Chapter 13

Learning to Shape Errors with a Confusion Objective

Jason Scholz

13.1 Introduction

Automated systems typically operate on their own only if they are simple, or within a closed and managed environment. In situations when things can go wrong with serious consequences, control is typically introduced through a human-in-the-loop. Trusted Autonomous Systems (TAS) are different in that they will be required to operate in open and unmanaged environments and when human guidance may not be available. To achieve machine perception in open environments, machine learning classifiers are likely critical components in TAS. However, in order for such systems to be worthy of their trust, users and designers will want to ensure that classification errors that may result in high-consequence impact can be managed; in effect, to introduce a bias in the machine analogous to a learned human bias.

Much recent statistical machine learning [11] involves learning weights in complex, modular non-linear artificial neural networks. These weights are derived from the gradient of the objective (or loss) function. This gradient is back-propagated (Rummelhart et al. 1986) throughout the whole system. In supervised classification, the objective function typically uses some relationship between the machine's output estimated class and the training set true class. The squared error and cross entropy are popular example functions. The choice of objective function instructs the machine by reward for a correct estimate, and lack thereof for incorrect estimates. These objective functions reward all correct choices and punish all incorrect choices equally. The choice of objective function affects error distribution and in high consequence situations some types of errors may be undesirable. When errors are made, a user wants them to be of least consequence, in effect, the *lesser of evils*. There may be more at stake than simply choosing the 'best' overall performance.

Consider as illustrative, the following humorous exchange from the 20th Century Fox film 'Master and Commander: The Far Side of the World', when in the officers mess Captain Aubrey points to a plate of bread and asks Dr Maturin,

J. Scholz (✉)

Defence Science and Technology Group, Joint and Operations Analysis Division,
Box 1500 Edinburgh, SA, Australia
e-mail: jason.scholz@defence.gov.au

Do you see those two weevils doctor?

I do.

Which would you choose?

(sighs annoyed) Neither; there is not a scrap a difference between them. They are the same species of Curculio.

If you had to choose. If you were forced to make a choice. If there was no other response...

(Exasperated) Well then if you are going to push me... (the doctor studies the weevils briefly)

...I would choose the right hand weevil; it has... significant advantage in both length and breadth.

(the captain thumps his fist in the table) There, I have you! You're completely dished! Do you not know that in the service... one must always choose the lesser of two weevils.

(the officers burst out in laughter)

The consequences of a deep network incorrectly classifying a pedestrian as a rubbish bin may have dire consequences in a potential driver-less vehicle accident, yet incorrectly classifying a post box as a rubbish bin may be of lesser concern. Further, the nature of those consequences will be application specific and thus empirically driven. In mission and safety critical applications when the consequences of error are high, the semantic coding of errors for a machine to learn more and less important costs is an important problem. One would ideally like the designer and/or user of systems to be able to specify a profile for acceptability of error consequences for the system application at hand and is critical to trust justifications. Errors will be a reality for deployed network solutions. Despite the performance claims of Deep Networks and the rush to multi-layer engineering with GPU implementations, the vast majority of techniques remain brittle. Small changes can result in magnified errors undermining trustworthiness in the accuracy and competence of such systems. This brittleness is evident when a network trained on given data sets is exposed to independently-derived dataset. Seewald [12] illustrates an order of magnitude in error degradation when a classifier trained on MNIST and USPS datasets is tested on the independently-derived DIGITS handwritten digits dataset. Further, it appears that adversarial examples may be created that have deleterious error effects on practically all forms of network architecture, both deep and shallow [6]. Although adversarial forms appear to be rare in naturally-occurring data, they can be easily generated by an adversary in ways that are indistinguishable to humans [15], which may undermine trustworthiness in the integrity of the system.

The literature on ‘cost-sensitive classification’ (CSS) is concerned with class distribution imbalance, typical of situations when the training data is limited or not necessarily statistically representative, and for classifying rare and important classes (e.g. medical diagnosis of rare diseases). Despite this specific focus, the techniques may be applicable to objective functions for more general deliberate error shaping, as they are generally formulated for a cost C of confusing the actual class i with estimated class j and probability of class j given training example x ,

$$L(x, i) = \sum_j P(j|x)C(i, j) \quad (13.1)$$

Approaches to CSS include reweighting (stratifying) available training data so that more costly errors will incur a larger overall cost [5], cost-sensitive boosting [13], [4] that combine multiple weak or diverse learners, or by changing the learning algorithm. Those that alter the learning algorithm appear to focus on either decision tree classifiers where cost sensitivity is achieved by pruning [10], or Support Vector Machines (SVM) that all use a hinge loss approximation of the cost-sensitive loss function [9]. A cost sensitive learning algorithm applicable as an augmentation to deep networks appeared only recently in the literature [7] and as with all the CSS has not considered the broader problem of shaping the entire error distribution, but only to make up for imbalance in the input class distribution.

The feasibility of a simple augmentation to networks to learn user-defined error profiles is examined next on the basis of a new proposed maximum likelihood objective function. The new technique is developed and explained in the context of binomial and multinomial regression. An implementation using Google's TensorFlow is then studied. A range of experiments are conducted with several independent data sets to ascertain the degree of control over error distributions where the prior distribution is unknown, on a shallow and a deep network architecture and findings are discussed.

13.2 Foundations

A derivation of maximum likelihood classifiers to match a target error profile follows. Noting, in this preliminary study, there is not yet any established theory as to how the overlaps in classification can be traded off, so the examination will necessarily be empirical.

In supervised learning, the Error Matrix, or Confusion Matrix indicates the correct and error classifications across categories from either training or test data. Each row represents the number of instances in an actual (true) class and each column represents the number of instances in an estimated class. The term confusion is used, as this representation quickly shows how often one class is confused with another. Type I errors refer to a true class X being incorrectly classified as a different class Y and is indicated in the non-diagonal values in the rows of the confusion matrix. Type II errors refer to a classified class being X when the true class was Y , and is indicated in the non-diagonal column values of the confusion matrix.

