Schemes for classification of MNIST digits

Kevin A. Fischer
Stanford University
kevinf@stanford.edu

Abstract

Learning to classify handwritten digits is a difficult problem, well suited to modern techniques in machine learning. This report compares three basic schemes for classifying the MNIST digits: softmax regression, a 4-layer convolutional neural network, and a one-vs-one support vector classifier.

1 Introduction

The basic principle of supervised learning is to train a classifier, that given a test input x can accurately predict the output class y the input belongs to. Such an example task is classifying handwritten digits, where an input image is given and an output digit is expected. The process of training involves feeding the classifier examples of (x, y) pairs, and modifying the parameters of the classifier to accurately predict these examples. If the input data are well clustered together and well represent the distribution of future test data, it is possible to create a classifier that can easily generalize to future examples.

2 Neural network classifiers

Neural network classifiers are loosely based around three principles: a computation graph, backpropagation of error terms, and unconstrained gradient descent. Consider the most simple network classifier for the MNIST data, softmax regression (Fig. 1a). An input vector x of dimension $n = 28$ x $28 = 784$ is fed into the forward propagation equations

$$z_i = w_{ij}x_j + b_i \quad \text{and} \quad \hat{y}_i = \frac{\mathrm{e}^{-z_i}}{\sum_{j=1}^{10} \mathrm{e}^{-z_j}}$$

to compute predictions, using the parameters $w \in \mathbb{R}^{10}$ x $\mathbb{R}^{784}$ and $b \in \mathbb{R}^{10}$. The output vector represents the network's estimates $\hat{y}_i = P(y_i|x)$. To assess the quality of the predictions, the cross-entropy loss function

$$L = -y_i \log(\hat{y}_i)$$

is used for each training example. With each additional batch of training examples, the goal is to update the parameters of the network to minimize the expected value of the loss function. This is done by using stochastic gradient descent

$$w_{ij} \to w_{ij} - \eta \frac{\partial \bar{L}}{\partial w_{ij}} \quad \text{and} \quad b_i \to b_i - \eta \frac{\partial \bar{L}}{\partial b_i}$$

where $\bar{L}$ denotes the average loss over a batch of training examples and η is the learning rate. In order to efficiently compute the derivatives of the loss with respect to the training examples, the backpropagation equations

$$\delta_i = \frac{\partial L}{\partial z_i} = \hat{y}_i - y_i, \quad \frac{\partial L}{\partial b_i} = \delta_i, \quad \text{and} \quad \frac{\partial L}{\partial w_{ij}} = x_j \delta_i$$

are used.

