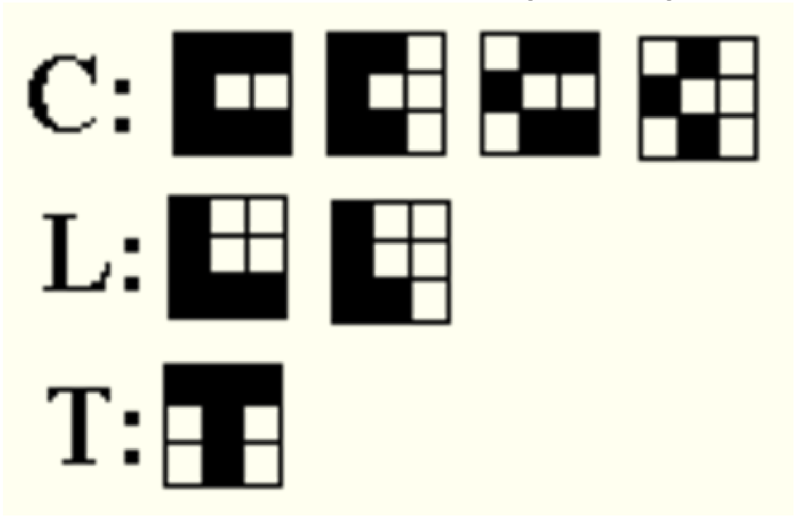
Tutorial 7

Monday 17th October, 2016

Problem 1. Design a multilayer perceptron which will learn to recognize various forms of the the letters C,L,T placed on a 3x3 grid through backpropagation algorithm.



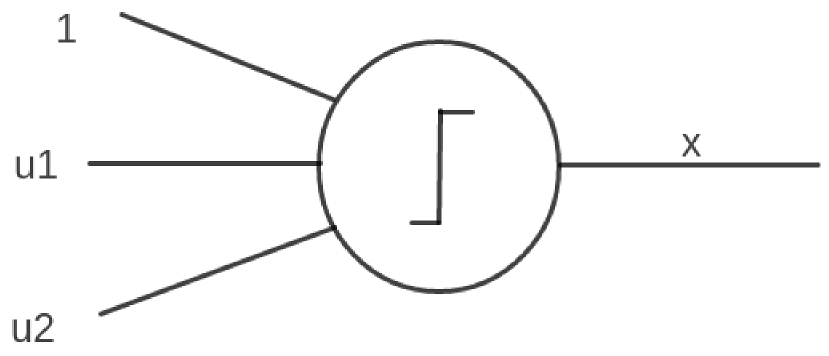
1. Design a one layer network indicating what should be applied at the input layer and what should be expected at the output layer showing the number of neurons, the connections between them and the neurons output function.

2. repeat (a) for two layer network by adding a hidden layer

   Solution: See Problem 5 of `http://www.eee.metu.edu.tr/~halici/courses/543LectureNotes/questions/qch6/index.html` for the solution.

Problem 2. Consider a perceptron for which $u \in R^2$ and

$$f(a) = \begin{cases} 1 & a > 0 \\ 0 & a = 0 \\ -1 & a < 0 \end{cases}$$



Let the desired output be 1 when elements of class A = {(1,2),(2,4),(3,3),(4,4)} is applied as input and let it be -1 for the class B = {(0,0),(2,3),(3,0),(4,2)}. Let the initial connection weights $w_0(0) = +1, w_1(0) = -2, w_2(0) = +1$ and learning rate be h = 0.5.

This perceptron is to be trained by perceptron convergence procedure, for which the weight update formula is $w(t+1) = w(t) + \eta(y^k - x^k(t))u^k$

1. (a) Mark the elements belonging to class A with x and those belonging to class B with o on input space.
   (b) Draw the line represented by the perceptron considering the initial connection weights w(0).
   (c) Find out the regions for which the perceptron output is +1 and 1
   (d) Which elements of A and B are correctly classified, which elements are misclassified and which are unclassified?
2. If u=(4,4) is applied at input, what will be w(1) ?
3. Repeat a) considering w(1).
4. If u=(4.2) is then applied at input, what will be w(2)?
5. Repeat 1) considering w(2).
6. Do you expect the perceptron convergence procedure to terminate? Why?

Solution: See Problem 8 of `http://www.eee.metu.edu.tr/~halici/courses/543LectureNotes/questions/qch6/index.html` for the solution.

