CD notes

Richard Turner

Notes on CD taken from: Hinton's lectures on POEs and his technical report, Mackay's 'Failures of the 1-Step Learning Algorithm', Welling's 'Learning in Markov Random Fields with Contrastive Divergence', and various other papers.

1 A Summary of Contrastive Divergence

Contrastive divergence is an approximate ML learning algorithm proposed by Hinton (2001). The Hinton network is a determinsitic mapping from observable space $\mathbf{x}$ of dimension D to an energy function $E(\mathbf{x};\mathbf{w})$ parameterised by parameters $\mathbf{w}$. The energy defines a probability via the Boltzmann distribution:

$$P(\mathbf{x}|\mathbf{w}) = \frac{\exp[-E(\mathbf{x},\mathbf{w})]}{Z(\mathbf{w})} \tag{1}$$

$$Z(\mathbf{w}) = \int d^D\mathbf{x} \exp[-E(\mathbf{x},\mathbf{w})] \tag{2}$$

Z is the normalising constant or partition function, which is very hard to evaluate.

(Note that models composed of an energy which is a sum of terms correspond to products of probability distributions - so products of experts naturally fall into this framework. Undirected graphical models are another example, in which the energy is specified through non-directional compatibilities.)

Differentiating the log-likelihood of the parameters we have;

$$\frac{\partial \log P(X|\mathbf{w})}{\partial \mathbf{w}} = \sum_n \frac{\partial}{\partial \mathbf{w}} \log\left[\frac{1}{Z(\mathbf{w})}\exp(-E[\mathbf{x}_n,\mathbf{w}])\right] \tag{3}$$

$$= \sum_n \left[\frac{1}{Z(\mathbf{w})}\frac{\partial Z(\mathbf{w})}{\partial \mathbf{w}} - \frac{\partial E(\mathbf{x}_n,\mathbf{w})}{\partial \mathbf{w}}\right] \tag{4}$$

$$= \sum_n \left[-\frac{1}{Z(\mathbf{w})}\int d^D\mathbf{x}\frac{\partial E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}}\exp(-E[\mathbf{x},\mathbf{w}]) - \frac{\partial E(\mathbf{x}_n,\mathbf{w})}{\partial \mathbf{w}}\right] \tag{5}$$

$$= N\left[\left\langle \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}}\right\rangle_{P(\mathbf{x}|\mathbf{w})} - \left\langle \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}}\right\rangle_{n}\right] \tag{6}$$

A maximum-likelihood learning algorithm for this density could therefore take steps:

$$\Delta \mathbf{w} \propto -\langle \mathbf{g} \rangle_0 + \langle \mathbf{g} \rangle_\infty \tag{7}$$

Where $\langle \mathbf{g} \rangle_0$ is the average of the gradient $\mathbf{g} = \partial E / \partial \mathbf{w}$ evaluated at the data points (drawn from the data density) and $\langle \mathbf{g} \rangle_\infty$ is the average gradient for the points $\mathbf{x}$ drawn from the prior $P(\mathbf{x}|\mathbf{w})$.

The algorithm coverges when the 'dreams' of the model match 'reality'. If the 'dreams' do not match reality, the parameters are altered so that the next round of dreams will be more like reality.

We know how to update the parameters, but how do we calculate the two expectations? In practice we have to approximate $\langle \mathbf{g} \rangle_\infty$: we guess the model's dreams (the guesses are 'confabulations').

If T is a Markov chain operator in $\mathbf{x}$ space that has $P(\mathbf{x}|\mathbf{w})$ as its unique invariant density then we can approximate $\langle \mathbf{g} \rangle_\infty$ by taking the data points and hitting each of them K times with T (typically K has to be large) and evaluating the gradient $\mathbf{g}_K$ at each resulting transformed point. For each data point a step:

$$\delta \mathbf{w} \propto -\mathbf{g}_0 + \mathbf{g}_K \tag{8}$$

is taken. Empirically Hinton has found that even for very small K the learning algorithm converges (close) to the maximum likelihood answer.

Hinton has an intuition for this: If we start at the data the Markov chain wanders away from the data to something it likes more. We can see which direction it wanders off in after only a few steps and its a big wste of time to let it go to equilibrium.

Indeed, with small K the algorithm can converge more rapidly because the difference $\mathbf{g}_0 - \mathbf{g}_K$ can be a less noisy quantity than $\mathbf{g}_0 - \mathbf{g}_\infty$. In the 1-step learning algorithm of Hinton, we set K to 1.

If our distribution is multimodal, initialising on the data should mean we explore all the modes close to the data. However, the drawbacks are two fold: 1. there is no pressure to remove modes from the model which are not close to any data and 2. although we can find the local shape of the mode, it is hard to estimate its volume.

The fact that $\mathbf{g}_0 - \mathbf{g}_K$ is a biased estimate of the gradient of the likelihood $\mathbf{g}_0 - \mathbf{g}_\infty$ is not important so long as the algorithm converges to the maximum-likehood parameters.

