

Distributed optimization of deeply nested systems

Miguel Á. Carreira-Perpiñán Weiran Wang
 EECS, University of California, Merced
<http://eecs.ucmerced.edu>

December 24, 2012

Abstract

In science and engineering, intelligent processing of complex signals such as images, sound or language is often performed by a parameterized hierarchy of nonlinear processing layers, sometimes biologically inspired. Hierarchical systems (or, more generally, nested systems) offer a way to generate complex mappings using simple stages. Each layer performs a different operation and achieves an ever more sophisticated representation of the input, as, for example, in a deep artificial neural network, an object recognition cascade in computer vision or a speech front-end processing. Joint estimation of the parameters of all the layers and selection of an optimal architecture is widely considered to be a difficult numerical nonconvex optimization problem, difficult to parallelize for execution in a distributed computation environment, and requiring significant human expert effort, which leads to suboptimal systems in practice. We describe a general mathematical strategy to learn the parameters and, to some extent, the architecture of nested systems, called the *method of auxiliary coordinates (MAC)*. This replaces the original problem involving a deeply nested function with a constrained problem involving a different function in an augmented space without nesting. The constrained problem may be solved with penalty-based methods using alternating optimization over the parameters and the auxiliary coordinates. MAC has provable convergence, is easy to implement reusing existing algorithms for single layers, can be parallelized trivially and massively, applies even when parameter derivatives are not available or not desirable, and is competitive with state-of-the-art nonlinear optimizers even in the serial computation setting, often providing reasonable models within a few iterations.

1 Introduction

The continued increase in recent years in data availability and processing power has enabled the development and practical applicability of ever more powerful models in statistical machine learning, for example to recognize faces or speech, or to translate natural language (Bishop, 2006). However, physical limitations in serial computation suggest that scalable processing will require algorithms that can be massively parallelized, so they can profit from the thousands of inexpensive processors available in cloud computing. We focus on hierarchical processing architectures such as deep neural nets (fig. 1), which were originally inspired by biological systems such as the visual and auditory cortex in the mammalian brain (Riesenhuber and Poggio, 1999; Serre et al., 2007; Gold and Morgan, 2000), and which have been proven very successful at learning sophisticated tasks, such as recognizing faces or speech, when trained on data. A typical neural net defines a hierarchical, feedforward, parametric mapping from inputs to outputs. The parameters (*weights*) are learned given a dataset by numerically minimizing an objective function. The outputs of the hidden units at each layer are obtained by transforming the previous layer’s outputs by a linear operation with the layer’s weights followed by a nonlinear elementwise mapping (e.g. sigmoid). Deep, nonlinear neural nets are universal approximators, that is, they can approximate any target mapping (from a wide class) to arbitrary accuracy given enough units (Bishop, 2006), and can have more representation power than shallow nets (Bengio and LeCun, 2007). The hidden units may encode hierarchical, distributed features that are useful to deal with complex sensory data. For example, when trained on images, deep nets can learn low-level features such as edges and T-junctions and high-level features such as parts decompositions. Other examples of hierarchical processing systems are cascades for object recognition and scene understanding in

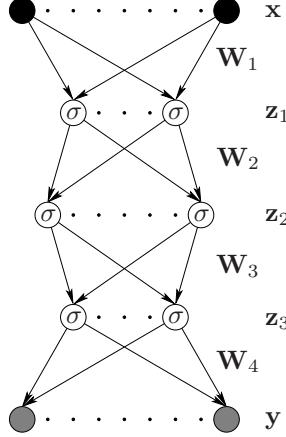


Figure 1: Net with $K = 3$ hidden layers (\mathbf{z}_k : auxiliary coordinates, \mathbf{W}_k : weights).

computer vision (Serre et al., 2007; Ranzato et al., 2007a) or for phoneme classification in speech processing (Gold and Morgan, 2000; Saon and Chien, 2012), wrapper approaches to classification or regression (e.g. based on dimensionality reduction; Wang and Carreira-Perpiñán, 2012), or kinematic chains in robotics (Craig, 2004). These and other architectures share a fundamental design principle: *mathematically, they construct a deeply nested mapping from inputs to outputs.*

The ideal performance of a nested system arises when all the parameters at all layers are jointly trained to minimize an objective function for the desired task, such as classification error (indeed, there is evidence that plasticity and learning probably occurs at all stages of the ventral stream of primate visual cortex; Riesenhuber and Poggio, 1999; Serre et al., 2007). However, this is challenging because nesting (i.e., function composition) produces inherently nonconvex functions. Joint training is usually done with the backpropagation algorithm (Rumelhart et al., 1986a; Werbos, 1974), which recursively computes the gradient with respect to each parameter using the chain rule. One can then simply update the parameters with a small step in the negative gradient direction as in gradient descent and stochastic gradient descent (SGD), or feed the gradient to a nonlinear optimization method that will compute a better search direction, possibly using second-order information (Orr and Müller, 1998). This process is repeated until a convergence criterion is satisfied. Backprop in any of these variants suffers from the problem of vanishing gradients (Rögnvaldsson, 1994; Erhan et al., 2009), where the gradients for lower layers are much smaller than those for higher layers, which leads to tiny steps, slowly zigzagging down a curved valley, and a very slow convergence. This problem worsens with the depth of the net and led researchers in the 1990s to give up in practice with nets beyond around two hidden layers (with special architectures such as convolutional nets (LeCun et al., 1998) being an exception) until recently, when improved initialization strategies (Hinton and Salakhutdinov, 2006; Bengio et al., 2007) and much faster computers—but not really any improvement in the optimization algorithms themselves—have renewed interest in deep architectures. Besides, backprop does not parallelize over layers (and, with nonconvex problems, is hard to parallelize over minibatches if using SGD), is only applicable if the mappings are differentiable with respect to the parameters, and needs careful tuning of learning rates. In summary, after decades of research in neural net optimization, simple backprop-based algorithms such as stochastic gradient descent remain the state-of-the-art, particularly when combined with good initialization strategies (Orr and Müller, 1998; Hinton and Salakhutdinov, 2006). In addition, selecting the best architecture, for example the number of units in each layer of a deep net, or the number of filterbanks in a speech front-end processing, requires a combinatorial search. In practice, this is approximated with a manual trial-and-error procedure that is very costly in effort and expertise required, and leads to suboptimal solutions (where often the parameters of each layer are set irrespective of the rest of the cascade).

We describe a general optimization strategy for deeply nested functions that we call *method of auxiliary coordinates (MAC)*, which partly alleviates the vanishing gradients problem, has embarrassing parallelization, and can reuse existing algorithms (possibly not gradient-based) that optimize single layers or individual units. Section 2 describes MAC, section 3 describes related work, section 4 gives experimental results that illustrate

the different advantages of MAC, and the appendix gives formal theorem statements and proofs.

2 The method of auxiliary coordinates (MAC)

2.1 The nested objective function

For definiteness, we describe the approach for a deep net such as that of fig. 1. Later sections will show other settings. Consider a regression problem of mapping inputs \mathbf{x} to outputs \mathbf{y} (both high-dimensional) with a deep net $\mathbf{f}(\mathbf{x})$ given a dataset of N pairs $(\mathbf{x}_n, \mathbf{y}_n)$. A typical objective function to learn a deep net with K hidden layers has the form (to simplify notation, we ignore bias parameters):

$$E_1(\mathbf{W}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n; \mathbf{W})\|^2 \quad \mathbf{f}(\mathbf{x}; \mathbf{W}) = \mathbf{f}_{K+1}(\dots \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}; \mathbf{W}_1); \mathbf{W}_2) \dots; \mathbf{W}_{K+1}) \quad (1)$$

where each layer function has the form $\mathbf{f}_k(\mathbf{x}; \mathbf{W}_k) = \sigma(\mathbf{W}_k \mathbf{x})$, i.e., a linear mapping followed by a squashing nonlinearity ($\sigma(t)$ applies a scalar function, such as the sigmoid $1/(1+e^{-t})$, elementwise to a vector argument, with output in $[0, 1]$). Our method applies to loss functions other than squared error (e.g. cross-entropy for classification), with fully or sparsely connected layers each with a different number of hidden units, with weights shared across layers, and with regularization terms on the weights \mathbf{W}_k . The basic issue is the deep nesting of the mapping \mathbf{f} . The traditional way to minimize (1) is by computing the gradient over all weights of the net using backpropagation and feeding it to a nonlinear optimization method.

2.2 The method of auxiliary coordinates (MAC)

We introduce one auxiliary variable per data point and per hidden unit and define the following equality-constrained optimization problem:

$$E(\mathbf{W}, \mathbf{Z}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}_{K+1}(\mathbf{z}_{K,n}; \mathbf{W}_{K+1})\|^2 \text{ s.t. } \begin{cases} \mathbf{z}_{K,n} = \mathbf{f}_K(\mathbf{z}_{K-1,n}; \mathbf{W}_K) \\ \dots \\ \mathbf{z}_{1,n} = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1) \end{cases} n = 1, \dots, N. \quad (2)$$

Each $\mathbf{z}_{k,n}$ can be seen as the coordinates of \mathbf{x}_n on an intermediate feature space, or as the hidden unit activations for \mathbf{x}_n . Intuitively, by eliminating \mathbf{Z} we see this is equivalent to the nested problem (1); we can prove under very general assumptions that both problems have exactly the same minimizers (see appendix A.2). Problem (2) seems more complicated (more variables and constraints), but each of its terms (objective and constraints) involve only a small subset of parameters and no nested functions. Below we show this reduces the ill-conditioning caused by the nesting, and partially decouples many variables, affording an efficient and distributed optimization.

2.3 MAC with quadratic-penalty (QP) optimization

The problem (2) may be solved with a number of constrained optimization approaches. To illustrate the advantages of MAC in the simplest way, we use the quadratic-penalty (QP) method (Nocedal and Wright, 2006). We optimize the following function over (\mathbf{W}, \mathbf{Z}) for fixed $\mu > 0$ and drive $\mu \rightarrow \infty$:

$$E_Q(\mathbf{W}, \mathbf{Z}; \mu) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}_{K+1}(\mathbf{z}_{K,n}; \mathbf{W}_{K+1})\|^2 + \frac{\mu}{2} \sum_{n=1}^N \sum_{k=1}^K \|\mathbf{z}_{k,n} - \mathbf{f}_k(\mathbf{z}_{k-1,n}; \mathbf{W}_k)\|^2. \quad (3)$$

This defines a continuous path $(\mathbf{W}^*(\mu), \mathbf{Z}^*(\mu))$ which, under some mild assumptions (see proof in appendix B), converges to a minimum of the constrained problem (2), and thus to a minimum of the original problem (1). In practice, we follow this path loosely.

The QP objective function can be seen as breaking the functional dependences in the nested mapping \mathbf{f} and unfolding it over layers. Every squared term involves only a shallow mapping; all variables (\mathbf{W}, \mathbf{Z}) are equally scaled, which improves the conditioning of the problem; and the derivatives required are simpler: we require no backpropagated gradients over \mathbf{W} , and sometimes no gradients over \mathbf{W}_k at all.