May, 2018

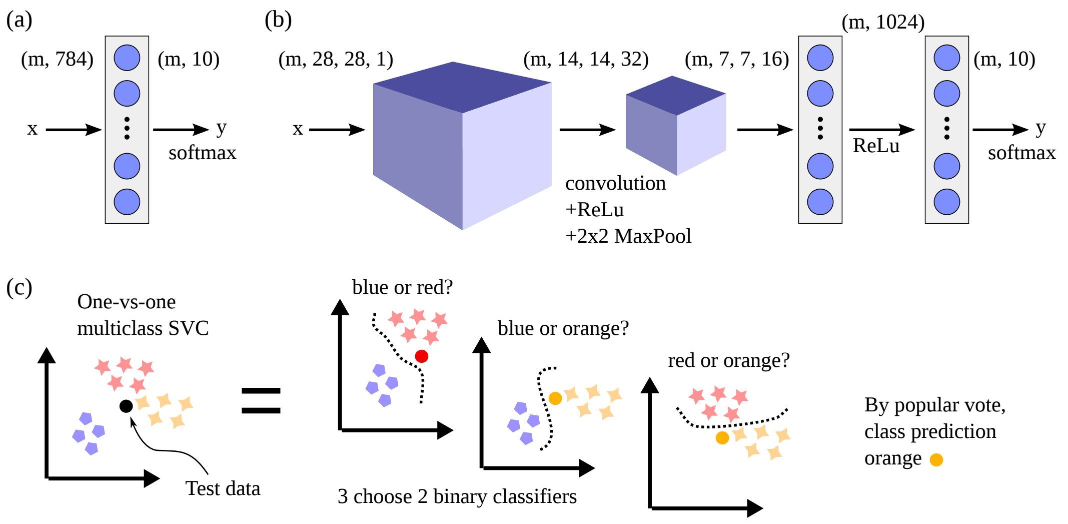


Figure 1: Classification schemes for MNIST digits. Each image contains 784 = 28 x 28 pixels values, normalized range from 0 to 1. (a) Softmax classifier. (b) 4-layer convolutional neural network. (c) Multiclass support vector classifier, shown schematically for three classes. For the MNIST data there is one class for each digit (10 total) and hence $\binom{10}{2} = 45$ binary SVCs need to be used.

Given that the only nonlinearity in the network is determined by the softmax unit, the ability of this network to separate arbitrary input distributions is limited. Because the input MNIST digits to the network exist in an extremely small and clustered subspace of all possible 28 x 28 images, it is possible to construct a network that takes advantage of this order. Convolutional networks do just this, by learning a structure of hierarchical maps that represent translationally invariant features, and in the process regularize the parameters to help prevent over-fitting. After a few convolutional layers in a network, there are almost always a few fully connected network layers (similar to the softmax regression) in order to sort through the features detected in the convolutional layers. The 4-layer network shown in Fig. 1b is such a network.

The convolutional blocks map input volumes (width, height, features) to output volumes (width, height, new features) via

$$\begin{aligned} z^{l+1}_{ijc} &= w_{xykc} a^{l}_{i+x,j+y,k} + b_c \\ a^{l+1}_{ijc} &= \max(z^{l+1}_{ijc}, 0), \end{aligned}$$

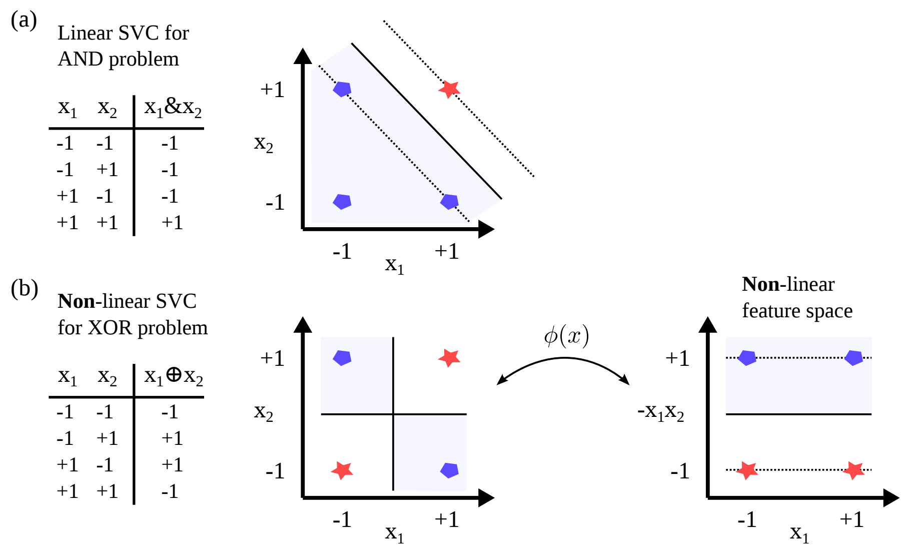
with a pool that decreases the dimensionality by a factor of 2 x 2 from taking the maxima of two-by-two blocks along the first and second dimensions of the activations a^{l}_{ijc}. Similar to the forward propagation equations, backpropagation for the convolutional blocks feature convolutions.

