1 Introduction

¡++¿ Deep neural networks have repeatedly surpassed state of the art benchmarks in various use cases, and they have been deployed in many commercial and industry applications. The surge in demand for deep learning has motivated the development of neural network libraries such as tensorflow and pytorch as well as greater abstractions such as keras, which allow more black box model training and development. The associated demand for data scientists and machine learning engineers have led to "transplants" from other fields migrating into machine learning, and much of the knowledge of deep learning has been distilled into the composition of functional, accessible components.

In contrast, the question of what deep learning models are actually doing is unclear and perhaps ill-posed. We seek to present partial answers presented from the fields of statistics, probabilistic graphical models, optimization, and operator theory. Many of these connections are well-known in their respective fields or at least are heavily implicit in their formulations. We are especially interested in the limitations of these interpretations and their intersections, especially where a secondary interpretation offers insight beyond the scope of a more restrictive interpretation.

We begin by describing the exponential family and special properties of the regression of residuals for learning exponential family models, thereby fully describing the output layer of most machine learning models. Second, we describe connect a few very simple probabilistic graphical models with well-known machine learning models, providing an interpretation of pairwise weights for intermediate deep learning layers. Third, we explore the optimization perspective of machine learning, starting with the probabilistic branch of optimization and then generalizing past by exploring other operatorswhich provide an interpretation of the rectified linear unit. Finally, we transition into an operator perspective and the deep equilibrium model perpsective of deep learning and tie it into the particle perspective of our layers acting as operators, transforming our spaces.

2 Exponential Families

¡++¿ Exponential families are deeply intertwined into the concept of entropy and energy and can act as a probabilistic building block for density estimation. In fact, both regression and classification are special cases of target distributions defined by a specific exponential family. Moreover, exponential families provide a more specific definition of residuals, and gradient descent of exponential family losses can be shown to directly update the natural parameters. The form for exponential family of distribution of distributions is derived by the Pitman-Koopman-Dermois Theorem. The proof by Koopman built off Fisher's ideas about sufficient statistics to express a family of distributions which can be defined by the sufficient statistic, assuming certain other restrictions.

$$\begin{aligned} P\left(x|\eta\right) &= g(\eta)h(x)\exp\left\{\langle u(x),\eta\rangle\right\} < ++ > \\ P\left(x|\eta\right) &= h(x)\exp\left\{\langle u(x),\eta\rangle - G(\eta)\right\} < ++ > \end{aligned} \tag{1}$$

¡++¿

where $u(x)$ is the sufficient statistic (which must be additive) for a single element, η is the natural parameter for that sufficient statistic, $h(x)$ is an innate measure of X which is completely independent of the choice of the natural parameter, $g(\eta)$ is the normalization constant for the Pitman-Koopman-Dermois form, and $G(\eta)$ is an alternative form form for the normalization constant which is more consistent with thelog factor potential interpretation.

We perform an equivalent derivation for both cases, with $\frac{1}{g(\eta)} = \exp\{G(\eta)\}$:

$$\begin{aligned}
\int P(x|\eta)\,dx &= \int g(\eta)h(x)\exp\{\langle u(x),\eta\rangle\}\,dx \\
1 &= g(\eta)\int h(x)\exp\{\langle u(x),\eta\rangle\}\,dx \\
\frac{1}{g(\eta)} &= \int h(x)\exp\{\langle u(x),\eta\rangle\}\,dx \\
\int P(x|\eta)\,dx &= \int h(x)\exp\{\langle u(x),\eta\rangle - G(\eta)\}\,dx \\
1 &= \int h(x)\exp\{\langle u(x),\eta\rangle - G(\eta)\}\,dx \\
\exp\{G(\eta)\} &= \int h(x)\exp\{\langle u(x),\eta\rangle\}\,dx
\end{aligned} \tag{2}$$

¡++¿

One very important property relates the natural parameter to the expected sufficient statistic:

$$\begin{aligned}
\exp\{G(\eta)\} &= \int h(x)\exp\{\langle u(x),\eta\rangle\}\,dx \\
\nabla_\eta \exp\{G(\eta)\} &= \frac{\int u(x)h(x)\exp\{\langle u(x),\eta\rangle\}\,dx}{\int h(x)\exp\{\langle u(x),\eta\rangle\}\,dx} \\
&= \frac{\int u(x)h(x)\exp\{\langle u(x),\eta\rangle\}\,dx}{\exp\{G(\eta)\}} < ++ > \\
\nabla_\eta \exp\{G(\eta)\} &= E_{x\sim P(x|\eta)}[u(x)] < ++ > \\
\nabla_\eta [-\log g(\eta)] &= E_{x\sim P(x|\eta)}[u(x)] < ++ >
\end{aligned} \tag{3}$$