Logistic Regression for the binomial case (two class) and multinomial [3] have been well studied in the literature. However, in order for the reader to comprehend the gradient function for the pairing of the multi-class logistic function also termed 'softmax' [14] function with alternate objective functions, it is necessary to derive these from first principles. Multinomial logistic regression is also extended to the Gaussian case to gain insight and contrast the form of the gradient function with our proposed objective. The following forms a foundation for the proposed technique for multinomial softmax regression on the confusion matrix to follow in the next section.

13.2.1 Binomial Logistic Regression

Consider a network with input $\mathbf{x}_n$ for the n th data presentation, weight vector $\mathbf{w}$, and bias vector $\mathbf{b}$.

$$\mathbf{z}_n = \mathbf{w}^T \mathbf{x}_n + \mathbf{b} \quad (13.2)$$

Consider a simple binary decision classifier, with a logistic function non-linearity,

$$y_n = \sigma(z_n) = \frac{1}{1 + e^{-z_n}} \quad (13.3)$$

In training the network with N examples, the probability of the true class t given the estimated class y output for training case n is,

$$p(\mathbf{t}|\mathbf{y}) = \prod_{n=1}^N p(t_n|y_n) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n} \quad (13.4)$$

To maximise this probability, it is equivalent to minimise the negative log likelihood termed the loss function,

$$E = -\log p(\mathbf{t}|\mathbf{y}) = -\sum_{n=1}^N \log p(t_n|y_n) = \sum_{n=1}^N t_n \log y_n + (1 - t_n) \log(1 - y_n) \quad (13.5)$$

When the gradient of the loss function is zero, this corresponds to the minimum.

$$\frac{\partial E}{\partial y_n} = -\frac{t_n}{y_n} + \frac{1 - t_n}{1 - y_n} = \frac{y_n - t_n}{y_n(1 - y_n)} \quad (13.6)$$

Now consider the chain rule,

$$\frac{\partial E}{\partial \mathbf{w}} = \frac{\partial E}{\partial y_n} \frac{\partial y_n}{\partial z_n} \frac{\partial z_n}{\partial \mathbf{w}} \quad (13.7)$$

Differentiating (13.2),

$$\frac{\partial y_n}{\partial z_n} = y_n(1 - y_n) \quad (13.8)$$

Thus from (13.6),

$$\frac{\partial E}{\partial \mathbf{w}} = \frac{y_n - t_n}{y_n(1 - y_n)} y_n(1 - y_n) \mathbf{x}_n = (y_n - t_n) \mathbf{x}_n \quad (13.9)$$

The logistic function matches the negative log likelihood function to achieve a cancellation in terms and a very simple expression for the gradient and thus the minimum point, which occurs precisely when $y_n = t_n$.

13.2.2 Multinomial Logistic Regression

Consider the network as per (13.1), independently duplicated in structure for each class j ,

$$z_{nj} = \mathbf{w}_j^T \mathbf{x}_n + b_j \quad j = 1, 2, \dots, K \quad (13.10)$$

Consider a multi-class decision classifier, with the generalised logistic or 'softmax' function non-linearity, for data presentation n and class j as follows,

$$y_{nj} = \sigma(z_{nj}) = \frac{e^{z_{nj}}}{\sum_{c=1}^K e^{z_{nc}}} \quad j = 1, 2, \dots, K \quad (13.11)$$

Softmax is thus the logistic function extended to one output for each class. This generalises the previous binary case where $K = 2$. Noting each class output is normalised with respect to all other classes, and therefore is not independent.

As before, in training the network with N examples, the probability of the true class t (a 'one-hot' vector) given the estimated output y for training case n and class j might be described with a multinomial probability mass function, normalised such that $\sum_j t_j = 1$

$$p(\mathbf{t}|\mathbf{y}) = \prod_{n=1}^N p(\mathbf{t}_n|\mathbf{y}_n) = \prod_{n=1}^N \prod_{j=1}^K y_{nj}^{t_{nj}} \quad (13.12)$$

To maximise this likelihood, it is equivalent to minimise the negative log likelihood,

$$E(\mathbf{w}) = -\log p(\mathbf{t}|\mathbf{y}) = -\sum_{n=1}^N \log p(\mathbf{t}_n|\mathbf{y}_n) = -\sum_{n=1}^N \sum_{j=1}^K t_{nj} \log y_{nj}(\mathbf{w}_j) \quad (13.13)$$

This log-likelihood function is often termed cross entropy and can be derived from the Kullback-Liebler divergence. Where the gradient of the negative log likelihood function is zero, $\frac{\partial E}{\partial \mathbf{w}} = 0$ corresponds to the minimum. This is pursued in stages,

$$\frac{\partial E}{\partial y_{nk}} = -\frac{t_{nk}}{y_{nk}} \quad (13.14)$$

Recalling the softmax function normalises with regard to other classes, means that the derivative of the softmax has two cases,

$$\frac{\partial y_{nk}}{\partial z_{nj}} = \begin{cases} y_{nk}(1 - y_{nk}) & j = k \\ -y_{nj}y_{nk} & j \neq k \end{cases} \quad (13.15)$$

The chain rule must account for K class paths, however, the expression may be simplified according to $j = k$ and $j \neq k$ parts only,

$$\frac{\partial E}{\partial \mathbf{w}} = \frac{\partial E}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial z_{nk}} \frac{\partial z_{nk}}{\partial \mathbf{w}_k} + \sum_{j:j \neq k} \frac{\partial E}{\partial y_{nj}} \frac{\partial y_{nj}}{\partial z_{nj}} \frac{\partial z_{nj}}{\partial \mathbf{w}_j} \quad (13.16)$$

$$= -\frac{t_{nk}}{y_{nk}} y_{nk}(1 - y_{nk}) \mathbf{x}_n + \sum_{j:j \neq k} \frac{t_{nj}}{y_{nj}} y_{nj} y_{nk} \mathbf{x}_n \quad (13.17)$$

$$= t_{nk} \mathbf{x}_n (y_{nk} - 1) + y_{nk} \mathbf{x}_n \sum_{j:j \neq k}^K t_{nj} \quad (13.18)$$

$$= \mathbf{x}_n \left(-t_{nk} + y_{nk} \sum_{j=1}^K t_{nj} \right) \quad (13.19)$$

Thus,

$$\frac{\partial E}{\partial \mathbf{w}} = \mathbf{x}_n (y_{nk} - t_{nk}) \quad (13.20)$$

This is the same form as for the binary case. As Eq. (13.16) makes clear from the cancellation of terms, the gradient of the softmax function *perfectly* matches the gradient of the log likelihood objective function, conspiring so as to form a significant reduction. Only one element of the training and the output is used, representing the single true (k th) condition.

