Gradient-Based Manipulation of Non-Parametric Entropy Estimates

Nicol N. Schraudolph

***Abstract*—We derive a family of differential learning rules that optimize the Shannon entropy at the output of an adaptive system via kernel density estimation. In contrast to parametric formulations of entropy, this nonparametric approach assumes no particular functional form of the output density. We address problems associated with quantized data and finite sample size, and implement efficient maximum likelihood techniques for optimizing the regularizer. We also develop a normalized entropy estimate that is invariant with respect to affine transformations, facilitating optimization of the shape, rather than the scale, of the output density. Kernel density estimates are smooth and differentiable; this makes the derived entropy estimates amenable to manipulation by gradient descent. The resulting weight updates are surprisingly simple and efficient learning rules that operate on pairs of input samples. They can be tuned for data-limited or memory-limited situations, or modified to give a fully online implementation.**

***Index Terms*—Parzen windows, kernel density, maximum likelihood kernel, step size adaptation, expectation-maximization, overrelaxation, affine-invariant entropy, entropy manipulation**

I. Introduction

Since learning is by definition an acquisition of information, it is not surprising that information-theoretic objectives play an important role in machine learning [24]. Although they have been proposed for supervised learning problems as well [14, 24], their particular strength lies in unsupervised learning, due to their ability to quantify information without reference to a particular desired output or behavior. Consider the mutual information between two random variables, defined as the sum of their individual entropies minus their joint entropy:

$$I(A,B) \;=\; H(A) \;+\; H(B) \;-\; H(A,B)\,. \qquad (1)$$

In contrast to, *e.g.*, correlation, which measures the degree to which a linear relationship between two variables is present, mutual information provides a measure of relatedness between A and B which does not presuppose any particular form of relationship. This is obviously very useful when the specific relationship between A and B is *a priori* unknown.

1) Three Approaches: Researchers have used mutual information as an objective for machine learning in (at least) three different ways. Linsker [20] proposed maximizing the mutual information between input and output of an adaptive system. In this context, mutual information is usefully reformulated as

$$I(A,B) \;=\; H(A) \;-\; H(A|B)\,, \qquad (2)$$

Manuscript received January 2003; revised October 2003.

The author is an independent consultant in Zürich, Switzerland. (email: neo@schraudolph.org)

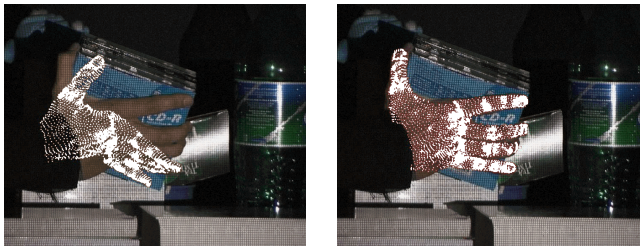


Fig. 1. Tracking an articulated 3D hand model in video frames. The correctly aligned model (right) has higher mutual information between the orientation of its surface normals and the intensity of the corresponding image pixels than the misaligned one (left).

Images courtesy of Matthieu Bray, ETHZ.

where $H(A|B) = H(A,B) - H(B)$ is the conditional entropy of A given B. The mutual information between input and output of a system can thus be interpreted as an information gain [33] at the output, *i.e.*, a drop in the entropy of the output density A when a specific input drawn from B is applied. Its maximization is known as the principle of maximum information preservation or "Infomax" [20], motivated by a desire to preserve the essence of a signal as it moves through a succession of processing stages.

A second approach is to minimize the mutual information between the outputs of an adaptive system in order to obtain a non-redundant code. This strategy has been variously rediscovered and named: redundancy reduction [2, 3], factorial code learning [4], predictability minimization [27, 28, 32], and independent component analysis [7, 10]; it can be used to perform blind separation and deconvolution of signals.

A third way to use mutual information in machine learning is to maximize it between the output of an adaptive system and some reference data which does not belong to its inputs (as in Infomax), but still relates to them in some (perhaps unknown) way. For instance, the reference may be a class label [37], or data from a different modality but describing the same object or event as the input. By maximizing mutual information, the adaptive system leverages this relationship to align its output with the reference. This can be a very powerful technique, and has been used extensively in computer vision [41–43]. A variation on this theme is to have two separate adaptive systems, receiving separate inputs, maximize the mutual information between their outputs [5].

2) Visual Tracking: Let us illustrate the above with an example. Say the objective is to track a user's hand in a video sequence, as shown in Fig. 1. To achieve this, an articulated hand model, with up to 30 degrees of freedom specifying position, orientation, and joint angles, must be aligned with the hand's image in each video frame. Conventionally, this could

be done by rendering the hand model, then using the difference between the predicted and actual image of the hand as an error signal. This has the major disadvantage, however, that rendering requires detailed information about the hand's color, texture, lighting, *etc.*, which is available only in the most controlled lab settings.

Instead, we can exploit the fact that mutual information does not presuppose a particular form of relationship: pick a random sample of points on the surface of the hand model, and maximize the mutual information between the orientation of their surface normals and the intensity of the corresponding pixel in the video image. As long as there is any consistent relationship between surface orientation and brightness (*i.e.*, the lighting is directional), mutual information will align the model with the image so as to uncover that relationship. A similar technique can also be used to align two images from different modalities, for example in medical image registration.

3) Simplifications: While information theory thus provides a very general, productive framework for deriving objectives for machine learning, its full generality and power is rarely realized, due to a number of difficulties. First, mutual information and the entropies from which it is composed are defined as functions of a probability density. In machine learning the input density is typically not available, and must first be estimated from data. This complication has caused many researchers to simplify matters by making strong distributional assumptions (such as gaussianity) about their data [20], or to restrict their adaptive system to only produce certain output distributions (*e.g.*, binary stochastic neurons [5]). Such parametrizations of the density sacrifice much of the generality of the information-theoretic approach, sometimes going so far as to effectively reduce entropy to a glorified measure of variance.

A second problem lies in the fact that for continuous distributions, entropy is not scale-invariant. A system capable of scaling its outputs can therefore maximize or minimize its output entropy in a trivial way. This is often countered by restricting the output range of the adaptive system, *e.g.*, with a sigmoid output nonlinearity [5, 7]. Finally, quite often only the joint entropy is optimized, instead of the full mutual information [7, 39]. This can lead to various problems; for instance, it is the reason why the independent component analysis algorithm of Bell and Sejnowski [7] could originally only handle supergaussian inputs.

In this paper, we try to avoid these kinds of simplifications in order to derive a completely general framework for estimating and manipulating entropies at the output of an adaptive system. In Section II we construct a differentiable estimate of the data density in nonparametric fashion, making no distributional assumptions other than smoothness. We pay particular attention to the optimization of the shape of the kernels used in this process. In Section III we formulate Shannon entropy in terms of this density estimate, and derive its gradient for purposes of entropy manipulation. We also provide an affine-invariant entropy that can safely be extremized by systems capable of scaling their outputs, and discuss a number of implementation issues. Section IV then summarizes and concludes our paper.

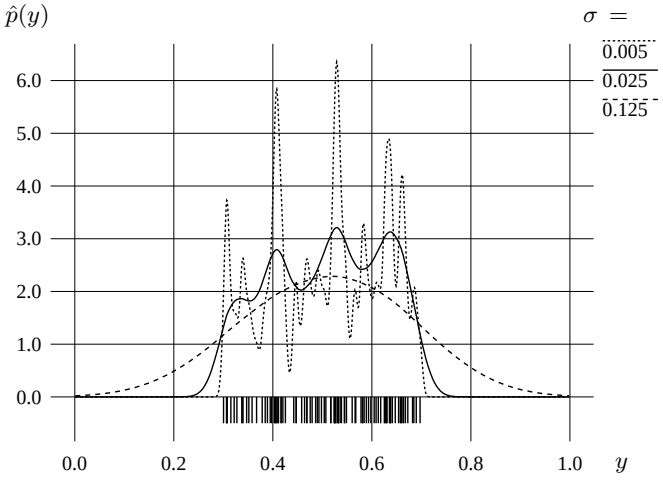


Fig. 2. Kernel density estimate $\hat{p}(y)$ for 100 points (shown as vertical bars) randomly sampled from a uniform distribution over the interval [0.3,0.7]. Depending on the kernel width σ, $\hat{p}(y)$ may be underregularized (dotted line), overregularized (dashed line), or "just right" (solid line).

