_\ell$ in the training set, we define the *conjugate* input by

$$\mathbf{v}_\ell = \left(\sum_{\ell=1}^{n} \mathbf{x}_\ell \mathbf{x}_\ell^\top \right)^{-1} \mathbf{x}_\ell. \tag{2.23}$$

The vectors $\mathbf{x}_\ell$ and $\mathbf{v}_\ell$ are conjugate in the sense that by construction, the sum $\sum_{\ell=1}^{n} \mathbf{x}_\ell \mathbf{v}_\ell^\top$ is equal to the $p \times p$ identity matrix, where p is the dimensionality of the input space. Combining this shorthand with the result in equation 2.19, we can rewrite the update in equation 2.22 as

$$\mathbf{W}_i \leftarrow \mathbf{W}_i \left(\sum_{\ell=1}^{n} \frac{Q_i(y_\ell | \mathbf{x}_\ell)}{P(y_\ell | \mathbf{x}_\ell)}\, \mathbf{x}_\ell \mathbf{v}_\ell^\top \right). \tag{2.24}$$

[1] This point is more subtle than we have indicated: it has been shown that the EM algorithm can be accelerated in gaussian latent variable models by representing the variance explicitly (Liu, Rubin, & Wu, 1998). However, we do not pursue that path here.

The term inside the parentheses of this update is a $p \times p$ matrix. Thus, from equation 2.24, we can also see that the EM algorithm is performing a *multiplicative* update on each capsule's weight matrix.

The update in equation 2.24 is guaranteed to increase the log-conditional likelihood in equation 2.6 except at stationary points. In practice, however, we have found it useful to modify this update in two ways. These modifications do not strictly preserve the EM algorithm's guarantee of monotonic convergence in the likelihood, but they yield other practical benefits without seeming to compromise the algorithm's stability. We discuss these next.

First, we modify the update in equation 2.24 to learn models that yield more accurate classifiers. We have noticed that the log-conditional likelihood in equation 2.6 does not always track the model's accuracy as a classifier in the later stages of learning. To counter this problem, we can modify the update in equation 2.24 to focus more on the examples that the model predicts least accurately. Such examples can be identified by computing the ratio

$$r_\ell = \frac{\max_{y \neq y_\ell} P(y|\mathbf{x}_\ell)}{P(y_\ell|\mathbf{x}_\ell)}. \tag{2.25}$$

This ratio is greater than one for inputs that are misclassified and less than one for inputs that are classified correctly. In addition, the ratio is close to zero for inputs that are classified correctly with high certainty. To focus on inputs that are misclassified, we choose a threshold $\nu \in [0, 1]$ and identify the inputs with $r_\ell \leq \nu$; next, for this subset of correctly classified inputs, we replace the means $\mathrm{E}[\mathbf{h}_i|\mathbf{x}_\ell, y_\ell]$ in the update of equation 2.24 by the prior means $\mathrm{E}[\mathbf{h}_i|\mathbf{x}_\ell]$. This modification leads to the thresholded update:

$$\mathbf{W}_i \leftarrow \mathbf{W}_i \left(\sum_{r_\ell \leq \nu} \mathbf{x}_\ell \mathbf{v}_\ell^\top + \sum_{r_\ell > \nu} \frac{Q_i(y_\ell|\mathbf{x}_\ell)}{P(y_\ell|\mathbf{x}_\ell)} \mathbf{x}_\ell \mathbf{v}_\ell^\top \right). \tag{2.26}$$

By setting $\nu = 0$, we recover the previous update in equation 2.24. For positive values of ν, however, the thresholded update focuses the model on those examples whose predictions are least accurate and/or most uncertain.

We emphasize that the correctly classified examples with $r_\ell \leq \nu$ are not dropped from the update altogether; they still appear in the first sum on the right-hand side of equation 2.22. The goal is not to ignore these examples, only to reduce their influence on the model after they are correctly classified with high certainty. The thresholded update can be viewed as a heuristic for large-margin classification (Boser, Guyon, & Vapnik, 1992), focusing on incorrect examples and/or correct examples near the decision boundary. Though motivated differently, it also resembles certain incremental variants

of the EM algorithm that have been explored for latent variable modeling (Neal & Hinton, 1998).

In addition to equation 2.26, we explore one other modification whose goal is to accelerate the algorithm's rate of convergence. There is a large literature on methods for accelerating EM algorithms while preserving their guarantees of monotonic convergence in the likelihood (Jamshidian & Jennrich, 1993, 1997; Liu & Rubin, 1994; Lange, 1995; Liu et al., 1998; Salakhutdinov, Roweis, & Ghahramani, 2003; Varadhan & Roland, 2008; Yu, 2012). For this work, we have adopted the simple approach of incorporating a momentum term (Qian, 1999) with an additional hyperparameter $\gamma > 0$. Such terms have a long and successful history with gradient-based methods (Rumelhart, Hinton, & Williams, 1986). Our approach does not preserve the theoretical guarantee of convergence, but in practice, we have observed that it behaves remarkably well. With this further modification, the update takes the form

$$\mathbf{W}^{(t+1)} = \mathrm{EM}_\nu\left(\mathbf{W}^{(t)}\right) + \gamma\left(\mathbf{W}^{(t)} - \mathbf{W}^{(t-1)}\right), \tag{2.27}$$

where $\mathrm{EM}_\nu(\mathbf{W})$ refers to the thresholded update in equation 2.26 and $\mathbf{W}^{(t)}$ refers to the weight matrices estimated after t iterations of this procedure. We obtained our main results in section 3 with this modified update, but we compare the behavior of equation 2.24 versus equations 2.26 and 2.27 further in appendix B.

2.4 Initialization. EM algorithms do not converge in general to a global maximum of the likelihood, and as a result, the models they discover can depend on how the model parameters are initialized. This is true for the update in equation 2.24 as well as the modified updates in equations 2.26 and 2.27.

After fixing $\sigma^2 = 1$, we compared two approaches for initializing the model's weight matrices $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_m$. In the first approach, we randomly sampled each matrix element from a gaussian distribution with zero mean and small variance. In the second approach, we initialized the matrices based on a singular value decomposition of the training examples in each class. We refer to the first approach as *random* initialization and the second as *subspace* initialization.

Roughly speaking, the goal of subspace initialization is to maximize $\|\mathbf{W}_i\mathbf{x}\|$ for inputs $\mathbf{x}$ that belong to the ith class. In particular, let $n_i < n$ denote the number of training examples with label i, and let $\mathbf{X}_i$ denote the $p \times n_i$ matrix of inputs with this label. We assume that for each class label, we have at least as many examples as the capsule dimensionality: $n_i \geq d$. The subspace initialization attempts to seed the matrix $\mathbf{W}_i$ with an informative projection of these n_i inputs; intuitively, such a projection seems more likely to encode the variability of patterns recognized by the ith capsule. To this end,

we compute the leading d eigenvectors $\{\mathbf{v}_{i\alpha}\}_{\alpha=1}^{d}$ and eigenvalues $\{\xi_{i\alpha}\}_{\alpha=1}^{d}$ of the $p \times p$ matrix $n_i^{-1}\mathbf{X}_i\mathbf{X}_i^\top$. Then we initialize the ith capsule's weight matrix as

$$\mathbf{W}_i = \frac{1}{\sqrt{d}} \begin{pmatrix} \frac{1}{\sqrt{\xi_{i1}}}\mathbf{v}_{i1} \\ \frac{1}{\sqrt{\xi_{i2}}}\mathbf{v}_{i2} \\ \vdots \\ \frac{1}{\sqrt{\xi_{id}}}\mathbf{v}_{id} \end{pmatrix}. \tag{2.28}$$

The factors of $\frac{1}{\sqrt{d}}$ and $\frac{1}{\sqrt{\xi_{i\alpha}}}$ in this initialization attempt to control for the dimensionality of the latent space and the dynamic range of the inputs. We obtained our main results in section 3 with the initialization in equation 2.28, but we compare the effects of random versus subspace initializations further in appendix B.