1.1 Where would you use contrastive divergence?

Let's give a concrete example for when you would want to use contrastive divergence. Consider Hinton and Sejnowski's Boltzmann machine. The Boltzmann machine specifies a joint distribution over observed $\mathbf{y}$ and latent $\mathbf{x}$ binary variables (+1/-1), through an energy function. Let $\mathbf{z}^T = [\mathbf{y}^T, \mathbf{x}^T]$ then:

$$P(\mathbf{z}|W) = \frac{1}{Z(W)} \exp(-\mathbf{z}^T W \mathbf{z}) \quad (9)$$

$$Z(W) = \sum_{\mathbf{z}} \exp(-\mathbf{z}^T W \mathbf{z}) \quad (10)$$

We can learn the weights W by maximum-likelihood as follows:

$$\frac{d \log P(\{\mathbf{y}^{(n)}\}_{n=1}^N | W_{ij})}{dW} = \frac{d}{dW_{ij}} \sum_n \log \left[\sum_{\mathbf{x}^{(n)}} \frac{1}{Z(W)} \exp(-[\mathbf{z}^{(n)}]^T W \mathbf{z}^{(n)}) \right] \quad (11)$$

$$= \sum_n \left[\langle z_i z_j \rangle_{P(\mathbf{x}|\mathbf{y}^{(n)}, W)} - \langle z_i z_j \rangle_{P(\mathbf{x},\mathbf{y}|W)} \right] \quad (12)$$

In the first term we clamp the visible units and average over the hiddens. In the second we average over all the imagined data and latents which the model could dream up.

These expectations can be approximated by Gibbs sampling:

$$P(x_i = 1|(x)_{\neq i}) = \frac{1}{1 + \exp(-2 \sum_j W_{ij} x_j)} \quad (13)$$

The Boltzmann machine is time consuming to simulate as the gradient of the loglikelihood is given by the difference in two terms which we have to use MCMC methods to approximate.

One way to speed things up is to use K-step learning, whereby we only take K-steps of the Gibbs sampler (T) to estimate the expectation. K might be as small as 1:

$$\frac{d \log P(\{\mathbf{y}^{(n)}\}_{n=1}^N | W_{ij})}{dW} = \sum_n \left[\langle z_i z_j \rangle_{P(\mathbf{x}|\mathbf{y}^{(n)}, W)} - \langle z_i z_j \rangle_{T^K P_0(\mathbf{y}^{(n)})} \right] \quad (14)$$

Where $T^K P_0(\mathbf{y}^{(n)})$ means: hit the data distribution K times with the Gibbs sampler.

We are interested in the properties of this new learning rule. In general it is very hard to analyse, but in the next section we look at a simple example.

1.2 Toy example: contrastive divergence convergence

Let's analyse a simple example of CD learning (where, for instance, there are no latent variables). Imagine we're going to learn the mean μ of a one-dimensional Gaussian distribution of unit variance.

The framework above gives us a method for finding the maximum-likelihood value of the mean without us having to calculate (potentially nasty) normalising constants.

The model specifies the energy:

$$E(x, \mu) = \frac{1}{2}(x - \mu)^2 \tag{15}$$

And the two gradients we need for exact ML learning are therefore $g_0 = \mu - \langle x \rangle_0$ and $g_\infty = \mu - \langle x \rangle_\infty$ so $\Delta\mu \propto \langle x \rangle_0 - \langle x \rangle_\infty = \mu_0 - \mu_\infty$.

We *approximate* the second term using sampling. We use the N data points to initialise the sampler. This will give us back N paired gradients, whose average gives us the noisy estimate of the dreams. An intuitive operator with the correct equilibrium distribution is the drift and diffuse operator:

$$x' = \mu + \alpha(x - \mu) + (1 - \alpha^2)^{1/2} n \tag{16}$$

where n is Gaussian noise of unit variance.

The first term causes the samples to drift to the mean of the model distribution, which by itself would reduce the variance of the distribution. The second term counteracts this by diffusing the points by just the right amount to make the equilibrium variance unity. This operator is not unlike those used in the Langevin method of sampling.

It's easy to see this operator has the correct equilibrium distribution as:

$$\begin{aligned}
P(x'|x) &= \text{Norm}(\mu(1-\alpha) + \alpha x, 1 - \alpha^2) && (17)\\
\langle x' \rangle &= \mu(1-\alpha) + \alpha\langle x \rangle && (18)\\
\mu_{eq} &= \mu(1-\alpha) + \alpha\mu_{eq} && (19)\\
\mu_{eq} &= \mu && (20)\\
\langle (x')^2 \rangle - \langle x' \rangle^2 &= \alpha^2\langle x^2 \rangle - \langle x \rangle^2 + 1 - \alpha^2 && (21)\\
\sigma_{eq}^2 &= \alpha^2\sigma_{eq}^2 + 1 - \alpha^2 && (22)\\
\sigma_{eq}^2 &= 1 && (23)
\end{aligned}$$