¡++¿

It can be proven that this is a one-to-one mapping which in one direction is known as the canonical-link function, and it has many useful properties. One specific case is when we generate samples from an unknown distribution, in which case the η that maximizes the log probability of seeing those samples would be the η that has the same sufficient statistic (assuming that is possible):

$$\begin{aligned}
\nabla_\eta \frac{1}{N}\sum_{n=1}^{N}\log P(x_n|\eta) &= \frac{1}{N}\sum_{n=1}^{N}\left(u(x_n) - \nabla_\eta G(\eta)\right) < ++ > \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(u(x_n) - E_{x\sim P(x|\eta)}[u(x)]\right) < ++ >
\end{aligned} \tag{4}$$

$$
\begin{aligned}
0 &= \frac{1}{N}\sum_{n=1}^{N}\left(u(x_n) - E_{x\sim P(x|\eta)}\left[u(x)\right]\right) \\
\sum_{n=1}^{N}\frac{u(x_n)}{N} &= E_{x\sim P(x|\eta)}\left[u(x)\right]
\end{aligned}
\tag{5}
$$

¡++¿

The first form shows that gradient descent on the log probabilities, which is the form of both regression and classification loss functions, with respect to the natural parameters is exactly the same as the difference between the sufficient statistic and the expected sufficient statistic. Hence, the concept of a residual for square regression loss is generalized to residuals of sufficient statistics, and the residual can be used as a gradient update to the natural parameters. This is quite remarkable, and when we consider that the samples are taken from some unknown distribution $p_{data}(x)$, we see that maximizing the negative log likelihood is the same as minimizing the cross-entropy, which is the same as minimizing the KL-divergence:

$$
\begin{aligned}
KL\left(p_{data}(x)\|p(x|\eta)\right) &= H\left(p_{data}(x)\right) - \int p_{data}(x)\log p(x|\eta)dx \\
\min_{\eta} KL\left(p_{data}(x)\|p(x|\eta)\right) &= \min_{\eta} -\int p_{data}(x)\log p(x|\eta)dx \\
\nabla_{\eta} KL\left(p_{data}(x)\|p(x|\eta)\right) &= \nabla_{\eta} E_{x\sim p_{data}(x)}\left[-\log p(x|\eta)\right] \\
&= E_{x\sim p_{data}(x)}\left[u(x_n)\right] - E_{x\sim p(x|\eta)}\left[u(x)\right]
\end{aligned}
\tag{6}
$$

¡++¿

This is called an M-projection, and it intertwines exponential families with the concept of entropy. In fact, Boltzmann's derivation of entropy is a special case with a multinoulli distribution. Intuitively, this connection is sensible, since entropy acts upon log probabilities, and the form of an exponential family distribution is in exponential space.

Finally, in the case of conditional models, or discriminative models, we may have some set of non-linear feature mappings for the input, and we may try to model the natural parameters as a linear combination of the features. In the case of a gaussian conditional distribution, this is known as regression, and for classification (multinomial) logistic regression.

$$
\begin{aligned}
\eta &= \sum_{j} w_j \phi_j(x) \\
\nabla_w \log p(x|w) &= \nabla_w \eta \nabla_\eta \log p(x|\eta) \\
&= \begin{pmatrix}\phi_1(x) & \cdots & \phi_M(x)\end{pmatrix} \nabla_\eta \log p(x|\eta)
\end{aligned}
\tag{7}
$$

¡++¿

This is also known as a generalized linear model (GLM), with the canonical link functions defining the nonlinearities. Hence, for these regression models, gradient updates for the weights can be done almost directly through the residuals.

There are many more remarkable properties of exponential families, and due to those, it is preferable to try to model distributions as exponential families. In fact, many derivations for regression and classification can be generalized to any exponential family.

3 Probabilistic Graphical Models

¡++¿

The properties of exponential families and their connection to KL-divergence and entropy motivates the parametrization of probabilistic graphical models (PGMs), specifically the undirected case, Markov networks.

For the case of positive PGMs, that is PGMs for which the probability density/probability mass is never 0, they can be written in the form of a Gibbs distribution:

$$P_\Phi(\mathcal{X}) \propto \prod_i \phi_i(C_i) \tag{8}$$

¡++¿

where $\mathcal{X} = \{X_1, \ldots, X_N\}$, i enumerates the set of cliques of the undirected graph, ϕ_i is a real-valued function called the factor potential, and $C_i = Scope[\phi_i]$.

This result is known as the Hammersley-Clifford theorem, and for positive distributions it connects the concepts of global conditional independence properties with pairwise conditional independence and local, Markov blanket conditional independence.

In the case of positive distributions, the Gibbs distribution can be written in terms of log factor potentials, and one parametrization called log-linear models involves linear weights along with pre-specified log factor potentials:

$$P_\Phi(\mathcal{X}) \propto \prod_i e^{-w_i E_i(D_i)} \tag{9}$$

¡++¿

where $-\log \phi_i = w_i E_i(D_i)$, and E_i are called energy functions, and these unnormalized parametrizations are called energy-based models.

