PCA and VQ in neural networks

Sebastian Seung

9.641 Lecture 24: Dec. 6, 2000

1 Pattern coding and recognition
For each class, use VQ to find a set of prototypes. To classify a novel input, find the closest prototype in memory, and assign the novel input to the same class as this best match.

For each class, use PCA to find an approximation by an affine or linear subspace. To classify a novel input, find the closest subspace.

2 Stochastic approximation
\[ w_{t+1} = w_t + \eta_t (x_t - w_t) \]
\[ = (1 - \eta_t) w_t + \eta_t x_t \]

If \( \eta \) is small, then we can approximate this as
\[ \dot{w} + w = \langle x \rangle \]
so \( w \) converges to the mean. But what about fluctuations?

According to stochastic approximation theory, if the \( x_t \) are drawn i.i.d. from some probability density \( P(x) \), and
\[ \sum_{t=0}^{\infty} \eta_t = \infty \]
\[ \sum_{t=0}^{\infty} \eta^2_t < \infty \]
then \( w_t \) converges to the mean.

3 Competitive learning
The \( k \)-means algorithm was a batch method, making use of all the examples to update the cluster centers. The competitive learning algorithm is an online method, which does not have to store more than one example at a time. Suppose there are \( k \) cluster centers \( w_a \).
1. minimize the distance $|x - w_a|$ to find the winner

2. update $w'_a = w_a + \eta(x - w_a)$ for the winner only.

In this rule, $w_a$ converges to the mean of the examples to which it was assigned. This can be seen by taking the time average of the update rule, and seeing that its steady state is $w_a = \langle x \rangle$. This can be interpreted as online gradient descent on the $k$-means cost function.

Could we implement this as a neural network? Note that $\arg\min_a |x - w_a|$ is the same as $\arg\max_a w_a \cdot x$ if the $w_a$ all have the same norm.

\[
\dot{z}_a + z_a = \Theta \left( b_a + \alpha z_a - \beta \sum_a z_a \right)
\]  
\[
b_a = \sum_i W_{ai} x_i
\]  

Let’s assume that the $b_a$ are bounded above by $b_{max}$. If $\beta > b_{max}$, and $\beta < \alpha < 2/\beta - b_{max}$, then there is only one neuron active at a steady state. If the network is initialized at $z_a = 0$, then the active neuron is the one with the largest feedforward input. The network has to be reset after each input.

Therefore, performing the weight update

\[
w'_{ai} = (1 - \eta) w_{ai} + \eta z_a x_i
\]

after convergence to a steady state is an implementation of competitive learning. This is Hebbian, since it is driven by the correlation of postsynaptic ($z_a$) and presynaptic ($x_i$) activity.

4 Oja’s rule

For a neural implementation of PCA, let’s try the same learning rule

\[
w' = (1 - \eta) w + \eta z x
\]

with a linear neuron that computes $z = w \cdot x$.

If the learning rate $\eta$ is low, then we can time average this to obtain $w' = (1 - \eta) w + \eta C w$. The continuous time approximation to this is

\[
\dot{w} + w = Cw
\]

The principal component will dominate this dynamics, but the problem is that $w$ will generally diverge or go to zero.

Oja’s rule makes the weight decay dependent on the output of the neuron,

\[
w' = (1 - \eta z^2) w + \eta z x
\]
The time average is
\[ w' = w + \eta(Cw - (w^T Cw)w) \]
Any steady state of this equation is an eigenvector of \( C \) with unity norm. To prove convergence, we consider the continuous time version
\[ \dot{w} = Cw - (w^T Cw)w \]
In the eigenvector basis, this looks like
\[ \dot{w} = \Lambda w - (w^T \Lambda w)w \]
which is equivalent to
\[ \frac{d}{dt} (\log w_i) = \Lambda_i - (w^T \Lambda w) \]
This is reminiscent of our winner-take-all network, since it has a global inhibition term. If we compare two components, we see that
\[ \frac{d}{dt} \log(w_i/w_j) = \Lambda_i - \Lambda_j \]
which proves that the largest eigenvalue wins.

5 Models of development

The Kohonen map is a variant of competitive learning that is designed to give topographic organization to the
update all neurons in neighborhood of winner topographic map
learning from noise developmental models Linsker

6 Matrix factorization