Pattern Classification and Machine Learning: Project Report

Nicolas Voirol (186268), Damien Firmenich(178474)
EPFL, School of Computer and Communication Sciences

Spring 2013

1 Introduction

Pattern classification and machine learning provide us with a large set of powerful tools and techniques. The sheer number and the diversity of these tools can make selecting the ideal technique for a given problem quite difficult. We could simply decide we want to use the best techniques available to make sure our system is optimal, but with complexity comes time consumption. Therefore, when using an elaborate tool to solve a classification problem, it is generally useful to make sure it actually performs better than a simpler technique. This project compares the Multi-Layer Perceptron technique against Least Squares Estimation and Logistic Regression on the NORB dataset.

Methods

The theory places gradient descent in batch mode above stochastic online descent in terms of the optimality of each update. We therefore implemented partial (or mini) batch mode with varying batch size to compare their actual impact on convergence speed and final error rate. We implemented this feature in our MLP and Logistic Regression classifiers since they are both based on gradient descent.

Model and parameters

1.1 Binary MLP and multi-way MLP

Backpropagation The vectorized backpropagation equations that we implemented for the binary MLP are as follows :

$$
\begin{aligned}
\mathbf{r}^{(3)} &= \sigma\left(\mathbf{a}^{(3)}\right) - \frac{1}{2}\left(\mathbf{t}+1\right) && \Rightarrow \quad \nabla_{\mathbf{w}^{(3)}} E_i = r^{(3)}\left(z^{(2)}\right)^T \\
\mathbf{r}_L^{(2)} &= \mathbf{a}_{LR}^{(2)} \sigma'\left(\mathbf{a}_L^{(2)}\right) \sigma\left(\mathbf{a}_R^{(2)}\right) \mathbf{w}^{(3)} \mathbf{r}^{(3)} && \Rightarrow \quad \nabla_{\mathbf{w}_L^{(2)}} E_i = \mathbf{r}_L^{(2)}\left(\mathbf{z}_L^{(1)}\right)^T \\
\mathbf{r}_R^{(2)} &= \mathbf{a}_{LR}^{(2)} \sigma\left(\mathbf{a}_L^{(2)}\right) \sigma'\left(\mathbf{a}_R^{(2)}\right) \mathbf{w}^{(3)} \mathbf{r}^{(3)} && \Rightarrow \quad \nabla_{\mathbf{w}_R^{(2)}} E_i = \mathbf{r}_R^{(2)}\left(\mathbf{z}_R^{(1)}\right)^T \\
\mathbf{r}_{LR}^{(3)} &= \sigma\left(\mathbf{a}_L^{(2)}\right) \sigma\left(\mathbf{a}_R^{(2)}\right) \mathbf{w}^{(3)} \mathbf{r}^{(3)} && \Rightarrow \quad \nabla_{\mathbf{w}_{LR}^{(2)}} E_i = \mathbf{r}_{LR}^{(2)}\left[\mathbf{z}_L^{(1)}\,\mathbf{z}_R^{(1)}\right]^T \\
\mathbf{r}_L^{(1)} &= \left(\operatorname{diag} g'\left(\mathbf{a}_L^{(2)}\right)\right)\left(\left(\mathbf{w}_L^{(2)}\right)^T \mathbf{r}_L^{(2)} + \left(\mathbf{w}_{LR[L]}^{(2)}\right)^T \mathbf{r}_{LR}^{(2)}\right) && \Rightarrow \quad \nabla_{\mathbf{w}_L^{(1)}} E_i = \mathbf{r}_L^{(1)}\left(\mathbf{x}_L\right)^T \\
\mathbf{r}_R^{(1)} &= \left(\operatorname{diag} g'\left(\mathbf{a}_R^{(2)}\right)\right)\left(\left(\mathbf{w}_L^{(2)}\right)^T \mathbf{r}_L^{(2)} + \left(\mathbf{w}_{LR[R]}^{(2)}\right)^T \mathbf{r}_{LR}^{(2)}\right) && \Rightarrow \quad \nabla_{\mathbf{w}_R^{(1)}} E_i = \mathbf{r}_{\mathbf{R}}^{(\mathbf{1})}\left(\mathbf{x}_R\right)^T
\end{aligned}
$$

Notation	Description
$\mathbf{w}^{(2)}_{LR[L]}$, $\mathbf{w}^{(2)}_{LR[R]}$	portion of the weight matrix corresponding to the left, respectively right, image
$g(\cdot)$	transfer function $tanh(\cdot)$
$\left[\mathbf{z}^{(1)}_L \; \mathbf{z}^{(1)}_R\right]$	concatenation of $\mathbf{z}^{(1)}_L$ and $\mathbf{z}^{(1)}_R$.