To discern what these distributions are and what energy is, we note that log-linear models are in fact a type of exponential family distribution, with the energy function for each scope being sufficient statistics, and the weights being the natural parameters. This means that once we define the energy functions, the interpretation of the energy functions is the distribution that minimizes KL divergence (maximizes entropy) while matching expected energies (expected sufficient statistics).

In the case of a single clique potential/energy function for the entire model, the associated exponential family model is known as the Boltzmann distribution. In this specific case, we see that the expected energy is the scaled entropy of the model, up to a term involving the normalization constant:

$$
\begin{aligned}
P_{E,\eta}(\mathcal{X}) &\propto e^{-E(\mathcal{X})\eta} \\
H\left(P_{E,\eta}(\mathcal{X})\right) &= -\eta \int E(\mathcal{X}) P_{E,\eta}(\mathcal{X}) + G(\eta) \\
&= -\eta E_{x\sim P_{E,\eta}(\mathcal{X})}\left[E(\mathcal{X})\right] + G(\eta) \\
\frac{1}{\eta} H\left(P_{E,\eta}(\mathcal{X})\right) &= E_{x\sim P_{E,\eta}(\mathcal{X})}\left[E(\mathcal{X})\right] + \frac{G(\eta)}{\eta}
\end{aligned}
\tag{10}
$$

¡++¿

In the case of thermodynamics, they use $\frac{1}{T}$ or $\frac{1}{kT}$ instead of η, and they call it the temperature. If we take the expectation of the log of the unnormalized probability distribution, $\tilde{P}_{E,\eta}(\mathcal{X}) = e^{-E(\mathcal{X})\eta}$, we can ignore the normalization factor, and we get the form of Helmholtz free energy, which motivates the energy functional that is optimized in in variational inference and other PGM inference and learning algorithms.

3.1 Machine learning PGMs

¡++¿

It turns out even very simple Markov network parametrizations can be very good approximations to useful, real world cases. The simplest log factor potentials would involve polynomials of the scope variables. Somewhat independently, it is preferable to have clique potentials with a small number of scope variables due to the exponential complexity of marginalization, with respect to the number of variables. For that reason, let us mostly consider the case of factor potentials involving two variables, an edge, or a single variable, a vertex:

$$
\begin{aligned}
P_{\Phi}(\mathcal{X}) \propto \exp\left\{ -\sum_{i\sim j} w_{i,j} \left\langle X_i, X_j \right\rangle - \sum_i u_i X_i \right\} \\
= e^{-\mathcal{X}^\intercal W \mathcal{X} - U^\intercal \mathcal{X}}
\end{aligned}
\tag{11}
$$

¡++¿

where in the equation we have replaced the quadratic coefficients with a single weight matrix, and the linear coefficients with a single vector.

This is known as the Ising model, in the specific case where all the variables are discrete variables of values $+1$ or -1, and it is the model for electron spins.

In the more general case where variables may take on values of $\{0, 1\}$ or perhaps continuous values, this is called the Boltzmann machine.

When the PGM can be divided into a bipartite graph, this is called a Restricted Boltzmann Machine, and when both sets of variables have the same size, the weight matrix is an $N \times N$ matrix.

¡++¿ Placeholder, I can't justify that it will be equivalent to a Markov network.

For a Boltzmann machine, let's assume the variable $Y_i \in \{0, 1\}$ is a bernoulli variable, X_j are other nodes in $\mathcal{X}$, and Nb_{Y_i} is the Markov blanket, that is the undirected neighbors of Y_i. If we isolate the terms in the Boltzmann machine to the ones involving Y_i, we end up with

$$
\begin{aligned}
P_\Phi(Y_i, Y_i^-) &\propto \exp\left\{ \sum_{j \in Nb_{Y_i}} w_{i,j} \langle Y_i, X_j \rangle + u_i Y_i \right\} \\
&= \exp\left\{ Y_i \sum_{j \in Nb_{Y_i}} (w_{i,j} X_j + u_i \ ) \right\}
\end{aligned}
\tag{12}
$$

¡++¿

The exponential family form of a Bernoulli variable is

$$
P(Y|\eta) = e^{Y\eta - G(\eta)} \tag{13}
$$

¡++¿

so the functional form of our Boltzmann machine motivates us to treat $\eta = w_{i,j} X_j + u_i$ and assume that is the conditional distribution for $Y_i | Nb_{Y_i}$:

$$
\begin{aligned}
\eta &= w_{i,j} X_j + u_i \\
P(Y|\eta) &= E_{Y \sim P(Y|\eta)}[Y] = \sigma(Y)
\end{aligned}
\tag{14}
$$

¡++¿

Notably, linear combination of other input is functional form of a feed-forward neural network. A caveat is that we have ignored all factors involving X_j, so technically we would have to pass messages through X_j, which would involve passing messages from the rest of the network.

