580.439/639 Solutions to Homework 9

Problem 1

Part a) The matrix **U** is $\mathbf{U}=\left[\vec{u}^1\,\vec{u}^2\,\vec{u}^3\ldots\vec{u}^P\right]$. By inspection, **Q=U**^{T}**U**. Note that each element of **Q** is the dot product of the corresponding stimulus patterns and **Q** is *PxP*. Now applying input pattern $\vec{u}^l$

$$
\begin{aligned}
v &= \left(\sum_{j=1}^{P}\sum_{k=1}^{P} v^j \left(\mathbf{Q}^{-1}\right)_{jk} \vec{u}^k\right)\bullet \vec{u}^l \\
&= \left(\sum_{j=1}^{P} v^j \sum_{k=1}^{P}\left(\mathbf{Q}^{-1}\right)_{jk} \vec{u}^k \bullet \vec{u}^l\right) = \sum_{j=1}^{P} v^j \sum_{k=1}^{P}\left(\mathbf{Q}^{-1}\right)_{jk}(\mathbf{Q})_{kl} \\
&= \sum_{j=1}^{P} v^j \left(\mathbf{Q}^{-1}\mathbf{Q}\right)_{jl} = \sum_{j=1}^{P} v^j \delta_{jl} = v^l .
\end{aligned}
$$

Thus the weights defined as in the problem set store these patterns in the perceptron.

Part b) Note that **Q** must not be singular. A sufficient condition is that the pattern vectors are linearly independent. This means that no more than *N* patterns can be stored.

Part c) Differentiating *E* w.r.t. $\vec{w}$ gives

$$\vec{\nabla}E = \frac{1}{2}\sum_{j=1}^{P} 2\left(\vec{w}\cdot\vec{u}^j - v^j\right)u^j = \sum_{j=1}^{P}\left(\vec{w}\cdot\vec{u}^j - v^j\right)u^j \; .$$

Then $\vec{w}_{new} = \vec{w}_{old} - \varepsilon\vec{\nabla}E$ is essentially the same as the PLR derived in class, as long as the bias term is absorbed into the weights. Also, this expresses the average over the whole training set, whereas the formula given in class is the update for each input pattern by itself.

The gradient descent method can be applied to more general perceptrons with a squashing function, where

$$v = S(\vec{w}\bullet\vec{u}-\gamma) \quad \text{and} \quad S(\cdot) \text{ is a differentiable sigmoidal function}$$

The result is similar to the PLR.

Problem 2

Part a) This is mainly a problem in keeping track of subscripts. Using the hint, suppose only the state of the kth neuron changes, from S_k to S_k', consistent with the update rule. The change in *H* can be written as below, where only terms involving neuron *k* are included. All other terms will cancel in the difference:

$$H' - H = -\frac{1}{2}\sum_{j=1}^{N} w_{kj}\left(S_k' - S_k\right)S_j - \frac{1}{2}\sum_{j=1}^{N} w_{jk}S_j\left(S_k' - S_k\right) + \frac{1}{2}w_{kk}(S_k'^2 - S_k^2) \quad (***)$$

The third term is included because the terms in w_{kk} are counted twice in the first two terms. Because $S_k = \pm 1$, this term is zero. Consider the first term and note that it can be rewritten as

$$-\frac{1}{2}\sum_{j=1}^{N} w_{kj}(S_k' - S_k)S_j = -\frac{1}{2}\left(\sum_{j=1}^{N} w_{kj}S_j\right)(S_k' - S_k)$$

The summation in brackets on the right-hand side is the argument of sgn() in the update rule for the k^{th} neuron and thus has the same sign as $S_k' - S_k$; because S is either +1 or -1, this means that the first term in (***) is negative. This leaves only the second term which has the same value as the first, since the weights are symmetric (w_{jk}=w_{kj}). By these arguments, the r.h.s. of Eqn. (***) is negative and the change in H is negative, consistent with a Lyapunov function.

Note that H has not been shown to be a Lyapunov function because we haven't shown that it is positive definite.

Part b) The function H in Eqn. (**) in the problem set is minimized when the summation in the brackets is maximized. This is a sum of N terms, each of which can be either 1 or –1. The maximum N^2 occurs when all terms are +1 or all terms are –1, that is when S_j=u_j or S_j=-u_j. Thus the minimum of H occurs at the pattern stored in the net or at minus the pattern. It turns out that this is true in general, if a pattern $\vec{u}$ is a stable state of a Hopfield net, then $-\vec{u}$ is also stable.

To make the Lyapunov function positive, just add a constant, so the actual Lyapunov is $N/2+H$ which is positive definite and has the same monotonic decreasing property as Eqn. (*).

To store a single pattern in a Hopfield net, the weights are set as $w_{jk}=u_ju_k/N$. Writing out the square in Eqn (**) and using this definition gives

$$H = -\frac{1}{2N}\left(\sum_{i=1}^{N} S_iu_i\right)^2 = -\frac{1}{2N}\sum_{i=1}^{N} S_iu_i \sum_{j=1}^{N} S_ju_j = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} S_iS_j\frac{u_iu_j}{N} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} S_iS_jw_{ij}$$

which matches Eqn. (*) as desired.

For the general case of P patterns stored in a net, the summation in Eqn. (**) could be averaged over the P patterns. Some thought will show that this is not guaranteed to have even a local minimum at all of the stored patterns. In practice Hopfield nets fail to store some of the patterns used to compute the weights when the number of patterns P exceeds about 0.14*N. This point is shown in Hertz et al. p. 35ff.