3 Experiments

We implemented the model described in section 2 and evaluated its performance in different architectures for multinomial classification. In some especially simple settings, we also sought to understand the latent representations learned by capsule regression. The organization of this section is as follows. In section 3.1, we describe the data sets that we used for benchmarking and the common setup for all of our experiments. In section 3.2, we present a visualization of results from the model in Figure 1 with two-dimensional capsules ($d = 2$). This visualization reveals how the latent spaces of different capsules are organized in tandem by the updates of section 2.3 to learn an accurate classifier. In section 3.3, we examine the internal representations learned by the model in Figure 1 with eight-dimensional capsules ($d = 8$). Here, we explore how distinct patterns of variability are encoded by different elements of the model's latent variables. Finally, in section 3.4, we present our main results—a systematic comparison of classifiers obtained from capsules of varying dimensionality as well as different capsule-based architectures (e.g., multiclass, one versus all, all versus all) for multinomial classification.

3.1 Setup. We experimented on two data sets of images—one of handwritten digits (LeCun, Bottou, Bengio, & Haffner, 1998) and the other of fashion items (Xiao, Rasul, & Vollgraf, 2017). Both data sets contain 60,000 training examples and 10,000 test examples of 28×28 grayscale images drawn from $m = 10$ classes; they have also been extensively benchmarked.

To speed up our experiments, we began by reducing the dimensionality of these images by a factor of four. Specifically, for each data set, we used a singular value decomposition of the training examples to identify the linear projections with largest variance, and then we used these projections to map each image into an input $\mathbf{x} \in \mathbb{R}^p$ for capsule regression, where $p = 196$. This was done for all the experiments in this letter.

We followed a single common protocol for training, validation, and testing (with one exception, mentioned at the end of the section, where we trained on a much larger data set of 1 million digit images). We trained our models on the first 50,000 examples in each training set while holding out the last 10,000 examples as a validation set. We monitored the classification error rate on the validation set in an attempt to prevent overfitting. We used the subspace initialization in equation 2.28 to seed the weight matrices before training, and we used a fixed momentum hyperparameter of $\gamma = 0.9$ in the update of equation 2.27. We did not use a fixed value for the thresholding hyperparameter ν. Instead, we divided the learning into five rounds with ν taking on a fixed value of 0.8 in the first round, 0.6 in the second, 0.4 in the third, 0.2 in the fourth, and 0 in the fifth. We terminated the first round when the error rate on the validation set had not improved for 128 consecutive iterations, and we terminated the next rounds when it had not improved for 64, 32, 16, and 8 consecutive iterations, respectively. Thus, each round had a fixed minimum number of iterations, though not a fixed maximum. Finally, we initialized each subsequent round by the best model obtained in the previous one (as measured by the error rate on the validation set). We present some empirical motivation for these choices of hyperparameters in appendix B.

3.2 Visualization of Results with $d = 2$ Capsules. We begin by examining one of the simplest models of capsule regression on the MNIST images of handwritten digits. Figure 2 visualizes the latent representations learned by each capsule in Figure 1 where $\mathbf{h}_i \in \mathbb{R}^2$. (This model for capsule regression has a fairly high test error rate of 6.21% on the digit images, but it is also by far the simplest to visualize.) As before, let $\mathrm{E}[\mathbf{h}_i|\mathbf{x}] = \mathbf{W}_i\mathbf{x}$ denote the expected (vector) value of the latent variable in the ith capsule given the input $\mathbf{x}$. The ith panel in the figure shows a scatterplot of the squashed two-dimensional vectors,

$$\mathbf{\Psi}_i(\mathbf{x}) = \frac{\mathrm{E}[\mathbf{h}_i|\mathbf{x}]}{\sqrt{\sum_{j=1}^m \left\| \mathrm{E}[\mathbf{h}_j|\mathbf{x}] \right\|^2}}, \tag{3.1}$$

where $\mathbf{x}$ is an input from the test set of 10,000 images. Note that by construction, we have that $\|\mathbf{\Psi}_i(\mathbf{x})\|^2 = P(y = i|\mathbf{h} = \mathrm{E}[\mathbf{h}|\mathbf{x}])$. It follows that each such vector $\mathbf{\Psi}_i(\mathbf{x})$ lies within the unit circle, and the larger its magnitude,

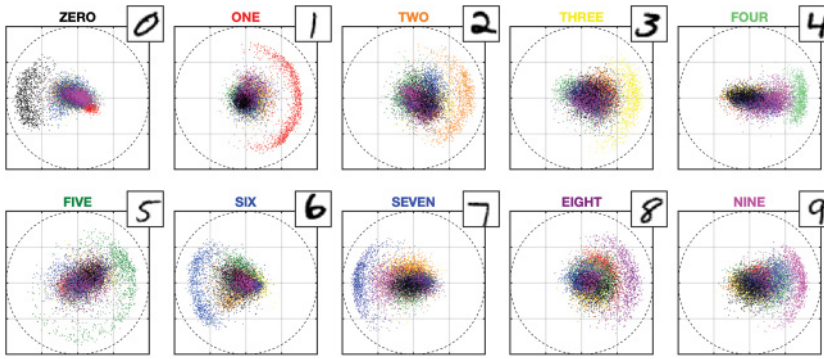

Figure 2: Visualization of the two-dimensional (squashed) latent representations in equation 3.1 for test images of MNIST hand-written digits. Best viewed in color.

the larger the probability $P(y = i|\mathbf{x})$ that the model assigns to the input $\mathbf{x}$. Each vector in these plots is also color-coded by its class label as displayed in the panel titles. Thus, for example, in the upper left-most panel, we see that test images of `zeros` (color-coded as black) are represented by vectors that lie near the unit circle and therefore receive large probabilities for being labeled as `zeros`. In the other panels, however, these same images (again, color-coded as black) are represented by vectors that lie closer to the origin and therefore receive small probabilities for being labeled as anything else. In fact, this pattern repeats itself for all the classes of digits: in each panel, the vectors that lie nearest the unit circle are those that share the same color as the class label in the panel's title. (At the top right of each panel, we display the most confidently recognized image of each class.)

Figure 3 shows the corresponding result for the same model trained on images of fashion items. The test error rate (15.14%) is higher for this data set, but the same pattern is evident. From the plots in these panels, we can also see which classes of images are most confusable. For example, the black latent variables (representing images of T-shirts) have large radii not only in the upper left-most panel but also in the bottom panel, second from the left. These two panels—corresponding to the capsules for T-shirts and shirts—show that these two classes of images are among the likeliest to be confused.

Naturally it is more difficult to visualize the results from models of capsule regression with higher-dimensional ($d > 2$) latent variables. But conceptually it is clear what happens: the circles of unit radius in the panels of Figures 2 and 3 are replaced by hyperspheres of unit radius in d dimensions. With latent variables of higher dimensionality, we might also expect the capsules to discover richer internal representations of the variability within each class of images. This is what we explore in the next section.