We now apply alternating optimization of the QP objective over \mathbf{Z} and \mathbf{W} :

W-step Minimizing over \mathbf{W} for fixed \mathbf{Z} results in a separate minimization over the weights of each hidden unit—each a single-layer, single-unit problem that can be solved with existing algorithms. Specifically, for the unit (k, h) , for $k = 1, \dots, K + 1$ (where we define $\mathbf{z}_{K+1,n} = \mathbf{y}_n$) and $h = 1, \dots, H_k$ (assuming there are H_k units in layer k), we have a nonlinear, least-squares regression of the form $\min_{\mathbf{w}_{kh}} \sum_{n=1}^N (z_{kh,n} - f_{kh}(\mathbf{z}_{k-1,n}; \mathbf{w}_{kh}))^2$, where \mathbf{w}_{kh} is the weight vector (h th row of \mathbf{W}_k) that feeds into the h th output unit of layer k , and $z_{kh,n}$ the corresponding scalar target for point \mathbf{x}_n .

Z-step Minimizing over \mathbf{Z} for fixed \mathbf{W} separates over the coordinates \mathbf{z}_n for each data point $n = 1, \dots, N$ (omitting the subindex n and weights):

$$\min_{\mathbf{z}} \frac{1}{2} \|\mathbf{y} - \mathbf{f}_{K+1}(\mathbf{z}_K)\|^2 + \frac{\mu}{2} \sum_{k=1}^K \|\mathbf{z}_k - \mathbf{f}_k(\mathbf{z}_{k-1})\|^2 \quad (4)$$

and can be solved using the derivatives w.r.t. \mathbf{z} of the single-layer functions $\mathbf{f}_1, \dots, \mathbf{f}_{K+1}$.

Thus, the **W**-step results in many independent, single-layer single-unit problems that can be solved with existing algorithms, without extra programming cost. The **Z**-step is new, however it always has the same form (4) of a “generalized” proximal operator (Rockafellar, 1976; Combettes and Pesquet, 2011). MAC reduces a complex, highly-coupled problem—training a deep net—to a sequence of simple, uncoupled problems (the **W**-step) which are coordinated through the auxiliary variables (the **Z**-step). For a large net with a large dataset, this affords an enormous potential for parallel, distributed computation. And, because each **W**- or **Z**-step operates over very large, decoupled blocks of variables, the decrease in the QP objective function is large in each iteration, unlike the tiny decreases achieved in the nested function. These large steps are effectively shortcuts through (\mathbf{W}, \mathbf{Z}) -space, instead of tiny steps along a curved valley in \mathbf{W} -space.

Rather than an algorithm, the method of auxiliary coordinates is a mathematical device to design optimization algorithms suited for any specific nested architecture, that are provably convergent, highly parallelizable and reuse existing algorithms for non-nested (or shallow) architectures. The key idea is the judicious elimination of subexpressions in a nested function via equality constraints. The architecture need not be strictly feedforward (e.g. recurrent nets). The designer need not introduce auxiliary coordinates at every layer: there is a spectrum between no auxiliary coordinates (full nesting), through hybrids that use some auxiliary coordinates and some semi-deep nets, to every single hidden unit having an auxiliary coordinate. An auxiliary coordinate may replace any subexpression of the nested function (e.g. the input to a hidden unit, rather than its output, leading to a quadratic **W**-step). Other methods for constrained optimization may be used (e.g. the augmented Lagrangian rather than the quadratic-penalty method). Depending on the characteristics of the problem, the **W**- and **Z**-steps may be solved with any of a number of nonlinear optimization methods, from gradient descent to Newton’s method, and using standard techniques such as warm starts, caching factorizations, inexact steps, stochastic updates using data minibatches, etc. In this respect, MAC is similar to other “metaalgorithms” such as expectation-maximization (EM) algorithms (Dempster et al., 1977) and alternating-direction method of multipliers (Boyd et al., 2011), which have become ubiquitous in statistics, machine learning, optimization and other areas.

Fig. 2 illustrates MAC learning for a sigmoidal deep autoencoder architecture, introducing auxiliary coordinates for each hidden unit at each layer (see section 4.2 for details). Classical backprop-based techniques such as stochastic gradient descent and conjugate gradients need many iterations to decrease the error, but each MAC/QP iteration achieves a large decrease, particularly at the beginning, so that it can reach a pretty good network pretty fast. While MAC/QP’s serial performance is already remarkable, its parallel implementation achieves a linear speedup on the number of processors (fig. 6).

Stopping criterion Exactly optimizing $E_Q(\mathbf{W}, \mathbf{Z}; \mu)$ for each μ follows the minima path strictly but is unnecessary, and one usually performs an inexact, faster optimization. Unlike in a general QP problem, in our case we have an accurate way to know when we should exit the optimization for a given μ . Since our real goal is to minimize the nested error $E_1(\mathbf{W})$, we exit when its value increases or decreases less than a set tolerance in relative terms. Further, as is common in neural net training, we use the validation error (i.e., $E_1(\mathbf{W})$ measured on a validation set). This means we follow the path $(\mathbf{W}^*(\mu), \mathbf{Z}^*(\mu))$ not strictly but only inasmuch as we approach a nested minimum, and our approach can be seen as a sophisticated way of taking a descent step in $E_1(\mathbf{W})$ but derived from $E_Q(\mathbf{W}, \mathbf{Z}; \mu)$. Using this stopping criterion maintains our theoretical convergence guarantees, because the path still ends in a minimum of E_1 and we drive $\mu \rightarrow \infty$.

The postprocessing step Once we have finished optimizing the MAC formulation with the QP method, we can apply a fast post-processing step that both reduces the objective function, achieves feasibility and eliminates the auxiliary coordinates. We simply satisfy the constraints by setting $\mathbf{z}_{kn} = \mathbf{f}_k(\mathbf{z}_{k-1,n}; \mathbf{W}_k)$, $k = 1, \dots, K$, $n = 1, \dots, N$, and keep all the weights the same except for the last layer, where we set \mathbf{W}_{K+1} by fitting \mathbf{f}_{K+1} to the dataset $(\mathbf{f}_K(\dots(\mathbf{f}_1(\mathbf{X}))), \mathbf{Y})$. One can prove the resulting weights reduce or leave unchanged the value of $E_1(\mathbf{W})$.

2.4 Jointly learning all the parameters in heterogeneous architectures

Another important advantage of MAC is that it is easily applicable to heterogeneous architectures, where each layer may perform a particular type of processing for which a specialized training algorithm exists, possibly not based on derivatives over the weights (so that backprop is not applicable or not convenient). For example, a quantization layer of an object recognition cascade, or the nonlinear layer of a radial basis function (RBF) network, often use a k -means training to estimate the weights. Simply reusing this existing training algorithm as the \mathbf{W} -step for that layer allows MAC to learn jointly the parameters of the entire network with minimal programming effort, something that is not easy or not possible with other methods.

Fig. 3 illustrates MAC learning for an autoencoder architecture where both the encoder and the decoder are RBF networks, introducing auxiliary coordinates only at the coding layer (see section 4.3 for details). In the \mathbf{W} -step, the basis functions of each RBF net are trained with k -means, and the weights in the remaining layers are trained by least-squares. As before, MAC/QP achieves a large error decrease in a few iterations.

2.5 Model selection

A final advantage of MAC is that it enables an efficient search not just over the parameter values of a given architecture, but (to some extent) over the architectures themselves. Traditional model selection usually involves obtaining optimal parameters (by running an already costly numerical optimization) for each possible architecture, and then evaluating each architecture based on a criterion such as cross-validation or a Bayesian Information Criterion (BIC), and picking the best (Hastie et al., 2009). This discrete-continuous optimization involves training an exponential number of models, so in practice one settles with a suboptimal search (e.g. fixing by hand part of the architecture based on an expert’s judgment, or selecting parts separately and then combining them). With MAC, model selection may be achieved “on the fly” by having the \mathbf{W} -step do model selection separately for each layer, and then letting the \mathbf{Z} -step coordinate the layers in the usual way. Specifically, consider a model selection criterion of the form $E_1(\mathbf{W}) + C(\mathbf{W})$, where E_1 is the nested objective function (1) and $C(\mathbf{W})$ is additive over the layers of the net:

$$C(\mathbf{W}) = C_1(\mathbf{W}_1) + \dots + C_K(\mathbf{W}_K). \quad (5)$$

This is satisfied by many criteria, such as BIC, AIC or minimum description length (Hastie et al., 2009), in which $C(\mathbf{W})$ is essentially proportional to the number of free parameters. While optimizing $E_1(\mathbf{W}) + C(\mathbf{W})$ directly involves testing M^K deep nets if we have M choices for each layer, with MAC the \mathbf{W} -step separates over layers, and requires testing only MK single-layer nets at each iteration. While these model selection tests are still costly, they may be run in parallel, and they need not be run at each iteration. That is, we may alternate between running multiple iterations that optimize \mathbf{W} for a given architecture, and running a model-selection iteration, and we still guarantee a monotonic decrease of $E_Q(\mathbf{W}) + C(\mathbf{W})$. In practice, we observe that a near-optimal model is often found in early iterations. Thus, the ability of MAC to decouple optimizations reduces a search of an exponential number of complex problems to an iterated search of a linear number of simple problems.

Fig. 5 illustrates how to learn the architecture with MAC for the RBF autoencoder (see section 4.4 for details) by trying 50 different values for the number of basis functions in each of the encoder and decoder (a search space of $50^2 = 2500$ architectures). Because, early during the optimization, MAC/QP settles on an architecture that is quite smaller than the one used in fig. 3, the result is in fact achieved in even less time.

3 Related work

We believe we are the first to propose the MAC formulation in full generality for nested function learning as a provably equivalent, constrained problem that is to be optimized jointly in the space of parameters and auxiliary coordinates using quadratic-penalty, augmented Lagrangian or other methods. However, there exist several lines of work related to it, and MAC/QP can be seen as giving a principled setting that justifies previous heuristic but effective approaches, and opening the door for new, principled ways of training deep nets and other nested systems.

Updating the activations of hidden units separately from the weights of a neural net has been done in the past, from early work in neural nets (Grossman et al., 1988; Saad and Marom, 1990; Krogh et al., 1990; Rohwer, 1990) to recent work in learning sparse features (Olshausen and Field, 1996, 1997; Ranzato et al., 2007b; Kavukcuoglu et al., 2008) and dimensionality reduction (Carreira-Perpiñán and Lu, 2008, 2010, 2011; Wang and Carreira-Perpiñán, 2012). Interest in using the activations of neural nets as independent variables goes back to the early days of neural nets, where learning good internal representations was as important as learning good weights (Grossman et al., 1988; Saad and Marom, 1990; Krogh et al., 1990; Rohwer, 1990). In fact, backpropagation was presented as a method to construct good internal representations that represent important features of the task domain (Rumelhart et al., 1986b). This necessarily requires dealing explicitly with the hidden activations. Thus, while several papers proposed objective functions of both the weights and the activations, these were not intended to solve the nested problem or to achieve distributed optimization, but to help learn good representations. These algorithms typically did not converge at all, or did not converge to a solution of the nested problem, and were developed for a single-hidden-layer net and tested in very small problems. More recent variations have similar problems (Ma et al., 1997; Castillo et al., 2006; Erdogmus et al., 2005). Nearly all this early work has focused on the case of a single hidden layer, which is easy enough to train by standard methods, so that no great advantage is obtained, and it does not reveal the parallel processing aspects of the problem, which become truly important in the deep net case. When extracting features and using overcomplete dictionaries, sparsity is often encouraged, which sometimes requires an explicit penalty over the features, but this has only been considered for a single layer (the one that extracts the features) and again does not minimize the nested problem (Olshausen and Field, 1996, 1997; Ranzato et al., 2007b; Kavukcuoglu et al., 2008). Some work for a single hidden layer net mentions the possibility of recovering backpropagation in a limit (Krogh et al., 1990; Kavukcuoglu et al., 2008), but this is not used to construct an algorithm that converges to a nested problem optimum. Recent works in deep net learning, such as pretraining (Hinton and Salakhutdinov, 2006) or greedy layerwise training (Bengio et al., 2007; Ngiam et al., 2011), do a single pass over the net from the input to the output layer, fixing the weights of each layer sequentially, but without optimizing a joint objective of all weights. While these heuristics can be used to achieve good initial weights, they do not converge to a minimum of the nested problem.