The residuals are computed using the derivative of the *tanh* transfer function, and the derivative of the logistic loss function. Then the gradients at each layer and for each stereo image (left, right) are computed using the residuals.

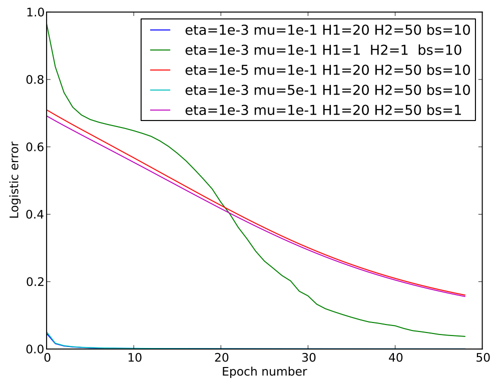
Multi-way classification Most of the MLP we implemented for binary classification could be reused for multi-way classification. Only two changes were necessary :

1. The computation of the error on the top-layer had to be modified to squared error on *1 of K* coded target vector $\boldsymbol{t}$, which implied the following change to the top-layer residual :

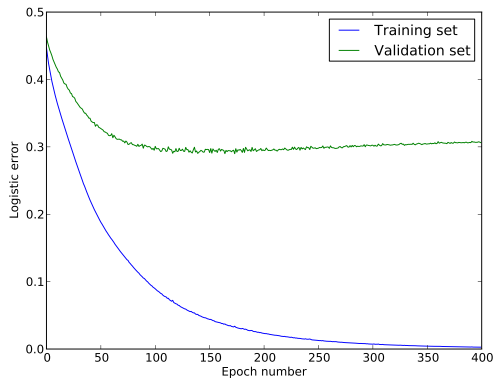
$$\mathbf{r}^{(3)} = \mathbf{a}^{(3)} - \tilde{\mathbf{t}}$$

2. Since we now have 5 outputs instead of a single one, the computation of the class associated with the input image is becomes $\underset{i=1,\dots,5}{argmax}\left(\mathbf{a}^{(3)}_i\right)$ instead of $sign\left(a^{(3)}\right)$.

Interestingly, the equations for backpropagation don't need to be changed as they are vectorized and automatically take into account the multi-dimensionality of the labels $\tilde{\mathbf{t}}$.



(a) Comparative plot showing the effect of the MLP and gradient descent parameters



(b) Example of overfitting for the multi-way MLP used for early-stopping

Figure 1: MLP training plots

Model selection Our MLP classifier had two native parameters (namely H_1 and H_2 which we considered separately to ensure analysis completeness), and relies on gradient descent to converge to the optimal solution, which adds learning rate η, momentum term μ and mini-batch size to the parameter list. The impact of each parameter on the final performance of the MLP is unfortunately extremely unintuitive and difficult to predict, especially because plausible ranges are highly inter-dependent. Because of this, we couldn't perform standard model selection by computing errors on the cartesian product of a reasonable range of parameter values.

We circumvented the problem by running the algorithm with reduced training and validation sets on a large combinations of values to get a reasonable estimation. This enabled us to narrow down the search space and perform better tests with the full training set. See Figure 1a for examples of how the parameters influences the convergence of the MLP. For the learning rate η we tested with constant values (in the order of 10^{-2} to 10^{-5}) as well as variations of $10^{-n}x^{-\frac{1}{a}}$ with $n \in [0;3]$ and $a \in [1;3]$. The momentum term varied from 0.05 to 0.25, and the mini-batch size ranged from 1 to 50. As explained in the problem statement, we tested configuration of MLP with 1 to 80 activation units in each hidden layers.

The optimal parameters for the binary MLP are $\eta = 10^{-3}$, $\mu = 10^{-1}$, with mini-batch size of 20. The numbers of units in the first layer is 20 and for the second layer is 50. For the multi-way MLP, we found $\eta = 10^{-3}$, $\mu = 10^{-1}$, mini-batch size of 5, and number of units in the first and second layer of 60 and 10 respectively.