II. Nonparametric Density Estimation

Since entropy is defined in terms of probability density, a nonparametric estimate of the density of a given data sample must be obtained first. To facilitate gradient-based entropy manipulation the density estimate should be smooth and differentiable; this excludes from consideration techniques that produce piecewise constant density estimates, such as histograms and sample spacings [6]. For large data samples, semi-parametric density estimation methods such as expectation-maximization (EM) [8, 11] or the self-organizing maps of van Hulle [39, 40] can be useful; for our purposes, however, kernel density estimation provides a simpler, fully nonparametric alternative.

A. Kernel Density Estimation

Parzen [23] window or kernel density estimation assumes that the probability density is a smoothed version of the empirical sample [12, chapter 4.3]. Its estimate $\hat{p}(\boldsymbol{y})$ of the density $p(\boldsymbol{y})$ of a random variable Y is simply the average of radial kernel functions K centered on the points in a sample T of instances of Y:

$$\hat{p}(\boldsymbol{y}) \;=\; \frac{1}{|T|} \sum_{\boldsymbol{y}_j \in T} K(\boldsymbol{y} - \boldsymbol{y}_j)\,. \tag{3}$$

This kernel density is an unbiased estimate for the true density of Y corrupted by noise with density equal to the kernel (or window) function K. We will use the multivariate Gaussian kernel

$$K(\boldsymbol{y}) \;=\; N(\boldsymbol{0}, \boldsymbol{\Sigma}) \;=\; \frac{\exp(-\frac{1}{2}\,\boldsymbol{y}^T \boldsymbol{\Sigma}^{-1}\,\boldsymbol{y})}{(2\pi)^{\frac{n}{2}}\,|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \tag{4}$$

with dimensionality n and covariance matrix $\boldsymbol{\Sigma}$. Other choices for K are possible but will not be pursued here, though we will augment (4) for quantized data.

It can be shown that under the right conditions $\hat{p}(\boldsymbol{y})$ will converge to the true density $p(\boldsymbol{y})$ as $|T| \to \infty$. For our Gaussian

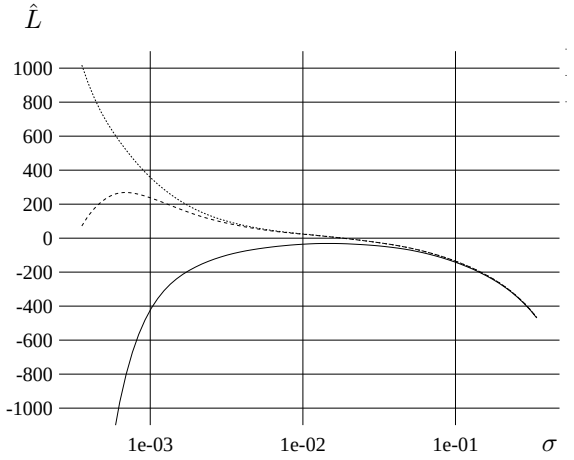


Fig. 3. Estimated log-likelihood $\hat{L}$ *vs.* kernel width σ for 1 000 points drawn from a uniform scalar density (true likelihood: $L = 0$), quantized to 3 decimal places ($b = 0.001$). Solid line is the correct estimate, using (5) with $T_i = S \setminus \{y_i\}$ and kernel (6) with $\kappa = 1/6$. Dashed line shows the distortion caused when $T_i = S$, failing to omit the diagonal terms from the double summation in (5). Dotted line results from using the simpler kernel (4) which fails to take the quantization of the data into account.

kernels, these conditions can be met by letting the covariance $\mathbf{\Sigma}$ of the kernel shrink to zero slowly enough as the sample size approaches infinity. The covariance is an important regularization parameter in any event, as it controls the smoothness of the kernel density estimate. Fig. 2 illustrates this in one dimension: When the kernel width $\sigma = \sqrt{\Sigma}$ is too small (dotted line), $\hat{p}(y)$ overly depends on the particular sample T from which it was computed, and the density estimate is underregularized. Conversely, when σ is too large (dashed line), $\hat{p}(y)$ is overregularized — it becomes insensitive to T, taking on the shape of the kernel function regardless of the true density $p(y)$. Between these extremes, the kernel that best regularizes the density estimate (solid line) can be found via the maximum likelihood approach.

An empirical estimate of the maximum likelihood kernel is the kernel that makes a second sample S drawn independently from $p(\boldsymbol{y})$ most likely under the estimated density $\hat{p}(\boldsymbol{y})$ computed from the first sample, T. For numerical reasons it is preferable to maximize the empirical log-likelihood

$$\begin{aligned} \hat{L} &= \ln \prod_{\boldsymbol{y}_i \in S} \hat{p}(\boldsymbol{y}_i) = \sum_{\boldsymbol{y}_i \in S} \ln \hat{p}(\boldsymbol{y}_i) \\ &= \sum_{\boldsymbol{y}_i \in S} \ln \sum_{\boldsymbol{y}_j \in T} K(\boldsymbol{y}_i - \boldsymbol{y}_j) - |S| \ln |T| \,. \end{aligned} \tag{5}$$

B. Quantization and Sampling Issues

The estimated kernel log-likelihood (5) assumes two independent samples from a continuous density. In practice, empirical data does not conform to these conditions: we are likely to be given a finite set of quantized data points to work with. Both quantization and finite sample size can severely distort $\hat{L}$; we now introduce ways to correct for these two problems.

Consider the plot of estimated log-likelihood *vs.* kernel width shown in Fig. 3. The uniform density we are sampling from is given by $p(y) = \text{const.} = 1$, and so the true log-likelihood is zero, yet its estimate computed according to (5), using the kernel (4), monotonically increases for ever smaller kernel widths (dotted line). The culprit is quantization: the sample points were given with three significant digits, that is, quantized into bins of width $b = 0.001$. When several samples fall into the same bin, they end up being considered *exactly* the same. In a real-valued space, such a coincidence would be nothing short of miraculous, and in an attempt to explain this miracle, maximum likelihood infers that the density must be a collection of delta functions centered on the sample points. This is of course unsatisfactory.

We can correct this problem by explicitly adding the quantization noise back into our quantized sample. That is, for a quantization bin width of $\boldsymbol{b}$, we replace (4) with

$$K(\boldsymbol{y}) = \frac{\exp[-\frac{1}{2}\,(\boldsymbol{y}^T \mathbf{\Sigma}^{-1}\,\boldsymbol{y} + \kappa\,\boldsymbol{b}^T \mathbf{\Sigma}^{-1} \boldsymbol{b})]}{\sqrt{(2\pi)^n\,|\mathbf{\Sigma}|}} \tag{6}$$

where κ must be chosen to appropriately reflect the quantization: to evaluate the kernel density at an arbitrary (*i.e.*, non-quantized) point $\boldsymbol{y}$, we assume that each data point $\boldsymbol{y}_j$ is uniformly distributed over its bin; the resulting variance is accounted for by setting $\kappa = 1/12$. This value must be doubled to $\kappa = 1/6$, however, if the point $\boldsymbol{y}$ where the density is evaluated is likewise a quantized data point, as is the case for the log-likelihood (5) *resp.* empirical entropies (24), (26) considered here.[1]

Fig. 3 shows that with the addition of quantization noise variance (dashed line), the log-likelihood estimate does acquire an optimum. However, it is located at a kernel width less than b, so that there is no significant smoothing of the quantized density. Furthermore, the empirical log-likelihood at the optimum is far from the true value $L = 0$. This indicates that another problem remains to be addressed, namely that of finite sample size. The log-likelihood estimate (5) assumes that S and T are *independently* drawn samples from the same density. In practice we may be given a single, large sample from which we subsample S and T. However, if this is done with replacement, there is a non-zero probability of having the same data point in both S and T. This leads to a problem similar to that of quantization, in that the coincidence of points biases the maximum likelihood estimate towards small kernel widths. The dashed line in Fig. 3 shows how the diagonal terms in $S \times T$ distort the estimate when both samples are in fact identical ($S = T$).

