Bernoulli model with deformations

Gustav Larsson

August 31, 2012

1 Introduction

MNIST is a dataset of well-posed (centered) and clean (little noise) digits of 10 classes. The task is to determine which class an image belongs to, by training several prototypes of each class and comparing an image to the prototypes. The image is converted to binary features, which are assumed to be drawn independently from Bernoulli distributions (the prototype). The prototypes are also allowed to deform using a wavelet basis to match the image better, to the cost of a Gaussian prior over the parameters of the deformation.

2 Method

2.1 Definitions

This document largely follows the same notation as in Yali Amit's book (Chapter 5). Here is a condensed form:

Definition 1. *Let Y be the set of all classes; K the set of all coefficient indices for a wavelet transform; $Q = \{1, 2\}$ the two axes in an image; J the set of all directed edge features for each pixel.*

Definition 2. *$L \in Z^d$ is a fixed image grid of d points $z \in Z = \mathbb{R}^2$, making up the pixel locations of an undeformed image.*

Definition 3. *Let the family of images, converted to edge features, be $\mathcal{X} = \{(X_1, \ldots, X_{|J|}) \mid X_j : L \to \{0, 1\}\}$, where the function value represents the presence of an edge.*

Definition 4. *Let the family of prototypes be $\mathcal{F} = \{(F_1, \ldots, F_{|J|}) \mid F_j : Z \to [0, 1]\}$, where the function value represents a Bernoulli probability. Notice that the functions are defined on the entire space Z, and not just the image grid L.*

2.2 Data model

Now, assume that each edge feature in the image $X \in \mathcal{X}$ was generated from a deformed prototype image $F \in \mathcal{F}$. The deformation is parameterized by u as

$$U(x) = (\Psi^{-1}(u^{(1)}), \Psi^{-1}(u^{(2)})),$$

where Ψ denotes a wavelet transform, and $u^{(q)}$ all necessary coefficients for axis q. We have

$$\Psi^{-1}(u) = \sum_{k \in K} u_k \psi_k(x),$$

where ψ_k is the wavelet basis functions associated with $k \in K$ the set of coefficients.

Now, the deformation and the image are assumed to be drawn from the following distributions:

$$u_k^{(q)} \sim \mathcal{N}(\mu_k^{(q)}, (\lambda_k^{(q)})^{-1}), \qquad q \in Q, k \in K, \tag{1}$$

$$X_j(x) \sim \text{Bern}(F_j(\tilde{x})), \qquad x \in L, j \in J, \tag{2}$$

where we introduce the short-hand $\tilde{x} = x + U(x)$. The hyperparameters μ and λ are specific to the prototype F.

We now set a cost function to the negative posterior, ignoring any additive constants, and get (not be confused with the set J)

$$\begin{aligned} J(u) =& \frac{1}{2} \sum_{q \in Q} \sum_{k \in K} \lambda_k^{(q)} (u_k^{(q)} - \mu_k^{(q)})^2 - \\ & \sum_{j \in J} \sum_{x \in L} (X_j(x) \log(F_j(\tilde{x})) + (1 - X_j(x)) \log(1 - F_j(\tilde{x}))) \end{aligned}$$

Taking partial derivatives of this we get

$$\begin{aligned} \frac{\partial J(u)}{\partial u_k^{(q)}} =& \lambda_k^{(q)} (u_k^{(q)} - \mu_k^{(q)}) - \\ & \sum_{j \in J} \sum_{x \in L} \left(\frac{X_j(x)}{F_j(\tilde{x})} - \frac{1 - X_j(x)}{1 - F_j(\tilde{x})} \right) \partial_q F_j(\tilde{x}) \psi_k(x), \end{aligned}$$

where ∂_q indicates the partial derivative along the q axis.

2.3 Minimization

We define the best deformation as

$$\hat{u} = \arg\min_u J(u).$$

This is determined using a Quasi-Newton search algorithm (specifically BFGS), which uses repeated evaluations of $J(u)$ and $\nabla J(u)$ to find the minimum. The process is done in a coarse-to-fine manner, by initially using S_0 coefficient levels, letting them converge, and then increase the number of levels to and including S.

The minimum cost $J(\hat{u})$ now gives us a value that we can compare between prototypes F, to determine the most likely one.