Problem 3. Recall the Regularized (Logistic) Cross-Entropy Loss function (minimized wrt $\mathbf{w} \in \Re^p$):

$$E(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} \log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \frac{\lambda}{2m}\|\mathbf{w}\|_2^2 \tag{1}$$

Now prove that minimizing the following dual kernelized objective[1] (minimized wrt $\alpha \in \Re^m$) is equivalent to minimizing the regularized cross-entropy loss function:

$$E_D(\alpha) = \left[\sum_{i=1}^{m}\left(\sum_{j=1}^{m} -y^{(i)}K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\alpha_j + \frac{\lambda}{2}\alpha_i K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\alpha_j\right) + \log\left(1 + \sum_{j=1}^{m}\alpha_j K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\right)\right] \tag{2}$$

where, decision function $f_{\mathbf{w}}(\mathbf{x}) = \dfrac{1}{1 + \exp\left(\sum_{\mathbf{j=1}}^{\mathbf{m}} \alpha_{\mathbf{j}}\mathbf{K}\left(\mathbf{x}, \mathbf{x}^{(\mathbf{j})}\right)\right)}$

Solution:

We will prove this result and in the process, also motivate (and later prove) the more general **Representer Theorem**.

1. **Some preliminary steps:**

   Recall another form of the regularized cross entropy[2] equivalent to (1)

$$E(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - \log\left(1 + \exp\left(\mathbf{w}^T\phi(\mathbf{x}^{(i)})\right)\right)\right)\right] + \frac{\lambda}{2m}||\mathbf{w}||^2 \tag{3}$$

   First of all, we will drop the common term $\frac{1}{m}$ from the primal optimization problem and equivalently minimize the unscaled version

$$E(w) = -\left[\sum_{i=1}^{m}\left(y^{(i)}\log f_w\left(\mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \tag{4}$$

$$\nabla E(\mathbf{w}) = \left[\sum_{i=1}^{m}\left(y^{(i)}\nabla\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right)\nabla\log\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \lambda\mathbf{w} \tag{5}$$

2. $\nabla\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) = \phi(\mathbf{x}^{(i)})e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}\left(\frac{1}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)^2$ and

   $\nabla\log\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right) = -\phi(\mathbf{x}^{(i)})\left(\frac{1}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)^2$

[1] `http://perso.telecom-paristech.fr/~clemenco/Projets_ENPC_files/kernel-log-regression-svm-boosting.pdf`

[2] Slide 5 of `https://www.cse.iitb.ac.in/~cs725/notes/lecture-slides/lecture-17-annotated.pdf`

3. $\Rightarrow$

$$\nabla E(\mathbf{w}) = \left[\sum_{i=1}^{m} \left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\phi(\mathbf{x}^{(i)})\right] + \lambda\mathbf{w} \tag{6}$$

At optimality, a necessary condition is that $\nabla E(\mathbf{w}) = 0$ and therefore,

$$\mathbf{w} = \frac{1}{\lambda}\left[\sum_{i=1}^{m} \left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\phi(\mathbf{x}^{(i)})\right] \tag{7}$$

4. **The main idea:**

We first recap the main optimization problem

$$E(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - \log\left(1+\exp\left(\mathbf{w}^T\phi(\mathbf{x}^{(i)})\right)\right)\right)\right] + \frac{\lambda}{2m}||\mathbf{w}||^2 \tag{8}$$

and an expression for $\mathbf{w}$ at optimality

$$\mathbf{w} = \frac{1}{\lambda}\left[\sum_{i=1}^{m} \left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\phi(\mathbf{x}^{(i)})\right] \tag{9}$$

To completely prove this specific case of KLR, let $\mathcal{X}$ be the space of examples such that $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\right\} \subseteq \mathcal{X}$ and for any $\mathbf{x} \in \mathcal{X}$, $K(.,\mathbf{x}) : \mathcal{X} \to \Re$ be a function such that $K(\mathbf{x}', \mathbf{x}) = \phi^T(\mathbf{x})\phi(\mathbf{x}')$. Recall that $\phi(\mathbf{x}) \in \Re^n$ and

$$f_{\mathbf{w}}(\mathbf{x}) = p(Y=1|\phi(\mathbf{x})) = \frac{1}{1+\exp\left(-\mathbf{w}^{\mathbf{T}}\phi(\mathbf{x})\right)}$$

For the rest of the discussion, we are interested in viewing $-\mathbf{w}^T\phi(\mathbf{x})$ as a function $h(\mathbf{x})$

$$f_{\mathbf{w}}(\mathbf{x}) = p(Y=1|\phi(\mathbf{x})) = \frac{1}{1+\exp\left(\mathbf{h}(\mathbf{x})\right)}$$