In Figure 1b is an example of overfitting where the error on the validation set initially decreases but starts to increase as soon as the MLP is overfitting the training data. We used early stopping to avoid this behavior in all our gradient-descent based algorithms.

1.2 Logistic regression

The logistic regression technique doesn't have any model parameters, but it relies on gradient descent which does (namely the learning rate η, momentum term μ, and mini batch size). Unfortunately, as in the MLP case, it is difficult to perform standard model selection on these parameters.

After having manually determined plausible parameter combinations, we performed model selection by comparing convergence speed and final error rate on these values. During this process, we realized that logistic regression was extremely sensitive to initialization and training order, so we ran our model selection multiple times on each parameter combination and considered mean, best case and worst case performance instead of individual results when comparing models (see Figure 2), resulting in the optimal configuration of $\eta = 2e^{-2}$, $\mu = 5e^{-2}$, with block size of 5.

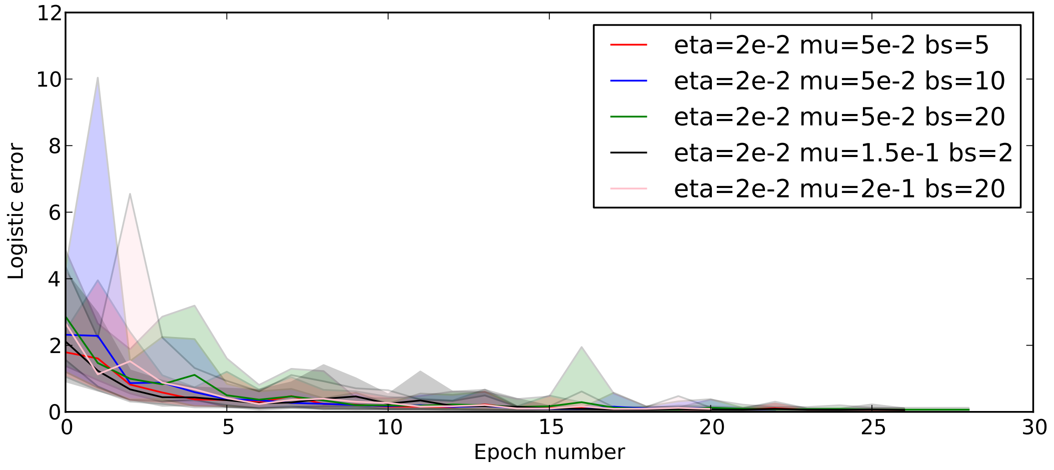


Figure 2: Comparative plot of promising logistic regression model parameter combinations. The red and black curves both show good initial convergence and stability, but the red performs better in the tail.

1.3 Least squares estimation

Error function The analytical solution to the least squares estimation with Tikhonov regularization problem is obtained by solving the normal equations problem. In the case of multi-way classification, the solution is trivially extended by solving the equations for each of the k vectors obtained by using *1 of K* encoding for the target vector $\boldsymbol{t}$. But this also means we multiply the impact of the Tikhonov regularizer by k. To compensate for this, we modified the regularized error function slightly :

$$E(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{N} \parallel \mathbf{y}(\mathbf{x}_i) - \tilde{\mathbf{t}}_i \parallel^2 + \frac{1}{k}\frac{\nu}{2}\sum_{k=1}^{K} \parallel \mathbf{w}_k \parallel^2$$

Model selection The model selection process for the least squares estimation technique was by far the simplest since we were dealing with a single parameter and a constant convergence time. All computations were performed using 10-fold cross-validation, so we combined the 10 values obtained after each parameter validation run into boxplots for clarity.

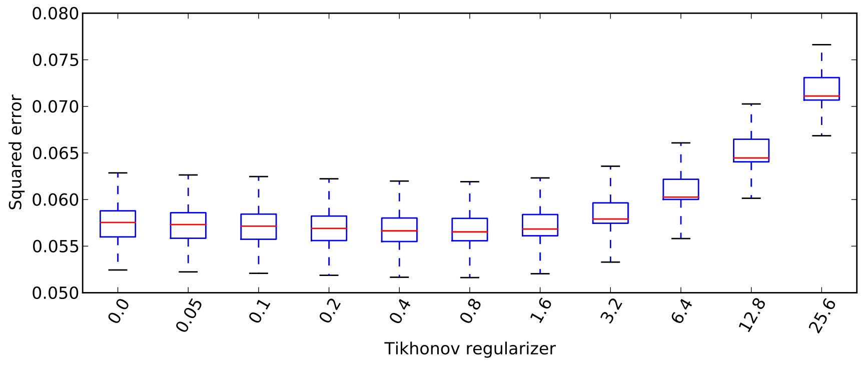


Figure 3: Interval selection for Tikhonov regularization parameter.