Having established this foundation, alternative objective functions are considered next.

13.2.3 Multinomial Softmax Regression for Gaussian Case

Consider the case where y are Gaussian distributed. Such a distribution may not be an unreasonable approximation in the case of the limit for very large training sets. This will allow investigation of the less-than-perfect match of softmax function with this new objective function.

$$p(\mathbf{t}|\mathbf{y}) = \prod_{n=1}^N p(\mathbf{t}_n|\mathbf{y}_n) = \prod_{n=1}^N \prod_{j=1}^K e^{\frac{-(y_{nj}-t_{nj})^2}{2\pi\sigma^2}} \quad (13.21)$$

To maximise this likelihood, it is equivalent to minimise the negative log likelihood,

$$E(\mathbf{w}) = -\log p(\mathbf{t}|\mathbf{y}) = -\sum_{n=1}^N \log p(\mathbf{t}_n|\mathbf{y}_n) = \frac{1}{2\pi\sigma^2} \sum_{n=1}^N \sum_{j=1}^K (t_{nj} - y_{nj}(\mathbf{w}_j))^2 \quad (13.22)$$

Instead of the logarithmic relationship to the outputs y , previously, there is now a square law relationship.

As previously, the gradient of the negative log likelihood function is zero, $\frac{\partial E}{\partial \mathbf{w}} = 0$ corresponds to the minimum. Which is pursued in stages,

$$\frac{\partial E}{\partial y_{nk}} = t_{nk} - y_{nk} \quad (13.23)$$

Using Eqs. (13.14) and (13.15) as before,

$$\frac{\partial E}{\partial \mathbf{w}} = (t_{nk} - y_{nk}) y_{nk} (1 - y_{nk}) \mathbf{x}_n - \sum_{j:j \neq k}^K (t_{nj} - y_{nj}) y_{nj} y_{nk} \mathbf{x}_n \quad (13.24)$$

$$= -\mathbf{x}_n y_{nk} \left((y_{nk} - t_{nk}) - \sum_{j=1}^K (y_{nj} - t_{nj}) y_{nj} \right) \quad (13.25)$$

Firstly, note the sign of the gradient is reversed with respect to (13.19). Second, as the minimum is achieved only when the gradient is zero, this occurs *iff* $t_{nj} = y_{nj}$, $\forall j$. This is notably different to the previous form (13.19), which showed no dependency to any other than the k^{th} estimates of y and training values t . Indeed, even if $t_{nk} = y_{nk}$ a gradient remainder is left over.

Notably, if a one-hot vector is used for training (which is usual for multinomial classifiers), then $t_{nj} = 0$ if $j \neq k$ so,

$$\frac{\partial E}{\partial \mathbf{w}} = -\mathbf{x}_n y_{nk} \left((y_{nk} - t_{nk})(1 - y_{nk}) - \sum_{j=1:j \neq k}^K y_{nj}^2 \right) \quad (13.26)$$

This clearly shows that the incorrect estimate outputs from the right-hand term in (13.25) introduce an irreducible offset into the gradient. However, if the classifier is working well then $y_{nj}^2 \ll 1$ and thus this term may be small.

13.3 Multinomial Softmax Regression on Confusion

A proposal to effect control over the confusion matrix is examined. The confusion matrix summarises the overall distribution of examples from trained classes $\mathbf{t}$ versus estimated classes $\mathbf{y}$. Consider the potential to learn a user-defined target confusion matrix distribution $\mathbf{u}$. Using Bayes theorem this means maximising,

$$p(\mathbf{u}|\mathbf{t}, \mathbf{y}) = \frac{p(\mathbf{t}, \mathbf{y}|\mathbf{u})p(\mathbf{u})}{p(\mathbf{t}, \mathbf{y})} \quad (13.27)$$

Noting the matrix product of $\mathbf{t}$ and $\mathbf{y}$ defines the confusion matrix for some representative data batch size N , writing the likelihood as,

$$p(\mathbf{t}, \mathbf{y}|\mathbf{u}) = \prod_{i=1}^K \prod_{j=1}^K \left(\sum_{m=1}^N t_{im} y_{mj} \right)^{u_{ij}} \quad (13.28)$$

Unlike previous formulations this likelihood directly mixes the estimator output and training classes within the argument of the logarithm. The negative log likelihood is thus,

$$E = -\log p(\mathbf{t}, \mathbf{y}|\mathbf{u}) = -\sum_{i=1}^K \sum_{j=1}^K u_{ij} \log \left(\sum_{m=1}^N t_{im} y_{mj} \right) \quad (13.29)$$

In this form it resembles the multinomial regression objective, with an important distinction in the log over a training-weighted sum of estimated $\mathbf{y}$ values. Where the gradient of the negative log likelihood function is zero, $\frac{\partial E}{\partial \mathbf{w}} = 0$ corresponds to the minimum. Which as previously is pursued in stages,

$$\frac{\partial E}{\partial y_{nj}} = -\sum_{i=1}^K \left(\frac{u_{ij} t_{in}}{\sum_{m=1}^N t_{im} y_{mj}} \right) \quad (13.30)$$