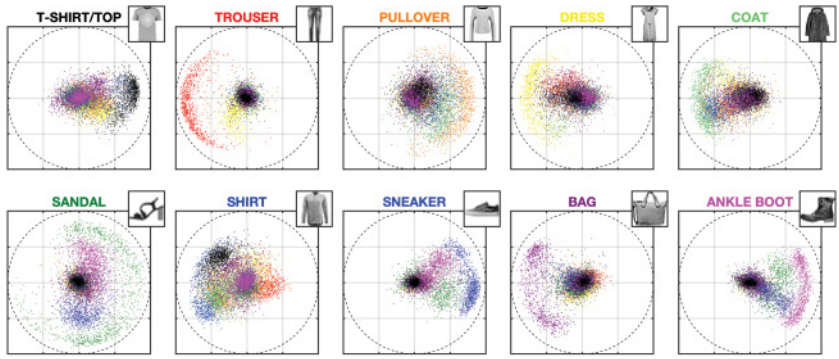

Figure 3: Visualization of the two-dimensional (squashed) latent representations in equation 3.1 for test images of fashion items. Best viewed in color.

3.3 Encoding of Variability by $d = 8$ Capsules. The latent variable $\mathbf{h}_i$ in the model of Figure 1 provides an internal representation of the variability of patterns in the ith class. Though difficult to visualize these patterns in higher-dimensional capsules, we can identify—for instance—the training example in the ith class for which the αth element of $\mathrm{E}[\mathbf{h}_i|\mathbf{x}]$ is largest in magnitude relative to the other elements of $\mathrm{E}[\mathbf{h}|\mathbf{x}]$. Specifically, for each class $i \in \{1, 2, \ldots, m\}$ and element $\alpha \in \{1, 2 \ldots, d\}$, we can identify the example

$$\mathbf{x}^*_{i\alpha} = \arg\max_{y_\ell = i} \left(\frac{\mathrm{E}[h_{i\alpha}|\mathbf{x}_\ell]}{\|\mathrm{E}[\mathbf{h}|\mathbf{x}_\ell]\|} \right)^2, \tag{3.2}$$

whose probability $P(y = i|\mathbf{x})$ receives its largest contribution from the αth element of latent variable $\mathbf{h}_i \in \mathbb{R}^d$. Such an example can be viewed as a prototype for the pattern of variability encoded by the αth latent variable of the ith capsule.

Figure 4 shows these prototypical examples for the model of Figure 1 with capsules of dimensionality $d = 8$. For the images of digits, the prototypes exhibit variations in orientation, thickness, and style (e.g., the presence of a loop in the digit 2 or an extra horizontal bar in the digit 7). For the images of fashion items, the prototypes exhibit variations in brightness, girth, and basic design. The examples in the figure are suggestive of the prototypes discovered by vector quantization (Lloyd, 1957), but in this case, they have emerged from internal representations of discriminatively trained capsules. It is clear that higher-dimensional capsules can represent a greater diversity of prototypes, and by doing so, they might be expected to yield more accurate classifiers. This is what we explore in the next section.

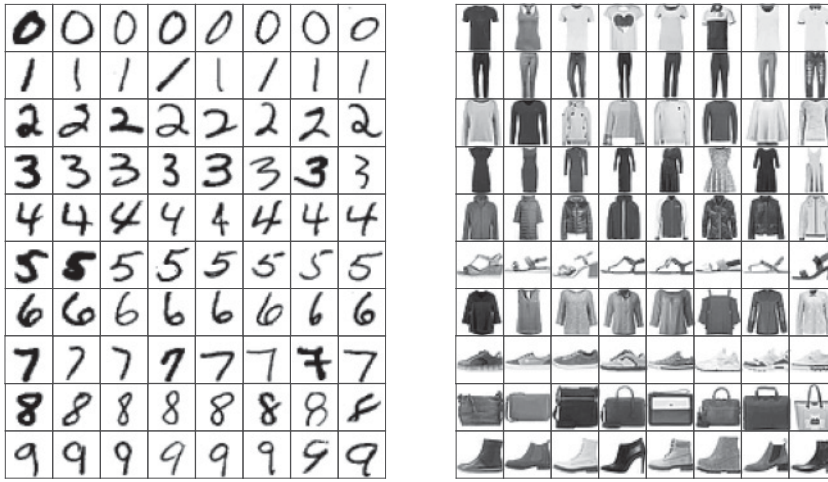

Figure 4: Prototypical training examples from different classes of digits (left) and fashion items (right). For each array, the image in the ith row and αth column shows the training example from the ith class whose probability $P(y = i|\mathbf{x})$ receives its largest contribution from the αth element of the ith capsule's latent variable $\mathbf{h}_i \in \mathbb{R}^d$. The results are shown for the model in Figure 1 with capsules of dimensionality $d = 8$.

3.4 Results for Classification. Our main experiments investigated the effect of the capsule dimensionality (d) on the model's performance as a classifier. We experimented with three types of architectures: (1) the multiclass-capsule architecture shown in Figure 1, in which we jointly train m capsules to recognize patterns from m different classes; (2) a one-versus-all architecture, in which we train m binary-capsule architectures in parallel and label inputs based on the most certain of their individual predictions; and (3) an all-versus-all architecture, in which we train $m(m-1)/2$ binary-capsule architectures in parallel and label inputs based on a majority vote. We note that these architectures employ different numbers of capsules, and therefore they have different numbers of weight matrices and learnable parameters even when all their capsules have the same dimensionality. In particular, the one-versus-all architecture has twice as many learnable parameters as the multiclass architecture, while the all-versus-all architecture has $m - 1$ times as many learnable parameters.

Figure 5 shows the results of these experiments. The red, blue, and black curves show, respectively, the test error rates of the multiclass, all-versus-one, and all-versus-all classifiers as a function of their capsule dimensionalities. For all these types of classifiers and for both data sets of images, the plots show that capsules with higher-dimensional latent variables are able

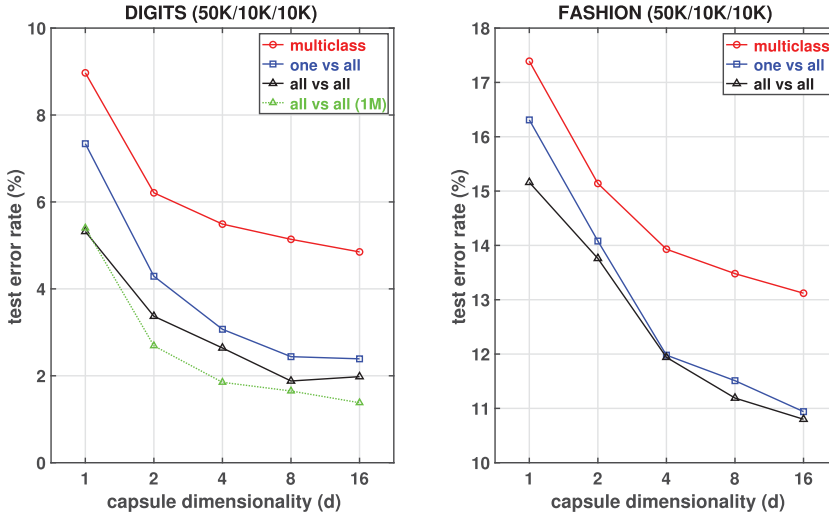

Figure 5: Effect of capsule dimensionality on test error rates for images of handwritten digits (left) and fashion items (right) from multiclass, one-versus-all, and all-versus-all classifiers.

to learn more accurate classifiers. For capsules of a fixed dimensionality, the plots also show that the architectures with more capsules (and hence more learnable parameters) are able to learn more accurate classifiers.