Comparatively, a feed-forward neural network is a Bayesian discriminative model, with an unspecified distribution for the input distribution. We have to make a stronger assumption that the forward weights define a conditional probability as above. The more formal definition we define a conditional random field from a previous layer to the next layer using a Boltzmann machine, with the first layer being the input nodes.

$$
\begin{aligned}
H^0 &= P_{data}(X) \\
P(H^{l+1}|H^l) &= \frac{1}{Z(H^l} RBM(H^l, H^{l+1}) \\
H^L &= Y
\end{aligned}
\tag{15}
$$

¡++¿

Since a conditional random field factors out any potentials involving the input,we do not have to worry about the previous layer's messages. We have had to make an additional assumption that we can define the conditional probability of the current layer, H^l, given the previous layer, H^{l-1}, using a CRF, and this means that we have to assume that the edge potentials connecting to the following layer, H^{l+1} do not contribute to the $P(H^l|H^{l-1}$.

One thing to note is that we are using undirected model as a network fragment ofa directed model. The directed version of this network fragment is called a conditional Bayesian network, and for the bernoulli variable case, we can define what is called a logistic conditional probability distribution (logistic CPD) almost exactly as above ¡++¿.

In fact, this can be generalized to any exponential family, through a generalized linear model with the parent variables, and we can change the target distribution family by changing the nonlinearity and the range of the target variable. There are a few specific assumptions about the structure we are using here worth explicitly stating: 1) The output variables for a layer are conditionally independent of each other, given the input variables, 2) the CPDs have a property called independence of causal influence, which theoretically speaking means that the information from the parents is combined in an associative (and in our case commutative as well) way, and practical speaking it means we can drop out input.

These properties are very strong, though perhaps intuitively desirable assumptions, and they somewhat fit the model for how neurons communicate with each other. Specifically, we can see the strengths of these connections through the complexity of the model - we can parametrize the conditional bayesian network between two layers by $(N^l + 1) \times N^{l+1}$ weights, whereas for unrestricted Bernoulli distributions, we would need to define a separate conditional probability for every conditional outcome, for a complexity of $\left(2^{N^l} - 1\right)^{2^{N^{l+1}}} = \mathcal{O}\left(2^{N^l N^{l+1}}\right)$ Intuitively, when we combine information about the conditioning variable outcomes in this additive way, the symmetries of addition are compressing information and we are reducing the complexity.

Since neural networks with sigmoid non-linearities can be written in this form and transformers are using multinomial logistic CPDs to define an attention variable, we finally have an interpretation of what the layers of a neural network are.

A specific case of the conditional Bayesian network is a conditional linear Gaussian, where we define the conditional probilities through linear Gaussians:

$$\begin{aligned} P(X^0) &= \mathcal{N}\left(X^0 | \mu^0, \Sigma^0\right) \\ P(X^{l+1}|X^l) &= \mathcal{N}\left(X^{l+1} | W_{l,l+1} X^l + B^{l+1}, \Sigma^{l+1}\right) \end{aligned} \tag{16}$$

¡++¿

Usually we assume $\Sigma^{l+1} = \left(\sigma^{l+1}\right)^2 \mathbf{I}$. The joint probability distribution from a linear Gaussian network can be solved analytically through a forward pass of message passing, and this is the same dynamic programming procedure which is used to solve for the optimal solution of model predictive control problems.

So far we have defined generalized linear models using affine functions of the parent variables. This has allowed us to define models by pairwise edge weights, specifically with log linear factors. What would happen if we relax this condition somewhat, so we are using linear combinations of various non-linear functions? It turns out that Gaussian joint probability distributions can always be described by pairwise edge-weights and variable potentials:

$$\begin{aligned} P(\mathcal{X}) &= \mathcal{N}\left(\mathcal{X} | \mu | \Sigma\right) \\ &= \exp\left\{\left\langle \left(\mathcal{X}, \mathcal{X}\mathcal{X}^\intercal\right), \left(\Sigma^{-1}\mu - \frac{1}{2}\Sigma^{-1}\right)\right\rangle\right\} \\ &= \exp\left\{-\frac{1}{2}\mathcal{X}^\intercal \Lambda \mathcal{X} + \mathcal{X}^\intercal \Lambda \mu\right\} \end{aligned} \tag{17}$$

