Solutions to exercises: week 2

(a) To prove that $e^{-x} \geq (1-x)$ it is good to first make a sketch. You will see that only for $x = 0$ we have equality. So first show that only for $x = 0$ we have equality. Now you can derive that the derivative of e^{-x} is always larger (i.e. less negative) than the derivative of $1 - x$ for $x > 0$, which means that for $x > 0$ will always hold $e^{-x} \geq (1-x)$. For $x < 0$ the opposite holds (the derivative of e^{-x} is more negative). Now because if only for $x = 0$ they are equal, then for $x < 0$ we need to have that $e^{-x} \geq (1-x)$.

Exercise 2.2
(a) To show the equivalence I will express the variables of the paper in terms of the variables of the slides. When the same variable is used (like y_i) I will use a tilde to indicate variables from the paper.

1. Trivially, the indices are different: $k, K \leftarrow t, T$
2. The labels are $\{0, 1\}$ instead of ± 1: $\tilde{y}_i = \frac{1}{2}(y_i + 1)$. This should also hold for the weak learners: $h_t(\mathbf{x}) = \frac{1}{2}(f_k(\mathbf{x}) + 1)$.
3. The error: $\tilde{\varepsilon}_t = \sum_i ((\tilde{w}_i^t / \sum_j \tilde{w}_j^t) |\frac{1}{2} f_k(\mathbf{x}_i) - \frac{1}{2} y_i| = \sum_i ((c \cdot w_i^k / \sum_j c \cdot w_j^k) \mathcal{I}(f_k(\mathbf{x}_i) \neq y_i) = \frac{\varepsilon_k}{\sum_j w_j^k}$ where I defined $\tilde{w}^t = c^k \cdot w^k$.
4. Now it becomes more interesting. To rewrite β, it is actually easier to take the logarithm: $\log \beta_t = \log(\frac{\varepsilon_k / \sum_j w_j}{1 - \varepsilon_k / \sum_j w_j}) = \log(\frac{\varepsilon_k}{\sum_j w_j - \varepsilon_k}) = -\log(\frac{\sum_j w_j - \varepsilon_k}{\varepsilon_k}) = -2\alpha_k$, so therefore $\beta_t = \exp(-2\alpha_k)$.
5. To see that the weight update in the article (step 5) is equivalent as in the slides (step 4), note that $1 - |h_t(\mathbf{x}_i) - y_i| = -y_i \alpha_k f_k(\mathbf{x}_i) - \alpha_k$. Note that the article added weak learner $t+1$, while in the slides weak learner K is added. Then use that $\beta^c = (\exp(-2\alpha))^c = \exp(-2c\alpha)$, and that $\exp(\sum_i c_i) = \prod_i \exp(c_i)$, and see that these update rules are equivalent upto a factor $\exp(\alpha_k)$: $\tilde{w}_i^{t+1} = \tilde{w}_i^t \exp(-y_i \alpha_K f_K(\mathbf{x}_i) - \alpha_K) = \prod_k \exp(-y_i \alpha_k f_k(\mathbf{x}_i)) \exp(-\alpha_k)$. It seems that the constant c^k in step 3. was actually $c^k = \exp(-\alpha_k)$.

Exercise 2.3
(a) My implementation is quite brute-force. The decision stump looks like:

```
function pred = decstump(x,bestf,bestt,bests)

if (bests>0)
  pred = (x(:,bestf)>=bestt);
else
  pred = (x(:,bestf)<bestt);
end
```

and the exhaustive search looks like:

```
function [bestfeat,besttheta,bestsign] = learndecstump(X,y,w)
p = w/sum(w);

besterr = inf;
bestfeat = 0;
besttheta = 0;
bestsign = +1;

[N,dim] = size(X);
% labels +-1
y = 2*y-1;
Nplus = sum(p(y>0));
```

```
Nmin = sum(p(y<0));

for i=1:dim
   [sx,I] = sort(X(:,i));
   sy = y(I); % sorted labels
   wsy = p(I).*sy; % weighted
   cy = cumsum([Nmin; wsy]);
   [mine,I] = min(cy);
   if (mine<besterr)
      besterr = mine;
      bestfeat = i;
      if I==1
         besttheta = sx(1)-10*eps;
      elseif (I>N)
         besttheta = sx(end)+10*eps;
      else
         besttheta = mean(sx((I-1):I));
      end
      bestsign = +1;
   end
   cy = cumsum([Nplus; -wsy]);
   [mine,I] = min(cy);
   if (mine<besterr)
      besterr = mine;
      bestfeat = i;
      if I==1
         besttheta = sx(1)-10*eps;
      elseif (I>N)
         besttheta = sx(end)+10*eps;
      else
         besttheta = mean(sx((I-1):I));
      end
      bestsign = -1;
   end
end
```

Exercise 2.4
(a) You should find that the best feature is 1, with a threshold around 1. Rescaling feature 2 should not have an influence.