Training the CNN is significantly more difficult than the softmax classifier because the CNN has nearly 100 times more parameters to learn. Hence, the adaptive gradient method is used, where the the updates are dependent on the past history of gradients.

3 Support vector classifiers

Support vector classifiers (SVC) find optimal decision boundaries between linearly separable data, and hence perform binary classifications. In order to perform classification of the MNIST digits with SVCs where there are ten different classes (0-9), it is necessary to train multiple binary classifiers (Fig. 1c). A commonly used scheme for using SVCs for multiclass identification is called 'one-vs-one' classification, where for c input classes $\binom{c}{2}$ classifiers are trained. In this scheme, each classifier is trained on a unique binary combination of input classes and then the final prediction is determined by the most popularly voted class.

Before discussing the MNIST SVC classifiers, consider the simpler two-dimensional Boolean problems, which are canonical for introducing the mechanics of support vector classifiers. In the


(a)
Linear SVC for AND problem
+1
x_2
-1
-1
+1
x_1
(b)
Non-linear SVC for XOR problem
+1
x_2
-1
-1
+1
x_1
$\phi(x)$
Non-linear feature space
+1
$-x_1x_2$
-1
-1
+1
x_1

x_1	x_2	$x_1\&x_2$
-1	-1	-1
-1	+1	-1
+1	-1	-1
+1	+1	+1

x_1	x_2	$x_1 \oplus x_2$
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1

Figure 2: Mechanism behind a binary support vector classifier. (a) Table shows classification task. Input space plotted, with classes denoted by color and shape of the markers. The decision boundary is shown as the solid line and the hard margins as dotted lines. (b) Table shows classification task. Left plot shows input space, with classes again denoted by color and shape. The decision boundary is highly nonlinear in the input space, but after transformation to a new feature space via $\phi(x) = [1\ \sqrt{2}x_1\ \sqrt{2}x_2\ \sqrt{2}x_1x_2\ x_1^2\ x_2^2]^T$ the classes are now linearly separable.

2D Boolean operations, two binary inputs x_1 and x_2, taking on values $\{-1, +1\}$, are given while a single output y is computed. This output lies within the original Boolean space as well. For example, in the AND problem $y = x_1 \& x_2$ (as shown in Fig. 2a).

Suppose one is interested in creating a support vector machine operating on $\mathbb{R}^2$ that learns the AND Boolean operation by using its input/output pairs $[(x_1, x_2)^T, y]$ as training data. For positive y one class is assigned while for negative y the other. In the two-dimensional input space it is simple enough to drawn a single line dividing the two classes, where the distance or margins between the line and the nearest points of each class is maximized. This is called a maximal margin classifier, and the support vector machine arrives at this linear decision boundary. Because a single line divides the classes, the data is called linearly separable and the margins are referred to as hard. Here, the primal form of the necessary optimization problem defining the SVM is

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \frac{1}{2}||w||^2 \\ \text{subject to} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1,\ i = 1, \ldots, 4 \end{aligned}$$

for the i-th input/output pair (x, y), where $w \in \mathbb{R}^4$ x $\mathbb{R}^4$ is the weight matrix and $b \in \mathbb{R}^4$ is a bias vector. When the data are linearly separable, then this optimization problem is convex. One can trivially find the analytic solution to the AND problem, given by the classifier

$$x \mapsto \text{sgn}(x_1 + x_2 - 1.5)$$

which is plotted in Fig. 2a. The margins from the decision boundary $x_1 + x_2 = 1.5$ are shown as the dotted lines. The samples on these margins are called the support vectors; note that only three of the samples are support vectors.

However, many interesting problems have input classes that are not linearly separable in the original feature space. As an example, consider the XOR problem, where the output is given by $y = x_1 \oplus x_2$ (Fig. 2b). In this problem, no single line can be drawn in the input space to separate the two output classes. Hence, to find a classifier, the general principle applied is to map the input features into a higher-dimensional space where they are indeed linearly separable. However, the primal form of the optimization problem for an SVM is not conducive to computations in this new space because it may be highly non-convex.