We initially got a feeling for plausible values by setting the Tikhonov regularizer to the exponential range $\{0\} \cup \{5 \cdot 10^{-2} * 2^k \mid 0 \leq k < 10\}$ (see Figure 3). Once we had singled out a suitable interval, namely $[0; 1.2]$, we performed a more precise search by resorting to a linear parameter space of $\{5 \cdot 10^{-2} * k \mid 0 \leq k < 25\}$ (see Figure 4) and we can thus place the optimal regularizer at 0.6.

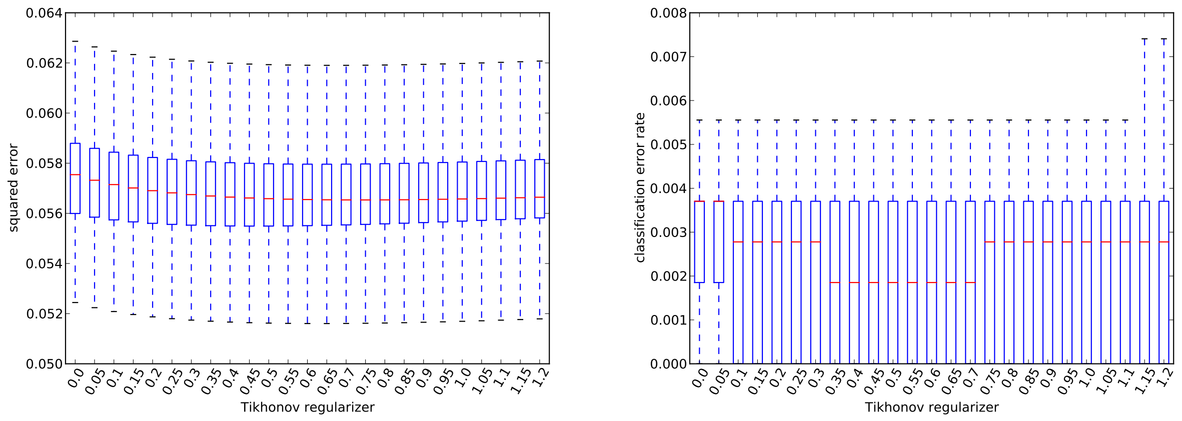


Figure 4: Error function value (left) and classification error rate (right) during parameter selection for Tikhonov regularizer. We can clearly see the curve minimum around 0.5 (the actual minimum is at 0.6).

2 Results and Discussion

2.1 Binary MLP and multi-way MLP

The graph in Figure 5 shows the logistic error computed with the optimal parameters for the binary MLP. Both the errors for the training and validation sets are very close and the MLP converges quickly to an optimal solution, as both the classification error for the training and the validation set are null.

The graph in Figure 6 shows the same metric for the multi-way MLP. The system converges slower than before and there is a better separation between the errors of the training and the validation set. We also

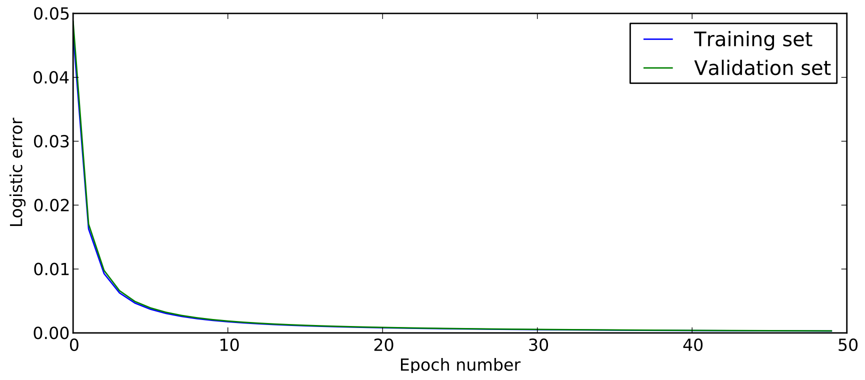


Figure 5: Logistic error for binary classification using MLP for the training and the validation set. The classification error rate is not very interesting in this setting since it drops to zero after one run over all training points.