This problem can be solved by prepartitioning the data set into two subsets, from which S and T are subsampled, respectively ("splitting data estimate" [6]). In a data-limited situation, however, we may not be able to create two sufficiently large sets of samples. In that case we can make efficient use of all available data while still ensuring $S \cap T = \emptyset$ (*i.e.*, no sample overlap) through a technique borrowed from leave-one-out crossvalidation: for each $y_i \in S$ in the outer sum of (5), let the inner sum range over $T_i = S \setminus \{y_i\}$ ("cross-validation estimate" [6]). The solid line in Fig. 3 shows the estimate obtained with this technique; at the optimum the estimated log-likelihood comes very close to the true value of zero.

[1] The value of $\kappa = 1/4$ given in [29] is incorrect.

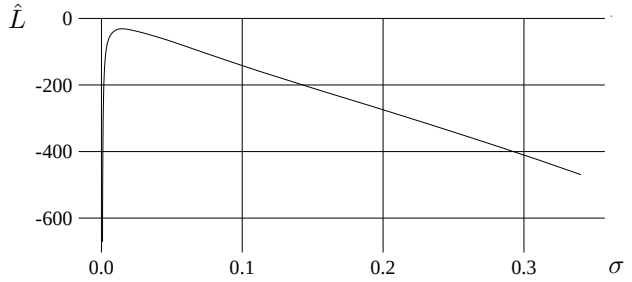


Fig. 4. The correct log-likelihood estimate $\hat{L}$ from Fig. 3, plotted against kernel width σ on a *linear* scale. This function shape is not amenable to maximization by ordinary gradient descent.

C. Gradient Ascent in Likelihood

Given that the kernel density estimate (3) is differentiable, an obvious way to optimize the kernel shape is by gradient ascent. Since the derivatives of $\hat{L}$ with respect to the elements of the full covariance matrix $\mathbf{\Sigma}$ are rather complicated, we restrict our discussion to diagonal covariance matrices, parametrized by a vector $\boldsymbol{\sigma}$ of kernel widths: $\mathbf{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma}^2)$. The derivative of the kernel (6) with respect to the kth kernel width parameters $[\boldsymbol{\sigma}]_k$ is then

$$\frac{\partial}{\partial[\boldsymbol{\sigma}]_k} K(\boldsymbol{y}) \;=\; \left(\frac{[\boldsymbol{y}]_k^2 \;+\; \kappa\,[\boldsymbol{b}]_k^2}{[\boldsymbol{\sigma}]_k^3} \;-\; \frac{1}{[\boldsymbol{\sigma}]_k} \right) K(\boldsymbol{y})\,, \qquad (7)$$

and that of the log-likelihood $\hat{L}$,

$$\frac{\partial \hat{L}}{\partial[\boldsymbol{\sigma}]_k} = \sum_{\boldsymbol{y}_i \in S} \sum_{\boldsymbol{y}_j \in T} \left(\frac{[\boldsymbol{y}_i - \boldsymbol{y}_j]_k^2 + \kappa\,[\boldsymbol{b}]_k^2}{[\boldsymbol{\sigma}]_k^3} \;-\; \frac{1}{[\boldsymbol{\sigma}]_k} \right) \pi_{ij}\,, \quad (8)$$

$$\text{where} \quad \pi_{ij} \;=\; \frac{K(\boldsymbol{y}_i - \boldsymbol{y}_j)}{|T|\,\hat{p}(\boldsymbol{y}_i)} \;=\; \frac{K(\boldsymbol{y}_i - \boldsymbol{y}_j)}{\displaystyle\sum_{\boldsymbol{y}_k \in T} K(\boldsymbol{y}_i - \boldsymbol{y}_k)\,.} \qquad (9)$$

For $\kappa = 0$, equations (7) and (8) simplify to those given in [29, 41, 42]. π_{ij} is a proximity factor that weighs how close $\boldsymbol{y}_j$ is to $\boldsymbol{y}_i$ relative to all other points in T. For Gaussian kernels, it is equivalent to the *softmax* nonlinearity [9] operating on the squared Mahalonobis distance [12]

$$D(\boldsymbol{u}) \;=\; \boldsymbol{u}^T \mathbf{\Sigma}^{-1} \boldsymbol{u}\,. \qquad (10)$$

Thus if $\boldsymbol{y}_j$ is significantly closer (in the Mahalonobis metric) to $\boldsymbol{y}_i$ than any other element of T, the proximity factor π_{ij} will approach one; conversely it will tend to zero if there is some other point in the sample that lies closer.

Using the above machinery, simple gradient ascent in the log-likelihood with step size η would update the kernel widths $\boldsymbol{\sigma}$ via

$$\boldsymbol{\sigma}(t{+}1) \;=\; \boldsymbol{\sigma}(t) + \eta\,\boldsymbol{g}(t)\,, \quad \boldsymbol{g}(t) \;=\; \frac{\partial \hat{L}}{\partial \boldsymbol{\sigma}}\bigg|_{\boldsymbol{\sigma}(t)}\,. \qquad (11)$$

However, this simplistic approach does not work well at all. Fig. 4 shows the same log-likelihood as Fig. 3, but plotted against kernel width on a linear scale. It is evident that the maximum of $\hat{L}$ lies near the semipole at $\sigma = 0$, where the gradient goes to infinity. (In general, the likelihood diverges at all degeneracies of the covariance matrix.) Step sizes η small enough to safely negotiate the neighborhood of the maximum will exhibit very slow convergence on the long, linear slope for larger σ (see Section II-F). Conversely, step sizes large enough to traverse the linear slope reasonably fast will overshoot the maximum and on the other side encounter large gradients, producing very large or even negative σ, with disastrous results.

D. Exponentiated Gradient and Step Size Adaptation

Scaling parameters such as $\boldsymbol{\sigma}$ are best adapted in log-space, where their gradients are much better behaved (*cf.* Fig. 3):

$$\ln \boldsymbol{\sigma}(t{+}1) \;=\; \ln \boldsymbol{\sigma}(t) \;+\; \eta\,\frac{\partial \hat{L}}{\partial \ln \boldsymbol{\sigma}}\,. \qquad (12)$$

Re-exponentiating (12) gives a multiplicative update rule ("exponentiated gradient" [19]) capable of adjusting $\boldsymbol{\sigma}$ over many orders of magnitude while keeping it strictly positive:

$$\boldsymbol{\sigma}(t{+}1) \;=\; \boldsymbol{\sigma}(t) \cdot e^{\,\eta\,\boldsymbol{g}(t)\cdot\boldsymbol{\sigma}(t)}\,, \qquad (13)$$

where $\cdot$ denotes Hadamard (*i.e.*, component-wise) multiplication. Note that the $\boldsymbol{\sigma}(t)$ in the exponent cancels against the denominator of the gradient (8). The exponentiated gradient update (13) can be further improved in terms of numerical stability and computational efficiency by re-linearizing the exponentiation via $e^u \approx \max(\frac{1}{2}, 1+u)$ [31], giving

$$\boldsymbol{\sigma}(t{+}1) \;=\; \boldsymbol{\sigma}(t) \cdot \max[\frac{1}{2},\, 1 + \eta\,\boldsymbol{g}(t)\cdot\boldsymbol{\sigma}(t)]\,. \qquad (14)$$