Exercise 2.5
(a) The decision stump gave me an apparent error of around 0.18, but a test error of 0.51.

Exercise 2.6
Something like: `function [bestf,bestt,bests,beta] = learnadaboost(x,y,T)`

```
N = size(x,1);
w = ones(N,1)/N;
bestf = zeros(T,1);
bestt = zeros(T,1);
bests = zeros(T,1);
err = zeros(T,1);

for t=1:T
   p = w/sum(w);
```

```
    [bestf(t),bestt(t),bests(t)] = learndecstump(x,y,p);
    pred = decstump(x,bestf(t),bestt(t),bests(t));
    df = abs(pred-y);
    err(t) = df'*p;
    if (err(t)==0)
        bestf = bestf(t,:);
        bestt = bestt(t,:);
        bests = bests(t,:);
        beta = 1/exp(1);
        break
    end
    beta(t) = err(t)/(1-err(t));
    w = w.*(beta(t).^(1-df));
end
```

Exercise 2.7

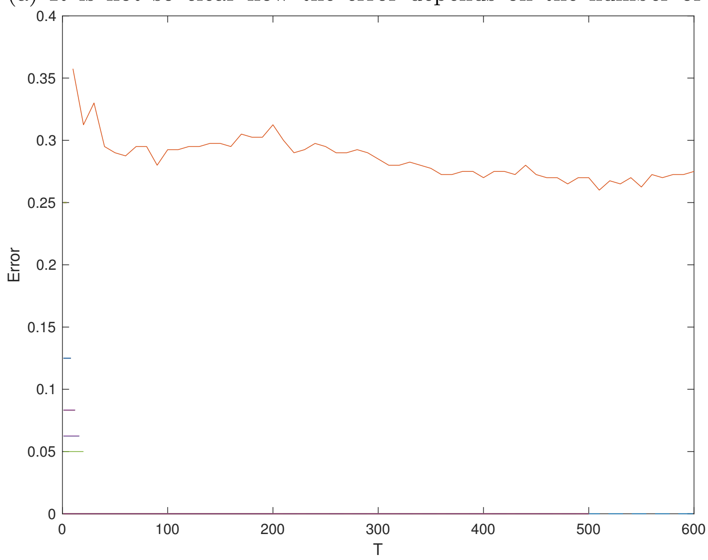
(a) When you test your implementation on a simple dataset, you should see that objects that are in a region of large class-overlap will obtain a high weight.

Exercise 2.8

(a) The classification error on the Fashion dataset is around 0.29, way better than a decision stump.

Exercise 2.9

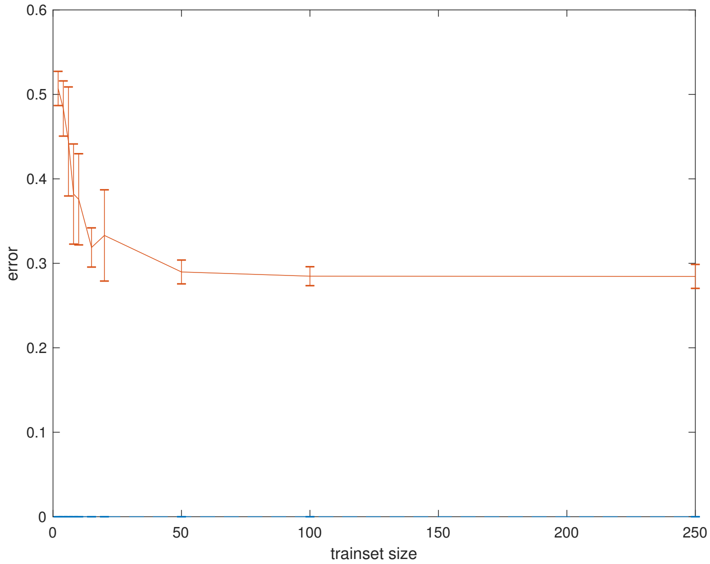
(a) It is not so clear how the error depends on the number of iterations T. I got this graph:



It seems that after already 300 decision stumps, not so much improves.

Exercise 2.10

(a) If you plot the learning curve for an Adaboost with `Tmax=100`:



you see that it converges for around $N = 100$. You also see that training errors are very small (basically 0 for `fashion57`!).