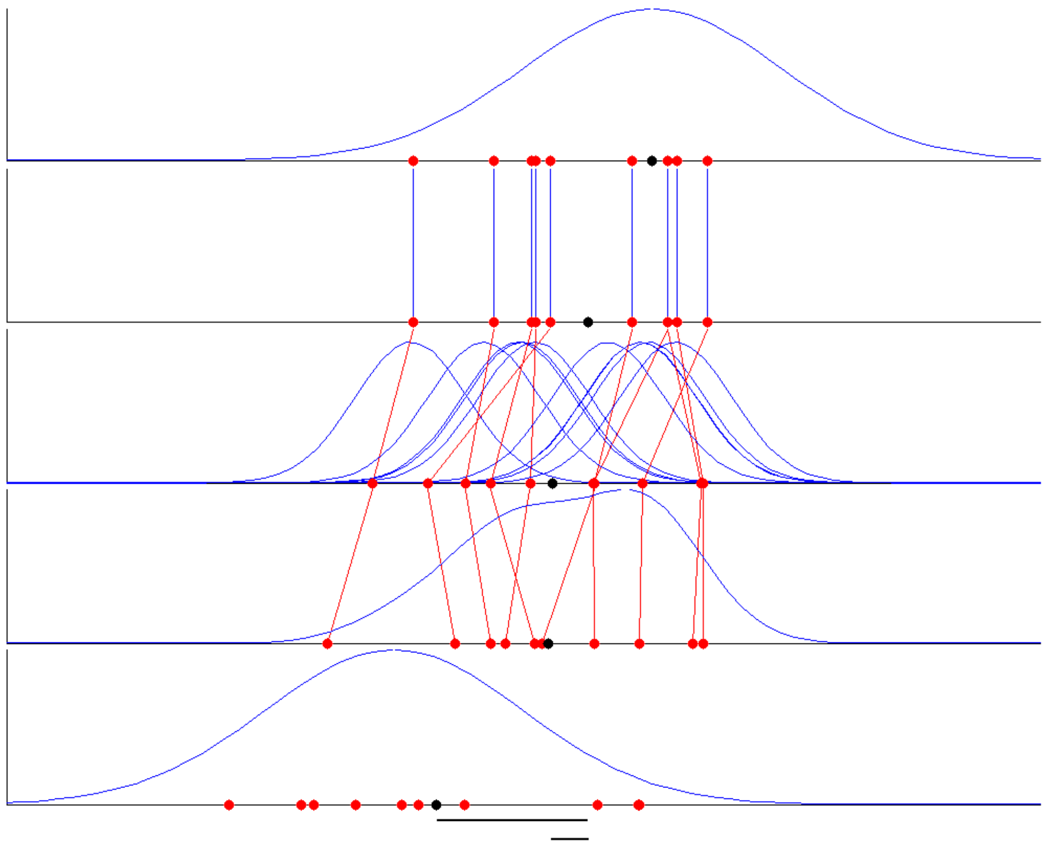
To be absolutely explicit, the algorithm is in pseudo-code:

```
Initialise $\mu$
```

```
for it=1:NumIts
for n=1:N
for k=1:K
$x^n_{k+1} = Tx^n_k$
end
end
$\mu_{it+1} = \mu_{it} + \sum_n (x^n_0 - x^n_K)$
end
```

So there is an outer loop of (noisy) gradient descent, and an inner loop of sampling (which can cause CD to be very slow). A further problem arises due to parameter dependent variance in the estimates of the gradients, which can swamp the actual signal and bias the programs outer loop.

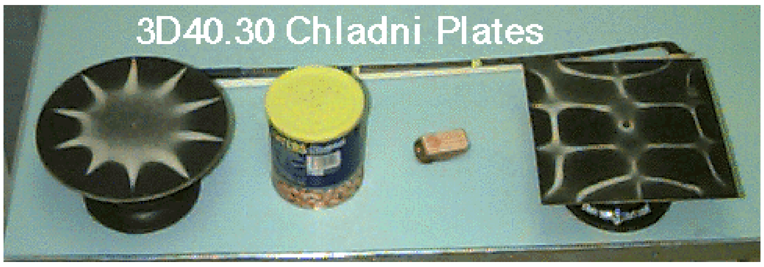
To 'do this to death', here's a picture of the algorithm. We imagine were at the start of learning when the current mean of the model is quite different from the mean of the data.



a. Data (red) generated from the forward model (blue) - a Gaussian with mean shown in black b. Data distribution (blue) with empirical mean (black) c. K=1 samples (red), drawn from conditional distributions (blue) d. K=2 samples (red), with the combination of the conditionals (blue) e. K = ∞ (equilibrium distribution) with mean shown in black and some typical samples (red).

Hinton's idea is that we set $K = 1$ to reduce the overhead. The gradients will clearly be biased (eg. Fig. 1), but intuitively, they should be in the correct direction, on average (again see Fig 1.). In the next section I'll run through David's proof of convergence in the above case, and then we'll modify the sampler and show CD no longer converges. An additional benefit of setting $K = 1$ is that the variance in the estimators will be reduced as each sample is paired to a nearby data point. There is a parameter dependent bias-variance trade-off controlled by K.