We will prove that for the optimization problem (8), $\mathbf{h}(\mathbf{x})$ can be equivalently expressed as $\sum_{\mathbf{j}=\mathbf{1}}^{\mathbf{m}} \alpha_{\mathbf{j}}\mathbf{K}\left(\mathbf{x}, \mathbf{x}^{(\mathbf{j})}\right)$, as a result of which we will obtain the following terms of (2):

$$\left[\sum_{i=1}^{m}\left(\sum_{j=1}^{m} -y^{(i)}K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\alpha_j\right) + \log\left(1 + \sum_{j=1}^{m}\alpha_j K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\right)\right] \tag{10}$$

Substituting (9) into $\frac{\lambda}{2m}||\mathbf{w}||^2$ term of (8) we will get the regularizer into the form

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\lambda}{2}\alpha_i K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\alpha_j$$

which forms the remaining term of (2)

5. Consider the set of functions $\mathcal{K} = \{K(., \mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$ and let $\mathcal{H}$ be the set of all functions that are **finite** linear combinations of functions in $\mathcal{K}$. That is, any function $h \in \mathcal{H}$ can be written as $\mathbf{h}(.) = \sum_{t=1}^{T} \alpha_t K(., \mathbf{x}_t)$ for some T and $\mathbf{x}_t \in \mathcal{X}, \alpha_t \in \Re$. One can easily verify that $\mathcal{H}$ is a vector space[3]

   Note that, in the special case when $f(\mathbf{x}') = K(\mathbf{x}', \mathbf{x})$, then $T = m$ and

$$f(\mathbf{x}') = K(\mathbf{x}', \mathbf{x}) = \sum_{i=1}^{n} \phi_i(\mathbf{x}') K(\mathbf{e}_i, \mathbf{x})$$

   where $\mathbf{e}_i$ is such that $\phi(\mathbf{e}_i) = \mathbf{u}_i \in \Re^n$, the unit vector along the i^{th} direction.

   Also, by the same token, if $\mathbf{w} \in \Re^n$ is in the search space of the regularized cross-entropy loss function (1), then

$$\phi^{\mathbf{T}}(\mathbf{x}')\mathbf{w} = \sum_{i=1}^{n} w_i K(\mathbf{e}_i, \mathbf{x})$$

   Thus, the solution to (1) is an $h \in \mathcal{H}$.

6. **Inner Product over $\mathcal{H}$:** For any $g(.) = \sum_{t=1}^{S} \beta_s K(., \mathbf{x}'_s) \in \mathcal{H}$ and $h(.) = \sum_{t=1}^{T} \alpha_t K(., \mathbf{x}_t) \in \mathcal{H}$, define the inner product[4]

$$\langle h, g \rangle = \sum_{s=1}^{S} \beta_s \sum_{t=1}^{T} \alpha_t K(\mathbf{x}'_s, \mathbf{x}_t) \tag{11}$$

   Further simplifying (11),

$$\langle h, g \rangle = \sum_{s=1}^{S} \beta_s \sum_{t=1}^{T} \alpha_t K(\mathbf{x}'_s, \mathbf{x}_t) = \sum_{s=1}^{S} \beta_s f(\mathbf{x}_s) \tag{12}$$

   One immediately observes that in the special case that $g() = K(., \mathbf{x})$,

$$\langle h, K(., \mathbf{x}) \rangle = h(\mathbf{x}) \tag{13}$$

7. **Orthogonal Decomposition:** Since $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\right\} \subseteq \mathcal{X}$ and $\mathcal{K} = \{K(., \mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$ with $\mathcal{H}$ being the set of all finite linear combinations of function in $\mathcal{K}$, we also have that

$$lin\_span\left\{K(., \mathbf{x}^{(1)}), K(.\mathbf{x}^{(2)}), \ldots, K(., \mathbf{x}^{(m)})\right\} \subseteq \mathcal{H}$$

[3]Try it yourself. Prove that $\mathcal{H}$ is closed under vector addition and (real) scalar multiplication.

[4]Again, you can verify that $\langle f, g \rangle$ is indeed an inner product following properties such as symmetry, linearity in the first argument and positive-definiteness: `https://en.wikipedia.org/wiki/Inner_product_space`

Thus, we can use orthogonal projection to decompose any $h \in \mathcal{H}$ into a sum of two functions, one lying in *lin_span* $\left\{K(.,\mathbf{x}^{(1)}), K(.\mathbf{x}^{(2)}), \ldots, K(.,\mathbf{x}^{(m)})\right\}$, and the other lying in the orthogonal complement:

$$h = h^{\parallel} + h^{\perp} = \sum_{i=1}^{m} \alpha_i K(.,\mathbf{x}^{(i)}) + h^{\perp} \tag{14}$$

where $\langle K(.,\mathbf{x}^{(i)}), h^{\perp}\rangle = 0$, for each $i = [1..m]$.