Equations (13.14) and (13.15) similarly apply,

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{w}} = & -\sum_{i=1}^K \left(\frac{u_{ik} t_{in}}{\sum_{m=1}^N t_{im} y_{mk}} \right) y_{nk} (1 - y_{nk}) \mathbf{x}_n \\ & + \sum_{j:j \neq k}^K \left\{ \sum_{i=1}^K \left(\frac{u_{ij} t_{in}}{\sum_{m=1}^N t_{im} y_{mj}} \right) y_{nj} y_{nk} \mathbf{x}_n \right\} \end{aligned} \quad (13.31)$$

$$\begin{aligned}
&= \mathbf{x}_n y_{nk} \sum_{i=1}^K \left(\frac{u_{ik} t_{in}}{\sum_{m=1}^N t_{im} y_{mk}} \right) (y_{nk} - 1) \\
&\quad + \mathbf{x}_n y_{nk} \sum_{j:j \neq k}^K \sum_{i=1}^K \left(\frac{u_{ij} t_{in}}{\sum_{m=1}^N t_{im} y_{mj}} \right) y_{nj}
\end{aligned} \tag{13.32}$$

$$\frac{\partial E}{\partial \mathbf{w}} = \mathbf{x}_n y_{nk} \left(\sum_{j=1}^K \sum_{i=1}^K \left(\frac{u_{ij} t_{in} y_{nj}}{\sum_{m=1}^N t_{im} y_{mj}} \right) - \sum_{i=1}^K \frac{u_{ik} t_{in}}{\sum_{m=1}^N t_{im} y_{mk}} \right) \tag{13.33}$$

This does not reduce further. If $t_{in} = 0$ for all but the $i = k$ training element (one hot training vector) then from (13.31),

$$\frac{\partial E}{\partial \mathbf{w}} = \mathbf{x}_n y_{nk} \left(\sum_{j=1: j \neq k}^K \left(\frac{u_{kj} t_{kn} y_{nj}}{\sum_{m=1}^N t_{km} y_{mj}} \right) + (y_{nk} - 1) \frac{u_{kk} t_{kn}}{\sum_{m=1}^N t_{km} y_{mk}} \right) \tag{13.34}$$

The right hand term is zero iff $y_{nk} = 1$. This will only be the case if all $y_{nj} = 0, \forall j \neq k$. Indeed the left hand term is zero if the latter condition is true, making the left and right hand terms coupled. An alternative condition that makes the left hand term equal to zero is if $u_{kj} = 0, \forall j \neq k$ that is, all non-diagonals of the user-defined matrix are zero (i.e. u is the identity matrix). Classification performance cannot be expected to equal that of the optimum multinomial case except in this special case. Similar to the Gaussian case prior, then the gradient cannot reach a minimum. Thus *any* choice of user-defined error confusions will compromise overall performance. A question remaining is to what degree will that performance be compromised and under what conditions might that compromise be acceptable?

13.4 Implementation and Results

An implementation was chosen to make an empirical study of classifier performance. The MNIST digits database [8], was chosen due to its requiring a multinomial classifier with modest computational requirements. The error objective that a user or designer might want could take a large variety of forms, so instead of attempting some exhaustive approach, a few likely error-shaping scenarios were chosen. The plan for the empirical testing to follow is then:

- Simple Classifier Evaluation
 - ‘Baseline’ to establish whether the confusion objective performs as well as the standard classifier under conditions when a user does not care about the error distribution.

- ‘Error trading’ to examine the capacity to trade error types I and II related to specific classification classes.
- Deep Network Classifier Evaluation
 - Repeat of ‘error trading’ used for the simple network to examine if a deeper classifier structure provides a greater capacity to trade error types I and II.
 - ‘Adversarial Errors’ examines the potential to thwart classification decisions by producing deliberate errors.

The first candidate was a simple regression network from the TensorFlow tutorial [1]. TensorFlow is an open source software library well suited to fast multidimensional data array (tensor) processing and allows deployment of computation on multiple CPUs or GPUs. The simple network uses 784 (28×28 pixels) inputs and $n = 10$ output nodes, each output employing a softmax non-linear function (corresponding to MNIST digit categories 0–9). As is typical with this dataset, 60,000 training images and 10,000 test images were used. A batch size of $m=200$ was maintained throughout. Only the objective function and optimizer were modified, with the 10×10 user-defined error distribution tensor, $\mathbf{u}$ according to the specific applied test. Given the batch of true labels $y_{\text{--}}$

```
y_ = tf.placeholder(tf.float32,[None,10])
```

Using the Adaptive Gradient optimizer,

```
train_step = tf.train.AdagradOptimizer(0.01).minimize(loss)
```

The following implements the extended square loss,

```
loss = tf.reduce_sum(tf.square(u - tf.matmul(y_,y,True,False)))
```

The following implements the extended cross-entropy loss,

```
loss = -tf.reduce_sum(u*tf.log(tf.matmul(y_,y,True,False)))
```

Noting $*$ represents element-wise multiplication of tensors.

Seewald [12] compared machine learning performance on three independently sourced handwritten digits databases: MNIST [8]; USPS, the US Postal Service ‘zip’ codes reduced to individual digits¹; and DIGITS, Seewald’s own collection from school students. The value of independent data rests in the fact that the off-line machine training may differ from the real world when a system is deployed. By studying the proposed error redistribution learner on independent data, its true robustness may be studied.

Table 13.1 shows the results of a baseline test to compare the simple classifier overall performance for cross entropy, square loss and the confusion objective functions as detailed in previous sections. The latter novel objective employed a 10×10 identity matrix as the user objective. The network was trained on the MNIST 60,000

¹<http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/zip.digits/>.