Auxiliary variables have been used before in statistics and machine learning, from early work in factor and homogeneity analysis (Gifi, 1990), to learn dimensionality reduction mappings given a dataset of high-dimensional points $\mathbf{x}_1, \dots, \mathbf{x}_N$. Here, one takes the latent coordinates \mathbf{z}_n of each data point \mathbf{x}_n as parameters to be estimated together with the reconstruction mapping \mathbf{f} that maps latent points to data space and minimize a least-squares error function $\sum_{n=1}^N \|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2$, often by alternating over \mathbf{f} and \mathbf{Z} . Various nonlinear versions of this approach exist where \mathbf{f} is a spline (LeBlanc and Tibshirani, 1994), single-layer neural net (Tan and Mavrovouniotis, 1995), radial basis function net (Smola et al., 2001), kernel regression (Meinicke et al., 2005) or Gaussian process (Lawrence, 2005). However, particularly with nonparametric functions, the error can be driven to zero by separating infinitely apart the \mathbf{Z} , and so these methods need ad-hoc terms on \mathbf{Z} to prevent this. The dimensionality reduction by unsupervised regression approach of Carreira-Perpiñán and Lu (2008, 2010, 2011) (generalized to supervised dimensionality reduction in Wang and Carreira-Perpiñán, 2012) solves this by optimizing instead $\sum_{n=1}^N \|\mathbf{z}_n - \mathbf{F}(\mathbf{x}_n)\|^2 + \|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2$ jointly over \mathbf{Z} , \mathbf{f} and the projection mapping \mathbf{F} (both RBF networks). This can be seen as a truncated version of our quadratic-penalty approach, where μ is kept constant, and limited to a single-hidden-layer net. Therefore, the resulting estimate for the nested mapping $\mathbf{f}(\mathbf{F}(\mathbf{x}))$ is biased, as it does not minimize the nested error.

In summary, these works were typically concerned with single-hidden-layer architectures, and did not solve the nested problem (1). Instead, their goal was to define a different problem (having a different solution): one where the designer has explicit control over the net's internal representations or features

(e.g. to encourage sparsity or some other desired property). In MAC, the auxiliary coordinates are purely a mathematical construct to solve a well-defined, general nested optimization problem, with embarrassing parallelism suitable for distributed computation, and is not necessarily related to learning good hidden representations. Also, none of these works realize the possibility of using heterogeneous architectures with layer-specific algorithms, or of learning the architecture itself by minimizing a model selection criterion that separates in the \mathbf{W} -step.

Finally, the MAC formulation is similar in spirit to the alternating direction method of multipliers (ADMM) (Boyd et al., 2011; Combettes and Pesquet, 2011) in that variables (the auxiliary coordinates) are introduced that decouple terms. However, ADMM splits an existing variable that appears in multiple terms of the objective function (which then decouple) rather than a functional nesting, for example $\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x})$ becomes $\min_{\mathbf{x}, \mathbf{y}} f(\mathbf{x}) + g(\mathbf{y})$ s.t. $\mathbf{x} = \mathbf{y}$, or \mathbf{x} is split into non-negative and non-positive parts. In contrast, MAC introduces new variables to break the nesting. ADMM is known to be very simple, effective and parallelizable, and to be able to achieve a pretty good estimate pretty fast, thanks to the decoupling introduced and the ability to use existing optimizers for the subproblems that arise. MAC also has these characteristics with problems involving function nesting.

4 Experiments

Section 4.1 describes how we implemented the \mathbf{W} - and \mathbf{Z} -steps and sections 4.2, 4.3 and 4.4 show how MAC can learn a homogeneous architecture (deep sigmoidal autoencoder), a heterogeneous architecture (RBF autoencoder) and the architecture itself, respectively. In all cases, we show the speedup achieved with a parallel implementation of MAC as well.

4.1 Optimization of the MAC-constrained problem using a quadratic penalty

We apply alternating optimization of the QP objective (3) over \mathbf{Z} and \mathbf{W} :

W-step Minimizing over \mathbf{W} for fixed \mathbf{Z} results in a separate nonlinear, least-squares regression of the form $\min_{\mathbf{w}_{kh}} \sum_{n=1}^N (z_{kh,n} - f_{kh}(\mathbf{z}_{k-1,n}; \mathbf{w}_{kh}))^2$ for $k = 1, \dots, K+1$ (where we define $\mathbf{z}_{K+1,n} = \mathbf{y}_n$) and $h = 1, \dots, H$, where \mathbf{w}_{kh} is the weight vector (h th row of \mathbf{W}_k) that feeds into the h th output unit of layer k (assuming there are H such units), and $z_{kh,n}$ the corresponding scalar target for point \mathbf{x}_n . We solve each of these KH problems with a Gauss-Newton approach (Nocedal and Wright, 2006), which essentially approximates the Hessian by linearizing the function f_{kh} , solves a linear system of $H \times H$ (the number of units feeding into z_{kh}) to get a search direction, does a line search (we use backtracking with initial step size 1), and iterates. In practice 1–2 iterations converge with high tolerance.

Z-step Minimizing over \mathbf{Z} for fixed \mathbf{W} separates over each \mathbf{z}_n for $n = 1, \dots, N$. The problem is also a nonlinear least-squares fit, formally very similar to those of the \mathbf{W} -step, because \mathbf{Z} and \mathbf{W} enter the objective function in a nearly symmetric way through $\sigma(\mathbf{W}_k \mathbf{z}_{k-1})$, but with additional quadratic terms $\|\mathbf{z}_{k,n} - \dots\|^2$. We optimize it again with the Gauss-Newton method, which usually spends 1–2 iterations.

This optimization of the MAC-constrained problem, based on a quadratic-penalty method with Gauss-Newton steps, produces reasonable results and is simple to implement, but it is not intended to be particularly efficient. A more efficient optimization can be achieved by (1) using other methods for constrained optimization, such as the augmented Lagrangian method instead of the quadratic penalty method; and (2) by using more efficient \mathbf{W} - or \mathbf{Z} -steps, by combining standard techniques (inexact steps with warm starts, caching factorizations, stochastic updates using data minibatches, etc.) with unconstrained optimization methods such as L-BFGS, conjugate gradients, gradient descent, alternating optimization or others. Exploring this is a topic of future research.

Parallel implementation of MAC/QP Our parallel implementation of MAC/QP is extremely simple at present, yet it achieves large speedups (about 6× faster if using 12 processors), which are nearly linear as a function of the number of processors for all experiments, as shown in fig. 6. Given that our code is in Matlab, we used the Matlab Parallel Processing Toolbox. The programming effort is insignificant: all we do

is to replace the “for” loop over weight vectors (in the \mathbf{W} -step) or over auxiliary coordinates (in the \mathbf{Z} -step) with a “parfor” loop. Matlab then sends each iteration of the loop to a different processor. We ran this in a shared-memory multiprocessor machine¹, using up to 12 processors (a limit imposed by our Matlab license) and obtained the results reported in the paper. While simple, the Matlab Parallel Processing Toolbox is quite inefficient. Larger speedups would be achievable with other parallel computation models such as MPI in C, and using a distributed architecture (so that cache and other overheads are reduced).

4.2 Homogeneous training: deep sigmoidal autoencoder

We use a dataset of handwritten digit images to train a deep autoencoder architecture that maps the input image to a low-dimensional coding layer and then tries to reconstruct the image from it. We used MAC/QP introducing auxiliary coordinates for each hidden unit at each layer. Fig. 2 shows the learning curves.

The USPS dataset (Hull, 1994), a commonly used machine learning benchmark, contains 16×16 grayscale images of handwritten digits, i.e., 256D vectors with values in $[0, 1]$. We use $N = 5\,000$ images for training and 5 000 for validation, both randomly selected equally over all digits.

The autoencoder architecture is 256–300–100–20–100–300–256, for a total of over 200 000 weights, with all $K = 5$ hidden layers being logistic sigmoid units and the output layer being linear. The initial weights are uniformly sampled from $[-1/\sqrt{f_k}, 1/\sqrt{f_k}]$ for layers $k = 1, \dots, K$, respectively, where f_k is the input dimension (fan-in) to each hidden layer (Orr and Müller, 1998). When using initial random weights, a large (but easy) decrease in the error can be achieved simply by adjusting the biases of the output layer so the network output matches the mean of the target data, and all algorithms attain this in the first few iterations, giving the impression of a large error decrease followed by a very slow subsequent decrease. To focus the plots on the subsequent, more difficult error decreases, we always apply a single step of gradient descent to the random initial weights, which mostly adjusts the biases as described. The resulting weights are given as initial weights to each optimization method.

MAC/QP runs the \mathbf{Z} -step with 1 Gauss-Newton iteration and the \mathbf{W} -step with up to 3 Gauss-Newton iterations. We also use a small regularization on \mathbf{W} in the first few iterations, which we drop once $\mu > 10^4$, since we find that this tends to lead to a better local optimum. For each value of μ , we optimize $E_Q(\mathbf{W}, \mathbf{Z}; \mu)$ in an inexact way as described in section 2, exiting when the value of the nested error $E_1(\mathbf{W})$, evaluated on a validation set, increases or decreases less than a set tolerance in relative terms. We use a tolerance of 10^{-2} and increase μ rather aggressively, to 10μ .

We show the learning curves for two classical backprop-based techniques, stochastic gradient descent and conjugate gradients. Stochastic gradient descent (SGD) has several parameters that should be carefully set by the user to ensure that convergence occurs, and that convergence is as fast as possible. We did a grid search for the minibatch size in $\{1, 10, 20, 50, 100, 200, 500, 1000, 5000\}$ and learning rate in $\{1, 10, 100, 1000\} \times 10^{-7}$. We found minibatches of 20 samples and a learning rate of 10^{-6} were best. We randomly permute the training set at the beginning of each epoch. For nonlinear conjugate gradients (CG), we used the Polak-Ribière version, widely regarded as the best (Nocedal and Wright, 2006). We use Carl Rasmussen’s implementation `minimize.m` (available at <http://learning.eng.cam.ac.uk/carl/code/minimize/minimize.m>), which uses a line search based on cubic interpolation that is more sophisticated than backtracking, and allows steps longer than 1. We found a minibatch of N (i.e., batch mode) worked best. We used restarts every 100 steps.

Figure 2(left) plots the mean squared training error for the nested objective function (1) vs run time for the different algorithms after the initialization. The validation error follows closely the training error and is not plotted. Markers are shown every iteration (MAC), every 100 iterations (CG), or every 20 epochs (SGD, one epoch is one pass over the training set). The change points of the quadratic penalty parameter μ are indicated using filled red circles at the beginning of each iteration when they happened. The learning curve of the parallelized version of MAC (using 12 processors) is also shown in blue. Fig. 2(right) shows some USPS images and their reconstruction at the end of training for each algorithm.

SGD and CG need many iterations to decrease the error, but each MAC/QP iteration achieves a large decrease, particularly at the beginning, so that it can reach a pretty good network pretty fast. While MAC/QP’s serial performance is already remarkable, its parallel implementation achieves a linear speedup on the number of processors (fig. 6).