A physical analogy (due to Hinton) helps make clear why parameter dependent variance in the inner loop of the program can bias the outer loop. Imagine scattering sand onto a metal plate. Once the sand has settled we could vibrate the plate via a motor, or by playing it with a violin bow. The plate will vibrate and it will have a pattern of nodes and antinodes, dependent upon which vibration modes are excited by the driving force. It is found that the sand is repelled from the antinodes, and comes to rest in striking patterns (Fig 2.). If we imagine the flat disc to be our true objective function and the vibrating disc to be noisy reports of this surface; CD could be repelled from high variance regions of paramter space even though the likelihood is maximised there.



However, I wouldn't push this analogy too far: the grains of sand are still minimising an objective function - their total energy; the sum of potential and kinetic energy. However, CD cannot have an objective function (see later).

Let's make this totally concrete with an example: imagine learning both the mean and the variance of a Gaussian distribution using K-step learning. Let's thikn about the large K situation first. There are regions of parameter space where the variance parameter is large. If the optima lies in such a region and we don't have many data points (and therefore not many samples) the estimate of the mean in such location from sampling will have large variance. The large variance in the gradients calculated in such regions will tend to throw us away from them and we might only wander back to such regions relatively slowly. For this reason, reducing K to 1, might improve the performance of the algorithm.

1.3 CD convergence

It would be nice to pin down conditions for which the one-step algorithm is a correctly convergent algorithm.

Let's prove 1-step learning converges to ML solution when we use the drift and diffuse operator:

$$\begin{aligned}
\Delta\mu &\propto \langle x_n \rangle_n - \langle x'_n \rangle && (24)\\
&= \mu^{ML} - \langle \alpha x_n + \mu(1-\alpha) + (1-\alpha^2)^{1/2}\eta \rangle && (25)\\
&= \mu^{ML} - \alpha\mu^{ML} + \mu(1-\alpha) && (26)\\
&= (1-\alpha)[\mu^{ML} - \mu] && (27)
\end{aligned}$$

So μ decays to μ^{ML} with a rate proportional to $1-\alpha$.

David now modifies T to provide an example of a model $E(\mathbf{x}, \mathbf{w})$ and Markov chain T for which the true likelihood is unimodal in the parameters, but *the one step learning algorithm does not converge to the ML parameters.*

The modified transition operator works as follows. 10% of the time x moves under drift and diffuse. 90% of the time it moves under the *right mixing* operator.

$$x' = \begin{cases} x & if\ x \le \mu \\ \mu + |v| & if\ x \ge \mu \end{cases} \tag{28}$$

v is a standard normal variate. This operator leaves points to the left of μ alone and mixes up points to the right.

Imagine we have two data points at ± 1, $\mu_{ML} = 0$. Let's assume we initialise at $\mu = 0$. Does the algorithm stay there? The mean of the distribution $|v|$ is $\sqrt{2/\pi} \simeq 0.8$, the expectation of the mean after one step is: $0.1(-1-\gamma+1+\gamma) + 0.9(-1+0.8) = 0.9 \times (-0.2)$ (γ being the amount the drift and diffuse opertator moves both the points towards the mean on average). So the one step algorithm will on average move μ to the right. The algorithm has a fixed point $\mu^* \ge 0$ whose precise location depends on the value of α in the drift and diffuse operator.

David points out that this is not an unrealistic example: MCMC algorithms making moves in $\mathbf{x}$ space, often make large steps in one part of the space and small steps in another.

1.3.1 A higher level perspective

David provides other examples where CD fails to converge. All the examples of failures of the one-step learning algorithm work because the data distribution (a cloud of delta functions) is not precisely realisable by the model (nor would we want it to be), and the sufficient statistics of the data are not invariant under the operator T. Let's pick apart exactly what this means and hence how David can up with the above examples.

The exponential family of models can be written:

$$P(x|\mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp\left[-\sum_i f_i(x) w_i\right] \tag{29}$$

$$P(\{x_n\}_{n=1}^N|\mathbf{w}) = \frac{1}{Z(\theta)^*} \exp\left[-N\sum_i f_i(\{x_n\}_{n=1}^N) w_i\right] \tag{30}$$

Let's find the CD estimates for the ML gradients for this class of model, defined by the energy: $E(\{x_n\}_{n=1}^N, \mathbf{w}) = \sum_i f_i(\{x_n\}_{n=1}^N) w_i$.

$$\frac{\partial \log P(X|\mathbf{w})}{\partial \mathbf{w}} = N\left[\left\langle \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}}\right\rangle_{TP_0(\mathbf{x})} - \left\langle \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}}\right\rangle_{P_0(\mathbf{x})}\right] \tag{31}$$

$$\tag{32}$$

For CD to have the ML parameters as fixed points, this means:

$$0 = N\left[\langle \mathbf{f}(\{x_n\}_{n=1}^N)\rangle_{TP_0(\mathbf{x})} - \langle \mathbf{f}(\{x_n\}_{n=1}^N)\rangle_{P_0(\mathbf{x})}\right] \tag{33}$$