Table 13.1 Comparison of classification performance, trained on MNIST data only

Objective/test data	Cross entropy	Square loss	Confusion objective as identity
MNIST	0.926	0.933	0.926
USPS	0.771	0.767	0.755
DIGITS	0.687	0.688	0.682

Table 13.2 Test 1. Confusion matrix for MNIST test data with a simple softmax regression network and standard cross-entropy objective function

Digit	0	1	2	3	4	5	6	7	8	9	Total	$p(i i)$
0	965	0	2	1	0	2	7	1	2	0	980	0.985
1	0	1106	4	5	1	2	4	2	11	0	1135	0.974
2	12	3	921	13	14	4	12	12	38	3	1032	0.892
3	2	1	22	916	0	21	4	10	25	9	1010	0.907
4	1	2	4	0	926	0	11	2	5	31	982	0.943
5	8	1	2	30	12	780	16	5	30	8	892	0.874
6	12	3	4	2	9	13	910	1	4	0	958	0.950
7	2	7	23	6	10	0	0	948	3	29	1028	0.922
8	4	3	7	15	10	21	10	12	888	4	974	0.912
9	11	5	2	11	32	14	0	13	8	913	1009	0.905
Total	1017	1131	991	999	1014	857	974	1006	1014	997	10000	N/A

image set only but tested on the MNIST 10,000 image test set, USPS 2,007 image test set and the entire DIGITS 4389 images. Table 13.1 demonstrates the expected degradation in performance for unseen data sets reported by Seewald, and that the performance of the novel objective function is comparable under baseline conditions.

The confusion matrix is shown in Table 13.2 with actual digit categories by row and estimated digit categories by column for MNIST training only on the 10,000 image test data set.

As illustrated in the row totals, not all digits are equally likely in the training data and, of course, not all were equally ‘easy’ for the network to recognise. Correct identification of a digit ‘0’ was highest at $p = 0.985$ and correct identification of a ‘5’ lowest at $p = 0.874$. The five most significant error confusions are (2,8), (9,4), (4,9), (5,8), (5,3).

13.4.1 Error Trading

In the following tests the capacity to trade errors related to a specific class (arbitrarily the digit 4) was examined. Table 13.3 summarises error trading tests with the simple network trained on MNIST data only and tested on MNIST, USPS and DIGITS

Table 13.3 Summary of five error-trading tests across three test data sets, noting the simple network was trained on MNIST training data *only*

Test objective	MNIST test data				USPS test data				DIGITS test data			
	Avg Acc	Prob (4,4)	T-I errs	T-II errs	Avg Acc	Prob (4,4)	T-I errs	T-II errs	Avg Acc	Prob (4,4)	T-I errs	T-II errs
Test 1. Standard x-entropy	0.93	0.94	56	88	0.77	0.79	42	17	0.69	0.82	79	91
Test 2. One-hot vector u(4,4) = 5	0.92	0.98	23	193	0.74	0.9	20	49	0.68	0.93	33	205
Test 3. u: r4 = 0 c4 = 0 u(4,4) = 5 flr = 0.05	0.89	0.97	34	196	0.74	0.84	33	33	0.63	0.88	53	157
Test 4. u: c4 = 0 u(4,4) = 1 flr = 0.05	0.89	0.77	229	10	0.74	0.75	50	15	0.63	0.77	101	45
Test 5. u: c4 = 0 u(4,4) = 2 flr = 0.05	0.89	0.87	126	32	0.74	0.78	44	19	0.63	0.84	72	84

Table 13.4 Test 2. Confusion matrix for MNIST test data illustrates the effect of increasing the one-hot vector for a digit ‘4’ with respect to others

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	960	0	1	2	5	2	7	1	2	0	980
1	0	1103	4	5	1	2	4	2	14	0	1135
2	10	1	913	13	25	4	13	11	38	4	1032
3	2	1	22	918	4	20	1	9	22	11	1010
4	1	1	2	0	959	0	6	2	3	8	982
5	8	2	1	30	26	773	16	5	25	6	892
6	15	3	5	2	13	19	896	0	5	0	958
7	2	7	25	5	19	0	1	943	0	26	1028
8	6	4	8	15	15	18	10	12	881	5	974
9	11	4	4	10	85	13	0	16	10	856	1009
Total	1015	1126	985	1000	1152	851	954	1001	1000	916	10000

data sets. Beginning with test 1, as the error benchmark, which is the standard cross entropy objective function, Table 13.1 shows the probability of correctly classifying a digit 4 given a digit 4 was presented, the Type I error related to the row for the digit 4 and Type II errors related to column for the digit 4, for each test data set. The results of test 1 clearly show that the nature of the errors across independent data sets is highly varied.

In test 2, the relative value of one diagonal cell in the user-defined matrix u with respect to all others was changed. This effect may be achieved by scaling one-hot vector unit values. The result of increasing the weight for digit 4 to be five times higher than any other individual digit is illustrated in Table 13.4. For the MNIST test, the average accuracy was 0.920 down slightly from 0.927. This reduced the errors across the row for the digit 4 from 56 to 23 errors, but in the column for digit 4, errors increased from 88 to 193. A reduction in type I errors is achieved at the expense of an increase in type II errors. This is shown in detail with the confusion matrix for the MNIST test data in Table 13.4.

As an analogy, consider a hunter who has a bias that improves detecting a ‘lion’ in long grass. This bias is better for the survival of the hunter to predation, but comes at the cost of more often perceiving a lion when in fact there was not one, which may have other consequences.

If this test is illustrative of the behaviour to be expected for a classifier, is it possible to exert more shaping control over type I and type II errors as trade for average error performance? Table 13.3 (test 3) summarises results of raising the ‘floor’ (f_{lr}) for diagonals in u as a compensatory allowance. A setting of $f_{lr} = 0.05$ was chosen to represent a desire to keep errors low everywhere (reasoning 0.05 is near to zero), but it is most desired to reduce errors in combinations of row ‘4’ and column ‘4’ where cells were set to zero. Table 13.4 illustrates the resulting confusion matrix for test. Test 3 was typical of attempts to change the same major row and column ($i = j$). The

Table 13.5 Test 5. Confusion matrix for MNIST test data illustrates the ability to reduce type II errors (column ‘4’) by trading a moderate increase in type I errors (row ‘4’)