To further accelerate convergence we give each kernel width $[\boldsymbol{\sigma}]_k$ its individual, time-varying step size $[\boldsymbol{\eta}]_k$, adapted via the following simple mechanism [34, 36]:

$$[\boldsymbol{\eta}]_k(t{+}1) \;=\; \begin{cases} \varrho\,[\boldsymbol{\eta}]_k(t) & \text{if } [\boldsymbol{g}]_k(t)\,[\boldsymbol{g}]_k(t{-}1) > 0\,, \\ [\boldsymbol{\eta}]_k(t)/\varrho & \text{otherwise}\,; \end{cases} \qquad (15)$$

we use $\varrho = 1.5$. This increases the step size along directions where the sign of the gradient remains the same, and decreases it where the sign changes. More sophisticated step size adaptation mechanisms have been developed [1, 30, 31], but we find (15) sufficient for our purposes here (see Section II-F).

E. Expectation-Maximization

While the accelerated gradient ascent in log-likelihood described above is very fast, it may still diverge occasionally, and optimizes only a diagonal covariance matrix. We can improve upon that by observing that the kernel density estimate is in form of a mixture of Gaussians, with $|T|$ equally weighted mixture components, each centered on a point $\boldsymbol{y}_j \in T$. This means that a simplified form of the highly efficient *expectation-maximization* (EM) algorithm [8, 11] can be used to find the full covariance matrix $\mathbf{\Sigma}$ that maximizes $\hat{L}$.

EM proceeds by alternating between two steps: in the *E-step* we calculate for a given $\mathbf{\Sigma}$ the expectation that a particular mixture component j is responsible for data point $\boldsymbol{y}_i$, given by the proximity factor π_{ij} (9). In the M-step we calculate the kernel covariance $\mathbf{\Sigma}$ that maximizes $\hat{L}$ for given π_{ij}. For Gaussian

kernels, this is simply the covariance of the proximity-weighted data:

$$\boldsymbol{\Sigma} \;=\; \frac{1}{|S|} \sum_{\boldsymbol{y}_i \in S} \sum_{\boldsymbol{y}_j \in T} \pi_{ij} \, (\boldsymbol{y}_i - \boldsymbol{y}_j)(\boldsymbol{y}_i - \boldsymbol{y}_j)^T . \qquad (16)$$

In contrast to EM proper, we do not update kernel centers or weights here, since these are fixed by our nonparametric approach to density estimation.

Since the calculation of $\boldsymbol{\Sigma}$ depends on the proximity factors, which in turn depend on the kernel shape, which is given by $\boldsymbol{\Sigma}$, the entire procedure must be iterated by alternating between E-step (9) and M-step (16). Convergence to the maximum of $\hat{L}$ is typically much faster than for gradient ascent; we hasten it further by employing after each M-step (16) the *overrelaxation*

$$\boldsymbol{\Sigma}(t) \;:=\; \boldsymbol{\Sigma}(t)\, \boldsymbol{\Sigma}(t-1)^{-1} \boldsymbol{\Sigma}(t) \,, \qquad (17)$$

where t indexes the EM iteration. This acceleration technique is safe in that it still guarantees convergence.

Finally, we need some initial values for the EM iterations to refine. A reasonable choice is to initialize $\boldsymbol{\Sigma}(0)$ to the covariance $\boldsymbol{\Sigma}_*$ of the data sample, which can be calculated as

$$\boldsymbol{\Sigma}_* \;=\; \frac{1}{2|S||T|} \sum_{\boldsymbol{y}_i \in S} \sum_{\boldsymbol{y}_j \in T} (\boldsymbol{y}_i - \boldsymbol{y}_j)(\boldsymbol{y}_i - \boldsymbol{y}_j)^T . \qquad (18)$$

This corresponds to invoking the M-step (16) with uniform proximity factors of $(\forall i, j)\; \pi_{ij} = (2|T|)^{-1}$.

F. Experimental Comparison

To illustrate the differences between the kernel shape optimization methods we have discussed, we compare their performance experimentally, using data from the hand tracking application shown in Fig. 1. Specifically, we optimized the kernel shape for mini-batches of 30 sample points from the 6-dimensional joint space spanned by the orientation of the hand model surface normal (x, y, and z components) and the intensity at the corresponding location of the video image (R, G, and B components). For the gradient methods, the kernel widths were initialized to the standard deviations of the data: $\boldsymbol{\sigma} = \mathrm{diag}(\boldsymbol{\Sigma}_*)$; the (initial) step sizes were roughly optimized by hand as follows: starting from $\eta = 1$, we repeatedly reduced η by a factor of 10, until stable convergence was achieved.

The results are shown in Fig. 5, which plots the determinant of the covariance matrix against the number of iterations in a typical kernel shape optimization. It is obvious that ordinary gradient ascent (dotted line) is intolerably slow to converge, due to the very small step size ($\eta = 10^{-5}$) that had to be employed to achieve stability; this validates our discussion in Section II-C above. Our re-linearized exponentiated gradient (dashed line), by contrast, converges within about 20 iterations. Step size adaptation (solid line) cuts this down to less than 10 iterations, at the cost of introducing some of the oscillatory ("twitchy") behavior characteristic of the simple sign-based technique we employed. More recent, sophisticated step size adaptation methods [1, 30, 31] could be used to ameliorate this problem.

The expectation-maximization (EM) algorithm takes about 25 iterations to converge, a very good result, considering that it

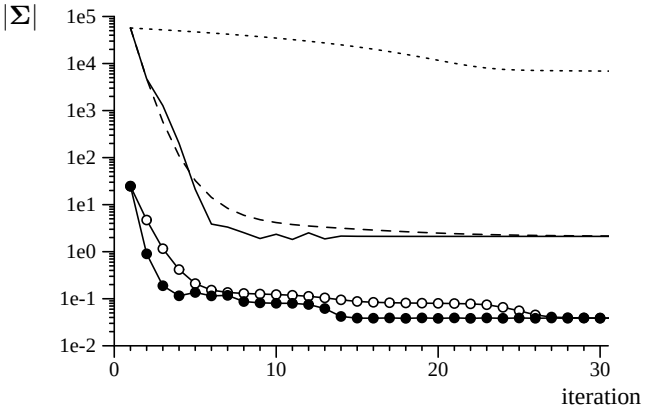


Fig. 5. Experimental comparison of kernel shape optimization methods. Dotted line: ordinary gradient ascent (11); dashed line: re-linearized exponentiated gradient (14); solid line: (14) with step size adaptation (15). Open circles: EM; solid disks: EM with overrelaxation (17).

optimizes a full covariance matrix (here: 36 parameters) instead of just the diagonal (here: 6 parameters), as the gradient methods do. Even better, our overrelaxation trick (17) brings this down to 15 iterations while still guaranteeing convergence. We conclude that EM with overrelaxation should be used whenever computationally feasible. In higher-dimensional spaces, where the cost of operating on full covariance matrices may be prohibitive, our re-linearized exponentiated gradient with step size adaptation offers an efficient diagonalized alternative.