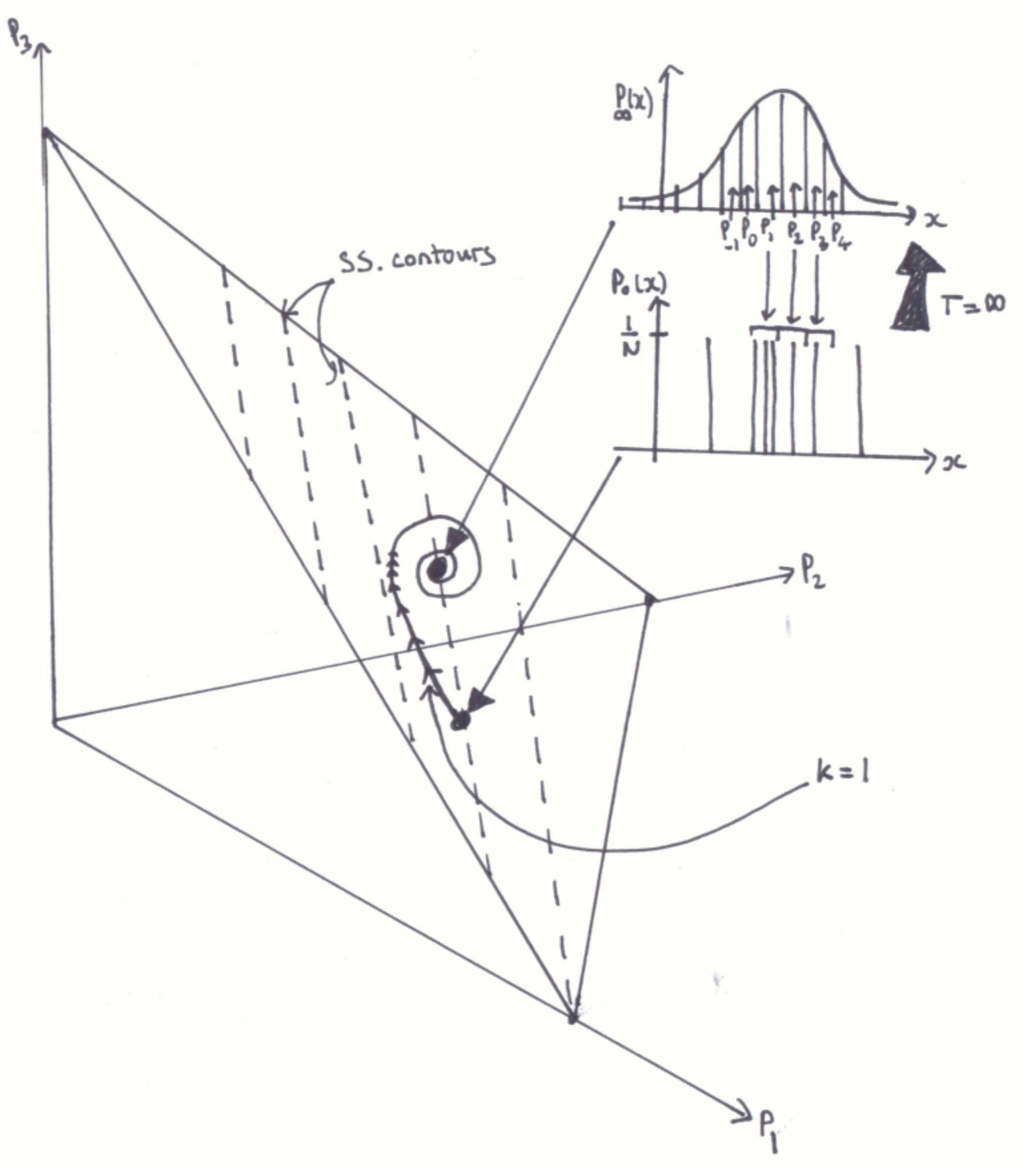
It's clear that if the sufficient statistics are not invariant under T the terms above cannot be equal and the fixed points of CD will not be the ML parameters.

However, something seems strange here: Imagine we are at the ML parameters. If we initialise our sampler on the data, why should it wander away from the data's sufficient statistics, only to come back to them after infinite time?

Let's think a little more about the dynamics of the probability density $Q_k = T^k P_0$ as a function of applications of T. We know Q_0 is just a sum of delta functions at the data points and Q_∞ is a Gaussian distribution of the correct mean and variance. But what happens for intermediate values of k? One way to visualise this is as follows: Imagine binning the probability density up into D small slots to make a histogram of probabilities. We could then plot the density as a point in a D dimensional space, and the evolution of the density would form a trajectory (from comb to Gaussian). (In the limit $D \to \infty$ we have a Hilbert space, in which the continuous density lives.) As probability distributions have to sum to 1, the dynamics will lie on a hyper-plane in this space (see the figure below).

