Forward and backward propagation in simple nets: `util/simpleNet`

Carl Edward Rasmussen

January 15th 1996

"feval.c" 2 ≡

```
#include <math.h>
#include "util.h"
#include <stdlib.h>

extern int no_hid, no_inp, no_tar, no_wts;
extern real *w, **wi, **wd, **wo, *hidden, *output;
extern real prior(real *dw);

void net(real *in, real *out)          /* compute network output given input */
{
  int  i, j;
  real x, *tmp;

  for (i=0; i<no_hid; i++) {             /* compute activity of hidden units */
    tmp = wi[i];
    x = tmp[no_inp];                                         /* hidden bias */
    for (j=0; j<no_inp; j++) x += in[j]*tmp[j];
    hidden[i] = tanh(x);
  }

  for (i=0; i<no_tar; i++) {           /* compute activities of output units */
    tmp = wo[i];
    x = tmp[no_hid];                                         /* output bias */
    for (j=0; j<no_hid; j++) x += hidden[j]*tmp[j];
    tmp = wd[i];
    for (j=0; j<no_inp; j++) x += in[j]*tmp[j];           /* direct weights */
    out[i] = x;
  }
}

real cost(ex, wc)     /* calc. error on examples, including penalty iff wc=1 */
  struct exampleset ex;
  int    wc;
{
  int  i, j;
  real x, *tmp, e = 0.0;

  for (i=0; i<ex.num; i++) {
    net(ex.inp[i], output);
    tmp = ex.tar[i];
    for (j=0; j<no_tar; j++) {
      x = output[j] - tmp[j];
      e += x*x;
    }
  }
  x = 0.0; if (wc == 1) x = prior(NULL);
  return (e+x)/ex.num;
}
```
◇

"fgeval.c" 3 ≡

```c
#include <math.h>
#include "util.h"

extern int no_inp, no_hid, no_tar, no_wts, nfgeval;
extern real *w, **wi, **wd, **wo, *output;
extern struct exampleset train;

real *hidden;                                   /* activity of hidden units */

extern real prior(real *dw);

real fgeval(dw)
  real *dw;
{
  int  i, j, k, l, m;
  real e = 0.0, x, y, *tmp, *pat;

  for (i=0; i<no_wts; i++) dw[i] = 0.0;                 /* zero derivatives */
  for (k=0; k<train.num; k++) {
    pat=train.inp[k];
    for (i=0; i<no_hid; i++) {                  /* compute hidden activities */
      tmp = wi[i];
      x = tmp[no_inp];                                        /* hidden bias */
      for (j=0; j<no_inp; j++)
        x += pat[j]*tmp[j];
      hidden[i] = tanh(x);
    }
    m = 0;
    for (i=0; i<no_tar; i++) {                  /* compute output activities */
      tmp = wo[i];
      x = tmp[no_hid]-train.tar[k][i];           /* output bias minus target */
      for (j=0; j<no_hid; j++) x += hidden[j]*tmp[j];
      tmp = wd[i];
      for (j=0; j<no_inp; j++) x += pat[j]*tmp[j];         /* direct weights */
      output[i] = x;
      e += x*x;                                  /* accumulate squared error */
      for (j=0; j<=no_hid; j++) dw[m++] += x*hidden[j];
      for (j=0; j<no_inp; j++) dw[m++] += x*pat[j];
    }
    for (j=0; j<no_hid; j++) {
      y=hidden[j];
      x = 0.0; for (i=0; i<no_tar; i++) x += output[i]*wo[i][j];
      y=x*(1.0-y*y);
      for (l=0; l<no_inp; l++)
        dw[m++] += y*pat[l];
      dw[m++] += y;
    }
  }
  nfgeval++;
  return 0.5*(e+prior(dw));
}
```

◇