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	940	1	1	6	0	6	8	3	13	2	980
1	0	1097	1	3	0	10	5	0	19	0	1135
2	7	17	869	23	3	14	9	30	46	14	1032
3	1	5	19	902	0	24	3	17	27	12	1010
4	5	21	9	11	856	6	16	6	23	29	982
5	6	8	5	21	2	772	18	17	30	13	892
6	9	6	7	2	6	33	883	2	8	2	958
7	3	28	21	11	2	4	3	910	7	39	1028
8	12	17	12	23	4	31	9	13	836	17	974
9	17	10	3	25	15	24	7	24	17	867	1009
Total	1000	1210	947	1027	888	924	961	1022	1026	995	10000

result was in increase in both type I *and* type II errors compared with manipulating one hot vector values only. Table 13.3 shows across test data sets a relative increase in Type I errors and decrease in Type II errors compared with test 2. For the MNIST test data the most errors related to cells (9,4) and/or (4,9). Driving down errors for one of these cells had the direct effect of raising errors in the other: a kind of ‘pivot’ effect. The strong and direct relationship between errors appeared to be only very weakly controllable. Evidence of these pivots appear in the unshaped confusion matrix in the form of the larger and symmetrical errors in Table 13.2. Pivots were observed at (4,9):(9,4), (5,8):(8,5), (5,3):(3,5), (7,9):(9,7) for MNIST tests. Notably these pivots were different, for each of the three test data sets.

In test 4, the capacity to target a reduction in type II errors *only* for a specific classification was examined. Table 13.3 summarises the result which had a strong desired effect, at the expense of an increase in type I errors. Test 5 demonstrates a final level of ‘tuning’ where the type II errors were generally lower than those of test 1, with only a small increase in Type I errors. The confusion matrix for test 5 with MNIST test data is shown in Table 13.5.

Tests 4 and 5 demonstrate a level of control over errors that is not possible by one-hot vector manipulation and could not be achieved without shaping the objective function. The set of tests also demonstrates an ability to target reduction in Type I or Type II errors at the expense of the other, and was sustained across two unforeseen independent data sets.

These results led to the question ‘if pivots identified in the initial confusion matrix are excluded from consideration, can significant control over errors in other arbitrary rows and columns be achieved?’ For the tests attempted, the answer appears to be yes. Table 13.6 illustrates the results of an attempt to minimise errors in the column for a digit ‘4’ (type II) and a row for a digit ‘7’ (type I) for MNIST test data. The user defined matrix, u column representing a ‘4’ contained zeros, and the row

Table 13.6 Test 6. Confusion matrix result for MNIST test data with the aim to ‘minimise errors in row 7 without excessive impact on column 4 and the overall error average’

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	925	0	1	5	0	6	4	25	11	3	980
1	0	1094	1	4	0	7	5	5	19	0	1135
2	6	15	855	20	3	9	12	72	34	6	1032
3	1	7	19	870	0	21	3	53	23	13	1010
4	7	21	12	7	848	7	19	10	22	29	982
5	7	8	6	16	2	740	14	63	29	7	892
6	7	7	7	1	5	35	862	27	7	0	958
7	3	8	11	7	5	0	1	979	0	14	1028
8	10	15	11	22	4	24	9	48	816	15	974
9	14	8	2	12	15	16	4	73	17	848	1009
Total	980	1183	925	964	882	865	933	1355	978	935	10000

representing a ‘7’ contained zeros, with $u(4,4)=2$, $u(7,7)=2$, $u(i,i)=1$ if $i \neq 4$ and a floor $u(i,j)$ otherwise equal to 0.05. The overall test accuracy was 0.88. Notably the most significant errors at (9,4) and (7,9) were suppressed as desired. The ability to push these errors lower would not come without more significant cost to the overall error rate. Type II errors increased in column ‘8’.

In practice, users will likely want to achieve a lower misclassification rate for some arbitrary chosen cell targets relevant to their problem choice. Empirical investigation appears to confirm this is possible providing data pivot points are avoided. This is discussed later.

13.4.2 Performance Using a Deep Network and Independent Data Sources

The deep convolutional network (Convnet) chosen was from the TensorFlow tutorial.² The Convnet chosen uses a hierarchy of three macro-levels, each level comprises a convolutional layer, rectified linear unit layer, max pool layer, and drop out layer. At the top of all this, there is an output processing layer termed ‘softmax’ or normalised exponential, making 13 layers in total. This provides significantly better performance than the simple network. When trained on MNIST data, this network provided a 99.2% average classification accuracy which is near to the current state of the art of 99.77% [2].

As a comparison, the earlier successful aim to ‘minimise errors in row 7 without excessive impact on column 4 and the overall error average’ (test 6) is revisited.

²<https://www.tensorflow.org/versions/r0.9/tutorials/mnist/pros/index.html>.

Table 13.7 MNIST test errors (softmax with logits)

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	751	0	32	0	5	2	6	2	2	3	803
1	1	754	52	0	72	2	13	0	7	0	901
2	0	0	834	2	1	0	3	1	4	0	845
3	1	0	50	719	2	47	0	8	4	2	833
4	0	0	7	0	793	0	0	0	3	9	812
5	5	0	11	7	2	683	3	4	14	2	731
5	10	0	0	0	85	12	679	0	3	0	789
7	1	4	117	2	5	0	0	714	4	10	857
8	8	3	27	6	9	16	5	3	709	4	790
8	0	5	24	9	31	2	0	18	26	716	831
Total	777	766	1154	745	1005	764	709	750	776	746	8192