What form does the operator T take in this discrete space? Well, as Q_k is a vector, T is a matrix; $\mathbf{q}_{k+1} = T\mathbf{q}_k$. Let's expand q_0 in terms of the eigenvectors of T: $\mathbf{q}_0 = \sum_d c_d \mathbf{e}_d$ then $\mathbf{q}_k = \sum_d \lambda_d^k c_d \mathbf{e}_d$. If the sampler is ergodic then only one of the eigenvalues is equal to 1 and the others have norm < 1. So the contribution of $d-1$ of the eigenvectors decay exponentially, and the remaining eigenvector is the equilibrium distribution - the Gaussian. Now, it is possible that pairs of eigenvalues are complex conjugates. This means the dynamics of $\mathbf{q}$ are decaying circles (spirals) towards the fixed point. What do the contours of constant expected sufficient statistics correspond to in our space? The expected sufficient statistics are given by $E(\mathbf{f}) = \sum_d f_d q_d$ ie. an inner product. An inner product is just the projection of one vector onto another, and so the contours of constant expected sufficient statistics are lines.

Now we have all we need to understand the dynamics of a system that would not have the ML fixed points. For such a system, when we initialise on the data, the Markov chain wanders away from this point, spiralling in toward another. The distribution initially wanders of the expected sufficient statistics contour, before exponentially decaying towards it. Of course, we could find operators that move along the contour (no imaginary eigenvalues), in which case the ML parameters would be fixed points. Furthermore, if the data distribution were actually realisable by the model, then Markov chain would stay put even if it had imaginary eigenvectors: the ML values of the parameters would again be fixed points.



In conclusion; the performance of CD is intimately tied to the sampler you use.

1.4 Objective functions

CD is empirically found to always converge and it might therefore have a Lyupanov function. However, it has not been found. Hinton has shown that it approximately minimises some objective function (which depends on the sampler).

We have shown that CD does noisy gradient ascent (noise due to averaging over a finite number of samples) on an objective function which approximates the the log-likelihood (approximates as we don't reach the equilibrium distribution of the sampler). To recap:

The gradients of the objective function we'd like to ascend are:

$$\frac{\partial \log P(X|\mathbf{w})}{\partial \mathbf{w}} = N\left[\left\langle \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}}\right\rangle_{P_\infty(\mathbf{x}|\mathbf{w})} - \left\langle \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}}\right\rangle_{P_D(\mathbf{x})}\right] \tag{34}$$

The actual gradients we actually compute are:

$$\frac{\partial \log P(X|\mathbf{w})}{\partial \mathbf{w}} = N\left[\left\langle \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}}\right\rangle_{Q(\mathbf{x}|\mathbf{w})} - \left\langle \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}}\right\rangle_{P_D(\mathbf{x})}\right] \tag{35}$$

So we approximate the 'dreams' of the model by 'confabulations' (things it might dream). In practice, the average over $Q = TP_D$ is conducted via sampling and is thus noisy too. Reiterating the two sorts of approximation here.

Hinton proposes another objective function which has similar gradients to those which we have used. The aim is the minimize the amount by which a step towards equilibrium improves the data distribution.

$$CD = KL(P_D(\mathbf{x})||P_\infty(\mathbf{x}|\mathbf{w})) - KL(Q(\mathbf{x}|\mathbf{w})||P_\infty(\mathbf{x}|\mathbf{w})) \tag{36}$$

where:

- $KL(P_D||P_\infty)$ = minimise the divergence between the data distribution and the model's distribution (objective function for normal ML learning [see later])
- $KL(Q||P_\infty)$ = maximise the divergence between the confabulations and the model's distribution. (I'm not sure why this is a good criteria for choosing your parameters, but it shows the intimate link between CD and the sampler)

CD is always positive unless $P_D = Q$ and thus $P_\infty = P_D$ meaning the model is perfect at the fixed points.

The gradients of this function wrt the parameters are:

$$
\begin{aligned}
-\frac{\partial CF}{\partial \mathbf{w}} &= -\frac{\partial}{\partial \mathbf{w}}[-\langle \log P_\infty(\mathbf{x}|\mathbf{w})\rangle_{P_D(\mathbf{x})} + \langle \log P_\infty(\mathbf{x}|\mathbf{w})\rangle_{Q(\mathbf{x}|\mathbf{w})}] && (37)\\
&= -\left\langle \frac{E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}} \right\rangle_{P_D(\mathbf{x})} + \left\langle \frac{E(\mathbf{x},\mathbf{w})}{\partial \mathbf{w}} \right\rangle_{Q(\mathbf{x}|\mathbf{w})} && (38)\\
&\quad + \frac{\partial Q(\mathbf{x}|\mathbf{w})}{\partial \mathbf{w}} \frac{\partial KL(Q(\mathbf{x}|\mathbf{w})||P_\infty(\mathbf{x}|\mathbf{w}))}{\partial Q(\mathbf{x}|\mathbf{w})} && (39)
\end{aligned}
$$

The nasty partition function cancels in the CF. However, as Q depends on the parameters of the model there is a problematic third term. Hinton says this is negligable for the models he's tried applying this to, so we recover the expression from the previous approach.