The data sets of MNIST hand-written digits and fashion items have been extensively benchmarked, so we can also compare these results to those of other classifiers (LeCun et al., 1998; Xiao et al., 2017). The one-dimensional ($d = 1$) capsule architectures in Figure 5 have test error rates comparable to those of other simple linear models; for example, Xiao et al. (2017) report test error rates of 8.3% and 15.8% for one-versus-all logistic regression on the data sets of digits and fashion items, respectively. Likewise, the models with higher-dimensional capsules offer similar improvements as other nonlinear approaches, such as k-nearest neighbors, random forests (Breiman, 2001), and fully connected neural nets; standardized implementations of these algorithms in Python's scikit-learn yield test error rates of roughly 2% to 3% on digits and 10% to 12% on fashion items (Xiao et al., 2017). However, none of the models in Figure 5 classify as well as the best-performing support vector machines (Cortes & Vapnik, 1995) or convolutional neural nets (LeCun et al., 1998). We believe that this gap in performance is mainly due to overfitting as opposed to an inability to learn sufficiently complex decision boundaries.

To test this hypothesis, we conducted another set of experiments where we trained the all-versus-all digit classifiers in Figure 5 on a much larger

data set of 1 million training images (Loosli, Cani, & Bottou, 2007). The 950,000 additional images for training were obtained from distortions (e.g., rotation, thickening) of the original MNIST training set. To facilitate a direct comparison with our previous results, we also used the same validation and test sets of 10,000 images. The results of these experiments are shown at the bottom (green) curve of the left panel of Figure 5. The results show that these all-versus-all capsule architectures have the capacity to learn better classifiers from larger amounts of training data. But even these classifiers are also plagued by overfitting: for example, the best of these classifiers (with capsule dimensionality $d = 16$) still exhibits a large gap between its test error rate (1.38%) on 10,000 images and its training error rate (0.0523%) on 1 million images.

4 Discussion

In this letter, we have introduced capsule regression as a higher-dimensional analog of simpler log-linear models such as logistic and softmax regression. We experimented with capsule regression in multiclass, one-versus-all, and all-versus-all architectures, and we showed that in all of these architectures, the model capacity grows in step with the capsule dimensionality. To learn these classifiers, we formulated capsule regression as a latent variable model, and we used the EM procedure to derive iterative least-squares updates for parameter estimation. Despite the squashing nonlinearity in our models, we showed that it remains tractable to perform exact inference over their continuous latent variables. One contribution of our work is to expand the family of tractable latent variable models that can learn meaningful distributed representations of high-dimensional inputs. Our work fits into a larger vision for probabilistic modeling: the "need to develop computationally-tractable representations of uncertainty" has been described as "one of the major open problems in classical AI" (Jordan, 2018).

Another contribution of our work is to highlight certain advantages of capsule regression for supervised learning with distributed representations. Traditional neural nets can learn more flexible decision boundaries than linear models, but this extra capacity comes at a cost: they involve much more complicated optimizations, with learning rates that must be tuned for convergence, and their internal representations (though effective for classification) can be fairly inscrutable. Models for capsule regression benefit equally from their distributed representations; as shown in Figure 5, with higher-dimensional capsules, these models acquire the capacity to learn increasingly accurate classifiers. But as shown in Figures 2 to 4, the internal representations of capsules also have a fairly interpretable structure, and as shown in section 2.3, these representations can be learned by simple least-squares updates. There are other plausible benefits to this structure that we have not yet explored. For example, we have only considered architectures in which all the capsules have the same dimensionality.

But these dimensionalities could be varied for classes that exhibit more diversity in their inputs and/or have larger numbers of training examples. It is harder to see how a traditional neural net could be purposefully adapted to reflect these forms of prior knowledge.

Several issues in our work deserve further study. The first is regularization: as previously mentioned, the models in section 2 are prone to overfitting even with early stopping on a validation set. It seems worthwhile to explore ℓ_1 and/or ℓ_2 regularization of the model parameters, as is common for other types of regression and to consider those forms of regularization—such as weight sharing (Nowlan & Hinton, 2004), dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), and reconstruction penalties (Sabour et al., 2017; Qin et al., 2020)—that have been widely used in deep learning. It should also help to incorporate some prior knowledge about images explicitly into the structure of the weight matrices, as is done in convolutional neural nets (LeCun et al., 1998).

Another issue is scaling: for very large data sets, it will be more practical to implement online or mini-batch versions of the updates in section 2.3. The least-squares form of equation 2.9 suggests certain possibilities beyond stochastic gradient descent for this purpose. There are, for example, passive-aggressive online algorithms for regression (Crammer, Dekel, Keshet, Shalev-Shwartz, & Singer, 2006) that could be adapted to this setting, with the posterior mean $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y]$ providing a target for the ith capsule's regression on the input $\mathbf{x}$ with label y. These posterior means also provide the sufficient statistics for faster incremental variants of the EM algorithm (Neal & Hinton, 1998).

A final issue is model building: Is it possible to extend the ideas for capsule regression in this letter to deeper and more sophisticated architectures? We have already seen that the one-versus-all and all-versus-all architectures in section 3.4 lead to more accurate classifiers than the basic model of capsule regression in Figure 1. But even these architectures are still too primitive for modeling (say) the part-whole relationships of complex visual objects; those relationships may need to be modeled more explicitly, as in the multilayer capsule networks (Sabour et al., 2017) whose squashing nonlinearities were the motivation for our own study. For deeper architectures with such nonlinearities, we believe that the methods in this letter may serve as useful building blocks. We started this letter by noting the promise of existing capsule networks, and it seems fitting, then, that we have come full circle. We conclude on the hopeful note that this work provides yet another bridge between the traditions of latent variable modeling and deep learning.

Appendix A: Supporting Calculations

In this appendix we present the more detailed calculations for inference that were omitted from section 2.2. In particular, in section A.1, we show how to calculate the interpolating coefficients $\lambda_0(\beta)$ and $\lambda_1(\beta)$, and in section A.2,

we show how to calculate the multidimensional integrals over the model's latent variables required for inference and learning.

A.1 Computing the Interpolating Coefficients. It is mostly straightforward to implement the EM algorithm in section 2, but some extra steps are needed to compute the interpolating coefficients, $\lambda_0(\beta)$ and $\lambda_1(\beta)$, defined in equations 2.13 and 2.14. In this section we show how to evaluate the one-dimensional integrals that appear in these definitions. We also show that $\lambda_0(\beta)$ and $\lambda_1(\beta)$ are monotonically increasing functions with values in the unit interval [0,1].

We begin by studying a family of integrals that include both equations 2.13 and 2.14 as a special case. In particular, for integer $s \geq 0$, let

$$I_s(\beta) = \beta e^{-\beta} \int_0^1 d\rho \, \rho^s \, e^{\rho\beta}. \tag{A.1}$$

With this definition, we see that $\lambda_0(\beta)$ in equation 2.13 corresponds to $I_s(\beta)$ with the choice $s = \frac{dm}{2}$, whereas $\lambda_1(\beta)$ in equation 2.14 corresponds to $I_s(\beta)$ with the choice $s = \frac{dm}{2} + 1$. Note that by restricting s to be integer valued, we are assuming that either the capsule dimensionality d or the number of labels m is an even integer. In practice, this is not an especially restrictive assumption because the capsule dimensionality d is a parameter of our own choosing.