For a specific training point $\mathbf{x}^{(j)}$, substituting from (14) into (13) for any $h \in \mathcal{H}$, using the fact that $\langle K(.,\mathbf{x}^{(i)}), h^{\perp}\rangle = 0$

$$h(\mathbf{x}^{(j)}) = \langle \sum_{i=1}^{m} \alpha_i K(.,\mathbf{x}^{(i)}) + h^{\perp}, K(.,\mathbf{x}^{(j)})\rangle = \sum_{i=1}^{m} \alpha_i \langle K(.,\mathbf{x}^{(i)}), K(.,\mathbf{x}^{(j)})\rangle = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) \tag{15}$$

which we observe is independent of $h^{\perp}$.

8. **Analysis of the Regularized Cross-Entropy Logistic Loss:**

   The Regularized Cross-Entropy Logistic Loss (8), has two parts (after ignoring the common $\frac{1}{m}$ factor), *viz.*, the **empirical risk**

$$-\left[\sum_{i=1}^{m} \left(y^{(i)}\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - \log\left(1 + \exp\left(\mathbf{w}^T\mathbf{x}^{(i)}\right)\right)\right)\right] \tag{16}$$

   Since the **empirical risk** in (16) is only a function of $h(\mathbf{x}^{(i)}) = \mathbf{w}^T\phi(\mathbf{x}^{(i)})$ for $i = [1..m]$, based on (15) we note that the value of the **empirical risk** in (16) will therefore be independent of $h^{\perp}$ and therefore one only needs to equivalently solve the following **empirical risk** by substituting from (15) *i.e.*, $h(\mathbf{x}^{(j)}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}^{(i)},\mathbf{x}^{(j)})$:

$$\left[\sum_{\mathbf{i=1}}^{\mathbf{m}} \left(\sum_{\mathbf{j=1}}^{\mathbf{m}} -\mathbf{y}^{(\mathbf{i})}\mathbf{K}\left(\mathbf{x}^{(\mathbf{i})},\mathbf{x}^{(\mathbf{j})}\right)\alpha_{\mathbf{j}}\right) + \log\left(\mathbf{1} + \sum_{\mathbf{j=1}}^{\mathbf{m}} \alpha_{\mathbf{j}}\mathbf{K}\left(\mathbf{x}^{(\mathbf{i})},\mathbf{x}^{(\mathbf{j})}\right)\right)\right]$$

9. **Safe with Regularizer?**

   Consider the regularizer function $||\mathbf{w}||_2^2$ which is a strictly monotonically increasing function of $||\mathbf{w}||$. Substituting $\mathbf{w} = \frac{1}{\lambda}\left[\sum_{i=1}^{m}\left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\phi(\mathbf{x}^{(i)})\right]$ from (7), one can view $\Omega(||h||)$ as a strictly monotonic function of $||h||$.

$$\Omega(||h||) = \Omega\left(||\sum_{i=1}^{m} \alpha_i K(.,\mathbf{x}^{(i)}) + h^{\perp}||\right) = \Omega\left(\sqrt{||\sum_{i=1}^{m} \alpha_i K(.,\mathbf{x}^{(i)})||^2 + ||h^{\perp}||^2}\right)$$

and therefore,

$$\Omega(||h||) = \Omega\left(\sqrt{||\sum_{i=1}^{m}\alpha_i K(.,\mathbf{x}^{(i)})||^2 + ||h^{\perp}||^2}\right) \geq \Omega\left(\sqrt{||\sum_{i=1}^{m}\alpha_i K(.,\mathbf{x}^{(i)})||^2}\right)$$

That is, setting $h^{\perp} = 0$ does not affect the first term of (8) while strictly increasing the second term. That is, any minimizer must have optimal $h^*(.)$ with $h^{\perp} = 0$. That is,

$$h(\mathbf{x}) = \sum_{i=1}^{m}\alpha_i K(\mathbf{x}^{(i)},\mathbf{x})$$

Problem 4. In the class, we discussed the probabilistic binary (class) logistic regression classifier. How will you extend logistic regression probabilistic model to multiple (say K) classes? Are their different ways of extending? What is the intuition behind each? Discuss and contrast advantages/disadvantages in each.

Solution: One might suggest handling multi-class (K) classification via K one-vs-rest probabilistic classifiers. But there is no obvious probabilistic semantics associated with such a classifier (question asked for a probabilistic MODEL for multiple classes).