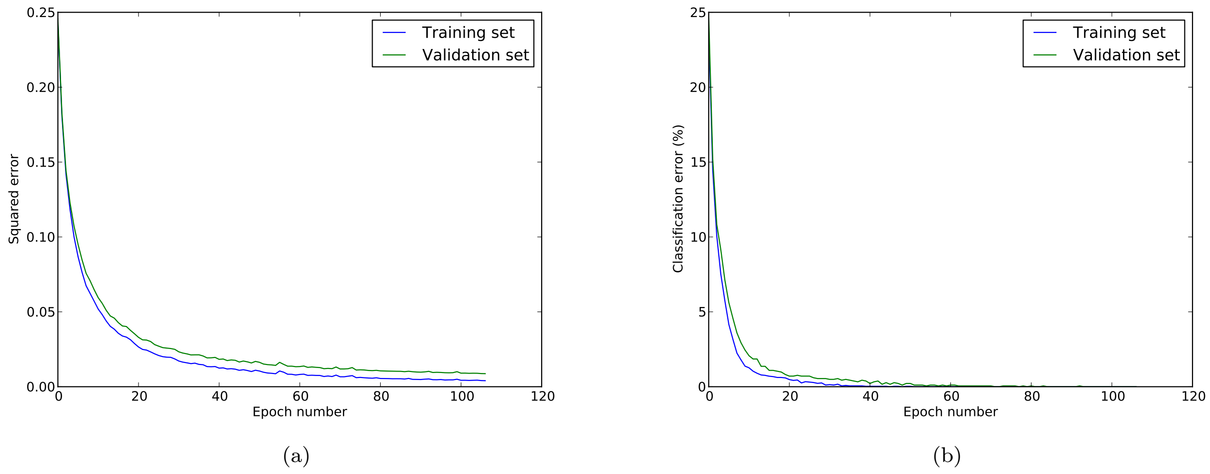


Figure 6: Squared error (a) and classification error (b) for multi-way classification using MLP for the training and the validation set.

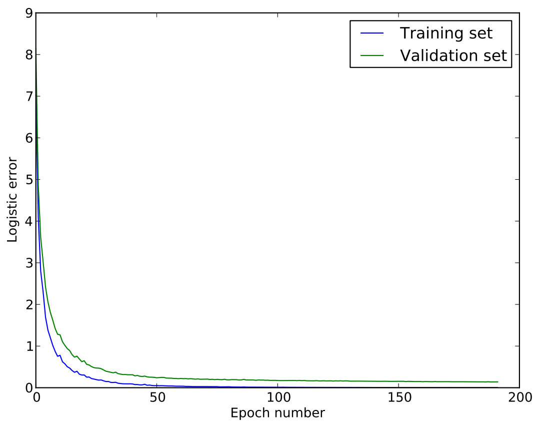
showed a graph representing the classification error for both sets, as they both converge to zero but not at the same speed. It is interesting to notice that both the binary and multi-way MLP converge to a classification error of zero after enough training.

2.2 Logistic regression

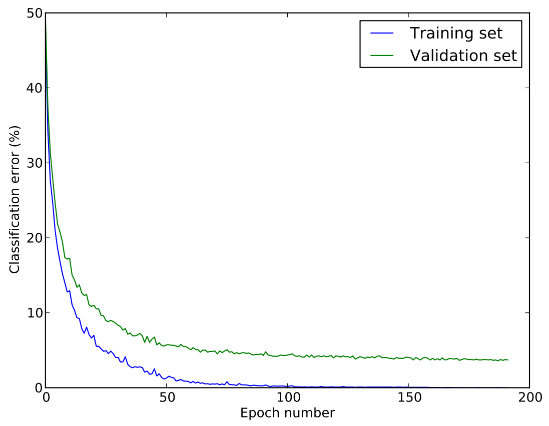
In Figure 7, we show the error for the logistic regression technique in function of the training epoch. We see right away that the convergence speed is close to the one we witnessed in the MLP case, but the classification error rate of the validation set is much lower using the multi-way MLP technique.