G. Heteroscedastic Kernel Density Estimate

Up to now he have tacitly assumed homoscedasticity, *i.e.*, that all kernels have the same shape. Many probability densities, in particular those with discontinuities, are better approximated by allowing each kernel its own shape, which can be fitted individually to the local data. Our approach is easily generalized to the heteroscedastic kernel density estimate

$$\hat{p}(\boldsymbol{y}) \;=\; \frac{1}{|T|} \sum_{\boldsymbol{y}_j \in T} K_j(\boldsymbol{y} - \boldsymbol{y}_j) \,, \qquad (19)$$

where each kernel K_j has its own covariance parameters $\boldsymbol{\Sigma}_j$, *resp.* $\boldsymbol{\sigma}_j$. The proximity factors must now be redefined as

$$\pi_{ij} \;=\; \frac{K_j(\boldsymbol{y}_i - \boldsymbol{y}_j)}{|T|\, \hat{p}(\boldsymbol{y}_i)} \;=\; \frac{K_j(\boldsymbol{y}_i - \boldsymbol{y}_j)}{\sum_{\boldsymbol{y}_k \in T} K_k(\boldsymbol{y}_i - \boldsymbol{y}_k) \,.} \qquad (20)$$

For gradient ascent, each sample $\boldsymbol{y}_j \in T$ is also given its own vector $\boldsymbol{\eta}_j$ of individually adapted step sizes, and the overall gradient (8) of the log-likelihood is replaced by the sample gradient

$$[\boldsymbol{g}_j]_k \;=\; \sum_{\boldsymbol{y}_i \in S} \left(\frac{[\boldsymbol{y}_i - \boldsymbol{y}_j]_k^2 + \kappa\, [\boldsymbol{b}]_k^2}{[\boldsymbol{\sigma}_j]_k^3} \;-\; \frac{1}{[\boldsymbol{\sigma}_j]_k} \right) \pi_{ij} \,. \qquad (21)$$

Similarly, for heteroscedastic expectation-maximization the E-step employs (20) instead of (9), while the M-step becomes

$$(\forall \boldsymbol{y}_j \in T) \quad \boldsymbol{\Sigma}_j \;=\; \frac{|T|}{|S|} \sum_{\boldsymbol{y}_i \in S} \pi_{ij} \, (\boldsymbol{y}_i - \boldsymbol{y}_j)(\boldsymbol{y}_i - \boldsymbol{y}_j)^T . \qquad (22)$$

III. Optimization of Empirical Entropy

Now that we have developed reliable and efficient techniques for nonparametric density estimation with optimal kernel shape, we shall use them to calculate and optimize the entropy produced from given empirical data by a parametrized differentiable map, such as a feedforward neural network. For complete generality, we will present equations for the heteroscedastic kernel density estimate (19); the homoscedastic versions are trivially obtained by dropping the sample index j from the relevant entities (K, D, and $\mathbf{\Sigma}$).

A. Nonparametric Entropy Estimate

In previous work [29, 41, 42] we approximated the entropy of a random variable Y empirically, based on a sample S of instances of Y:

$$\begin{aligned} H(Y) &= -\int p(\boldsymbol{y}) \ln p(\boldsymbol{y})\, d\boldsymbol{y} \\ &\approx -\frac{1}{|S|} \sum_{\boldsymbol{y}_i \in S} \ln p(\boldsymbol{y}_i)\,, \end{aligned} \tag{23}$$

where $p(\boldsymbol{y})$ is the probability density of Y. In a machine learning setting, $p(\boldsymbol{y})$ is normally not explicitly available — in general, we are given only empirical data, *i.e.*, a supply of instances of Y. However, we can infer an estimated kernel density $\hat{p}(\boldsymbol{y})$ from these samples via (3), and use that to obtain a nonparametric estimate of the empirical entropy of Y:

$$\begin{aligned} \hat{H}(Y) &= -\frac{1}{|S|} \sum_{\boldsymbol{y}_i \in S} \ln \hat{p}(\boldsymbol{y}_i) \\ &= -\frac{1}{|S|} \sum_{\boldsymbol{y}_i \in S} \ln \sum_{\boldsymbol{y}_j \in T} K_j(\boldsymbol{y}_i - \boldsymbol{y}_j) \;+\; \ln |T|\,. \end{aligned} \tag{24}$$

Note the close similarity to the empirical likelihood (5) — in fact, all the points we have made above regarding the adverse effects of quantization and finite sample size, and how to overcome them, equally apply to the estimation and optimization of $\hat{H}$ here.

B. Gradient of Estimated Entropy

Consider the situation where Y is in fact produced by a parametrized mapping $N_{\boldsymbol{w}}$ from another multivariate random variable X, *i.e.*, the k^{th} instance of Y is $\boldsymbol{y}_k = N_{\boldsymbol{w}}(\boldsymbol{x}_k)$. The adaptive mapping $N_{\boldsymbol{w}}$ might for example be a feedforward neural network with weights $\boldsymbol{w}$. Our goal is to manipulate the estimated entropy of Y by adjusting the parameters $\boldsymbol{w}$. The gradient of $\hat{H}(Y)$ with respect to these weights is

$$\begin{aligned} &\frac{\partial}{\partial \boldsymbol{w}} \hat{H}(N_{\boldsymbol{w}}(X)) \;= \\ &= -\frac{1}{|S|} \sum_{\boldsymbol{x}_i \in S} \frac{\partial}{\partial \boldsymbol{w}} \ln \sum_{\boldsymbol{x}_j \in T} K_j(N_{\boldsymbol{w}}(\boldsymbol{x}_i) - N_{\boldsymbol{w}}(\boldsymbol{x}_j)) \\ &= -\frac{1}{|S|} \sum_{\boldsymbol{x}_i \in S} \frac{\sum_{\boldsymbol{x}_j \in T} \frac{\partial}{\partial \boldsymbol{w}} K_j(\boldsymbol{y}_i - \boldsymbol{y}_j)}{\sum_{\boldsymbol{x}_k \in T} K_k(\boldsymbol{y}_i - \boldsymbol{y}_k)} \\ &= \frac{1}{2\,|S|} \sum_{\boldsymbol{x}_i \in S} \sum_{\boldsymbol{x}_j \in T} \pi_{ij}\, \frac{\partial}{\partial \boldsymbol{w}} D_j(\boldsymbol{y}_i - \boldsymbol{y}_j) \\ &= \frac{1}{|S|} \sum_{\boldsymbol{x}_i \in S} \sum_{\boldsymbol{x}_j \in T} \pi_{ij} \left(\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{w}} - \frac{\partial \boldsymbol{y}_j}{\partial \boldsymbol{w}} \right) \mathbf{\Sigma}_j^{-1} (\boldsymbol{y}_i - \boldsymbol{y}_j) \end{aligned} \tag{25}$$

where π_{ij} is the proximity factor from (9), and D_j the squared Mahalonobis distance (10) for the covariance $\mathbf{\Sigma}_j$.

Provided that $N_{\boldsymbol{w}}$ is differentiable with respect to its parameters, we can thus lower (raise) the entropy $\hat{H}(Y)$ by gradient descent (ascent) in $\boldsymbol{w}$ as prescribed by (25). Note that the update rule is differential, *i.e.*, it always operates on the difference between two sample points. It minimizes (maximizes) entropy by reducing (increasing) the Mahalonobis distance between neighboring points, where neighborhoods are defined in a soft, probabilistic manner by the proximity factors (9).

C. Affine-Invariant Entropy Estimate and Gradient

As described above, the optimization of entropy in the output of an adaptive mapping suffers from a severe drawback: entropy is not invariant under linear transformations of the output, but varies with the log-determinant of its covariance [7], as given by $\mathbf{\Sigma}_*$ in (18). The easiest way for an adaptive system to increase (decrease) its output entropy is therefore to modify its output covariance by a suitable linear transformation.