¹An Aberdeen Stirling 148 computer having 4 physical CPUs (Intel Xeon CPU L7555@ 1.87GHz), each with 8 individual processing cores (thus a total of 32 actual processors), and a total RAM size of 64 GB.

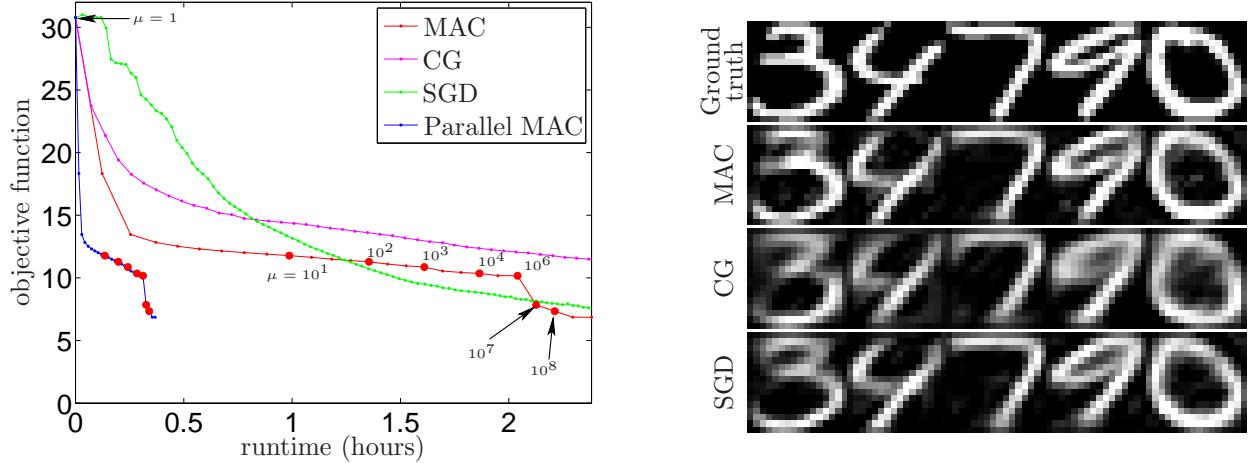


Figure 2: Training a deep autoencoder to reconstruct images of handwritten digits. *Left:* nested function error (1) for each algorithm, with markers shown every iteration (MAC), every 100 iterations (CG), or every 20 epochs (SGD). For MAC/QP, we incremented the quadratic penalty μ as indicated at the red solid markers. All these experiments were run in the same computer using a single processor, except the parallel MAC training curve, which used 12 processors sharing the same memory using the Matlab Parallel Processing Toolbox. *Right:* reconstruction of sample training images by different methods.

4.3 Heterogeneous training: radial basis functions autoencoder

We use a dataset of object images to train an autoencoder architecture where both the encoder and decoder are RBF networks but, rather than using a gradient-based optimization for each subnet, we use k -means to train the basis functions of each RBF in the \mathbf{W} -step (while the weights in the remaining layers are trained by least-squares). We used MAC/QP introducing auxiliary coordinates only at the coding layer. Backprop-based algorithms are incompatible with k -means training, so instead we compare with alternating optimization. Fig. 3 shows the learning curves.

The COIL-20 image dataset (Nene et al., 1996), commonly used as a benchmark to test dimensionality reduction algorithms, contains rotation sequences of 20 different objects every 5 degrees (i.e., 72 images per object), each a grayscale image with pixel intensity in $[0, 1]$. We resize the images to 32×32 . Thus, the data contain 20 closed, nonlinear 1D manifolds in a 1 024-dimensional space. We pick half of the images from objects 1 (duck) and 4 (cat) as validation set, which leaves a training set containing $N = 1368$ images.

The autoencoder architecture is as follows. The bottleneck layer of low-dimensional codes has only 2 units, so that we can visualize the data. The encoder reduces the dimensionality of the input image to 2D, while the decoder reconstructs the image as best as possible from the 2D representation. Both the encoder and the decoder are radial basis function (RBF) networks, each having a single hidden layer. The first one (encoder) has the form $\mathbf{z} = \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}; \mathbf{W}_1); \mathbf{W}_2) = \mathbf{W}_2 \mathbf{f}_1(\mathbf{x}; \mathbf{W}_1)$, where the vector $\mathbf{f}_1(\mathbf{x}; \mathbf{W}_1)$ has $M_1 = 1368$ elements (basis functions) of the form $\exp(-\|(\mathbf{x} - \mathbf{w}_{1i})/\sigma_1\|^2)$, $i = 1, \dots, M_1$, with $\sigma_1 = 4$, and maps an image \mathbf{x} to a 2D space \mathbf{z} . The second one (decoder) has the form $\mathbf{x}' = \mathbf{f}_4(\mathbf{f}_3(\mathbf{z}; \mathbf{W}_3); \mathbf{W}_4) = \mathbf{W}_4 \mathbf{f}_3(\mathbf{z}; \mathbf{W}_3)$, where the vector $\mathbf{f}_3(\mathbf{z}; \mathbf{W}_3)$ has $M_3 = 1368$ elements (basis functions) of the form $\exp(-\|(\mathbf{x} - \mathbf{w}_{3i})/\sigma_3\|^2)$, $i = 1, \dots, M_3$, with $\sigma_3 = 0.5$, and maps a 2D point \mathbf{z} to a 1 024D image. Thus, the complete autoencoder is the concatenation of the two Gaussian RBF networks, it has $K = 3$ hidden layers with sizes 1 024–1 368–2–1 368–1 024, and a total of almost 3 million weights. As is usual with RBF networks, we applied a quadratic regularization to the linear-layer weights with a small value ($\lambda_2 = \lambda_4 = 10^{-3}$). The nested problem is then to minimize the following objective function, which is a least-squares error plus a quadratic regularization

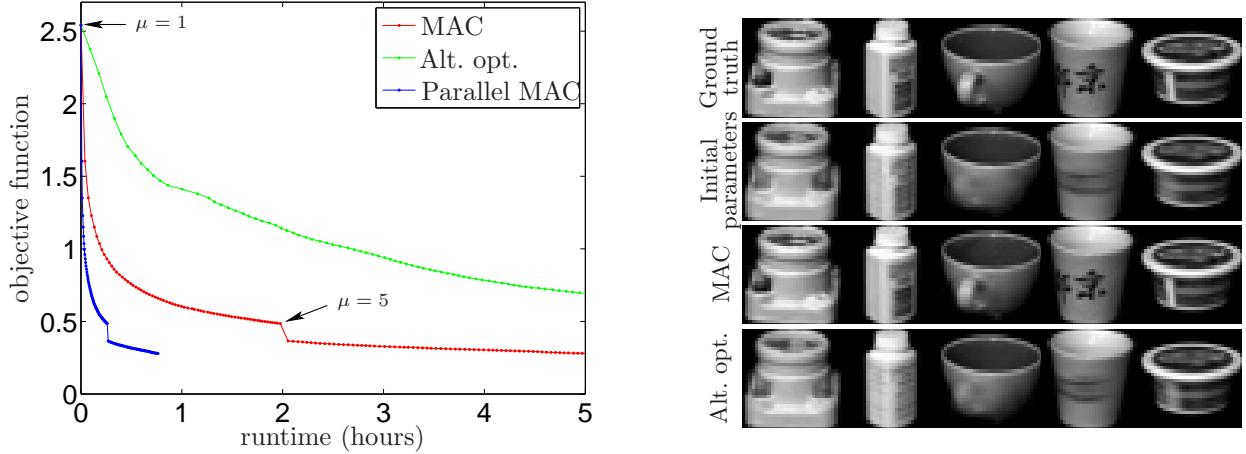


Figure 3: Training an RBF autoencoder to reconstruct images of objects. *Left*: nested function error (1) for each algorithm, with markers shown every iteration (MAC, alternating optimization). Other details as in fig. 2. *Right*: reconstruction of sample training images by different methods.

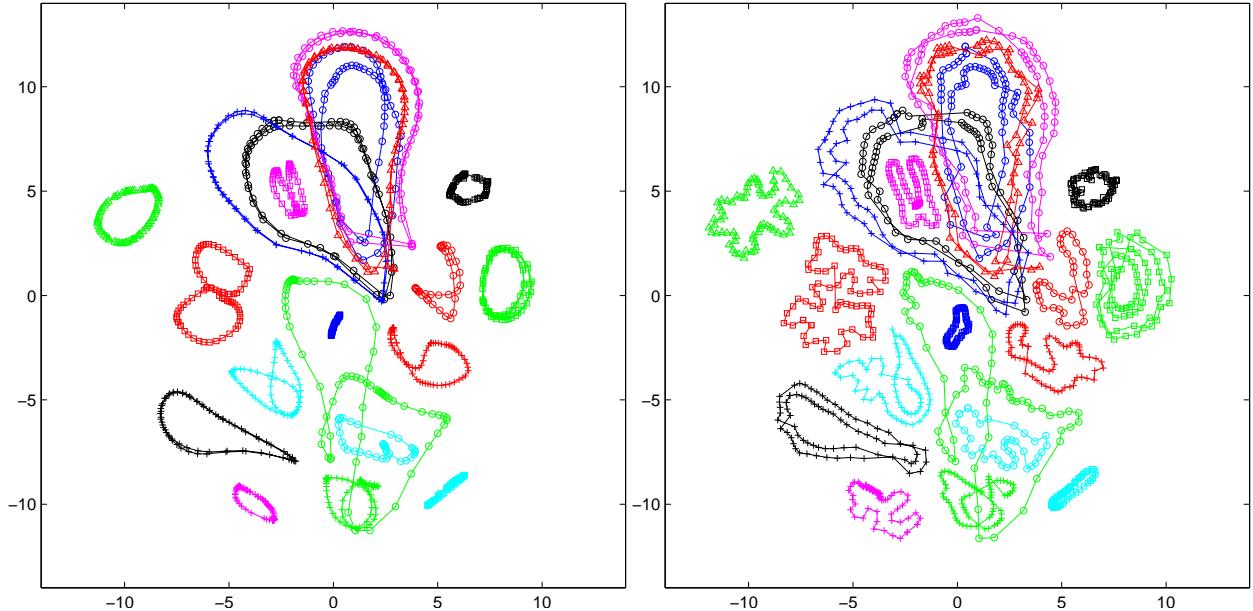


Figure 4: Values of the \mathbf{Z} coordinates (2D latent space) for the COIL-20 data set. *Left*: initialization obtained from the elastic embedding algorithm. *Right*: results from MAC at the end of training (section 4.3).

on the linear-layer weights:

$$E_1(\mathbf{W}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n; \mathbf{W})\|^2 + \lambda_2 \|\mathbf{W}_2\|_F^2 + \lambda_4 \|\mathbf{W}_4\|_F^2 \quad \mathbf{f}(\mathbf{x}; \mathbf{W}) = \mathbf{f}_4(\mathbf{f}_3(\mathbf{f}_2(\mathbf{f}_1(\mathbf{x}; \mathbf{W}_1); \mathbf{W}_2); \mathbf{W}_3); \mathbf{W}_4). \quad (6)$$

In practice, RBF networks are trained in two stages (Bishop, 2006). Consider the encoder, for example. First, one trains the centers \mathbf{W}_1 using a clustering algorithm applied to the inputs $\{\mathbf{x}_n\}_{n=1}^N$, typically k -means or (when the number of centers is large) simply by fixing them to be a random subset of the inputs. Second, having determined \mathbf{W}_1 , one obtains \mathbf{W}_2 from a linear least-squares problem, by solving a linear system.