2.3 Classification results

The results for the misclassification rate on the test set are shown in Figure 8. Both implementations of MLP show higher performance than linear classifiers. On the binary classification dataset, the binary MLP misclassified **3.65**%($\sigma = 1.37$). On the 5-way classification dataset, the MLP had an average error percentage of **11.29**% ($\sigma = 1.11$), the logistic regression performed worse with an average of **19.84**%



(a)



(b)

Figure 7: Logistic error (a) and classification error (b) for multi-way classification using logistic regression for the training and the validation set.

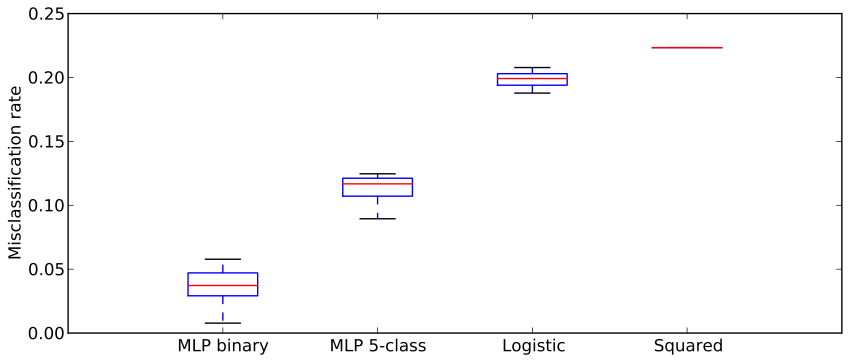
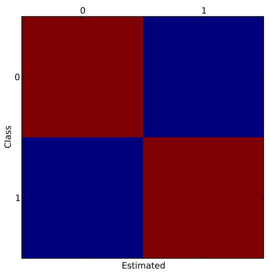


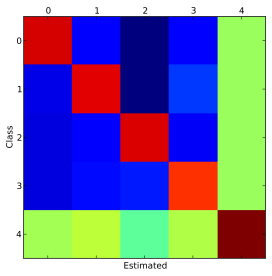
Figure 8: Comparative boxplot for each classifier

($\sigma = 0.61$) classification errors. All those results were computed on 10 runs. Linear regression with squared error had the worse classification error percentage with **22.34**%.

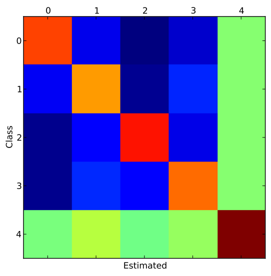
The confusion matrices are shown in Figure 9. As expected and confirmed by the boxplot in Figure 8, the binary MLP performs very well with very few errors. The multi-class classification methods are more confused by the input images as the classes are closer to each other in terms of feature space. This qualitative comparison shows a decay in classification performance from the MLP to the squared error regression, confirming the results of Figure 8.



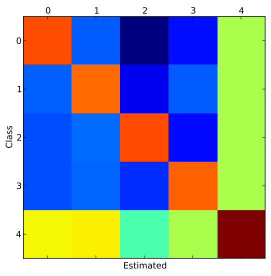
(a) Binary MLP



(b) Multi-way MLP

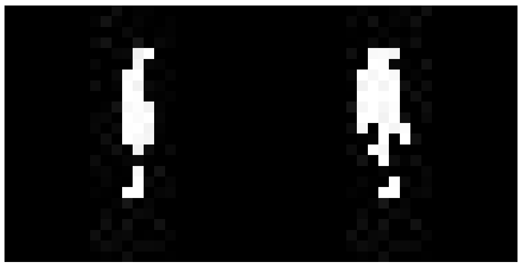


(c) Logistic error regression

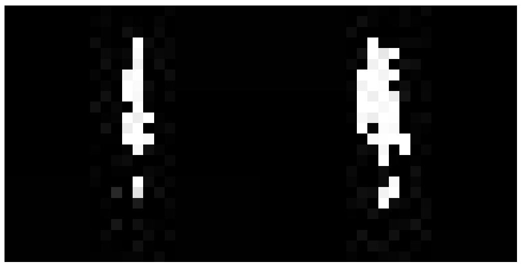


(d) Squared error regression

Figure 9: Confusion matrices (qualitative comparison)



(a) Negative $t_i a_i^{(3)}$ close to zero



(b) Large negative $t_i a_i^{(3)}$

Figure 10: Misclassified image examples with the multi-way MLP