Suppose, then, that s in equation A.1 is integer valued. In that case, we can obtain a closed form (of sorts) for this integral by the method of differentiating under the integral sign:

$$I_s(\beta) = \beta e^{-\beta} \left(\frac{\partial}{\partial \beta}\right)^s \int_0^1 d\rho \, e^{\rho\beta} = \beta e^{-\beta} \left(\frac{\partial}{\partial \beta}\right)^s \left(\frac{e^\beta - 1}{\beta}\right). \tag{A.2}$$

Thus, we have $I_0(\beta) = 1 - e^{-\beta}$ for the base case $s = 0$ and $I_1(\beta) = 1 - \frac{1-e^{-\beta}}{\beta}$ for the case $s = 1$. Equation A.2 shows that the integral can be evaluated analytically, but this formula rapidly becomes unwieldy even for moderate values of s. Thus, another approach is required.

A better strategy is to integrate the right-hand side of equation A.1 by parts. This yields the recursive formula

$$I_s(\beta) = 1 - \frac{s}{\beta} I_{s-1}(\beta). \tag{A.3}$$

In principle, this recursion can be used to compute $I_s(\beta)$, starting from the base case at $I_0(\beta) = 1 - e^{-\beta}$ and iterating equation A.3 for as many steps as needed. In practice, however, a problem arises for large values of s or small values of β (which typically occur in the early stages of learning). In

this case, the recursion becomes numerically unstable; in particular, if early steps of the recursion have already introduced some numerical error, then the right-hand side of equation A.3 will tend to amplify this error whenever $\beta < s$. In fact, equation A.3 is a textbook example of numerical instability (Hill, 2016).

To compute $I_s(\beta)$ when $\beta \leq s$, we adopt an equally well-known solution to this problem. Our first step is to derive fairly tight bounds on $I_s(\beta)$. It is clear from equation A.1 that $I_s(\beta) \leq I_{s+1}(\beta)$ for all β. We can obtain lower and upper bounds on $I_s(\beta)$ by substituting this inequality into equation A.3. In this way, for $s \geq 1$, we find that

$$\frac{\beta}{\beta + s + 1} \;\leq\; I_s(\beta) \;\leq\; \frac{\beta}{\beta + s}. \tag{A.4}$$

As an aside, we note that these bounds imply $0 \leq I_s(\beta) < 1$ for all values of β. It follows at once, as claimed in section 2.2, that the interpolating coefficients $\lambda_0(\beta)$ and $\lambda_1(\beta)$ also lie in the unit interval.

The bounds in equation A.4 provide a starting point to compute $I_s(\beta)$ when $\beta \leq s$. To see this, we invert equation A.3 to obtain the backward recursion

$$I_s(\beta) = \frac{\beta}{s+1}\big(1 - I_{s+1}(\beta)\big). \tag{A.5}$$

The backward recursion in equation A.5 is numerically stable in exactly the opposite regime as the forward recursion in equation A.3; we can therefore use it to compute $I_s(\beta)$ when $\beta \leq s$. We do this in two steps. First, we start by computing lower and upper bounds on $I_{s+k}(\beta)$ for some sufficiently large value of k. Then we use the backward recursion k times to transform our upper and lower bounds on $I_{s+k}(\beta)$ into lower and upper bounds on $I_s(\beta)$. Since by assumption $\beta \leq s$, these bounds get tighter at every step of the recursion; it follows that by choosing k to be sufficiently large, we can compute $I_s(\beta)$ to whatever numerical accuracy is desired. As a practical matter, in our experiments, we set $k = 64$ and verified for each evaluation that $I_s(\beta)$ was computed to within a factor of 1 ± 10^{-4}.

To summarize, then, we use the forward recursion in equation A.3 to compute $I_s(\beta)$ when $\beta > s$, and we use the backward recursion in equation A.5 to compute $I_s(\beta)$ when $\beta \leq s$. In practice, we only invoke these recursions to compute $\lambda_0(\beta)$, setting $s = \frac{dm}{2}$, because the forward recursion can then be used to compute $\lambda_1(\beta)$ with $s = \frac{dm}{2} + 1$. Note that the value of $s = \frac{dm}{2}$ is fixed in advance by the capsule architecture. Thus, before the outset of learning, it may also be possible to compile lookup tables for $I_s(\beta)$ and use interpolation schemes for even faster inference. However, we have not pursued that approach here.

We claimed in section 2.2 that the interpolating coefficients $\lambda_0(\beta)$ and $\lambda_1(\beta)$ were monotonically increasing functions of β. To prove this, we note that

$$I_s'(\beta) = \frac{d}{d\beta}\left[\beta e^{-\beta}\left(\frac{\partial}{\partial\beta}\right)^s\left(\frac{e^\beta - 1}{\beta}\right)\right], \tag{A.6}$$

$$= \left(\frac{1}{\beta} - 1\right) I_s(\beta) + I_{s+1}(\beta), \tag{A.7}$$

$$= 1 - \left(\frac{\beta + s}{\beta}\right) I_s(\beta) \tag{A.8}$$

$$\geq 0, \tag{A.9}$$

where in the third line we have used equation A.5 to express $I_{s+1}(\beta)$ in terms of $I_s(\beta)$, and in the last line, we have appealed to the upper bound in equation A.4. Since $\lambda_0(\beta) = I_{\frac{dm}{2}}(\beta)$ and $\lambda_1(\beta) = I_{\frac{dm}{2}+1}(\beta)$, this proves the claim.

A.2 Integrating over the Model's Latent Variables. In this section we show how to calculate the conditional probability $P(y = j|\mathbf{x})$ in equation 2.4 and the posterior mean $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ in equation 2.8. Both calculations involve multidimensional integrals over all of the model's latent variables, which we denote collectively by $\mathbf{h} \in \mathbb{R}^D$.

We begin by considering the simpler but closely related integral given by

$$J = \int_{\mathbf{h}\in\mathbb{R}^D} \frac{1}{(2\pi\sigma^2)^{D/2}}\, e^{-\frac{\|\mathbf{h}-\boldsymbol{\mu}\|^2}{2\sigma^2}}\, \frac{\sigma^2}{\|\mathbf{h}\|^2}. \tag{A.10}$$

This integral is nontrivial due to the last term in the integrand, where the squared magnitude $\|\mathbf{h}\|^2$ appears in the denominator. We include the factor of σ^2 in the numerator of this term so that J is a dimensionless quantity. For now, we also assume that $D > 2$, since for $D \leq 2$ the integral in equation A.10 is not well-defined. After evaluating this integral, we will see that the results for $P(y = j|\mathbf{x})$ and $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ follow from fairly mechanical calculations.