Such linear rescaling is often undesirable, and in any event could be achieved by much simpler (*e.g.*, Hebbian *resp.* anti-Hebbian) means. Here we want to focus on the nonlinear problem of optimizing the *shape*, rather than the *scale*, of the output density. Parra [22] achieves this by constraining the adaptive system to implement only symplectic (*i.e.*, volume-preserving) maps. This technique, however, complicates the architecture of the system, and is limited in scope: even where scaling *per se* must not be rewarded, the adaptive systems may still be required scale the data, *e.g.*, to match given targets. In such cases the symplectic map approach cannot be used.

Instead of constraining the architecture, we constrain the objective function to be invariant with respect to affine transformations. Such an affine-invariant entropy measure can be constructed by means of subtracting out the dependence of entropy on scale:

$$\tilde{H}(Y) \;=\; \hat{H}(Y) \;-\; \frac{1}{2} \ln |\mathbf{\Sigma}_*|\,. \tag{26}$$

The gradient of the correction term with respect to the parameters $\boldsymbol{w}$ can be derived as follows:

$$\frac{\partial \ln |\mathbf{\Sigma}_*|}{\partial \mathbf{\Sigma}_*} \;=\; \frac{(\operatorname{adj} \mathbf{\Sigma}_*)^T}{|\mathbf{\Sigma}_*|} \;=\; \mathbf{\Sigma}_*^{-T}, \tag{27}$$

as shown in [7]. From Equation (18) the gradient of $\mathbf{\Sigma}_*$ with respect to a single parameter w is found to be

$$\begin{aligned} \frac{\partial \mathbf{\Sigma}_*}{\partial w} &= \frac{1}{2|S||T|} \sum_{\boldsymbol{y}_i \in S} \sum_{\boldsymbol{y}_j \in T} \left[\boldsymbol{u}_{ij} (\boldsymbol{y}_i - \boldsymbol{y}_j)^T + (\boldsymbol{y}_i - \boldsymbol{y}_j)\, \boldsymbol{u}_{ij}^T \right], \\ &\text{where} \quad \boldsymbol{u}_{ij} \equiv \frac{\partial \boldsymbol{y}_i}{\partial w} - \frac{\partial \boldsymbol{y}_j}{\partial w}\,. \end{aligned} \tag{28}$$

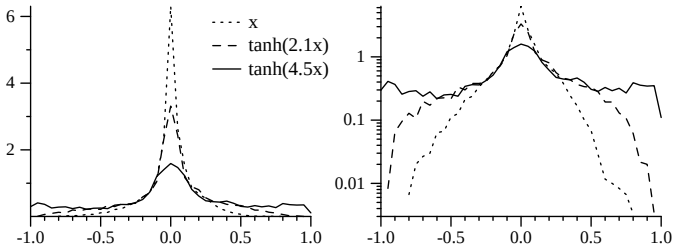


Fig. 6. Histogram-estimated density of sampled speech x (dotted lines) and its nonlinear transformation $\tanh(wx)$ with gains of $w = 2.1$ (dashed) and $w = 4.5$ (solid), plotted on a linear (left) *resp.* logarithmic (right) scale.

Putting the two together gives

$$\frac{\partial}{\partial w} \ln|\boldsymbol{\Sigma}_*| = \operatorname{trace}\left(\frac{\partial \ln|\boldsymbol{\Sigma}_*|}{\partial \boldsymbol{\Sigma}_*} \frac{\partial \boldsymbol{\Sigma}_*}{\partial w}\right) = \tag{29}$$
$$= \frac{1}{|S||T|} \sum_{\boldsymbol{x}_i \in S} \sum_{\boldsymbol{x}_j \in T} \boldsymbol{u}_{ij}^T \boldsymbol{\Sigma}_*^{-1} (\boldsymbol{y}_i - \boldsymbol{y}_j)\,,$$

which can be extended to the entire vector $\boldsymbol{w}$ of parameters and combined with (25) to finally yield

$$\frac{\partial}{\partial \boldsymbol{w}} \tilde{H}(N_{\boldsymbol{w}}(X)) =$$
$$= \frac{\partial}{\partial \boldsymbol{w}} \hat{H}(N_{\boldsymbol{w}}(X)) - \frac{1}{2} \frac{\partial \ln|\boldsymbol{\Sigma}_*|}{\partial \boldsymbol{w}} \tag{30}$$
$$= \frac{1}{|S|} \sum_{\boldsymbol{x}_i \in S} \sum_{\boldsymbol{x}_j \in T} \left(\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{w}} - \frac{\partial \boldsymbol{y}_j}{\partial \boldsymbol{w}}\right) \boldsymbol{M}_{ij} (\boldsymbol{y}_i - \boldsymbol{y}_j)\,,$$
$$\text{where} \quad \boldsymbol{M}_{ij} \equiv \pi_{ij} \boldsymbol{\Sigma}_j^{-1} - \frac{1}{2|T|} \boldsymbol{\Sigma}_*^{-1}\,.$$

Comparing this with (25) we see that an affine-invariant entropy estimate $\tilde{H}$ can be optimized by gradient methods at the small additional cost of including the subtractive correction term $\boldsymbol{\Sigma}_*^{-1}/(2|T|)$.

We illustrate the difference between optimizing $\hat{H}$ and $\tilde{H}$ on 3 seconds of speech data (male voice sampled at 4kHz). Fig. 6 (dotted) shows the typical Laplacian distribution of our speech signal, evident in the linear tails on the log-scale plot. We take this signal through an adaptive gain and hyperbolic tangent nonlinearity, then maximize the output entropy with respect to the gain w by gradient ascent (on mini-batches of 100 samples). Maximizing the empirical entropy $\hat{H}$ via (25) produces a gain of $w \approx 4.5$; Fig. 6 (solid) shows that at this gain, the output density is as uniform as possible, and thus has maximal entropy given the fixed range of the $\tanh$ function. Maximizing the affine-invariant entropy $\tilde{H}$ via (30), by contrast, gives a gain of $w \approx 2.1$, at which the output has maximal entropy for a given variance, *i.e.*, is as Gaussian as possible. This can be seen in the near-quadratic tails on the log-scale plot in Fig. 6 (dashed), which are characteristic of a Gaussian. Thus even in a range-limited adaptive system, the two types of entropy measure can produce quite different behavior.

D. Update Strategies

A straightforward stochastic approximation gradient algorithm to manipulate $\hat{H}$ *resp.* $\tilde{H}$ can be implemented by iterating over the following three steps:

1) Pick a reference point $\boldsymbol{x}_i \in S$ from the data and calculate $\boldsymbol{y}_i = N_{\boldsymbol{w}}(\boldsymbol{x}_i)$.
2) Loop through a set of data points $\boldsymbol{x}_j \in T_i$ and accumulate the sums of various relevant terms.
3) Use the accumulated sums to update $\boldsymbol{w}$ in proportion to (25) *resp.* (30), and $\boldsymbol{\Sigma}$ via either EM with overrelaxation (17), or re-linearized exponentiated gradient (14) with step size adaptation (15).

Note that we are interleaving kernel shape optimization steps with entropy manipulation weight updates. This computational shortcut is permissible since the kernel shape only needs to track gradual weight changes, not each particular data sample; it is thus neither necessary nor desirable to iterate the kernel shape optimization to convergence prior to each weight update.

For the estimation and manipulation of Shannon entropy, by contrast, it is necessary to accumulate statistics over a batch T_i of sample points before each update of the weights $\boldsymbol{w}$ and kernel shape $\boldsymbol{\Sigma}$. This is because the updates depend on the ratio of accumulated sums, for which in general there is no satisfactory stochastic approximation. It is possible though to employ relatively small "mini-batches" here to achieve a near-online weight update (see below). In Section III-E we will discuss alternative objectives that are amenable to a true online implementation.

An important practical issue is how the sets S and T_i of samples are generated. We have already mentioned that there must not be any overlap between them — that is, the reference point $\boldsymbol{x}_i$ must not be contained in T_i. This still leaves many possible ways of sampling from a supply of data; which one is most effective will depend on the particular application.