Instead, the method of Lagrange multipliers is used to find a new dual form of the SVM where the optimization problem is convex. The dual optimization for 2D Boolean problems is given by

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2}\sum_{i,j=1}^{n} y^{(i)}y^{(j)}\alpha_i\alpha_j K(x^{(i)},x^{(j)}) - \sum_{i=1}^{n}\alpha_i \\ \text{subject to} \quad & 0 \le \alpha_i,\ i=1,\dots,n \\ & \sum_{i=1}^{n}\alpha_i y^{(i)} = 0, \end{aligned} \tag{1}$$

where there are $n = 4$ input features, the α_i are the Lagrange multipliers, and $K(x^{(i)}, x^{(j)}) = \langle\phi(x^{(i)}), \phi(x^{(j)})\rangle$ is the nonlinear kernel function for the feature map $\phi(x)$. Notably, the dual optimization problem depends only on the inner product of samples in the transformed feature space.

After solving the dual optimization problem, the decision boundary is similarly computed from the optimal weights and bias

$$x \mapsto \text{sgn}(w^{*T}x + b^*).$$

In the nonlinear feature space, the weight matrix is given by

$$w^* = \sum_{i=1}^{n}\alpha_i y^{(i)}\phi(x^{(i)}),$$

with the bias

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T}x^{(i)} + \min_{i:y^{(i)}=+1} w^{*T}x^{(i)}}{2}.$$

Therefore, in this nonlinear space the optimization algorithm finds a linear decision boundary, whose support vectors are a subset of $\phi(x)$ and each weighted via an α_i.

For the XOR problem, it is widely known that the feature map $\phi(x) = [1\ \sqrt{2}x_1\ \sqrt{2}x_2\ \sqrt{2}x_1x_2\ x_1^2\ x_2^2]^T$ makes the input data linearly separable: then the dual optimization has a nice analytic solution. In the nonlinear feature space, only the α_i corresponding to $\sqrt{2}x_1x_2$ is nonzero. Hence, the polynomial machine that solves the XOR decision problem is

$$x \mapsto \text{sgn}(-x_1x_2).$$

In many practical optimization problems such as classifying MNIST data, however, nice analytic solutions do not exist. Here, it is very important that the inner product in the nonlinear feature space is often simple to calculate, despite the space being higher dimensional. For example, although the map for the XOR problem is in a 6-dimensional feature space (rather than the original 4), the inner product is given by $K(x, z) = (x^Tz + 1)^2$ which is very simple to compute. Then, one can apply the kernel trick to compute

$$w^{*T}z = \sum_{i=1}^{n}\alpha_i y^{(i)}K(x^{(i)}, z)$$

without ever performing any calculations in the nonlinear space.

Finally, note that the end result of using l_1 regularization on the Lagrange multipliers in the dual optimization problem (addition of slack variables) simply changes the constraint

$$0 \le \alpha_i \ \rightarrow\ 0 \le \alpha_i \le C, \tag{2}$$

where C is the penalty for samples violating a normalized decision margin. For Boolean logic, the input space is restricted so different values of C do not alter the decision boundary at all. Another way to say this is that for the polynomial machine trained using the entire Boolean input sample space of the 2D XOR problem, there is zero empirical error.

Now we are prepared to discuss the SVC used on the MNIST data. $\binom{10}{2}$ binary classifiers are trained from the dual optimization problem Eq. 1 and the constraint of Eq. 2. The input data are $n = 28$ x $28 = 784$ dimensional vectors, with pixel values ranging between 0-1. For this classification problem, the radial basis function (RBF) kernel $K(x, z) = \exp(-\gamma||x - z||^2)$ with $\gamma = 0.05$ has been found particularly effective at separating the input classes in the nonlinear space.