We can evaluate the integral in equation A.10 by a series of simple transformations. First, we use the integral identity

$$\frac{1}{\|\mathbf{h}\|^2} = \int_0^\infty \frac{d\eta}{\eta^3}\, e^{-\frac{\|\mathbf{h}\|^2}{2\eta^2}} \tag{A.11}$$

to lift the denominator of $\|\mathbf{h}\|^2$ in equation A.10 into the exponent. This identity introduces an auxiliary variable η over which we must also integrate to

obtain J. But now we can reverse the order of integration and perform the resulting (gaussian) integral over $\mathbf{h}$. In this way we find that

$$J = \sigma^2 e^{-\frac{\|\boldsymbol{\mu}\|^2}{2\sigma^2}} \int_0^\infty \frac{d\eta}{\eta^3} \left(\frac{\eta^2}{\sigma^2+\eta^2}\right)^{\frac{D}{2}} e^{\frac{\eta^2\|\boldsymbol{\mu}\|^2}{2(\sigma^2+\eta^2)}}, \tag{A.12}$$

thus replacing the multidimensional integral in equation A.10 by a much simpler one-dimensional integral. Next we make the change of variables $\eta = \sigma \tan\theta$, which yields

$$J = e^{-\frac{\|\boldsymbol{\mu}\|^2}{2\sigma^2}} \int_0^{\pi/2} d\theta \, \cos\theta \, (\sin\theta)^{D-3} \, e^{\frac{\|\boldsymbol{\mu}\|^2 \sin^2\theta}{2\sigma^2}}. \tag{A.13}$$

A more recognizable form emerges from one further change of variables $\rho = \sin^2\theta$. In this way we find

$$J = \frac{1}{2} \int_0^1 d\rho \, \rho^{\frac{D}{2}-2} \, e^{-\frac{(1-\rho)\|\boldsymbol{\mu}\|^2}{2\sigma^2}}. \tag{A.14}$$

At this point, readers may already glimpse the origin of the integrals in equations 2.13 and 2.14, in terms of which we have expressed our results for $P(y = j|\mathbf{x})$ and $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$.

To evaluate the integrals in equations 2.4 and 2.8, we need one more intermediate result. Let $\mathbf{h} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I})$ be normally distributed with mean $\boldsymbol{\mu} \in \mathbb{R}^D$, and let

$$p_\alpha = \mathrm{E}\left[\frac{h_\alpha^2}{\|\mathbf{h}\|^2}\right] \tag{A.15}$$

denote the expected value of the proportion of its squared magnitude from its αth component. Note that $\sum_\alpha p_\alpha = 1$ so that we can view p_α as a distribution. To compute p_α, we must evaluate the integral

$$p_\alpha = \int_{\mathbf{h}\in\mathbb{R}^D} \frac{1}{(2\pi\sigma^2)^{D/2}} \, e^{-\frac{\|\mathbf{h}-\boldsymbol{\mu}\|^2}{2\sigma^2}} \, \frac{h_\alpha^2}{\|\mathbf{h}\|^2}. \tag{A.16}$$

The integral in equation A.16 differs from the one in equation A.10 by the numerator of the rightmost term in the integrand. But we can relate these two integrals by the method of differentiating under the integral sign:

$$p_\alpha = \sigma^2 \frac{\partial^2 J}{\partial \mu_\alpha^2} + 2\mu_\alpha \frac{\partial J}{\partial \mu_\alpha} + \left(1 + \frac{\mu_\alpha^2}{\sigma^2}\right) J. \tag{A.17}$$

Using equation A.17, we can evaluate the integral for p_α by the more mechanical process of differentiation. We proceed by substituting the result in equation A.14 into the right-hand side of equation A.17. After simplifying, we find

$$p_\alpha = \frac{1}{D} + \left(\frac{\mu_\alpha^2}{\|\boldsymbol{\mu}\|^2} - \frac{1}{D} \right) \cdot \frac{\|\boldsymbol{\mu}\|^2}{2\sigma^2} \cdot \int_0^1 d\rho \, \rho^{\frac{D}{2}} \, e^{-\frac{(1-\rho)\|\boldsymbol{\mu}\|^2}{2\sigma^2}}. \tag{A.18}$$

By summing both sides of equation A.18 over α, it is also easy to verify (as required) that $\sum_\alpha p_\alpha = 1$.

With the result in equation A.18, we can now derive the expression for $P(y = j|\mathbf{x})$ in equation 2.16. Suppose that the dimensionality D of the integral in equation A.16 is equal to the product dm, where d is the capsule dimensionality and m is the number of labels. Then we can write $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m)$ where each $\mathbf{h}_i \in \mathbb{R}^d$, and we can set $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_m)$ where $\boldsymbol{\mu}_i = \mathbf{W}_i\mathbf{x}$. With this notation, we can rewrite the marginalization in equation 2.4 as

$$P(y = j|\mathbf{x}) = \int_{\mathbf{h} \in \mathbb{R}^{dm}} \left[\prod_i \frac{e^{-\frac{1}{2\sigma^2}\|\mathbf{h}_i - \boldsymbol{\mu}_i\|^2}}{(2\pi\sigma^2)^{d/2}} \right] \frac{\|\mathbf{h}_j\|^2}{\|\mathbf{h}\|^2}. \tag{A.19}$$

Comparing equations A.16 and A.19, we see that the latter is obtained by summing over d elements of the former—namely, the d elements that select the vector $\mathbf{h}_j \in \mathbb{R}^d$ from the larger vector $\mathbf{h} \in \mathbb{R}^{dm}$. Performing this sum, we recover the result in equation 2.16.

Finally, we show how to derive the result in equation 2.19 for the posterior means $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$. Again, using the shorthand notation $\boldsymbol{\mu}_i = \mathbf{W}_i\mathbf{x}$, we can rewrite the calculations in equations 2.7 and 2.8 as

$$\mathrm{E}[\mathbf{h}_i|y = j, \mathbf{x}] = \frac{\int_{\mathbf{h} \in \mathbb{R}^{dm}} \left[\prod_i e^{-\frac{1}{2\sigma^2}\|\mathbf{h}_i - \boldsymbol{\mu}_i\|^2} \right] \frac{\|\mathbf{h}_j\|^2}{\|\mathbf{h}\|^2} \, \mathbf{h}_i}{\int_{\mathbf{h} \in \mathbb{R}^{dm}} \left[\prod_i e^{-\frac{1}{2\sigma^2}\|\mathbf{h}_i - \boldsymbol{\mu}_i\|^2} \right] \frac{\|\mathbf{h}_j\|^2}{\|\mathbf{h}\|^2}}. \tag{A.20}$$

To evaluate this expectation, we observe that it can be related to the marginalization in equation A.19 by the method of differentiating under the integral sign. In particular, we have that

$$\mathrm{E}[\mathbf{h}_i|y = j, \mathbf{x}] = \boldsymbol{\mu}_i + \sigma^2 \frac{\partial}{\partial \boldsymbol{\mu}_i} \big[\log P(y = j|\mathbf{x}) \big]. \tag{A.21}$$

We proceed by substituting the result for $P(y = j|\mathbf{x})$ in equation 2.16 into the right-hand side of equation A.21. After some tedious but straightforward calculation, this yields the result for $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ in equation 2.19.

As noted earlier, the above derivations assume that $D > 2$, since the integral in equation A.10 is divergent for $D = 2$. But the case $D = 2$ is also of interest: it arises for binary classification ($m = 2$) with scalar capsules ($d = 1$). In this case, we can replace equation A.10 with the ε-regularized integral,

$$J_\epsilon = \int_{\mathbf{h}\in\mathbb{R}^D} \frac{1}{(2\pi\sigma^2)^{D/2}}\, e^{-\frac{\|\mathbf{h}-\boldsymbol{\mu}\|^2}{2\sigma^2}}\, \frac{\sigma^2}{\|\mathbf{h}\|^2+\varepsilon^2}. \tag{A.22}$$

We can then follow the same steps as before to obtain expressions that reduce to the conditional probability $P(y = j|\mathbf{x})$ and posterior mean $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ in the limit $\varepsilon \to 0$. These limits are well defined even in the case $D = 2$, and in this case, they also yield the results for $P(y = j|\mathbf{x})$ in equation 2.16 and $\mathrm{E}[\mathbf{h}_i|\mathbf{x}, y = j]$ in equation 2.19.