Basic idea is that each class c can have a different weight vector $[w_{c,1}, w_{c,2}, \ldots, w_{c,k}, \ldots, w_{c,K}]$
.

Extension to multi-class logistic

1. Each class $c = 1, 2, \ldots, K-1$ can have a different weight vector $[\mathbf{w}_{c,1}, \mathbf{w}_{c,2}, \ldots, \mathbf{w}_{c,k}, \ldots, \mathbf{w}_{c,K-1}]$ and

$$p(Y = c|\phi(\mathbf{x})) = \frac{e^{-(\mathbf{w}_c)^T\phi(\mathbf{x})}}{1 + \sum_{k=1}^{K-1} e^{-(\mathbf{w}_k)^T\phi(\mathbf{x})}}$$

for $c = 1, \ldots, K-1$ so that

$$p(Y = K|\phi(\mathbf{x})) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{-(\mathbf{w}_k)^T\phi(\mathbf{x})}}$$

Alternative (equivalent) extension to multi-class logistic

1. Each class $c = 1, 2, \ldots, K$ can have a different weight vector $[\mathbf{w}_{c,1}, \mathbf{w}_{c,2} \ldots \mathbf{w}_{c,p}]$ and

$$p(Y = c|\phi(\mathbf{x})) = \frac{e^{-(\mathbf{w}_c)^T\phi(\mathbf{x})}}{\sum_{k=1}^{K} e^{-(\mathbf{w}_k)^T\phi(\mathbf{x})}}$$

for $c = 1, \ldots, K$.

This function is also the called the **softmax**[5] function.

[5]https://en.wikipedia.org/wiki/Softmax_function

Problem 5. Suppose you are provided a multi-layer neural network for a binary classification problem. Can you convert this network into an equivalent kernel perceptron (single node neural network only)? What will be the exact steps?

Solution:

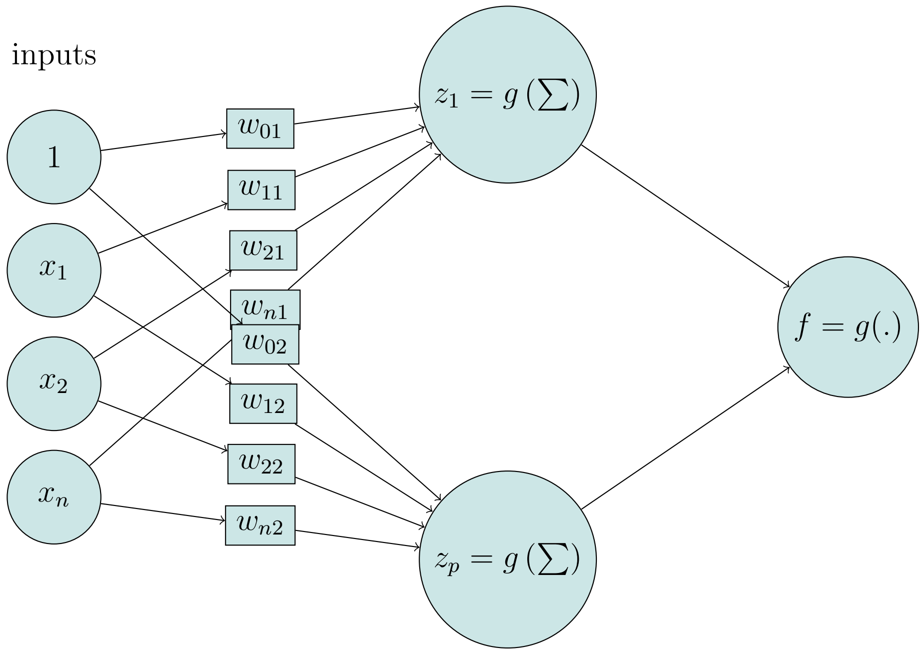
1. Yes. You are already provided the NN classifier. So you need not worry about separability etc. In fact, the question was only about an equivalent classifier (even if not-separating completely) which can be constructed as follows:

2. Let the neural network be as shown below with p nodes $(z_1, z_2, \dots, z_p)$ in the last layer. You can develop an equivalent kernel preceptron by specifying

$$\phi_z(x) = [z_1(x), z_2(x), \dots, z_p(x)]$$

and using the following Kernel:

$$K_z(x, x') = \phi_z^T(x)\phi_z(x')$$



inputs
1
x_1
x_2
x_n
w_{01}
w_{11}
w_{21}
w_{n1}
w_{02}
w_{12}
w_{22}
w_{n2}
$z_1 = g\left(\sum\right)$
$z_p = g\left(\sum\right)$
$f = g(.)$