In this case it was performed by training the Convnet on *USPS training data only* and then tested on USPS test data, MNIST test data and DIGITS test data. For the experiment 1280 USPS training images only were used. Test data used included 8192 MNIST test images, 1024 USPS test images, and 4096 DIGITS test images. The results are shown in Tables 13.7, 13.8, 13.9, 13.10, 13.11 and 13.12 inclusive. Tables 13.7, 13.9, and 13.11 show the confusion matrix achieved using the standard classifier on MNIST, USPS and DIGITS test databases with average error rates 0.897, 0.957, and 0.692, respectively. These errors are contrasted with Tables 13.8, 13.10, and 13.12 which show the achieved error performance for test 6 using the new objective function on MNIST, USPS and DIGITS test databases with average error rates of 0.784, 0.893 and 0.538 respectively. As expected, test 6 resulted in some increase in the average errors in all cases. Notably, the misclassifications for each of these databases show significantly *different* characteristics to what is now seen trained on USPS data. Pivot points for the MNIST test are different to those earlier, now with confusions (7,2) and (6,4) as most significant. Pivot points for the DIGITS test have significant confusions (7,2), (9,3), (1,4) and (1,7). Examining the objective to minimise errors in row 7, this is demonstrated in all three cases. Error confusion (7,2) were reduced from 117 to 31, and 220 to 130 in the MNIST and DIGITS tests and remained the same at 2 for the USPS test. Errors for column 4 increased from 212 to 460 and 204 to 278 for MNIST and DIGITS tests, and from 3 to 20 for the USPS test. This largely fulfilled the aim of the test, however, there are several notable 'surprises'. Table 13.8 shows significant error increases for (0,2), (0,6), and (1,6) compared with Table 13.7. Table 13.12 shows significant error increases for (0,2), (2,6) and (4,9).

Table 13.8 MNIST test errors for the objective to minimise row 7 errors (test 6)

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	475	2	192	0	23	2	113	1	1	13	822
1	0	636	5	0	159	1	135	0	10	0	946
2	0	0	757	8	31	1	17	14	14	1	843
3	3	1	101	618	3	47	3	21	4	16	817
4	0	0	0	0	775	0	1	10	1	11	798
5	6	0	5	35	5	613	26	11	0	17	718
6	0	0	5	0	157	1	625	0	0	3	791
7	0	3	31	5	11	1	0	765	8	19	843
8	3	1	20	44	23	64	93	27	485	30	790
9	1	3	16	6	48	3	0	58	12	677	824
Total	488	646	1132	716	1235	733	1013	907	535	787	8192

Table 13.9 USPS test errors (softmax with logits)

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	197	0	0	0	0	0	1	1	1	0	200
1	0	128	0	0	0	0	2	1	0	0	131
2	0	1	87	0	1	1	0	1	3	0	94
3	0	0	2	79	0	4	0	0	0	0	85
4	1	2	1	0	99	0	0	0	0	3	106
5	2	0	0	1	0	73	0	0	0	0	76
6	0	0	0	0	1	2	80	0	0	0	83
7	0	0	1	0	1	0	0	58	0	0	60
8	1	2	1	1	0	1	0	0	83	0	89
9	0	1	0	0	0	0	0	2	1	96	100
Total	201	134	92	81	102	81	83	63	88	99	1024

13.4.3 Adversarial Errors

Control over the occurrence of type I errors is most strikingly demonstrated by training a network to create a deliberate confusion. This may be a requirement for a cyber operations system. Table 13.13 illustrates the result to cause a deliberate confusion of a digit ‘7’ for a ‘0’ created by a user matrix. As expected the average error drops by 10% to 0.829. Notably this network learns never to output a ‘7’ and instead classifies a ‘7’ as a ‘0’ precisely as desired. Further, the “normal” detection statistics of other categories is largely unaffected.

To produce type II errors in a specific cell of row i and column j , proved more difficult. It is necessary to maintain at least some small value for $u(j, j)$ to ensure correct classifications of i are allowed as reducing $u(j, j)$ to zero will have the

Table 13.10 USPS test errors for the objective to minimise row 7 errors (test 6)

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	165	2	1	1	3	0	13	1	4	0	190
1	0	130	0	0	1	0	4	0	0	0	135
2	3	0	72	6	6	0	0	0	11	0	98
3	0	0	2	73	1	3	0	2	3	0	84
4	0	0	0	0	88	0	0	4	0	1	93
5	2	0	0	0	1	67	6	2	4	1	83
6	0	0	0	0	3	1	87	0	0	0	91
7	0	0	0	0	2	0	0	72	0	0	74
8	2	0	0	2	1	1	1	1	81	1	90
9	0	0	0	0	2	0	0	4	1	79	86
Total	172	132	75	82	108	72	111	86	104	82	1024

Table 13.11 DIGITS errors (softmax with logits)

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	372	0	17	0	1	3	3	1	11	0	408
1	1	66	8	0	169	0	0	152	1	11	408
2	6	1	366	0	1	1	13	2	20	0	410
3	0	15	13	298	3	63	0	13	4	1	410
4	0	1	0	0	383	9	7	2	2	3	407
5	2	1	2	4	0	371	6	6	16	5	413
6	5	3	0	0	4	27	363	0	2	0	404
7	0	13	220	0	9	0	0	136	10	16	404
8	2	14	42	12	7	23	11	1	304	1	417
9	6	8	22	120	10	10	0	28	33	178	415
Total	394	122	690	434	587	507	403	341	403	215	4096

effect of reducing the entire column to zero, as per the example in Table 13.13, with the resulting distribution of the classification estimates redistributed without user control, into the next most similar categories. Table 13.14 illustrates a best effort test (error average 0.806) achieved with the aim of ‘creating errors in cell (2,5) and minimising correct classifications in cell (5,5)’. As can be seen in Table 13.14, the desired effect is achievable, however, the consequence of the desired weakness in correctly classifying (5,5) produces more significant errors at both (5,3) and (5,8), which may be the ‘next weakest’ points.