1) Data-limited: Consider the case where we only have access to a relatively small, predetermined set of data points. To make the best use of this limited supply, all pairs of samples should enter into each computation of the entropy and likelihood gradients. This can be achieved by a technique akin to leave-one-out cross-validation: use all available data as set S, and set $T_i = S \setminus \{\boldsymbol{x}_i\}$, *i.e.*, omit only the reference point from the inner summation. Note that in order to implement this data-limited approach efficiently, we must be able to hold all samples in memory at the same time.

2) Memory-limited: The situation is quite different when the supply of data exceeds our memory capacity. An embedded controller, for instance, may be required to process an infinite stream of incoming samples with as little memory capacity as possible. We can accommodate this constraint by recognizing that with the exception of the reference point, there is no need to ever reuse the same data sample. A memory-limited implementation can therefore simply pick a fresh reference point from the data stream for each update, then collect the required statistics over the next $|T|$ samples. Memory is required only to store the reference point and the three sums that are accumulated over the batch.

3) Intermediate: In practice an implementation may well fall somewhere between these two extremes. For instance, if the supply of data is large but not cheap, the best strategy may be to obtain a fresh batch of samples for each update, small enough to be comfortably held in memory. All pairs of points in this batch can then be used to compute the update, as in the data-limited case. If online weight updates (*i.e.*, after each sample)

are required, each new sample can be used as a reference point, and statistics gathered over a sliding window of the $|T|$ most recent past samples [13].

This raises a valid question: what is an appropriate size for $|T|$ in general? How many points should our statistics be accumulated over before we perform an update and pick a new reference point? It is well known that the error in empirical statistics decreases with the square root of the size of the sample. That is, the longer we sample before making an update, the more accurate that update will be. On the other hand, this means that more computation has to be performed for each update. Moreover, the noise associated with small sample sizes can in fact be very effective in helping gradient descent escape from local minima. In practice we have found small batch sizes of 20 to 50 samples to provide the best performance. Where highly accurate asymptotic results are required, the batch size can be increased after initial stochastic convergence.

E. Online Variants

As described above, the estimation and manipulation of Shannon entropy requires collecting the data into batches for learning. There are, however, related objectives that can be implemented as true online algorithms that adjust their weights in response to each new data pattern as it becomes available, using only minimal memory of past data:

1) Projection Pursuit and Renyi Entropy: Projection pursuit [15, 16] is concerned with finding linear, low-dimensional projections of high-dimensional data that optimize a given index function. Our work here can be understood as projection pursuit with Shannon entropy as index function, extended to nonlinear systems. We can apply kernel density estimation to other projection pursuit indices as well, such as

$$\begin{aligned} Q(Y) &= -\int p(\boldsymbol{y})^2\, d\boldsymbol{y} \;\approx\; -\frac{1}{|S|}\sum_{\boldsymbol{y}_i\in S}\hat{p}(\boldsymbol{y}_i) \\ &= -\frac{1}{|S||T|}\sum_{\boldsymbol{y}_i\in S}\sum_{\boldsymbol{y}_j\in T}K(\boldsymbol{y}_i-\boldsymbol{y}_j)\,, \end{aligned} \tag{31}$$

shown by Huber [16, p. 446] to be closely related to the original projection pursuit index proposed by Friedman and Tukey [15]. $Q(Y)$ is in fact the quadratic Renyi [25] entropy, which belongs to a class of generalized entropy measures [18] which when extremized under given constraints produce the same result as extremizing the Shannon entropy [17].

Assume again that Y is produced from X by the adaptive mapping $N_{\boldsymbol{w}}$, and differentiate $Q(Y)$ with respect to the parameters $\boldsymbol{w}$:

$$\frac{\partial}{\partial \boldsymbol{w}}\,\hat{Q}(N_{\boldsymbol{w}}(X)) \;= \tag{32}$$

$$\frac{1}{|S||T|}\sum_{\boldsymbol{y}_i\in S}\sum_{\boldsymbol{y}_j\in T}K(\boldsymbol{y}_i-\boldsymbol{y}_j)\,(\boldsymbol{y}_i-\boldsymbol{y}_j)^T\boldsymbol{\Sigma}^{-1}\frac{\partial}{\partial \boldsymbol{w}}\,(\boldsymbol{y}_i-\boldsymbol{y}_j)$$

This differs from (25) only in that the terms in the outer sum over S are no longer normalized by the $\hat{p}(\boldsymbol{y})$ in the denominator of (9). This has an important consequence in that it makes the gradient amenable to true online implementation: $\boldsymbol{w}$ can now be updated according to

$$\Delta\boldsymbol{w} \;\propto\; K(\Delta\boldsymbol{y})\,(\Delta\boldsymbol{y})^T\boldsymbol{\Sigma}^{-1}\frac{\partial}{\partial \boldsymbol{w}}\,(\Delta\boldsymbol{y}) \tag{33}$$

where the Δ operator denotes temporal finite differencing. The simple form of (33) makes quadratic Renyi entropy attractive for entropy manipulation purposes [14, 24, 37].

Viola [41, p. 65] notes that $\hat{H}$ overestimates the true Shannon entropy while $\hat{Q}$ underestimates it, and proposes combining the two to obtain an improved empirical estimate of Shannon entropy. One way to do this — albeit at the cost of again requiring batching of data — is to add a suitable constant to the denominator of (9), as used in (25).

2) Parametric Reference Distribution: A generic strategy to obtain a true online algorithm is to replace $\hat{p}(\boldsymbol{y})$ in the denominator of (9) by a *parametric* approximation $q(\boldsymbol{y})$ that can be evaluated without reference to a specific set of data points. The online update rule then becomes

$$\Delta\boldsymbol{w} \;\propto\; \frac{K(\Delta\boldsymbol{y})}{q(\boldsymbol{y})}\,(\Delta\boldsymbol{y})^T\boldsymbol{\Sigma}^{-1}\frac{\partial}{\partial \boldsymbol{w}}\,(\Delta\boldsymbol{y}) \tag{34}$$

Note that this optimizes the objective

$$\begin{aligned} Q(Y) &= -\int p(\boldsymbol{y})\ln q(\boldsymbol{y})\, d\boldsymbol{y} \\ &= -\int p(\boldsymbol{y})\ln p(\boldsymbol{y})\, d\boldsymbol{y} \;+\int p(\boldsymbol{y})\ln\frac{p(\boldsymbol{y})}{q(\boldsymbol{y})}\, d\boldsymbol{y} \\ &= H(Y)+D(p||q) \;\geq\; H(Y) \end{aligned} \tag{35}$$

which differs from the entropy $H(Y)$ by the Kullback-Leibler (KL) divergence between p and q, $D(p||q)$. Since the KL-divergence is always non-negative, $Q(Y)$ is an upper bound on the entropy, which can be kept tight by continually adapting the reference distribution q to the output $Y=N_{\boldsymbol{w}}(X)$ of our adaptive map. This can be achieved through online density tracking methods such as Kalman filtering for a single Gaussian, online EM [21, 35] for a mixture of Gaussians, or the self-organizing maps of van Hulle [39, 40]. Note that quadratic Renyi entropy manipulation (33) is in fact the special case of (34) with a uniform reference density.

IV. Summary

The optimization of entropy in the output of an adaptive system [29] requires access to the density function, which must therefore be estimated empirically. This is commonly achieved by resorting to parametric methods which impose strong modeling assumptions upon the data. We have based our entropy estimate on a non-parametric alternative, kernel density estimation, instead, and provided corrections for problems that can occur when the data sample is finite or quantized. Both gradient ascent and expectation-maximization techniques for maximum likelihood optimization of the shape of the regularizing kernel have been developed, compared empirically, and generalized to heteroscedastic kernels.