¡++¿

This means that we should be able to do a parametrization of the nodes and edgesto define the PGM, even if we don't assume a linear-gaussian bayesian factorization. We can also define a conditional probability distribution this way, so given N input variables, $X_1, \ldots, X_N$, and N output variables, $Y_1, \ldots, Y_N$, we can define a conditional random field through a mean and covariance. This is the PGM for a Gaussian Process:

$$P(Y|X) = GP\left(\mu(x), K(x, x')\right) \tag{18}$$

¡++¿

The benefit is that since we know the conditional joint probability distribution, we can condition on observing subsets of Y:

$$P(Y_{unobs}|X, Y_{obs}) = \mathcal{N}\left(\mu_{unobs} + K_{unobs,obs}\Sigma_{obs,obs}^{-1}\left(Y_{obs} - \mu_{obs}\right), \Sigma_{unobs,unobs} - \Sigma_{unobs,obs}\Sigma_{obs,obs}^{-1}\Sigma_{obs,unobs}\right) \tag{19}$$

¡++¿

A caveat is that this is not exactly a PGM, since the edge weights are dependent on the value of the inputs, whereas with a PGM the parametrizations are independent of values. But from a high level, when defining a CPD from a set of N inputs to N outputs, we have machine learning model where we define it through an $N \times N$ matrix representing the edge weights for an RBM, a conditional Bayesian network through a generalized linear model, or an $N \times N$ matrix representing the covariance matrix for a Gaussian Process.

4 Optimization

¡++¿

In our exponential families discussion, we saw that our maximum likelihood objectives is equivalent to minimizing the KL divergence, and we are minizing weights that define our parametrization for our probability distribution. For exponential family models, this is amounts to minimizing over the weights the expected energy of the model. When we add L2 weight-regularization, it is equivalent to adding an isotropic Gaussian prior on the weights with covariance $\frac{\lambda}{C}\mathbf{I}$, where C is a multiplicative factor of the log likelihood loss that is often left out, and finding the MAP estimate for the weight. In addition, PGM inference algorithms, having broken out of a discriminative modeling objective, involve optimizing a functional known as the Helmholtz free energy functional, which can be considered a relaxation of KL minimization.

4.1 Maximum Entropy

¡++¿

Another optimization objective, which connects particularly well with the concepts of exponential families, is the maximum entropy objective. The concept has been used almost as a philosophical criterion for choosing distributions given observation, and we briefly restate it in terms of KL-divergence.

Suppose we have a set of observations from our samples or statistics:

$$\begin{aligned} \bar{U}_i &= \frac{1}{N}\sum_{n=1}^{N} U_i(x_n), \forall i \in \{1,\ldots,M\} \\ E_{X\sim P_{Data}(x)}\left[U_i(x)\right], &\forall i \in \{1,\ldots,M\} \end{aligned} \tag{20}$$

¡++¿

In the former case, we can use our samples as an estimate for expectations, so henceforth we will just treat our observed data as expectations of certain statistics.

We can now define sufficient statistics as log-linear factors for a Bolztmann distribution of those factors, which we will call a maximum entropy distribution:

$$\tilde{P_{MaxEnt}}(X) = \exp\left\{\sum_{i=1}^{M} w_i U_i(X)\right\} \tag{21}$$

¡++¿

Now suppose we try to find the I-projection of that distribution:

$$\begin{aligned} KL\left(Q(X)\|P_{MaxEnt}(X)\right) &= \sum_{i=1}^{M}\int Q(X)w_iU_i(X)dX - H\left(Q(X)\right) \\ &= \sum_{i=1}^{M} w_i E_{Q(X)}\left\{U_i(X)\right\} - H\left(Q(X)\right) \\ &= \sum_{i=1}^{M} w_i\bar{U}_i - H\left(Q(X)\right) \\ \arg\min_Q KL\left(Q(X)\|P_{MaxEnt}(X)\right) &= \arg\min_Q -H\left(Q(X)\right) \\ &= \arg\max_{Q(X)} H\left(Q(X)\right) \end{aligned} \tag{22}$$

¡++¿

So the distribution which minimizes the KL divergence with Boltzmann distribution, is the distribution that maximizes entropy given those constraints of observing data. The distribution that minimizes KL divergence with the Boltzmann distribution defined by the statistics is that same Boltzmann distribution, so maximizing entropy with respect to observed data is the same as minimizing KL divergence, which is consistent with our maximum likelihood formulations.

4.2 A possible perspective on layers

¡++¿

In fact, if we are have a GLM model with an associated exponential family distribution, our optimization objective is convex, so we are performing a convex optimization problem. However, the deeper layers change it into a non-convex problem.

There are some interesting tools in convex optimization that may be of help here.

One is under certain restrictions, we can compose functions together into convex functions. We can compose convex/concave functions with real-valued monotonic convex/concave into convex or concave functions, where the result depends on whether the scalar functions is nonincreasing or nondecreasing. We can also do composition with vector-value component-wise monotonic functions and and guarantee convexity/concavity. This is suggestively useful, since affine functions are both linear and concave, and neural networks are built with compositions of affine functions and non-linearities - though the sigmoid nonlinearity is neither convex nor concave.