2.4 Classification

We shall denote the parameters of a prototype as $\theta = (F, \mu, \lambda, y)$, where $y \in Y$, denotes the class that the prototype is representing. Let Θ denote the set of all such parameter tuples, and allow multiple entries with the same class y.

It is appropriate to now write $J_\theta(u)$ as the cost function associated with θ and the image X.

Classification, i.e. determining the class of X, denoted $\hat{y}$, is done by taking

$$\hat{\theta} = \arg\min_{\theta \in \Theta} J_\theta(\hat{u}),$$

which of course contains $\hat{y}$.

2.5 Learning

Building Θ is done by taking N images of each class and running a Bernoulli mixture model for each with M components. The templates of the mixture model constitutes the prototypes F. For stability, allow only $F_j(x) \in [\delta, 1-\delta]$, for some small value $\delta > 0$.

The domain of the functions F_j is extended to the entire Z by bilinear interpolation. Values outside the grid are given the closest edge value in F_j.

The gradient ∇F_j is calculated by central differences with sample distance 1 in the middle and the first difference on the boundaries. Values outside L are evaluated again by bilinear interpolation, however this time the fill value outside the grid is 0.

The hyperparameters of the prior, μ and λ, are learned as follows. Each template is associated with a set of original training images (mixture component affinities are assumed to be degenerate for all images, meaning each image has contributed to only one mixture component). For each image, $\hat{u}$ is determined by the method above, using a predetermined and fixed μ_0 and λ_0. The values μ and λ associated with this template is now extracted as the mean and precision (inverse variance) of those $\hat{u}$ values.

The values μ_0 are set to $\mathbf{0}$, since the template is expected to match well with the identity deformation. The values λ_0 are set as following for both axes (omitting (q) from the notation)

$$\lambda_k = \lambda_{(\alpha, s, l_1, l_2)} = \eta 2^{\rho s}$$

where η is a fairly arbitrarily scaled penalty term and $\rho > 0$ is a smoothening term. The coefficient index k breaks up to $\alpha \in \{HG, GH, HH\}$, the dilations $0 \leq s \leq S$ and the translations $0 \leq l_1, l_2 < 2^s$. The value $s = 0$ represents the scaling function (as opposed to the wavelet functions) and thus has only one α value. The value S dictates how many levels of wavelet functions to use, which is determined beforehand. This means that setting $S = 0$, only the scaling function is used.

2.6 Experiments

Classification can be done without deformation, using only the mixture model (NoDeform) or with deformations (Deform). As a speed optimization, you can also employ deformation only if there are other prototypes within a factor α of the minimum cost without deformations, in which case all those become contendors and are deformed (SelDeform).

The MNIST dataset consists of images of size 28×28 with gray-level intensities. The images are zero-padded to 32×32 to work better with the wavelet transform, and converted to binary features ($J = 8$). The directed edge features are described in Yali Amit's book (Chapter 5.4) and we use $k = 5$ with feature inflation (the 8 neighbors of an edge are also reported as edges).

2.7 Conjugate prior

Learning λ values runs the risk of overfitting or getting the wrong scale since the likelihood term is underrated (because pixels are falsely assumed to be independent). In this section we investigate the effects of putting a Gamma prior over λ in the Gaussian distribution in (1). Since we have several values of λ, we will actually have a Gamma prior over each of those values. Instead of controlling the hyperparameters a and b, we will decide a reasonable value for what the most probable value of λ should be, and then adjust the variance by setting b. The relationship is given by deriving the Gamma distribution with regards to λ, setting the expression to zero. This gives

$$\arg\max_{\lambda} \mathrm{Gam}(\lambda | a, b) = \frac{a-1}{b}. \tag{3}$$

We want to set λ through η and ρ as described earlier, and then adjust b as needed, this gives us

$$a = b\eta 2^{\rho s} + 1$$

Omitting some calculations, this gives us expressions of a_N and b_N for the posterior distribution. From that we extract λ_N, the maximum of our posterior distribution according to (3) as

$$\lambda_N = \frac{b_0 \lambda_0 + \frac{N}{2}}{b_0 + \frac{N}{2\lambda_{ML}}},$$

where $\lambda_{ML} = \sigma_{ML}^{-2}$ is the sample precision and N the sample size.