The reason why this is preferred to a fully nonlinear optimization over centers \mathbf{W}_1 and weights \mathbf{W}_2 is that it achieves near-optimal nets with a simple, noniterative procedure. This type of two-stage noniterative strategy to obtain nonlinear networks is widely applied beyond RBF networks, for example with support vector machines (Schölkopf and Smola, 2001), kernel PCA (Schölkopf et al., 1998), slice inverse regression (Li, 1991) and others.

We wish to capitalize on this attractive property to train deep autoencoders constructed by concatenating RBF networks. However, backprop-based algorithms are incompatible with this two-stage training procedure, since it does not use derivatives to optimize over the centers. This leads us to the two following optimization methods: an alternating optimization approach, and MAC.

We can use an alternating optimization approach where we alternate the following two steps: (1) A step where we fix $(\mathbf{W}_1, \mathbf{W}_2)$ and train $(\mathbf{W}_3, \mathbf{W}_4)$, by applying k -means to \mathbf{W}_3 and a linear system to \mathbf{W}_4 . This step is identical to the \mathbf{W} -step in MAC over $(\mathbf{W}_1, \mathbf{W}_2)$. (2) A step where we fix $(\mathbf{W}_3, \mathbf{W}_4)$ and train $(\mathbf{W}_1, \mathbf{W}_2)$, by applying k -means to \mathbf{W}_1 and a nonlinear optimization to \mathbf{W}_2 (we use nonlinear conjugate gradients with 10 steps). This is because \mathbf{W}_2 no longer appears linearly in the objective function, but is nonlinearly embedded as the argument of the decoder. This step is significantly slower than the \mathbf{W} -step in MAC over $(\mathbf{W}_3, \mathbf{W}_4)$.

We define the MAC-constrained problem as follows. We introduce auxiliary coordinates only at the coding layer (rather than at all $K = 3$ hidden layers). This allows the \mathbf{W} -step to become the desired k -means plus linear system training for the encoder and decoder separately. It requires no programming effort; we simply call an existing, k -means-based RBF training algorithm for each of the encoder and decoder separately. We start with a quadratic penalty parameter $\mu = 1$ and increase it to $\mu = 5$ after 70 iterations.

Since we use as many centers as data points ($M_1 = M_3 = N$), the k -means step simplifies (for both methods) to setting each basis function center to an input point.

In this experiment, instead of using random initial weights, we obtained initial values for the \mathbf{Z} coordinates by running a nonlinear dimensionality reduction method, the elastic embedding (EE) (Carreira-Perpiñán, 2010); this gives significantly better embeddings than spectral methods; (Tenenbaum et al., 2000; Roweis and Saul, 2000). This takes as input a matrix of $N \times N$ similarity values between every pair of COIL images $\mathbf{x}_1, \dots, \mathbf{x}_N$, and produces a nonlinear projection \mathbf{z}_n in 2D for each \mathbf{x}_n . We used Gaussian similarities with a kernel width of 10 and run EE for 200 iterations using $\lambda = 100$ as its user parameter. All the optimization algorithms were initialized from these projections.

Fig. 3(left) shows the nested function error (6) (normalized per data point, i.e., divided by N). As before, MAC/QP achieves a large error decrease in a few iterations. Alternating optimization is much slower. Again, a parallel implementation of MAC/QP achieves a large speedup, which is nearly linear on the number of processors (fig. 6). Fig. 3(right) shows some COIL images and their reconstruction at the end of training for each algorithm. Fig. 4 shows the initial projections \mathbf{Z} (from the elastic embedding algorithm) and the final projections \mathbf{Z} (after running MAC/QP). Most of the manifolds have improved, for example opening up loops that were folded.

4.4 Learning the architecture: RBF autoencoder

We repeat the experiment of the RBF autoencoder of the previous section, but now we learn its architecture. We jointly learn the architecture of the encoder and of the decoder by trying 50 different values for the number of basis functions in each (a search space of $50^2 = 2500$ architectures). We define the following objective function over architectures and their weights:

$$\bar{E}(\mathbf{W}) = E_1(\mathbf{W}) + C(\mathbf{W}) \quad (7)$$

where $E_1(\mathbf{W})$ is the nested error from eq. (6) (including regularization terms), and $C(\mathbf{W}) = C(\mathbf{W}_1) + \dots + C(\mathbf{W}_4)$ is the model selection term. We use the well-known AIC criterion (Hastie et al., 2009). This is defined as

$$\bar{E}(\Theta) = \text{SSE}(\Theta) + C(\Theta) \quad C(\Theta) = 2\epsilon^2 |\Theta| \quad (8)$$

(times a constant $\frac{1}{N}$, which we omit) where $\text{SSE}(\Theta)$ is the sum of squared errors achieved in the training set with a model having parameters Θ , ϵ^2 is the mean squared error achieved by a low-bias model (typically

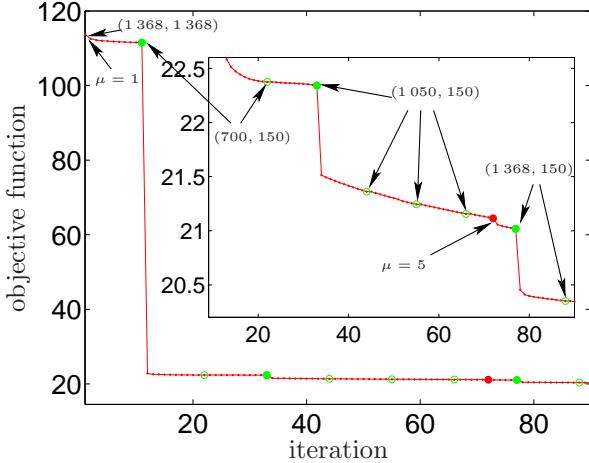


Figure 5: Learning the architecture of the RBF autoencoder for the dataset of fig. 3 using MAC. We show the total error $E_1(\mathbf{W}) + C(\mathbf{W})$ (the nested function error (1) plus the model cost) per point. Model selection steps are run every 10 iterations and are indicated with green markers (solid if the architecture changes and empty if it does not change). Other details as in fig. 3.

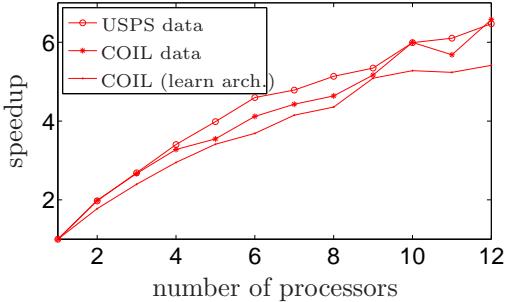


Figure 6: Parallel processing speedup of MAC/QP as a function of the number of processors for each of the experiments of figures 2, 3 and 5. The speedup in all our experiments was approximately linear, reaching about 6 \times with 12 processors.

estimated by training a model with many parameters) and $|\Theta|$ is the number of free parameters in the model. In our case, this means we use $C(\mathbf{W})$ defined as

$$C(\mathbf{W}) = 2\epsilon^2 |\mathbf{W}| = 2\epsilon^2(DM_1 + LM_1 + LM_3 + DM_3) = 2\epsilon^2(D + L)(M_1 + M_3) \quad (9)$$

where M_1 and M_3 are the numbers of centers for the encoder and decoder, respectively (first and third hidden layers), and $D = 1024$ and $L = 2$ are the input and output dimension of the encoder, respectively (equivalently, the output and input dimension of the decoder). The total number of free parameters (centers and linear weights) in the autoencoder is thus $|\mathbf{W}| = (D + L)(M_1 + M_3)$.

We choose each of the numbers of centers M_1 and M_3 from a discrete set consisting of the 50 equispaced values in the range 150 to 1368 (a total of $50^2 = 2500$ different architectures). We estimated $\epsilon^2 = 0.05$ from the result of the RBF autoencoder of section 4.3, which had a large number of parameters and thus a low bias. As in that section, the centers of each network are constrained to be equal to a subset of the input points (chosen at random). We set $\sigma_1 = 4$, $\sigma_3 = 2.5$ and $\lambda_1 = \lambda_2 = 10^{-3}$. We start the MAC/QP optimization from the most complex model, having $M_1 = M_3 = 1368$ centers (i.e., the model of the previous section). While every iteration optimizes the MAC/QP objective function (3) over (\mathbf{W}, \mathbf{Z}) , we run a model selection step only every 10 iterations. This selects separately for each net the best M_k value and potentially

changes the size of \mathbf{W} . Thus, every 11th iteration is a model selection step, which may or may not change the architecture.

Figure 5 shows the total error $\bar{E}(\mathbf{W}) = E_1(\mathbf{W}) + C(\mathbf{W})$ of eq. (7) (the nested function error plus the model cost). Model selection steps are indicated with green markers (solid if the architecture did change and empty if it did not change), annotated with the resulting value of (M_1, M_3) . Other details are as in fig. 3. The first change of architecture moves to a far smaller model $(M_1, M_3) = (700, 150)$, achieving an enormous decrease in objective. This is explained by the strong penalty that AIC imposes on the number of parameters, favoring simpler models. Then, this is followed by a few minor changes of architecture interleaved with a continuous optimization of its weights. The final architecture has $(M_1, M_3) = (1\,368, 150)$, for a total of 1.5 million weights. While this architecture incurs a larger training error than that of the previous section, it uses a much simpler model and has a lower value for the overall objective function of eq. (7). Because, early during the optimization, MAC/QP settles on an architecture that is quite smaller than the one used in fig. 3, the result is in fact achieved in even less time. And, again, the parallel implementation is trivial and achieves an approximately linear speedup on the number of processors (fig. 6).

5 Conclusion

MAC drastically facilitates, in runtime and human effort, the practical design and estimation of nonconvex, nested problems by jointly optimizing over all parameters, reusing existing algorithms, searching automatically over architectures and affording massively parallel computation, while provably converging to a solution of the nested problem. It could replace or complement backpropagation-based algorithms in learning nested systems both in the serial and parallel settings. It is particularly timely given that serial computation is reaching a plateau and cloud computing is becoming a commodity, and intelligent data processing is finding its way into mainstream devices (phones, cameras, etc.), thanks to increases in computational power and data availability. An important area of application may be the joint, automatic tuning of all stages of a complex, intelligent-processing system in data-rich disciplines, such as those found in computer vision and speech, in a distributed cloud computing environment. MAC also opens many questions, such as the optimal way to introduce auxiliary coordinates in a given problem, and the choice of specific algorithms to optimize the \mathbf{W} - and \mathbf{Z} -steps.

6 Acknowledgments

Work funded in part by NSF CAREER award IIS-0754089.

A Theorems and proofs

A.1 Definitions

Consider a regression problem of mapping inputs \mathbf{x} to outputs \mathbf{y} (both high-dimensional) with a deep net $\mathbf{f}(\mathbf{x})$ given a dataset of N pairs $(\mathbf{x}_n, \mathbf{y}_n)$. We define the *nested objective function* to learn a deep net with K hidden layers, like that in fig. 1, as (to simplify notation, we ignore bias parameters and assume each hidden layer has H units):

$$E_1(\mathbf{W}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n; \mathbf{W})\|^2 \quad \mathbf{f}(\mathbf{x}; \mathbf{W}) = \mathbf{f}_{K+1}(\dots \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}; \mathbf{W}_1); \mathbf{W}_2) \dots; \mathbf{W}_{K+1}) \quad (1)$$

where each layer function has the form $\mathbf{f}_k(\mathbf{x}; \mathbf{W}_k) = \sigma(\mathbf{W}_k \mathbf{x})$, i.e., a linear mapping followed by a squashing nonlinearity ($\sigma(t)$ applies a scalar function, such as the sigmoid $1/(1+e^{-t})$, elementwise to a vector argument, with output in $[0, 1]$).