The resulting nonparametric density estimate is smooth and differentiable, and can thus be used to manipulate Shannon entropies at the output of a parametrized mapping, such as a neural

network, by gradient methods. We also provide a normalization term that makes the entropy estimate invariant to affine transformations. The gradient of either entropy estimate yields a simple and efficient batch learning rule that operates on pairs of input samples. We have given data-limited and memory-limited implementations for this nonparametric entropy manipulation method, related our approach to projection pursuit and quadratic Renyi entropy, and described a generic way to obtain a fully online implementation.

ACKNOWLEDGMENT

We would like to thank Giorgio Panin for helpful discussions and Matlab code, Matthieu Bray for hand tracking data and images, and the anonymous reviewers for their valuable feedback.

REFERENCES

[1] L. B. Almeida, T. Langlois, J. D. Amaral, and A. Plakhov. Parameter adaptation in stochastic optimization. In D. Saad, editor, *On-Line Learning in Neural Networks*, Publications of the Newton Institute, chapter 6, pages 111–134. Cambridge University Press, 1999.

[2] J. J. Atick. Could information theory provide an ecological theory of sensory processing? *Network*, 3:213–251, 1992.

[3] H. B. Barlow. Possible principles underlying the transformation of sensory messages. In W. A. Rosenbluth, editor, *Sensory Communication*. MIT Press, Cambridge, MA, 1961.

[4] H. B. Barlow. Unsupervised learning. *Neural Computation*, 1 (3):295–311, 1989.

[5] S. Becker and G. E. Hinton. A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355: 161–163, 1992.

[6] J. Beirlant, E. J. Dudewicz, L. Györfi, and E. C. van der Meulen. Nonparametric entropy estimation: An overview. *Intl. Journal of Mathematical and Statistical Sciences*, 6(1):17–39, 1997.

[7] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[8] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, International Computer Science Institute, UC Berkeley, CA, 1997. `http://crow.ee.washington.edu/people/bulyko/papers/em.pdf`.

[9] J. S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 211–217. Morgan Kaufmann, San Mateo, CA, 1990.

[10] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society series B*, 39:1–38, 1977.

[12] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[13] D. Erdogmus, K. E. Hild, II, and J. C. Principe. Online entropy manipulation: Stochastic information gradient. *IEEE Signal Processing Letters*, 10(8):242–245, 2003.

[14] D. Erdogmus and J. C. Principe. Generalized information potential criterion for adaptive system training. *IEEE Transaction on Neural Networks*, 13(5):1035–1044, 2002.

[15] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23:881–889, 1974.

[16] P. J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2): 435–475, 1985.

[17] J. N. Kapur. *Measures of Information and their Applications*. John Wiley & Sons, 1992.

[18] J. N. Kapur and H. K. Kesavan. *Entropy Optimization Principles with Applications*. Academic Press, 1992.

[19] J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *Proc. 27th Annual ACM Symposium on Theory of Computing*, pages 209–218, New York, NY, May 1995. The Association for Computing Machinery.

[20] R. Linsker. Self-organization in a perceptual network. *Computer*, pages 105–117, March 1988.

[21] Masa-aki Sato and S. Ishii. On-line em algorithm for the normalized gaussian network. *Neural Computation*, 12(2):407–432, 2000.

[22] L. C. Parra. Symplectic nonlinear component analysis. In Touretzky et al. [38], pages 437–443.

[23] E. Parzen. On the estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[24] J. C. Principe, D. Xu, and J. W. Fisher III. Information-theoretic learning. In S. Haykin, editor, *Unsupervised Adaptive Filtering: Blind Source Separation*, volume 1, chapter 7, pages 265–319. John Wiley & Sons, 2000.

[25] A. Renyi. On measures of entropy and information. In *Proc. 4th Berkeley Symposium Mathematical Statistics and Probability*, pages 547–561. University of California Press, 1961. Reprinted in [26, pp. 565–580].

[26] A. Renyi. *Selected Papers of Alfred Renyi*, volume 2. Akademia Kiado, Budapest, 1976.

[27] J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

[28] J. Schmidhuber, M. Eldracher, and B. Foltin. Semilinear predictability minimization produces well-known feature detectors. *Neural Computation*, 8(4):773–786, 1996.

[29] N. N. Schraudolph. *Optimization of Entropy with Neural Networks*. PhD thesis, University of California, San Diego, 1995. `http://n.schraudolph.org/pubs/thesis.ps.gz`.

[30] N. N. Schraudolph. Local Gain Adaptation in Stochastic Gradient Descent. In *Proc. Intl. Conf. Artificial Neural Networks*, pages 569–574, Edinburgh, Scotland, 1999. IEE, London. `http://n.schraudolph.org/pubs/smd.ps.gz`.

[31] N. N. Schraudolph. Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent. *Neural Computation*, 14(7): 1723–1738, 2002.

[32] N. N. Schraudolph, M. Eldracher, and J. Schmidhuber. Processing Images by Semi-Linear Predictability Minimization. *Network: Computation in Neural Systems*, 10(2):133–169, 1999.

[33] N. N. Schraudolph and T. J. Sejnowski. Unsupervised Discrimination of Clustered Data via Optimization of Binary Information Gain. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 499–506. Morgan Kaufmann, San Mateo, CA, 1993.

[34] F. M. Silva and L. B. Almeida. Speeding up back-propagation. In R. Eckmiller, editor, *Advanced Neural Computers*, pages 151–158, Amsterdam, 1990. Elsevier.

[35] A. F. M. Smith and U. E. Makov. A quasi-Bayes sequential procedure for mixtures. *J. Royal Statistical Society, B*, 40(1):106–112, 1978.

[36] T. Tollenaere. SuperSAB: fast adaptive back propagation with good scaling properties. *Neural Networks*, 3:561–573, 1990.

[37] K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003.

[38] D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors. *Advances in Neural Information Processing Systems*, volume 8, 1996. The MIT Press, Cambridge, MA.

[39] M. M. van Hulle. The formation of topographic maps that maximize the average mutual information of the output responses to noiseless input signals. *Neural Computation*, 9(3):595–606,

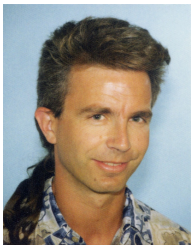
1997.

[40] M. M. van Hulle. Kernel-based equiprobabilistic topographic map formation. *Neural Computation*, 10(7):1847–1871, 1998.

[41] P. A. Viola. *Alignment by Maximization of Mutual Information*. PhD thesis, Massachusetts Institute of Technology, 1995.

[42] P. A. Viola, N. N. Schraudolph, and T. J. Sejnowski. Empirical Entropy Manipulation for Real-World Problems. In Touretzky et al. [38], pages 851–857.

[43] P. A. Viola and W. M. Wells III. Alignment by maximization of mutual information. In *Fifth International Conference on Computer Vision*, pages 16–23, Cambridge, MA, 1995. IEEE, Los Alamitos.



Nicol N. Schraudolph received the B.Sc. (Hons) degree in Computer Science from the University of Essex, England, in 1988, and the M.S. degree in Computer Science from the University of California, San Diego, in 1990. From 1991 to 1995 he was a fellow of the MacDonnell-Pew Center for Cognitive Neuroscience at the Computational Neurobiology Lab of the Salk Institute, earning a Ph.D. degree in Cognitive Science and Computer Science from the University of California, San Diego, in 1995.

From 1996 to 2001 he was senior research associate at the Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA) in Lugano, Switzerland. From 2001 to 2003 he has led the machine learning group at the Institute of Computational Sciences of the Federal Institute of Technology (ETH) in Zürich, Switzerland. His research in machine learning currently focuses on developing rapid stochastic gradient methods and their applications.