Another way is that we can relax the definition of convexity to quasiconvexity, which guarantees weakened properties of convexity. Essentially this allows a function to become concave, but only if it is decreasing, although we have to be careful about what decreasing means in multi-dimensional spaces. Practically speaking, quasiconvexity is a property about the sublevel sets being convex. Moreover, many optimization algorithm proofs assume that the sublevel sets are bounded, which means that whatever loss value we are currently at, we only have to search over a bounded, convex set to find the minimum. In addition, composition of functions does apply to quasiconvex functions as well. This is suggestive of the ability of optimization to provide an interpretation of multi-layered networks.

Though we are able to convert PGMs into optimization objectives, we quickly find departures from this ideal in machine learning. One example is the Rectified linear unit (ReLu), which on inspection does not seem to have a well-known exponential family distribution analog. A second is that often additional regularization terms are added, such as smooth differentiability regularization or consistency regularization, that do not known probability distributions.

In contrast, regularization is a very well-developed theory in optimization. The primary optimization objectives and each of the regularization terms become what is called a multicriterion optimization. Constrained optimization is one case of multicriterion optimization, where bounds on the constraints function values define the additional criteria, and for Lagrangian optimization, the regularization constants have to be optimized as well, through the dual problem. Regularization, or softly constrained problems, pre-define the regularization constants.

Overall, the field of optimization has developed algorithms for optimizing the dual problem to turn a hard-constrained problem into a soft-constrained optimization problem, as well as primal-dual algorithms.

The interpretation of ReLu functions is somewhat more complicated. As discussed above, we might have some hard-constraint on the space where we are optimizing over. Normally, we would use gradient descent, but once we reach the constraint boundaries, our objective function has changed into something that is not differentiable. One method for avoiding this non-differentiability is to project the update back into the constrained set of legal weight values, and these two steps are what is called the projected gradient method. In fact, vanilla gradient descent with stepsize t_k is equivalent to minimizing a quadratic model:

$$\theta^{k+1} := \arg\min_{\theta} f(\theta^k) + \left\langle \nabla_\theta f(\theta^k), \theta - \theta^k \right\rangle + \frac{t_k}{2}\|\theta - \theta^k\|^2 = \theta^k - t_k \nabla_\theta f(\theta^k) \tag{23}$$

¡++¿

If we minimize that quadratic model over a set C, we define our projection operator:

$$\begin{aligned} P_C(\theta) &= \arg\min_{\theta} \delta\left(\theta \geq 0\right) \\ & P_C(\theta^k - t_k \nabla_\theta f(\theta^k) \end{aligned} \tag{24}$$

¡++¿

One interesting projection we can apply is projection to the non-negative real numbers - that is in fact the ReLu function. There are useful optimization implications for this function - when dealing with Lagrange multipliers, for our inequality constraints we have to constrain our lagrange multipliers to have non-negative values.

Just as hard constraints are the analog to regularization, we can soften the hard constraints of a projection operator to regularization functions, like L1 regularization or other, possibly non-differentiable functions. This is called the proximal operator:

$$\begin{aligned} prox_g(x) &= \arg\min_{u} g(u) + \frac{1}{2}\|x-u\|^2 \\ \theta^{k+1} &:= prox_{t_k g}\left(\theta^k - t_k \nabla_\theta f(\theta^k)\right) \end{aligned} \tag{25}$$

¡++¿

The use of these operators as a secondary step on top of a gradient descent step opens up a more powerful field of optimization theory and algorithms, including the fast iterative shrinking and thresholding algorithm (FISTA), of which Nesterov momentum is a special case.

From an interpretative standpoint, we can see that components in our model that involve projections and other operators may be perfectly valid as an alternative way to perform an optimization step without having to perform differentiation.

5 Operator Theory

¡++¿

Optimization is a search procedure for local minima, in the probabilistic framework corresponding to maximum likelihood or maximum a priori estimates. Gradient descent is the basis for most differentiable optimization procedures, but we have seen that other operators can be applied as well. Generally in these cases there is a specific interpretation for choosing these operators, such as projection on to a constrained set or applying the proximal operator to a model composed of additive terms, such as regularization.

Regularization also involves differential operators on the loss function, although this does not fit well with the concept of adding a prior. A second concern is that for deep neural networks, gradient operators and projected/proximal operators are vertical operators for optimization, but the layers themselves are applied "horizontally" and have a very different interpretation compared to the gradient operators. In particular, ReLus for lagrangians as projected gradient

descent in the vertical, optimization sense in weight space, whereas the ReLus for neural networks are projections in state space.