In the *method of auxiliary coordinates (MAC)*, we introduce one auxiliary variable per data point and per hidden unit (so $\mathbf{Z} = (\mathbf{Z}_1, \dots, \mathbf{Z}_N)$, with $\mathbf{z}_n = (\mathbf{z}_{1,n}, \dots, \mathbf{z}_{K,n})$) and define the following equality-constrained

optimization problem:

$$E(\mathbf{W}, \mathbf{Z}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}_{K+1}(\mathbf{z}_{K,n}; \mathbf{W}_{K+1})\|^2 \text{ s.t. } \begin{cases} \mathbf{z}_{K,n} = \mathbf{f}_K(\mathbf{z}_{K-1,n}; \mathbf{W}_K) \\ \vdots \\ \mathbf{z}_{1,n} = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1) \end{cases} n = 1, \dots, N. \quad (2)$$

Sometimes, for notational convenience (in particular in theorem B.3), we will write the constraints for the n th point as a single vector constraint $\mathbf{z}_n - \mathbf{F}(\mathbf{z}_n, \mathbf{W}; \mathbf{x}_n) = \mathbf{0}$ (with an obvious definition for \mathbf{F}). We will also call Ω the feasible set of the MAC-constrained problem, i.e.,

$$\Omega = \{(\mathbf{W}, \mathbf{Z}): \mathbf{z}_n = \mathbf{F}(\mathbf{z}_n, \mathbf{W}; \mathbf{x}_n), n = 1, \dots, N\}. \quad (10)$$

To solve the constrained problem (2) using the quadratic-penalty (QP) method (Nocedal and Wright, 2006), we optimize the following function over (\mathbf{W}, \mathbf{Z}) for fixed $\mu > 0$ and drive $\mu \rightarrow \infty$:

$$E_Q(\mathbf{W}, \mathbf{Z}; \mu) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}_{K+1}(\mathbf{z}_{K,n}; \mathbf{W}_{K+1})\|^2 + \frac{\mu}{2} \sum_{n=1}^N \sum_{k=1}^K \|\mathbf{z}_{k,n} - \mathbf{f}_k(\mathbf{z}_{k-1,n}; \mathbf{W}_k)\|^2. \quad (3)$$

A.2 Equivalence of the MAC and nested formulations

First, we give a theorem that holds under very general assumptions. In particular, it does not require the functions to be smooth, it holds for any loss function beyond the least-squares one, and it holds if the nested problem is itself subject to constraints.

Theorem A.1. *The nested problem (1) and the MAC-constrained problem (2) are equivalent in the sense that their minimizers are in a one-to-one correspondence.*

Proof. Let us prove that any minimizer of the nested problem is associated with a unique minimizer of the MAC-constrained problem (\Rightarrow), and vice versa (\Leftarrow). Recall the following definitions (Nocedal and Wright, 2006): (i) For an unconstrained minimization problem $\min_{\mathbf{x} \in \mathbb{R}^n} F(\mathbf{x})$, $\mathbf{x}^* \in \mathbb{R}^n$ is a local minimizer if there exists a nonempty neighborhood $\mathcal{N} \subset \mathbb{R}^n$ of \mathbf{x}^* such that $F(\mathbf{x}^*) \leq F(\mathbf{x}) \forall \mathbf{x} \in \mathcal{N}$. (ii) For a constrained minimization problem $\min F(\mathbf{x})$ s.t. $\mathbf{x} \in \Omega \subset \mathbb{R}^n$, $\mathbf{x}^* \in \mathbb{R}^n$ is a local minimizer if $\mathbf{x}^* \in \Omega$ and there exists a nonempty neighborhood $\mathcal{N} \subset \mathbb{R}^n$ of \mathbf{x}^* such that $F(\mathbf{x}^*) \leq F(\mathbf{x}) \forall \mathbf{x} \in \mathcal{N} \cap \Omega$.

Define the “forward-propagation” function $\mathbf{g}(\mathbf{W})$ as the result of mapping $\mathbf{z}_{1,n} = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1), \dots, \mathbf{z}_{K,n} = \mathbf{f}_K(\mathbf{z}_{K-1,n}; \mathbf{W}_K)$ for $n = 1, \dots, N$. This maps each \mathbf{W} to a unique \mathbf{Z} , and satisfies $\mathbf{f}_{K+1}(\mathbf{z}_{K,n}; \mathbf{W}_{K+1}) = \mathbf{f}_{K+1}(\dots \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1); \mathbf{W}_2) \dots; \mathbf{W}_{K+1}) = \mathbf{f}(\mathbf{x}_n; \mathbf{W})$ for $n = 1, \dots, N$, and therefore that $E_1(\mathbf{W}) = E(\mathbf{W}, \mathbf{g}(\mathbf{W}))$ for any \mathbf{W} .

(\Rightarrow) Let \mathbf{W}^* be a local minimizer of the nested problem (1). Then, there exists a nonempty neighborhood \mathcal{N} of \mathbf{W}^* such that $E_1(\mathbf{W}^*) \leq E_1(\mathbf{W}) \forall \mathbf{W} \in \mathcal{N}$. Let $\mathbf{Z}^* = \mathbf{g}(\mathbf{W}^*)$ and call $\mathcal{M} = \{(\mathbf{W}, \mathbf{Z}): \mathbf{W} \in \mathcal{N} \text{ and } \mathbf{Z} = \mathbf{g}(\mathbf{W})\}$, which is a nonempty neighborhood of $(\mathbf{W}^*, \mathbf{Z}^*)$ in (\mathbf{W}, \mathbf{Z}) -space. Now, for any $(\mathbf{W}, \mathbf{Z}) \in \mathcal{M} \cap \Omega$ we have that $E(\mathbf{W}, \mathbf{Z}) = E(\mathbf{W}, \mathbf{g}(\mathbf{W})) = E_1(\mathbf{W}) \geq E_1(\mathbf{W}^*) = E(\mathbf{W}^*, \mathbf{g}(\mathbf{W}^*)) = E(\mathbf{W}^*, \mathbf{Z}^*)$. Hence $(\mathbf{W}^*, \mathbf{Z}^*)$ is a local minimizer of the MAC-constrained problem.

(\Leftarrow) Let $(\mathbf{W}^*, \mathbf{Z}^*)$ be a local minimizer of the MAC-constrained problem (2). Then, there exists a nonempty neighborhood \mathcal{M} of $(\mathbf{W}^*, \mathbf{Z}^*)$ such that $E(\mathbf{W}^*, \mathbf{Z}^*) \leq E(\mathbf{W}, \mathbf{Z}) \forall (\mathbf{W}, \mathbf{Z}) \in \mathcal{M} \cap \Omega$. Note that $(\mathbf{W}, \mathbf{Z}) \in \mathcal{M} \cap \Omega \Rightarrow \mathbf{Z} = \mathbf{g}(\mathbf{W}) \Rightarrow E(\mathbf{W}, \mathbf{Z}) = E_1(\mathbf{W})$, and this applies in particular to $(\mathbf{W}^*, \mathbf{Z}^*)$ (which, being a solution, is feasible and thus belongs to $\mathcal{M} \cap \Omega$). Calling $\mathcal{N} = \{\mathbf{W}: (\mathbf{W}, \mathbf{Z}) \in \mathcal{M} \cap \Omega\}$, we have that $\forall \mathbf{W} \in \mathcal{N}: E_1(\mathbf{W}) = E(\mathbf{W}, \mathbf{g}(\mathbf{W})) = E(\mathbf{W}, \mathbf{Z}) \geq E(\mathbf{W}^*, \mathbf{Z}^*) = E(\mathbf{W}^*, \mathbf{g}(\mathbf{W}^*)) = E_1(\mathbf{W}^*)$. Hence \mathbf{W}^* is a local minimizer of the nested problem.

Finally, one can see that the proof holds if the nested problem uses a loss function that is not the least-squares one, and if the nested problem is itself subject to constraints. \square

Obviously, the theorem holds if we exchange \geq with $>$ everywhere (thus exchanging non-strict with strict minimizers), and if we exchange “min” with “max” (hence the maximizers of the MAC and nested formulations are in a one-to-one correspondence as well). Figure 7 illustrates the theorem. Essentially, the nested objective function $E_1(\mathbf{W})$ stretches along the manifold defined by $(\mathbf{W}, \mathbf{Z} = \mathbf{g}(\mathbf{W}))$ preserving the minimizers and maximizers. The projection on \mathbf{W} -space of the part of $E(\mathbf{W}, \mathbf{Z})$ that sits on top of that manifold recovers $E_1(\mathbf{W})$.

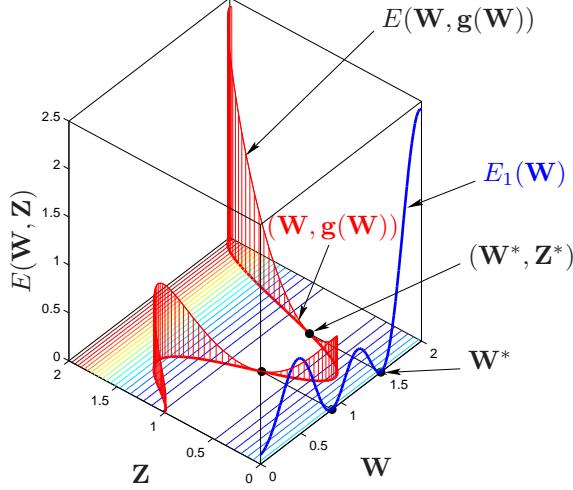


Figure 7: Illustration of the equivalence between the nested and MAC-constrained problems (see the proof of theorem A.1). The MAC objective function $E(\mathbf{W}, \mathbf{Z})$ is shown with contour lines in the (\mathbf{W}, \mathbf{Z}) -space, and with the vertical red lines on the feasible set $(\mathbf{W}, g(\mathbf{W}))$. The nested objective function $E_1(\mathbf{W})$ is shown in blue. Corresponding minima for both problems, \mathbf{W}^* and $(\mathbf{W}^*, \mathbf{Z}^*)$, are indicated.

A.3 KKT conditions

We now show that the first-order necessary (Karush-Kuhn-Tucker, KKT) conditions of both problems (nested and MAC-constrained) have the same stationary points. For simplicity and clarity of exposition, we give a proof for the special case of $K = 1$. The proof for $K > 1$ layers follows analogously. We assume the functions \mathbf{f}_1 and \mathbf{f}_2 have continuous first derivatives w.r.t. both its input and its weights. $\mathbf{J}_{\mathbf{f}_2}(\cdot; \mathbf{W}_2)$ indicates the Jacobian of \mathbf{f}_2 w.r.t. its input. To simplify notation, we sometimes omit the dependence on the weights; for example, we write $\mathbf{f}_2(\mathbf{f}_1(\mathbf{x}; \mathbf{W}_1); \mathbf{W}_2)$ as $\mathbf{f}_2(\mathbf{f}_1(\mathbf{x}))$, and $\mathbf{J}_{\mathbf{f}_2}(\cdot; \mathbf{W}_2)$ as $\mathbf{J}_{\mathbf{f}_2}(\cdot)$.