3 Preliminary results

All experiment here are on subsets of the MNIST dataset, so the number of classes is $|Y| = 10$. The number of training samples is given by $N \cdot |Y|$. The tables contain the following information:

Method	α	Miss rate	$F_{\rightarrow}T$	$T_{\rightarrow}F$	Deformed	#cont.	F undef.
NoDeform	1.0	7.50%	-	-	-	-	-
SelDeform	1.1	4.80%	3.40%	0.70%	17.90%	3.70	1.90%
SelDeform	1.2	3.90%	4.70%	1.10%	35.90%	5.60	0.40%
SelDeform	1.3	3.00%	5.50%	1.00%	52.20%	8.22	0.10%
SelDeform	1.4	2.90%	5.60%	1.00%	67.40%	11.55	0.00%
SelDeform	1.5	2.90%	5.60%	1.00%	78.00%	15.58	0.00%
Deform	∞	2.90%	5.60%	1.00%	100.00%	50.00	0.00%

Table 1: Shows improvement of deformations and influence of α.

$F_{\rightarrow}T$ Denote how many classifications that were True with the mixture model alone, but turned False as a result of deformations.

$T_{\rightarrow}F$ Analogous to $F_{\rightarrow}T$.

Deformed Percentage of test cases that employed deformations.

#cont. Average number of deformations made for all cases where deformations were employed.

F undef. Percentage of test cases that were classified F and did not use deformations (meaning, α might be too small if this is greater than zero).

3.1 Preliminary results 1 and influence of α

Setting $N = 100, M = 5, \eta = 100, \rho = 1, S_0 = 1, S = 3, \delta = 0.05$ and using Daubechies D8 wavelets for Ψ. This trial was tested on 1000 samples from the training set (disjoint from the subset used for training). We tried several values of α. Results in Tab. 1.

3.2 Conjugate prior

Tried $\eta = 100, \rho = 1$, which was used in Trial 1 above. Also tried $\eta = 10, \rho = 2.7$, which was very roughly chosen to be somewhat similar to the trained shape of the different coefficient levels, just to see how it would affect the results. Results can be seen in Figs. 1, 2, 3 and 4.

Since this a 3D search space in η, ρ and b_0, this is a very shallow investigation so far.

3.3 Iterative training of hyperparameters

Plugging μ and λ back in and training, using several iterations, causes the parameters to diverge (Fig. 5).

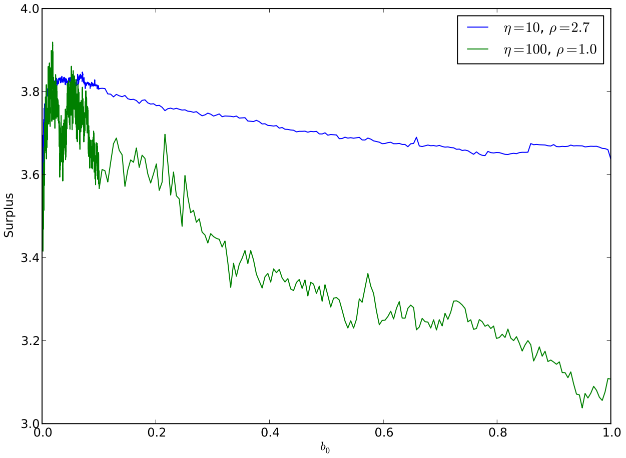

Figure 1: Two different values of η and ρ are tried. The granularity of b_0 changes after 0.1 to save some calculation time.

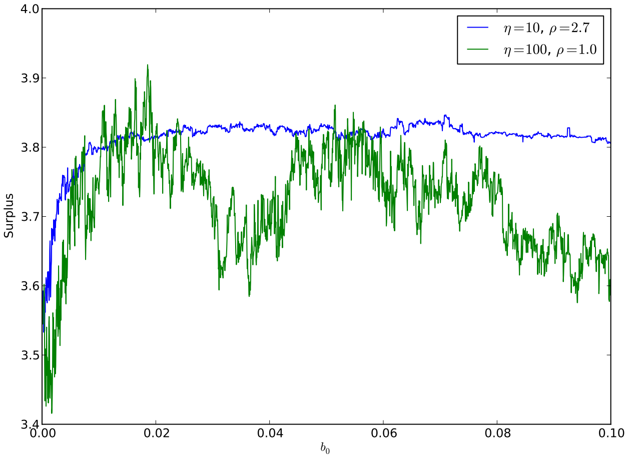

Figure 2: Same as Fig. 1, except zoomed in.