Appendix B: Supporting Experiments

We obtained the results in section 3 with the modified update in equation 2.27 and the subspace initialization in equation 2.28. Most of these results were devoted to comparing models that were identically trained but had different values of the capsule dimensionality, d. To make meaningful comparisons, though, it was first necessary to standardize how we initialized the models and which updates we used to train them. In this appendix, we describe some of the preliminary experiments that informed these choices. In particular, section B.1 explores the effect of different updates (with thresholding and/or momentum), and section B.2 explores the effect of random versus subspace initializations.

B.1 Effects of Momentum and Thresholding. First we consider the effect of the hyperparameters ν and γ on the course of learning. We used these hyperparameters to modify the EM update in equation 2.24, and these modifications led to the variants in equations 2.26 and 2.27. In this section, we consider how these modifications affect the log-conditional likelihood $\mathcal{L}(\mathbf{W}, \sigma^2)$ in equation 2.6 and the number of misclassified examples. To understand these effects, we experimented on the model in Figure 1 with capsule dimensionality $d = 4$. We trained this model on both data sets in four different ways: using the EM update ($\gamma = \nu = 0$) in equation 2.24, using the thresholded update ($\gamma = 0, \nu = 0.8$) in equation 2.26, and using the momentum update ($\gamma = 0.9$) in equation 2.27 both with thresholding ($\nu = 0.8$) and without thresholding ($\nu = 0$). For each experiment, we initialized the model's weight matrices by equation 2.28, and we reestimated them for 1000 iterations on all 60,000 training examples.

The panels in Figure 6 plot the normalized log-loss, computed as $-\frac{1}{n\log m}\cdot\mathcal{L}(\mathbf{W}, \sigma^2)$, versus the number of iterations of learning on images of hand-written digits and fashion items. In these plots, the blue curves

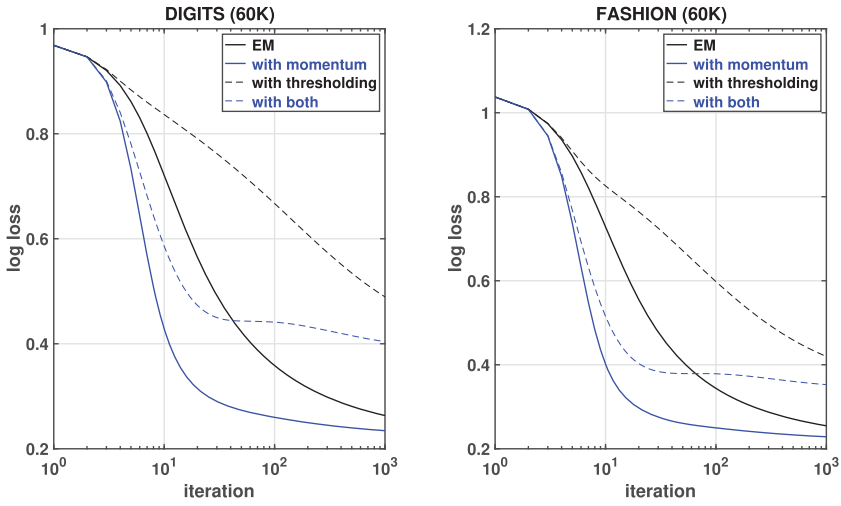

Figure 6: Effects of momentum ($\gamma = 0.9$) and thresholding ($\nu = 0.8$) on convergence of the EM algorithm for the model in Figure 1 with capsule dimensionality $d = 4$.

indicate the updates with momentum, and the dashed curves indicate the updates with thresholding. Two trends are clear: first, the blue curves lie below the black curves, indicating that the log-loss decreases more rapidly when a momentum term is included in the update; second, the dashed curves lie above the solid curves, indicating that the log-loss converges to a higher (worse) value with the thresholded update. The latter result is to be expected, as the thresholded update in equation 2.26 attempts to reduce the log-loss only on a subset of training examples. Though not reproduced here, we observed these trends consistently across models of many different cardinalities (m) and capsule dimensionalities (d).

The panels in Figure 7 plot the classification error rate on the training sets of digit and fashion images versus the number of iterations of learning. Two trends are clear in these plots as well. The first trend is the same as before: the blue curves lie below the black curves, indicating that the error rate decreases more rapidly when a momentum term is included in the update. But the second trend is different: in these plots, the dashed curves lie *below* the solid curves, with a significant gap emerging after fewer than 10 iterations and growing thereafter. In particular, on both data sets, we see that the error rates converge to a significantly lower value with the thresholded update. In practice, the thresholded update appears to trade the worse likelihoods in Figure 6 for the lower error rates in Figure 7. Again, though not reproduced here, we observed these trends consistently across models of many different cardinalities (m) and capsule dimensionalities (d).

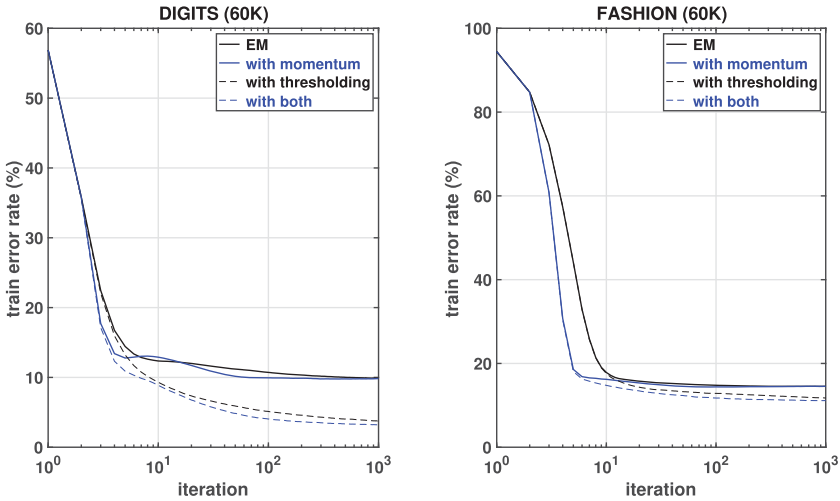

Figure 7: Effects of momentum ($\gamma = 0.9$) and thresholding ($\nu = 0.8$) on convergence of the EM algorithm for the model in Figure 1 with capsule dimensionality $d = 4$.

For the models in section 3, we were ultimately more interested in minimizing their error rates as classifiers than maximizing their log-conditional likelihoods. To be sure, the likelihood provides a useful surrogate for the error rate (as the former is differentiable in the model parameters whereas the latter is not). But for our main experiments—based on the above results—we adopted the modified update in equation 2.27 with the tunable hyperparameters ν and γ. Moreover, as shown in section 3, these hyperparameters did not require elaborate tuning to be effective in practice.

B.2 Effects of Initialization. We also compared the effects of different initializations. For these comparisons, we experimented with the same $d = 4$ model as in the previous section, but in addition, we trained models whose weight matrices were initialized by different levels of zero-mean gaussian noise. In particular, for each noise level, we generated 20 different initializations and trained the resulting models for 1000 iterations. Then for each of these models, we recorded the lowest classification error rate that it achieved on the 60,000 training examples over the course of learning.