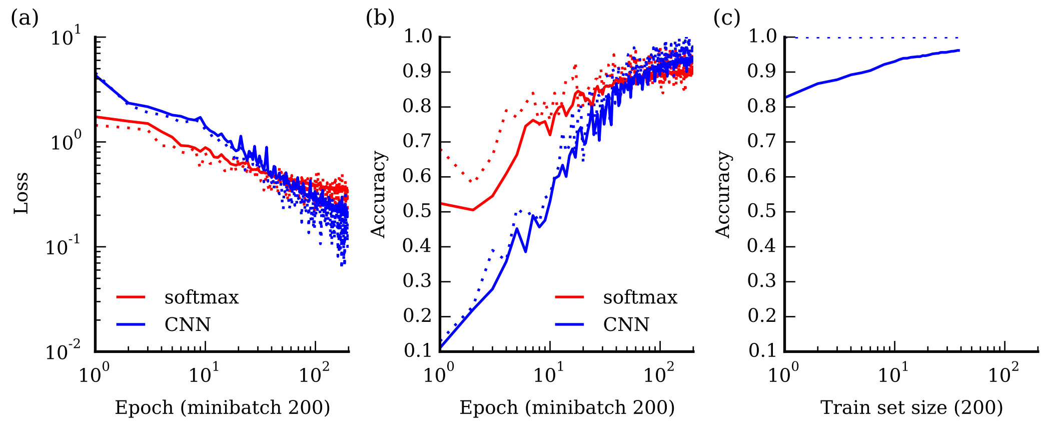


Figure 3: Training and test accuracy and loss for all three classifiers. (a) Training and test loss in dotted and solid curves, respectively. (b) Training and test accuracy in dotted and solid curves, respectively. Maximum accuracy for softmax regression and CNN are 90.7% and 94.3%, respectively. (c) Multiclass support vector classifier training and test accuracy in dotted and solid curves, respectively. Maximum accuracy achieved is 95.7%.

Taylor expanding the kernel shows that it transforms to an infinite-dimensional nonlinear feature space, while the constant γ sets the roll of in importance of higher-order polynomials. Rather than finding a separating line in this space, the SVM looks for a separating hyperplane defined by $w^{*T}x + b^* = 0$. For the MNIST data, the slack is taken to be $C = 5$, which results in Var$(\alpha_i) \approx$ E$[\alpha_i]^2$—almost all of the training examples provide some support to the decision boundary in the nonlinear feature space.

4 Results

In this section, the results from training the three classifiers: softmax, CNN, and SVC, are discussed. The stochastic gradient descent training for the neural networks (softmax and CNN) consumes training samples in minibatches of 200, and hence the cross-entropy loss (Eq. 2) can be estimated at each training step (Fig. 3a). For both the softmax network and CNN, the loss on the training samples at each step has much higher local variance compared to that of the test samples. This is expected, since the test set is 50 times larges than the training set. Comparing the loss between the softmax network and CNN, the softmax network is clearly easier to train, but has ultimately worse performance. This is expected since the CNN has nearly 1 million parameters to train compared to the roughly 10,000 of the softmax network. Consequently, the CNN has much more expressive power and, so long as the training method is efficient, one would expect it to have lower loss on the MNIST data. Meanwhile, the softmax network (multinomial logistic regression) can only learn relatively simple decision rules.

Next, consider the accuracy during training between the softmax network and the CNN (Fig. 3b). Compared with the loss a similar trend is observed, where the softmax regression learns to accurately classify the MNIST digits more quickly but loses out in the long run to the CNN.

Finally, consider the performance of the SVC (Fig. 3c). There are several substantive difference in training support vector machines for how to evaluate their resulting performance. These considerations are related to the fact that the most basic formulation of an SVM consumes the training examples simultaneously[10]. Further, training the dual optimization problem for an SVM is convex, and a set of convergence criterion exist for finding the optimal solution. Hence, it is not necessarily helpful to consider the behavior of the loss function nor monitor the training accuracy.