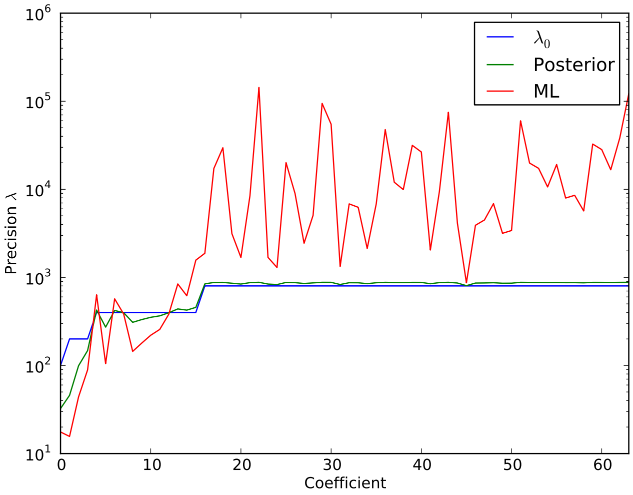

Figure 3: Prior (λ_0), likelihood (ML) and posterior of the coefficients for $\eta = 100$ and $\rho = 1$ at $b_0 = 0.05$. For digit 0, mixture component 0 and axis 0.

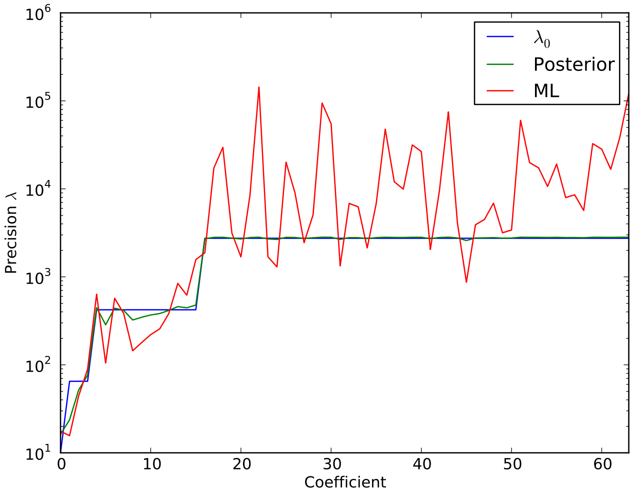

Figure 4: Prior (λ_0), likelihood (ML) and posterior of the coefficients for $\eta = 10$ and $\rho = 2.7$ at $b_0 = 0.05$. For digit 0, mixture component 0 and axis 0.

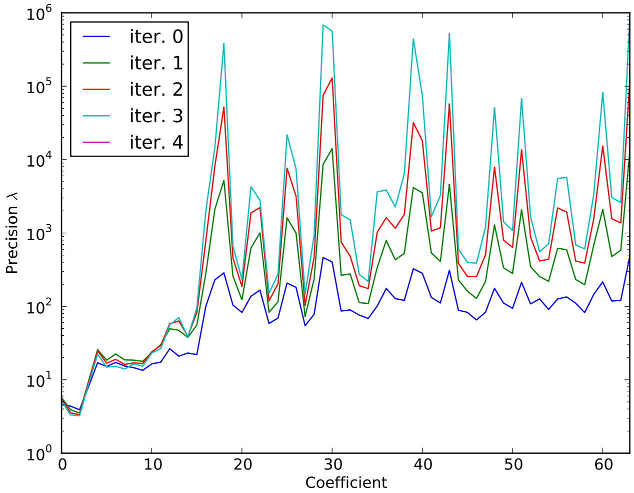

Figure 5: Divergence of iteratively training the precision λ.

3.4 Observations

The gradient $\nabla J(u)$ is not $\mathbf{0}$ at the point of convergence in the BFGS algorithm. Exact reason unknown.

4 Code

GitHub repositories: gustavla/vision-research, amitgroup/amitgroup