Figure 8 shows the results of these experiments as a box plot over these 20 different initializations. For comparison, we also show the result from the model with the subspace initialization in equation 2.28. As expected, there is some spread in the results obtained with the random initializations. However, we found that none of the randomly initialized models (on either data set) learned a classifier as accurate as the model with subspace

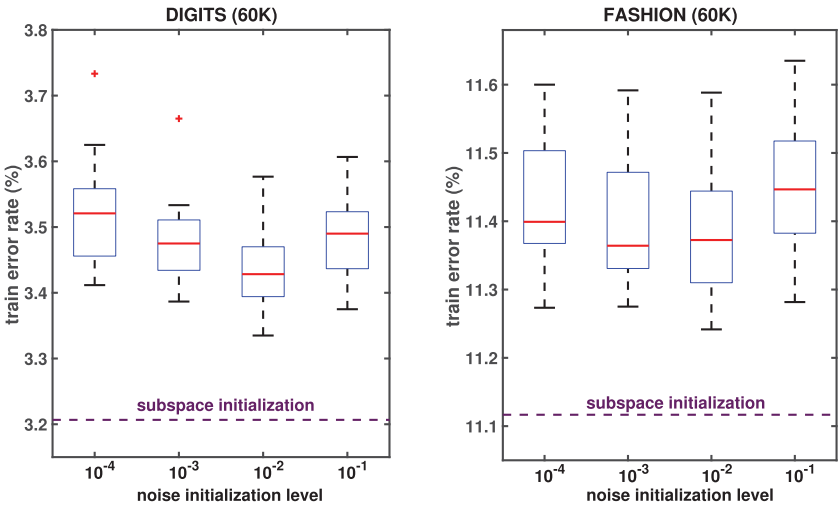

Figure 8: Comparison of subspace versus random initializations.

initialization. For these types of experiments, it is of course necessary to train a large ensemble of randomly initialized models, and therefore we cannot claim to have observed this behavior across a large number of models of different types and sizes. We also note that Figure 8 reports error rates on the training examples, not the test examples, so there may be some benefits of random initialization for preventing overfitting. Nevertheless, based on the above results, we adopted the subspace initialization in equation 2.28 for all of our experiments in section 3.

Acknowledgments

I am grateful to the anonymous reviewer whose suggestions improved this letter in many ways.

References

Ahmed, K., & Torresani, L. (2019). STAR-Caps: Capsule networks with straight-through attentive routing. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché, F. Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems, 32* (pp. 9101–9110). Red Hook. NY: Curran.

Bahadori, M. T. (2018). *Spectral capsule networks*. Workshop at the International Conference on Learning Representations (ICLR-18). http://openreview.net/pdf?id=HJuMyYPaM

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory* (pp. 144–152). New York: ACM Press.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, *7*, 551–585.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*(1), 1–38.

Duarte, K., Rawat, Y., & Shah, M. (2018). VideoCapsuleNet: A simplified network for action detection. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems, 31* (pp. 7610–7619). Red Hook, NY: Curran.

Ghahramani, Z., & Hinton, G. E. (1996). *The EM algorithm for mixtures of factor analyzers* (Technical Report CRG-TR-96-1). Department of Computer Science, University of Toronto.

Hahn, T., Pyeon, M., & Kim, G. (2019). Self-routing capsule networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, W. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems, 32* (pp. 7658–7667). Red Hook, NY: Curran.

Hill, C. (2016). *Learning scientific programming with Python*. Cambridge: Cambridge University Press.

Hinton, G. E., Krizhevsky, A., & Wang, S. D. (2011). Transforming auto-encoders. In *Proceedings of the International Conference on Artificial Neural Networks* (pp. 44–51). Berlin: Springer.

Hinton, G. E., Sabour, S., & Frosst, N. (2018). Matrix capsules with EM routing. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net.

Jamshidian, M., & Jennrich, R. I. (1993). Conjugate gradient acceleration of the EM algorithm. *Journal of the American Statistical Association*, *88*, 221–228.

Jamshidian, M., & Jennrich, R. I. (1997). Acceleration of the EM algorithm by using quasi-Newton methods. *Journal of the Royal Statistical Society, Series B*, *59*, 569–587.

Jeong, T., Lee, Y., & Kim, H. (2019). Ladder capsule network. In *Proceedings of the 36th International Conference on Machine Learning* (pp. 3071–3079). PMLR.

Jordan, M. I. (2018). *Artificial intelligence—the revolution that hasn't happened yet.* https://bit.ly/2XO7rhQ.

Kosiorek, A., Sabour, S., Teh, Y. W., & Hinton, G. E. (2019). Stacked capsule autoencoders. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems, 32* (pp. 15512–15522). Red Hook, NY: Curran.

Lange, K. L. (1995). A quasi-Newton acceleration of the EM algorithm. *Statistica Sinica*, *5*, 1–18.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, *86*(11), 2278–2324.

Liu, C., & Rubin, D. B. (1994). The ECME algorithm: A simple extension of EM and ECM with faster monotone convergence. *Biometrika*, *81*, 633–648.

Liu, C., Rubin, D. B., & Wu, Y. N. (1998). Parameter expansion to accelerate EM: The PX-EM algorithm. *Biometrika*, *85*, 755–770.

Lloyd, S. P. (1957). *Least squares quantization in PCM*. Bell Labs Technical Report RR-5497.

Loosli, G., Cani, S., & Bottou, L. (2007). Training invariant support vector machines using selective sampling. In L. Bottou, O. Chapelle, D. DeCoste, & J. Weston (Eds.), *Large scale kernel machines* (pp. 301–320). Cambridge, MA: MIT Press.

Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods* (Technical Report CRG-TR-93-1). Department of Computer Science, University of Toronto.

Neal, R. M., & Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan (Ed.), *Learning in graphical models* (pp. 355–368). Amsterdam: Kluwer.

Nowlan, S. J., & Hinton, G. E. (2004). Simplifying neural networks by soft weight-sharing. *Neural Computation*, *4*(4), 473–493.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, *12*, 145–151.

Qin, Y., Frosst, N., Sabour, S., Raffel, C., Cottrell, G., & Hinton, G. E. (2020). Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net.

Rubin, D. B., & Thayer, D. T. (1982). EM algorithms for ML factor analysis. *Psychometrika*, *47*(1), 69–76.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing* (1:318–362). Cambridge, MA: MIT Press.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems, 30* (pp. 3856–3866). Red Hook, NY: Curran.

Salakhutdinov, R. R., Roweis, S. T., & Ghahramani, Z. (2003). Optimization with EM and expectation-conjugate-gradient. In *Proceedings of the 20th International Conference on Machine Learning* (pp. 672–679). Palo Alto, CA: AAAI Press.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Tsai, Y.-H. H., Srivastava, N., Goh, H., & Salakhutdinov, R. (2020). Capsules with inverted dot-product attention routing. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net.

Varadhan, R., & Roland, C. (2008). Simple and globally convergent methods for accelerating the convergence of any EM algorithm. *Scandinavian Journal of Statistics*, *35*, 335–353.

Venkataraman, S. R., Balasubramanian, S., & Sarma, R. R. (2020). Building deep equivariant capsule networks. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, *1*(1–2), 1–305.

Wang, D., & Liu, Q. (2018). *An optimization view on dynamic routing between capsules*. Workshop at the International Conference on Learning Representations. OpenReview.net.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms.* arXiv:1708.07747.

Yu, Y. (2012). Monotonically overrelaxed EM algorithms. *Journal of Computational and Graphical Statistics*, *21*, 518–537.

Zhang, L., Edraki, M., & Qi, G.-J. (2018). CapProNet: Deep feature learning via orthogonal projections onto capsule subspaces. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems, 31* (pp. 5814–5823). Red Hook, NY: Curran.

Received June 3, 2020; accepted July 31, 2020.