[10]There has been some work towards extending support vector machines to minibatch training in order to handle extremely large datasets, e.g. see Takác et al. [2013].

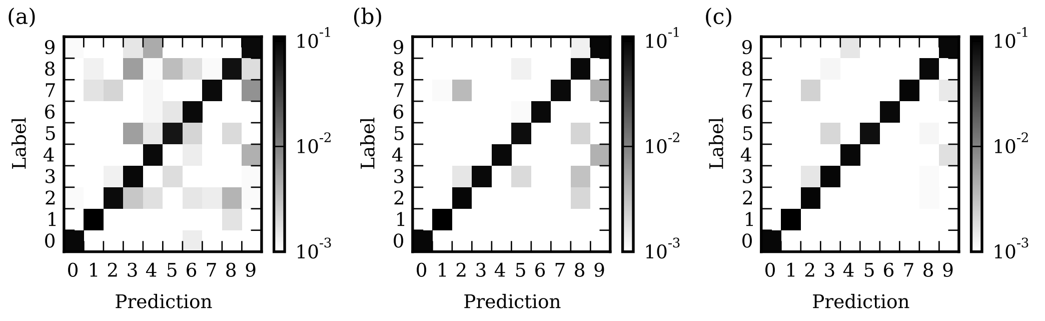


Figure 4: Normalized prediction and label correlation matrices. (a) Softmax classifer. (b) 4-layer convolutional neural network. (c) Multiclass support vector classifier.

As long as the kernel provided to an SVM is sufficiently high dimensional, it is very likely the optimization will find a hyperplane separating the input classes. Unlike for the neural networks, this means the SVC always classifies the training data with 100% accuracy (Fig. 3c). Now, compare the test accuracy, which reaches a maximum classification accuracy higher than either the softmax regression or CNN. If the entire dataset was consumed, this accuracy can reach 98.5% Sopyła [2017]. Here, the maximum training set size was cutoff to 8,000 training examples because of memory limitations of processing the SVCs. The `cvxopt` package used necessarily stores the entire kernel, which is an m x m matrix. Storing the values each with 8 bytes, then the kernel for the entire training set of 70,000 examples would require almost 40 GB of ram. However, `libsvm` provides a much more memory efficient implement of SVMs, which caches commonly used values—this allows the entire training set to be used more efficiently.

Clearly, for the MNIST problem the SVC is significantly better at generalizing than either the softmax or CNN networks, especially given that for just 200 samples the accuracy on the test set is already over 80 %. This suggests that the RBF kernel very efficiently separates different MNIST digits in a high-dimensional space. Given that it is possible to visualize this separation using techniques such as t-SNE [Maaten and Hinton, 2008], it is not surprising that the SVC has very strong predictive power here.

Last, briefly compare the correlation (confusion) matrices for the different classifiers (Fig. 4). Relatively hard digits for the softmax classifier to distinguish are 3 from 5 or 8, and 4 or 7 from 9. The CNN manages to improve all of these cases but still struggles a bit with the 4 or 7 from 9. The SVC really only struggles a tiny bit in comparison with 2 from 7. These matrices would potentially provide feedback for where to target changes in the learning architectures or algorithms to better improve classification accuracy.

5 Conclusion

This report compared three different classifiers of increasing predictive power on MNIST digits. First softmax regression, then a 4-layer convolutional neural network, and finally a one-vs-one support vector classifier. The SVC turned out to have the highest predictive and generalization power on the MNIST data set and was easier to train for small training sets.

References

Martin Takác, Avleen Singh Bijral, Peter Richtárik, and Nati Srebro. Mini-batch primal and dual methods for SVMs. In *ICML (3)*, pages 1022–1030, 2013.

Krzysztof Sopyła. SVM MNIST digit classification in python using scikit-learn, 2017. URL `https://github.com/ksopyla/svm_mnist_digit_classification`.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.