This state space operators interpretation is commonly used the Information Theory interpretation of deep learning, where the inputs and the hidden state outputs of each layer become random variables, and the layers themselves define transitions. Generally it is described as a Markov chain of probability distributions, just like Markov Chain Monte Carlo and Sequential Monte Carlo:

$$X^{(0)} \rightarrow X^{(1)} \rightarrow \cdots \tag{26}$$

where the superscript in parentheses is the notation for the index of a temporal model.

One acceptable transition is to define a mapping from $f_t : X^{(t)} \rightarrow X^{(t+1)}$:

$$x^{(t+1)} := f(x^{(t)}) \tag{27}$$

and this defines a deterministic transition from one probability space to another. To generalize to the non-deterministic case, consider the case of a Bayesian network, where we define a conditional probability distribution $P(X_i|Pa_{X_i})$, and for a Markov Chain, we could use the conditional probabilities with respect to each incoming state as a "column" in our transition matrix:

$$\begin{aligned} X^{(t+1)} &\sim P\left(X^{(t+1)}|X^{(t)}\right) < ++ > \\ X^{(t+1)} &= K_{t \rightarrow t+1} X^{(t)} \\ X^{(t+1)} &= \int K(X^{(t)}, X^{(t+1)}) X^{(t)} dX^{(t)} \end{aligned} \tag{28}$$

¡++¿

where we have defined a kernel matrix for the discrete case in the second equation and a kernel function for the continuous case in the third equation.

This is obviously a much more general treatment than the deterministic case, but one may wonder if this is able to encompass all methods worth considering of mapping one probability distribution to another. This question will lead us into the mathematics of distribution theory (CITE ¡++¿ Schwartz).

In order to lay the foundation an interpretation that will allow other operators we continue the discussion of layers acting as operators in deep learning theory. In "Can Recurrent Neural Networks Warp Time" and Deep Equilibrium Models (Bai et al, 2019 ¡++¿ CITE) for residual networks, the neural network instead defines a change in state, assuming the probability space shares the same outcome space:

$$X^{(t+1)} := X^{(t)} + \tau \circ X^{(t)} \tag{29}$$

¡++¿

This can then be interpreted as the discretization of a continuous time change in state:

$$\frac{d}{dt} X(t) = \tau \circ X(t) \tag{30}$$

¡++¿

We have now define a partial differential equation over time - the discrete case is known as the difference equation:

$$X^{(t+1)} - X^{(t)} = (\tau - \mathbf{I}) \circ X^{(t)} \tag{31}$$

¡++¿

and we have now defined the Laplacian on the right side, which has many interesting properties, one of which is a spectral graph theory proof that under certain assumptions, there are no eigenvectors/eigenfunctions with eigenvalue greater than 1, and that there is a unique non-negative eigenvector/eigenfunction with eigenvalue 1, which corresponds to the equilibrium distribution.

More importantly, we can now more generally define changes in time through a partial differential equation through an operator, and our neural network is describing a discretized version of a continuous time operator. The interpretation of a layer as a differential equation expands our definition of operators from marginalizations with conditional probabilities to differential operators, which is more appropriate for differential regularization terms. Moreover, while expanding the class of operators, it provides a less haphazard selection process to the choices of operators by an intuitive restriction to ones that correspond to meaningful differential equations.

$$\begin{aligned} X^{(t+1)} = \int P(X^{(t+1)}|X^{(t)})P(X^{(t)}dX^{(t)} \\ \frac{d}{dt}X(t) = \frac{\partial^2}{\partial x^2}X(t) \end{aligned} \tag{32}$$

¡++¿

The mathematical framework for this generalization is distribution theory (CITE ¡++¿). What we can integrate some function $g(x)$ of over a distribution of x. When the distribution is probability measure with a probability density function $P(x)$, we can write our expectation as

$$E_{X \sim P(x)}\left[g(x)\right] = \int g(x)P(x)dx \tag{33}$$

¡++¿

We can define this as an inner product over function space of the two functions, the function of a random variable and the probability measure. More generally, for each $P(x)$, we can define a linear mapping from the space of C^∞ functions (¡++¿ Footnote: technically smooth functions with compact support over an open subset U of X) to the reals:

$$\begin{aligned} F_P(g) = \int g(x)P(x)dx \\ F : C^\infty(X) \to R \\ < F, g >:= F(g) \end{aligned} \tag{34}$$

¡++¿

Hence, probability measures are members of the dual space, that is the space of linear functionals, of the space of smooth functions. The members of

this dual space are referred to as distributions or generalized functions, and it is denoted $\mathcal{D}'(X)$. However, there are examples of linear functionals which are not probability measures.

For example, the nth order differential operator evaluated at a specific point:

$$F(g) = \frac{\partial^n}{\partial x^n} g(x_0) \tag{35}$$

¡++¿

Distributions have various properties, but the most pertinent to us is when we can define linear mapping, T, from the space of distributions of one space, $\mathcal{D}'(X)$ to the space of distributions of another space, $\mathcal{D}'(Y)$.