Given the ML motivation for CD I'm not sure you need the CD 'objective function' given above. It does however explicitly indicate that the choice of sampler will effect the results of CD learning.

To conclude, contrastive divergence can be viewed as doing approximate noisy gradient ascent of the log likelihood or contrastive-divergence objective functions. One idea is to use CD to cheaply march near to the ML solution and then to compute exact gradients for the final slow crawl to the peak.

2 KL-ML learning

As we have seen in the last section, contrastive divergence is often motivated as approximately minimising an objective function. This objective function is itself motivated from a KL divergence which has the same fixed points as the likelihood. In the following sections we try to pick apart this approach. We then extend it using the framework of Welling et al 2004.

2.1 No latents

A model might specify the probability of a data point given the model parameters $P(y|\theta)$. In **maximum-likelihood** learning we find the parameters θ of the model given some data $\{y_n\}_{n=1}^N$ (or Y for short) by maximising the likelihood: $\arg\max_\theta \log P(Y|\theta) = \arg\max_\theta \sum_{n=1}^N \log P(y_n|\theta)$. I'm being explicit as to what exactly each distribution means to avoid getting into a tangle in a minute.

This can be recast as the **minimisation** of a **KL divergence** by defining a '**data distribution**' $P_D(y|\{y_n\}_{n=1}^N) = \sum \delta_{y,y_n}$ which is a **sum of delta functions on each of the data points**. Taking the

KL between the data distribution and the model distribution:

$$\begin{aligned}
\mathrm{KL}[P_D(y)||P_\theta(y)] &= \sum_y P_D(y)\log\frac{P_D(y}{P_\theta(y)} &(40)\\
&= -H[P_D(y)] - \sum_y P_D(y)\log P_\theta(y) &(41)\\
&= -\sum_y\sum_n \delta_{y,y_n}\log P_\theta(y) &(42)\\
&= -\sum_n \log P_\theta(y_n) &(43)
\end{aligned}$$

which is the negative of the log likelihood

Thus the maximum-likelihood parameters are given by:

$$\theta_{ML} = \operatorname*{argmin}_\theta \mathrm{KL}[P_D(y)||P_\theta(y)] \tag{44}$$

KL$[P_D(y)||P_\theta(y)]$ therefore has the same fixed points as the likelihood.

2.2 With latent variables

All interesting generative models have latent variables $P_\theta(y,h)$ and it possible to form a KL interpretation of learning for such models. Again we have to choose the data distribution which now over both the observables and the latents. The natural choice is $P_D(y,h) = P_D(y)P_\theta(h|y)$.

$$\begin{aligned}
\mathrm{KL}[P_D(y,h)||P_\theta(y,h)] &= \sum_{y,h} P_D(y,h)\log\frac{P_D(y,h)}{P_\theta(y,h)} &(45)\\
&= \sum_{y,h} P_D(y,h)\log\frac{P_D(y)P_\theta(h|y)}{P_\theta(y)P_\theta(h|y)} &(46)\\
&= \sum_y P_D(y)\log\frac{P_D(y)}{P_\theta(y)} &(47)\\
&= \mathrm{KL}[P_D(y)||P_\theta(y)] &(48)
\end{aligned}$$

Which is therefore equal to the negative of the log-likelihood from the previous result.

2.3 A mini-review of the free energy formulation

It's often useful in physics to express probability distributions in a particular form called the Boltzmann distribution:

$$\begin{aligned} P_\theta(y) &= \frac{1}{Z}\exp[-E(y,\theta)] && (49)\\ Z &= \sum_y \exp[-E(y,\theta)] && (50)\end{aligned}$$

Models therefore specify a probability distribution through a non-negative energy. For such a parameterisation the **free energy** is a useful quantity from which any thermodynamic quantity can be derived. It is defined in terms of the partition function $F = -\ln Z$ and it can be related to the entropy and average energy.

$$\begin{aligned} H(P_\theta) &= \langle \log Z + E(y,\theta)\rangle_{P_\theta} && (51)\\ &= \log Z + \langle E(y,\theta)\rangle_{P_\theta} && (52)\end{aligned}$$

Therefore: $F = \langle E(y,\theta)\rangle_{P_\theta} - H(P_\theta)$

The free energy is often intractable to compute and one way to approximate it is through the **variational free energy**:

$$\begin{aligned} \tilde{F} &= F + \mathrm{KL}[Q(Y)||P_\theta(Y)] && (53)\\ & && (54)\end{aligned}$$

Which is bounded below by the true free energy when $Q(y) = P_\theta(y)$. A manipulation shows why this approximation **does not require the nasty partition function to be evaluated**:

$$\begin{aligned} \tilde{F} &= F - \sum_y Q(y)\log\frac{Q(y)}{P_\theta(y)} && (55)\\ &= F + \log Z - \sum_Y Q(y)\log\frac{Q(y)}{\exp[-E(y,\theta)]} && (56)\\ &= -\sum_y Q(y)\log\frac{Q(y)}{P^*_\theta(y)} && (57)\end{aligned}$$

Usually Q depends on some variational parameters α which can be chosen to make the bound on the true free energy as small as possible. We effectively select the best approximation to P_θ in the family of distributions $Q(\alpha)$.