Table 13.12 DIGITS errors for the objective to minimise row 7 errors (test 6)

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	235	2	101	2	15	0	52	6	0	3	416
1	2	28	9	0	165	0	17	127	4	55	407
2	0	4	204	1	3	00	127	5	67	0	411
3	0	9	87	253	1	16	13	19	2	12	412
4	0	6	1	0	294	2	17	11	0	77	408
5	0	1	3	16	0	302	52	11	0	19	404
6	1	0	4	0	0	9	381	0	0	11	406
7	0	20	130	16	69	0	0	160	3	15	413
8	0	1	26	26	3	80	75	6	175	19	411
9	1	10	6	90	22	8	0	83	17	171	408
Total	239	81	571	404	572	417	734	428	268	382	4096

Table 13.13 Example confusion matrix for adversarial error creation. Most occurrences of a digit ‘7’ are classified as the digit ‘0’ as desired

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	956	0	1	1	0	4	11	0	7	0	980
1	3	1107	4	3	1	2	5	0	10	0	1135
2	10	2	933	14	12	3	15	0	39	4	1032
3	15	1	20	917	2	22	2	0	21	10	1010
4	2	2	4	0	927	0	11	0	6	30	982
5	30	1	2	26	15	762	15	0	32	9	892
6	19	3	5	0	6	13	907	0	5	0	958
7	929	9	24	9	13	3	1	0	4	36	1028
8	23	3	8	17	8	17	11	0	883	4	974
9	25	4	0	11	30	13	1	0	8	917	1009
Total	2012	1132	1001	998	1014	839	979	0	1015	184	10000

13.5 Discussion

The hypothesis that a more expressive deep network would be significantly more capable of supporting an arbitrary redistribution of errors than a shallow network was not demonstrated in these tests. It does appear possible to trade errors under limited conditions towards arbitrarily-chosen errors which, as expected from theoretical study of the gradients, comes at some cost to the total error rate. The degree of control that can be exercised appears limited by pivots in the presented data. These pivots are in turn dependent on training data and the actual data used in practice. As demonstrated by use of three independent data sets, the ‘confusion training’ to meet an objective *performed best* when the training and live data were drawn from independent sources.

Table 13.14 Adversarial error result for the aim to ‘create errors in cell (2,5) and minimise correct classifications in cell (5,5)’

Digit	0	1	2	3	4	5	6	7	8	9	Total
0	949	0	1	1	0	7	13	1	8	0	980
1	0	1096	3	8	0	6	5	1	16	0	1135
2	5	0	660	5	6	319	8	7	19	3	1032
3	2	1	18	893	1	43	5	10	28	9	1010
4	2	3	1	1	896	13	10	1	14	41	982
5	40	9	1	248	46	39	37	26	425	21	892
6	14	2	1	2	6	22	891	0	20	0	958
7	3	15	41	1	11	14	1	902	2	38	1028
8	13	10	5	22	14	33	9	10	842	16	974
9	8	6	9	14	42	8	0	15	18	889	1009
Total	1036	1142	740	1195	1022	504	979	973	1392	1017	10000

A critical question is ‘what other loss functions might one choose, and what are the implications for that choice?’ [16], Williamsion argues that a proper composite loss (objective) needs to control convexity (geometrical properties) and control statistical properties. Considering logistic loss, the deliberate *perfect* match of the logistic or softmax characteristic and the multinomial log likelihood characteristic that yields a collapse in complexity of the gradient was not afforded by our proposed choice of objective function with softmax. Yet, perhaps it remains possible to find an alternate functional composition that would allow a similar simplification?

The adversarial generation of confusions was convincingly demonstrated with the ‘user specified confusion’ objective function. The confusion objective function is very good at *creating* errors where they are wanted.

13.6 Conclusion

A technique has been demonstrated with the ability to learn to shape errors for both a shallow and deep network based on a novel maximum likelihood ‘confusion objective’ function. Results were demonstrated for some limited but useful cases in trading type I and type II errors, maintaining error objectives across independent and unforeseen data sets, and an ability to create adversarial confusions. Next steps might include tests on examples with a significantly larger number of classes, and the derivation of bounds of the gradient minima for the technique to provide further insights. The technique appears sufficiently promising to warrant more thorough statistical analysis of sensitivity of total errors to specific error objectives.

Acknowledgements Thanks to: Mr. Darren Williams, Dr. Glenn Moy and Dr. Darryn Reid for reviewing earlier drafts and encouragement; Prof. Hussein Abbass for timely and critical feedback on the draft; and Dr. Alexander Seewald for kindly supplying USPS and DIGITS data sets.

References

1. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems. (2016) *arXiv preprint* [arXiv:1603.04467](https://arxiv.org/abs/1603.04467)
2. D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification. *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2012), pp. 3642–3649
3. J.N. Darroch, D. Ratcliff, Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics* (1972), pp. 1470–1480
4. P. Domingos, Metacost: A general method for making classifiers cost-sensitive. *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 1999), pp. 155–164
5. C. Elkan, The foundations of cost-sensitive learning. *International Joint Conference on Artificial Intelligence*, vol. 17 (LAWRENCE ERLBAUM ASSOCIATES LTD, 2001), pp. 973–978
6. I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples (2014). *arXiv preprint* [arXiv:1412.6572](https://arxiv.org/abs/1412.6572)
7. S.H. Khan, M. Bennamoun, F. Soheli, R. Togneri, Cost sensitive learning of deep feature representations from imbalanced data (2015). *arXiv preprint* [arXiv:1508.03422](https://arxiv.org/abs/1508.03422)
8. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)
9. Y. Lee, Y. Lin, G. Wahba, Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *J. Am. Stat. Assoc.* **99**(465), 67–81 (2004)
10. S. Lomax, S. Vadera, A survey of cost-sensitive decision tree induction algorithms. *ACM Comput. Surv. (CSUR)* **45**(2), 16 (2013)
11. J. Schmidhuber, Deep learning in neural networks: An overview. *Neural Netw.* **61**, 85–117 (2015)
12. A.K. Seewald, On the brittleness of handwritten digit recognition models. *ISRN Machine Vision*, 2012 (2011)
13. Y. Sun, M.S. Kamel, A.K.C. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data. *Pattern Recogn.* **40**(12), 3358–3378 (2007)
14. R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction* (MIT press Cambridge, 1998)
15. P. Tabacof, E. Valle. Exploring the space of adversarial images (2015). *arXiv preprint* [arXiv:1510.05328](https://arxiv.org/abs/1510.05328)
16. R. Williamson. *Loss Functions*, vol. 1 (Springer, 2013), pp. 71–80

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.