Suppose there exists a reverse mapping $T' : C_c^\infty(Y) \to C_c^\infty(X)$ such that $\forall F \in \mathcal{D}'(X), g \in C_c^\infty(Y)$:

$$\langle TF, g \rangle = \langle F, T'g \rangle \tag{36}$$

¡++¿

This intuitively similar to the concept of an adjoint, except that it is not guaranteed to exist. When it is satisfied, it allows us to define the effect of an operator by the action of it's adjoint on the target function (¡++¿Footnote: bra-ket ¡f— T — g¿ in quantum physics).

The relevance of this mathematical framework is that we have defined distributions and we can also define differential operators and conditional probabilities as linear mappings from one space of distributions to another, which is exactly what we want for differential equations, Markov Chain Monte Carlo, and Probabilistic Graphical Models:

5.1 Example: Conditional Probabilities

¡++¿

For MCMC and Bayesian networks, we define can define mappings from one probability distribution, $P(x)$, to another distribution, $P(y)$, using a conditional probability distribution:

$$P(y) = \int P(y|x)P(x)dx \tag{37}$$

¡++¿

Since distributions are linear mappings of functions to the reals, we want to define the distribution by its action on a function of the random variable, y:

$$\begin{aligned} E_{y\sim P(y)}\left[f(y)\right] &= \int P(y)f(y)dy \\ &= \int\int P(y|x)P(x)dxf(y)dy \\ &= \int P(x)\int P(y|x)f(y)dy \\ &= \int P(x)\int K(x,y)f(y)dydx \\ &= \left\langle P(x), K^\dagger(y,x)f(y)\right\rangle \end{aligned} \tag{38}$$

¡++¿

where we define the kernel function $K(x, y) := P(y|x)$, and $K^\dagger$ is known as the adjoint of the kernel function. Conditional probabilties are in fact just defining this continuous kernel function, and applying it as an operator is just multiplying by this kernel function and rearranging the order of integration when we calculate expectations.

This means that conditional probabilties is simply the special case of multiplying distributions by a function, akin to multiplying by a weight matrix.

In the more general case where the kernel function does not correspond to a conditional distribution, we have to account for the change of measure with the change of variable formula:

$$\begin{aligned} \langle KP(x), g(y) \rangle &= \int K(x, y) P(y) f(y) dy \\ &= \frac{1}{det(K)} \left\langle P(x), S^\dagger f(y) \right\rangle \end{aligned} \tag{39}$$

¡++¿

Reversible conditional probability distributions are special cases where the determinant is equal to one, and that the sense of kernels used in Gaussian processes and Reproducible Kernel Hilbert Spaces are symmetric kernels, which are a special case of detailed balance in MCMC (¡++¿ CITE).

5.2 Differential operators

¡++¿

When we apply conditional distributions, particularly for Monte Carlo estimates, we are not explicitly concerned with probability distribution itself. Generally we are taking samples, then for each sample, sampling from the conditional probability distribution, and then using those to estimate some expectation.

However, when we apply the differential operator or are calculating entropy, we are interested in knowledge of the distribution itself beyond knowledge of the sample values. This is why we cannot calculate entropy directly for a data distribution - it would require us to know the probability density at our sample points.

However, it is still possible to calculate expectations of a distribution transformed by a differential operator:

$$\begin{aligned} \langle \partial^n P(x), g(x) \rangle &= \langle P(x), (-1)^n \partial^n g(x) \rangle \\ &= E_{x \sim P(x)} \left\{ (-1)^n \frac{\partial^n}{\partial x^n} g(x) \right\} < ++ > \end{aligned} \tag{40}$$

¡++¿

(¡++¿ FOOTNOTE, requires integration by parts and compact support forcing the integral to vanish at the boundaries).

We can construct more complicated operators by taking linear combinations of differental operators and multiplying by smooth functions, providing the form for most differential equations, and we can add stochasticity through the kernel functions through conditional probabilities of noise distributions.

5.3 Nonlinear operators

Our previous examples have all been linear operators on function space, and when we have been considering expectations, we are evaluating linear functionals through a function of a random variable.

However, there are both nonlinear functionals, such as the entropy function, and nonlinear operators, such as the ReLu operator, and neural networks usually apply a nonlinear operator at each layer. Nonlinear differential equations tend to not have closed-form solutions, but can be approximated numerically, and an infinite depth resnet with the same weights can be construed as an infinite depth calculation towards the fixed point (¡++¿). Moreover, the ReLu returns once again as being a plausible operator, like before in terms of a constraint. For example, we may have a differential equation for a mechanics problem, where the object is constrained to be above a plane. This equation is physically meaningful, though very difficult to solve, and often we have to pursue numeric solutions.