2.4 Applying the free energy formulation to KL-learning

We now apply the methods described above. Let's parameterise our model distribution via an energy:

$$P_\theta(Y,H) = \frac{1}{Z}\exp[-E(Y,H,\theta)] \tag{58}$$

$$\tag{59}$$

(The Y and H denote we have more than just one dimension of hidden variables.)

For simplicity, we can restrict the form of energies under consideration further, using the exponential family parameterisation:

$$E(Y,H,\theta) = -\sum_{i,j}\lambda_{ij}f_{ij}(h_i,y_i) \tag{60}$$

$$\tag{61}$$

Pluging this into the KL divergence:

$$\begin{aligned}
\mathrm{KL}[P_D(Y,H)||P_\theta(Y,H)] &= \sum_{Y,H} P_D(Y,H)\log\frac{P_D(Y,H)}{P_\theta(Y,H)} && (62)\\
&= \log Z + \langle E(Y,H,\theta)\rangle_{P_D} - H[P_D(Y,H)] && (63)\\
CF &= -F_\theta + F_0 && (64)
\end{aligned}$$

The so-called **contrastive free energy** is the difference between the free energy of the model $F_\theta = F_\infty$ and a pseudo-free energy $F_0 = \langle E(Y,H,\theta)\rangle_{P_D} - H[P_D(Y,H)] = \sum_{Y,H} P_D(Y,H)\log\frac{P_D(Y,H)}{P^*_\theta(Y,H)}$

We can learn the parameters by taking gradients of the CF:

$$\begin{aligned}
\frac{\partial\,\mathrm{KL}[P_D(Y)||P_{\lambda_k}(Y)]}{\partial\theta} &= \frac{1}{Z}\frac{\partial Z}{\partial\lambda_k} + \frac{\partial\langle E(Y,H,\lambda)\rangle_{P_D}}{\partial\lambda_k} && (65)\\
&= \langle f_{ik}(h_i,y_i)\rangle_{P_\theta} - \langle f_{ik}(h_i,y_i)\rangle_{P_D} && (66)
\end{aligned}$$

This is the usual relation that in exponential family model the sufficient statistics of the latents averaged over the data distribution must match the prior, when averaged over the data.

Learing proceeds as follows: find the pseudo-free energy of the system with the data points clamped to their observed values (involving inference, and then sampling, over the hidden units). Then remove the constraints on the observables, allowing the system to relax to a new distribution P_θ with a lower free energy (now involving sampling over both latents and observables). If in this process the sufficient statistics change, then we have an imperfect model and we need to change the parameters in such a way that they are better matched at the next iteration. At the end dreams match reality and we have found the ML solution.

However, in order to learn parameters in this way, we need to compute these two nasty expectations. Can we use the variational free energy approximations to make things easier?

2.5 Free energy approximations for CF-learning

We'll now form approximations to each of the free energies:

The 'classical' variational approximation for F_∞ is given by: $\tilde{F}_\infty = \mathrm{KL}[Q_\infty(Y,H)||P^*_\theta(Y,H)]$, which, to reiterate, does not depend on Z.

We can also use a variational approximation for the pseudo-free energy $\tilde{F}_0 = \sum_{Y,H} Q_D(Y,H) \log \frac{Q_D(Y,H)}{P^*_\theta(Y,H)}$. Where if we choose $Q_D(Y,H) = P_D(Y)Q(H|Y)$, $\tilde{F}_0$ is bounded below by F_0: $\tilde{F}_0 = F_0 + KL[Q(H|\hat{Y})||P_\theta(H|\hat{Y})]$

The variational distributions will be simpler (tree-like, fully-factored etc.) and they will have parameters that will be chosen by minimising the respective KL divergence terms.

The **approximate contrastive divergence** is thus:

$$\tilde{C}F \quad = \quad \tilde{F}_0 - \tilde{F}_\infty \tag{67}$$

There is an intuitive argument for why $CF \geq 0$:

Both the free energies have the same energy function (although the expectation is taken over different distributions). However a constrained entropy is always lower than an unconstrained entropy, and therefore the entropy of the data distribution is always lower than the entropy of the model distribution. Hence, you might expect $F_0 \geq F_\infty$ and we slowly tighten the distribution as we iterate.

I'm not sure the point of this paragraph: CF is certainly greater than or equal to zero (as it is a KL), the question is whether the approximation is (ie. has a Lyapunov function).

3 Variational CD

In CD we relax the data distribution toward the model distribution using a Markov chain operator. Taking inspiration from the previous section, an alternative is to relax the variational version by optimising the parameters of a variational distribution for K steps:

$$\tilde{C}F = \tilde{F}_0 - \tilde{F}_K \tag{68}$$