Theorem A.2. *The KKT conditions for the nested problem (1) and the MAC-constrained problem (2) are equivalent.*

Proof. The nested problem for a nested function $\mathbf{f}_2(\mathbf{f}_1(\mathbf{x}))$ is:

$$\min_{\mathbf{W}_1, \mathbf{W}_2} E_1(\mathbf{W}_1, \mathbf{W}_2) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1); \mathbf{W}_2)\|^2.$$

Then we have the stationary point equation (first-order necessary conditions for a minimizer):

$$\frac{\partial E_1}{\partial \mathbf{W}_1} = - \sum_{n=1}^N \frac{\partial \mathbf{f}_1^T}{\partial \mathbf{W}_1}(\mathbf{x}_n) \mathbf{J}_{\mathbf{f}_2}(\mathbf{f}_1(\mathbf{x}_n))^T (\mathbf{y}_n - \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}_n))) = \mathbf{0} \quad (11)$$

$$\frac{\partial E_1}{\partial \mathbf{W}_2} = - \sum_{n=1}^N \frac{\partial \mathbf{f}_2^T}{\partial \mathbf{W}_2}(\mathbf{f}_1(\mathbf{x}_n)) (\mathbf{y}_n - \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}_n))) = \mathbf{0} \quad (12)$$

which is satisfied by all the minima, maxima and saddle points.

The MAC-constrained problem is

$$\min_{\mathbf{W}_1, \mathbf{W}_2, \mathbf{Z}} E(\mathbf{W}_1, \mathbf{W}_2, \mathbf{Z}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}_2(\mathbf{z}_n; \mathbf{W}_2)\|^2 \text{ s.t. } \mathbf{z}_n = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1), n = 1, \dots, N,$$

with Lagrangian

$$\mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{Z}, \boldsymbol{\lambda}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}_2(\mathbf{z}_n; \mathbf{W}_2)\|^2 - \sum_{n=1}^N \boldsymbol{\lambda}_n^T (\mathbf{z}_n - \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1))$$

and KKT conditions

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{W}_1} = \sum_{n=1}^N \frac{\partial \mathbf{f}_1^T}{\partial \mathbf{W}_1}(\mathbf{x}_n) \boldsymbol{\lambda}_n = \mathbf{0} \quad (13)$$

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{W}_2} = - \sum_{n=1}^N \frac{\partial \mathbf{f}_2^T}{\partial \mathbf{W}_2}(\mathbf{f}_1(\mathbf{x}_n)) (\mathbf{y}_n - \mathbf{f}_2(\mathbf{z}_n)) = \mathbf{0} \quad (14)$$

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{z}_n} = -\mathbf{J}_{\mathbf{f}_2}(\mathbf{z}_n)^T (\mathbf{y}_n - \mathbf{f}_2(\mathbf{z}_n)) - \boldsymbol{\lambda}_n = \mathbf{0}, \quad n = 1, \dots, N \quad (15)$$

$$\mathbf{z}_n = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1), \quad n = 1, \dots, N. \quad (16)$$

Substituting $\boldsymbol{\lambda}_n$ from eq. (15) and \mathbf{z}_n from eq. (16):

$$\boldsymbol{\lambda}_n = -\mathbf{J}_{\mathbf{f}_2}(\mathbf{z}_n)^T (\mathbf{y}_n - \mathbf{f}_2(\mathbf{z}_n)), \quad n = 1, \dots, N \quad (15')$$

$$\mathbf{z}_n = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1), \quad n = 1, \dots, N \quad (16')$$

into eqs. (13)–(14) we recover eqs. (11)–(12), thus a KKT point of the constrained problem is a stationary point of the nested problem. Conversely, given a stationary point $(\mathbf{W}_1, \mathbf{W}_2)$ of the nested problem, and defining $\boldsymbol{\lambda}_n$ and \mathbf{z}_n as in eqs. (15')–(16'), then $(\mathbf{W}_1, \mathbf{W}_2, \mathbf{Z}, \boldsymbol{\lambda})$ satisfies eqs. (13)–(16) and so is a KKT point of the constrained problem. Hence, there is a one-to-one correspondence between the stationary points of the nested problem and the KKT points of the MAC-constrained problem. \square

From theorem A.1 and A.2, it follows that the minimizers, maximizers and saddle points of the nested problem are in one-to-one correspondence with the respective minimizers, maximizers and saddle points of the MAC-constrained problem.

B Convergence of the quadratic-penalty method for MAC

Let us first give convergence conditions for the general equality-constrained minimization problem:

$$\min f(\mathbf{x}) \text{ s.t. } c_i(\mathbf{x}) = 0, \quad i = 1, \dots, m \quad (17)$$

and the quadratic-penalty (QP) function

$$Q(\mathbf{x}; \mu) = f(\mathbf{x}) + \frac{\mu}{2} \sum_{i=1}^m c_i^2(\mathbf{x}) \quad (18)$$

with penalty parameter $\mu > 0$. Given a positive increasing sequence $(\mu_k) \rightarrow \infty$, a nonnegative sequence $(\tau_k) \rightarrow 0$, and a starting point \mathbf{x}_0 , the QP method finds an approximate minimizer \mathbf{x}_k of $Q(\mathbf{x}; \mu_k)$ for $k = 1, 2, \dots$, so that the iterate \mathbf{x}_k satisfies $\|\nabla_{\mathbf{x}} Q(\mathbf{x}_k; \mu_k)\| \leq \tau_k$. Given this algorithm, we have the following theorems:

Theorem B.1 (Nocedal and Wright, 2006, Th. 17.1). *Suppose that $(\mu_k) \rightarrow \infty$ and $(\tau_k) \rightarrow 0$. If each \mathbf{x}_k is the exact global minimizer of $Q(\mathbf{x}; \mu_k)$, then every limit point \mathbf{x}^* of the sequence (\mathbf{x}_k) is a global solution of the problem (17).*

Theorem B.2 (Nocedal and Wright, 2006, Th. 17.2). *Suppose that $(\mu_k) \rightarrow \infty$ and $(\tau_k) \rightarrow 0$, and that \mathbf{x}^* is a limit point of (\mathbf{x}_k) . Then \mathbf{x}^* is a stationary point of the function $\sum_{i=1}^m c_i^2(\mathbf{x})$. Besides, if the constraint gradients $\nabla c_i(\mathbf{x}^*)$, $i = 1, \dots, m$ are linearly independent, then \mathbf{x}^* is a KKT point for the problem (17). For such points, we have for any infinite subsequence \mathcal{K} such that $\lim_{k \in \mathcal{K}} \mathbf{x}_k = \mathbf{x}^*$ that $\lim_{k \in \mathcal{K}} -\mu_k c_i(\mathbf{x}_k) = \lambda_i^*$, $i = 1, \dots, m$, where $\boldsymbol{\lambda}^*$ is the multiplier vector that satisfies the KKT conditions for the problem (17).*

If now we particularize these general theorems to our case, we can obtain stronger theorems. Theorem B.1 is generally not applicable, because optimization problems involving nested functions are typically not convex and have local minima. Theorem B.2 is applicable to prove convergence in the nonconvex case. We assume the functions $\mathbf{f}_1, \dots, \mathbf{f}_{K+1}$ in eq. (1) have continuous first derivatives w.r.t. both its input and its weights, so $E(\mathbf{W}, \mathbf{Z})$ is differentiable w.r.t. \mathbf{W} and \mathbf{Z} .

Theorem B.3 (Convergence of MAC/QP for nested problems). *Consider the constrained problem (2) and its quadratic-penalty function $E_Q(\mathbf{W}, \mathbf{Z}; \mu)$ of (3). Given a positive increasing sequence $(\mu_k) \rightarrow \infty$, a nonnegative sequence $(\tau_k) \rightarrow 0$, and a starting point $(\mathbf{W}^0, \mathbf{Z}^0)$, suppose the QP method finds an approximate minimizer $(\mathbf{W}^k, \mathbf{Z}^k)$ of $E_Q(\mathbf{W}^k, \mathbf{Z}^k; \mu_k)$ that satisfies $\|\nabla_{\mathbf{W}, \mathbf{Z}} E_Q(\mathbf{W}^k, \mathbf{Z}^k; \mu_k)\| \leq \tau_k$ for $k = 1, 2, \dots$. Then, $\lim_{k \rightarrow \infty} (\mathbf{W}^k, \mathbf{Z}^k) = (\mathbf{W}^*, \mathbf{Z}^*)$, which is a KKT point for the problem (2), and its Lagrange multiplier vector has elements $\lambda_n^* = \lim_{k \rightarrow \infty} -\mu_k (\mathbf{Z}_n^k - \mathbf{F}(\mathbf{Z}_n^k, \mathbf{W}^k; \mathbf{x}_n))$, $n = 1, \dots, N$.*

Proof. It follows by applying theorem B.2 to the constrained problem (2) and by noting that $\lim_{k \rightarrow \infty} (\mathbf{W}^k, \mathbf{Z}^k) = (\mathbf{W}^*, \mathbf{Z}^*)$ exists and that the constraint gradients are linearly independent. We prove these two statements in turn.

The limit of the sequence $((\mathbf{W}^k, \mathbf{Z}^k))$ exists because the objective function $E(\mathbf{W}, \mathbf{Z})$ of the MAC-constrained problem (hence the QP function $E_Q(\mathbf{W}, \mathbf{Z}; \mu)$) are lower bounded and have continuous derivatives.

The constraint gradients are l.i. at any point (\mathbf{W}, \mathbf{Z}) and thus, in particular, at the limit $(\mathbf{W}^*, \mathbf{Z}^*)$. To see this, let us first compute the constraint gradients. There is one constraint $C_{nkh}(\mathbf{W}, \mathbf{Z}) = z_{nkh} - f_{kh}(\mathbf{z}_{n,k-1}; \mathbf{W}_k) = 0$ for each point $n = 1, \dots, N$, layer $k = 1, \dots, K$ and unit $h \in \mathcal{I}(k)$, where we define $\mathcal{I}(k)$ as the set of auxiliary coordinate indices for layer k and $\mathbf{z}_{n0} = \mathbf{x}_n$, $n = 1, \dots, N$. The gradient of this constraint is:

$$\begin{aligned}\frac{\partial C_{nkh}}{\partial \mathbf{W}_{k'}} &= -\delta_{kk'} \frac{\partial f_{kh}}{\partial \mathbf{W}_k}, \quad k = 1, \dots, K \\ \frac{\partial C_{nkh}}{\partial z_{n'k'h'}} &= \delta_{nn'} \left(\delta_{kk'} \delta_{hh'} - \delta_{k-1,k'} \frac{\partial f_{kh}}{\partial z_{n,k-1,h}} \right), \quad n = 1, \dots, N, \quad k = 1, \dots, K, \quad h \in \mathcal{I}(k).\end{aligned}$$

Now, we will show that these gradients are l.i. at any point (\mathbf{W}, \mathbf{Z}) . It suffices to look at the gradients w.r.t. \mathbf{Z} . Call $\alpha_{nkh} = \partial f_{kh} / \partial z_{n,k-1,h}$ for short. Constructing a linear combination of them and setting it to zero:

$$\sum_{n=1}^N \sum_{k=1}^K \sum_{h \in \mathcal{I}(k)} \lambda_{nkh} \frac{\partial C_{nkh}}{\partial \mathbf{Z}'} = \mathbf{0}.$$

This implies, for the gradient element corresponding to $z_{n'k'h'}$:

$$\begin{aligned}\sum_{n=1}^N \sum_{k=1}^K \sum_{h \in \mathcal{I}(k)} \lambda_{nkh} \delta_{nn'} (\delta_{kk'} \delta_{hh'} - \delta_{k-1,k'} \alpha_{nkh}) &= \lambda_{n'k'h'} - \sum_{h \in \mathcal{I}(k'+1)} \lambda_{n',k'+1,h} \alpha_{n',k'+1,h} = 0 \\ \implies \lambda_{n'k'h'} &= \sum_{h \in \mathcal{I}(k'+1)} \lambda_{n',k'+1,h} \alpha_{n',k'+1,h}.\end{aligned}$$

Applying this for $k' = K, \dots, 1$:

- For $k' = K$: $\lambda_{n'Kh'} = 0$, $n' = 1, \dots, N$, $h' \in \mathcal{I}(K)$.
- For $k' = K-1$: $\lambda_{n',K-1,h'} = \sum_{h \in \mathcal{I}(K)} \lambda_{n',K,h} \alpha_{n',K,h} = 0$, $n' = 1, \dots, N$, $h' \in \mathcal{I}(K-1)$.
- ...
- For $k' = 1$: $\lambda_{n',1,h'} = \sum_{h \in \mathcal{I}(2)} \lambda_{n',2,h} \alpha_{n',2,h} = 0$, $n' = 1, \dots, N$, $h' \in \mathcal{I}(1)$.

Hence, all the coefficients λ_{nkh} are zero and the gradients are l.i. \square

In practice, as with any continuous optimization problem, convergence may occur in pathological cases to a stationary point of the constrained problem rather than a minimizer.

In summary, MAC/QP defines a continuous path $(\mathbf{W}^*(\mu), \mathbf{Z}^*(\mu))$ which, under some mild assumptions (essentially, that we minimize $E_Q(\mathbf{W}, \mathbf{Z}; \mu)$ increasingly accurately as $\mu \rightarrow \infty$), converges to a stationary point (typically a minimizer) of the constrained problem (2), and thus to a minimizer of the nested problem (1).

We also have the following result (for simplicity, we give it for $K = 1$ layer).

Theorem B.4. *If a stationary point of the QP function for the problem of theorem A.2 satisfies $\mathbf{z}_n = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1)$, then it is also a stationary point of the nested problem for all $\mu \geq 0$.*

Proof. The QP function is:

$$E_Q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{Z}; \mu) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}_2(\mathbf{z}_n; \mathbf{W}_2)\|^2 + \frac{\mu}{2} \sum_{n=1}^N \|\mathbf{z}_n - \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1)\|^2.$$

A stationary point of E_Q must satisfy the equations:

$$\frac{\partial E_Q}{\partial \mathbf{W}_1} = -\mu \sum_{n=1}^N \frac{\partial \mathbf{f}_1^T}{\partial \mathbf{W}_1}(\mathbf{x}_n) (\mathbf{z}_n - \mathbf{f}_1(\mathbf{x}_n)) = \mathbf{0} \quad (19)$$

$$\frac{\partial E_Q}{\partial \mathbf{W}_2} = - \sum_{n=1}^N \frac{\partial \mathbf{f}_2^T}{\partial \mathbf{W}_2}(\mathbf{z}_n) (\mathbf{y}_n - \mathbf{f}_2(\mathbf{z}_n)) = \mathbf{0} \quad (20)$$

$$\frac{\partial E_Q}{\partial \mathbf{z}_n} = \mathbf{J}_{\mathbf{f}_2}(\mathbf{z}_n)^T (\mathbf{y}_n - \mathbf{f}_2(\mathbf{z}_n)) + \mu(\mathbf{z}_n - \mathbf{f}_1(\mathbf{x}_n)) = \mathbf{0}, \quad n = 1, \dots, N. \quad (21)$$

If $\mathbf{z}_n = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1)$ for $n = 1, \dots, N$, then from eq. (21) $\mathbf{J}_{\mathbf{f}_2}(\mathbf{z}_n)^T (\mathbf{y}_n - \mathbf{f}_2(\mathbf{z}_n)) = \mathbf{0}$ for $n = 1, \dots, N$ and $(\mathbf{W}_1, \mathbf{W}_2)$ satisfies eqs. (11)–(12), so it is a stationary point of the nested problem for all $\mu \geq 0$. \square

Remarks:

- Since the QP minimizer approaches the constraints from their infeasible side, the assumption $\mathbf{z}_n = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1)$ for $n = 1, \dots, N$ does not hold unless $E_1(\mathbf{W}_1, \mathbf{W}_2) = 0$ there.
- The converse of theorem B.4 is not generally true: if $(\mathbf{W}_1, \mathbf{W}_2)$ is a stationary point of the nested problem, then defining $\mathbf{z}_n = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1)$ for $n = 1, \dots, N$ we have that eqs. (19)–(20) hold but eq. (21) does not.
- Theorem B.4 does not imply that the function $E_Q(\mathbf{W}, \mathbf{Z}; \mu)$ (for $\mu > \bar{\mu}$) is an exact penalty function for the objective $E(\mathbf{W}, \mathbf{Z})$, for this we need the opposite: that any local solution of the MAC-constrained problem is a local minimizer of $E_Q(\mathbf{W}, \mathbf{Z}; \mu)$.

References

- Y. Bengio and Y. LeCun. Scaling learning algorithms toward AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*, Neural Information Processing Series, pages 321–360. MIT Press, 2007.
- Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 153–160. MIT Press, Cambridge, MA, 2007.
- C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Series in Information Science and Statistics. Springer-Verlag, Berlin, 2006.

- S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.
- M. Á. Carreira-Perpiñán. The elastic embedding algorithm for dimensionality reduction. In J. Fürnkranz and T. Joachims, editors, *Proc. of the 27th Int. Conf. Machine Learning (ICML 2010)*, pages 167–174, Haifa, Israel, June 21–25 2010.
- M. Á. Carreira-Perpiñán and Z. Lu. Dimensionality reduction by unsupervised regression. In *Proc. of the 2008 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR’08)*, Anchorage, AK, June 23–28 2008.
- M. Á. Carreira-Perpiñán and Z. Lu. Parametric dimensionality reduction by unsupervised regression. In *Proc. of the 2010 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR’10)*, pages 1895–1902, San Francisco, CA, June 13–18 2010.
- M. Á. Carreira-Perpiñán and Z. Lu. Manifold learning and missing data recovery through unsupervised regression. In *Proc. of the 12th IEEE Int. Conf. Data Mining (ICDM 2011)*, pages 1014–1019, Vancouver, BC, Dec. 11–14 2011.
- E. Castillo, B. Guijarro-Berdiñas, O. Fontenla-Romero, and A. Alonso-Betanzos. A very fast learning method for neural networks based on sensitivity analysis. *Journal of Machine Learning Research*, 7:1159–1182, July 2006.
- P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer Series in Optimization and Its Applications, pages 185–212. Springer-Verlag, 2011.
- J. J. Craig. *Introduction to Robotics. Mechanics and Control*. Prentice-Hall, third edition, 2004.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.
- D. Erdogmus, O. Fontenla-Romero, J. C. Principe, A. Alonso-Betanzos, and E. Castillo. Linear-least-squares initialization of multilayer perceptrons through backpropagation of the desired response. *IEEE Trans. Neural Networks*, 16(2):325–337, Mar. 2005.
- D. Erhan, P. A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Proc. of the 12th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2009)*, pages 153–160, Clearwater Beach, FL, Mar. 21–24 2009.
- A. Gifi. *Nonlinear Multivariate Analysis*. John Wiley & Sons, 1990.
- B. Gold and N. Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. John Wiley & Sons, New York, London, Sydney, 2000.
- T. Grossman, R. Meir, and E. Domany. Learning by choice of internal representations. *Complex Systems*, 2(5):555–575, 1988.
- T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning—Data Mining, Inference and Prediction*. Springer Series in Statistics. Springer-Verlag, second edition, 2009.
- G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 28 2006.
- J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

- K. Kavukcuoglu, M. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical Report CBLL-TR-2008-12-01, Dept. of Computer Science, New York University, Dec. 4 2008.
- A. Krogh, C. J. Thorbergsson, and J. A. Hertz. A cost function for internal representations. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems (NIPS)*, volume 2, pages 733–740. Morgan Kaufmann, San Mateo, CA, 1990.
- N. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, Nov. 2005.
- M. LeBlanc and R. Tibshirani. Adaptive principal surfaces. *J. Amer. Stat. Assoc.*, 89(425):53–64, Mar. 1994.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, Nov. 1998.
- K.-C. Li. Sliced inverse regression for dimension reduction. *J. Amer. Stat. Assoc.*, 86(414):316–327 (with comments, pp. 328–342), June 1991.
- S. Ma, C. Ji, and J. Farmer. An efficient EM-based training algorithm for feedforward neural networks. *Neural Networks*, 10(2):243–256, Mar. 1997.
- P. Meinicke, S. Klanke, R. Memisevic, and H. Ritter. Principal surfaces from unsupervised kernel regression. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(9):1379–1391, Sept. 2005.
- S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (COIL-20). Technical Report CUCS-005-96, Dept. of Computer Science, Columbia University, Feb. 1996.
- J. Ngiam, Z. Chen, P. W. Koh, and A. Ng. Learning deep energy models. In L. Getoor and T. Scheffer, editors, *Proc. of the 28th Int. Conf. Machine Learning (ICML 2011)*, pages 1105–1112, Bellevue, WA, June 28 – July 2 2011.
- J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, second edition, 2006.
- B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 13 1996.
- B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Res.*, 37(23):3311–3325, Dec. 1997.
- G. B. Orr and K.-R. Müller, editors. *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
- M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. of the 2007 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'07)*, pages 1–8, Minneapolis, MN, June 18–23 2007a.
- M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 1137–1144. MIT Press, Cambridge, MA, 2007b.
- M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nat. Neurosci.*, 2(11):1019–1025, Nov. 1999.
- R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control and Optim.*, 14(5):877–898, 1976.
- T. Rögnvaldsson. On Langevin updating in multilayer perceptrons. *Neural Computation*, 6(5):916–926, Sept. 1994.

- R. Rohwer. The ‘moving targets’ training algorithm. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems (NIPS)*, volume 2, pages 558–565. Morgan Kaufmann, San Mateo, CA, 1990.
- S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, Dec. 22 2000.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986a.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Computing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986b.
- D. Saad and E. Marom. Learning by choice of internal representations: An energy minimization approach. *Complex Systems*, 4(1):107–118, 1990.
- G. Saon and J.-T. Chien. Large-vocabulary continuous speech recognition systems: A look at some recent advances. *IEEE Signal Processing Magazine*, 29(6):18–33, Nov. 2012.
- B. Schölkopf and A. J. Smola. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, MA, 2001.
- B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, July 1998.
- T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(3):411–426, Mar. 2007.
- A. J. Smola, S. Mika, B. Schölkopf, and R. C. Williamson. Regularized principal manifolds. *Journal of Machine Learning Research*, 1:179–209, June 2001.
- S. Tan and M. L. Mavrovouniotis. Reducing data dimensionality through optimizing neural network inputs. *AIChE Journal*, 41(6):1471–1479, June 1995.
- J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, Dec. 22 2000.
- W. Wang and M. Á. Carreira-Perpiñán. Nonlinear low-dimensional regression using auxiliary coordinates. In N. Lawrence and M. Girolami, editors, *Proc. of the 15th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2012)*, pages 1295–1304, La Palma, Canary Islands, Spain, Apr. 21–23 2012.
- P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, University of Harvard, 1974.