Homework 4

Jacob Sachs

3 May 2013

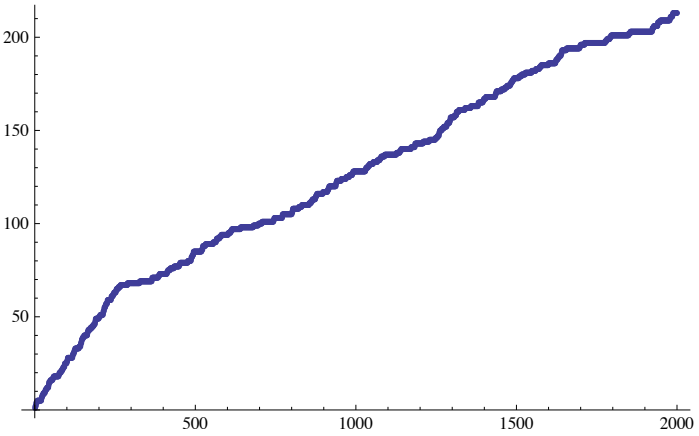
Mistakes vs Examples

Linear

In[21]:=
```
ldata = Import["/Users/Jacob/jsachs13-cs25010-spr-13/hw4/linear_online.txt", "Table"];
```

In[22]:=
```
ListPlot[ldata]
```

Out[22]=

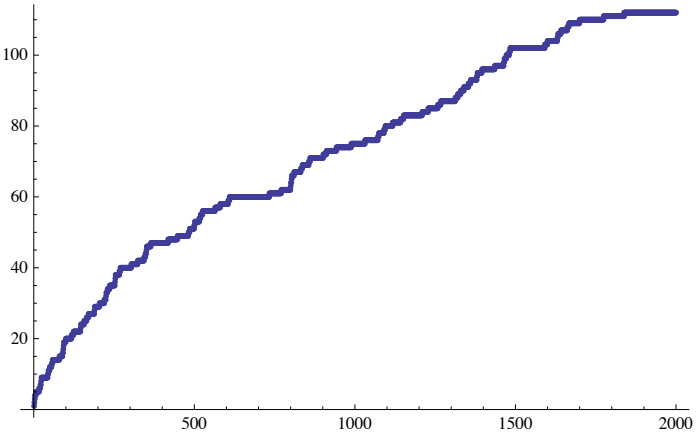


Kernel

In[23]:=
```
kdata = Import["/Users/Jacob/jsachs13-cs25010-spr-13/hw4/kernel_online.txt", "Table"];
```

In[25]:=
```
ListPlot[kdata]
```

Out[25]=



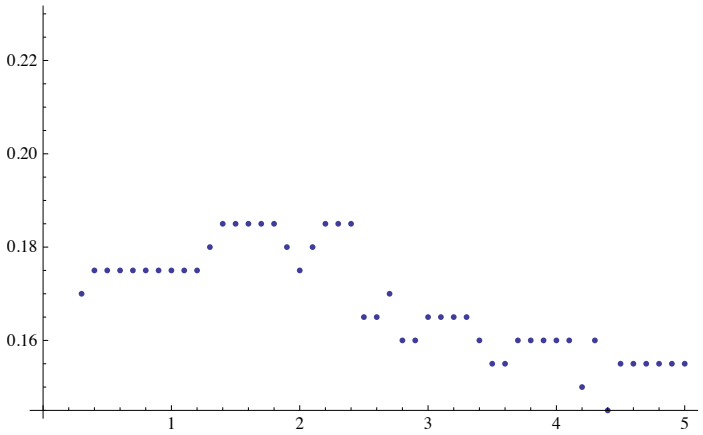
Cross-validation Error

- **On the first 200 values of the data set:**

In[19]:=
```
err = {{0.1, 0.5}, {0.2, 0.35}, {0.3, 0.17}, {0.4, 0.175}, {0.5, 0.175},
    {0.6, 0.175}, {0.7, 0.175}, {0.8, 0.175}, {0.9, 0.175}, {1., 0.175},
    {1.1, 0.175}, {1.2, 0.175}, {1.3, 0.18}, {1.4, 0.185}, {1.5, 0.185},
    {1.6, 0.185}, {1.7, 0.185}, {1.8, 0.185}, {1.9, 0.18}, {2., 0.175}, {2.1, 0.18},
    {2.2, 0.185}, {2.3, 0.185}, {2.4, 0.185}, {2.5, 0.165}, {2.6, 0.165}, {2.7, 0.17},
    {2.8, 0.16}, {2.9, 0.16}, {3.0, 0.165}, {3.1, 0.165}, {3.2, 0.165}, {3.3, 0.165},
    {3.4, 0.16}, {3.5, 0.155}, {3.6, 0.155}, {3.7, 0.16}, {3.8, 0.16}, {3.9, 0.16},
    {4., 0.16}, {4.1, 0.16}, {4.2, 0.15}, {4.3, 0.16}, {4.4, 0.145}, {4.5, 0.155},
    {4.6, 0.155}, {4.7, 0.155}, {4.8, 0.155}, {4.9, 0.155}, {5., 0.155}};
```

In[20]:=
```
ListPlot[err]
```

Out[20]=



The best result is $\sigma = 4.4$

Design Choices

- **Simple Wieghts**

  In my auxiliary research on papers written on perceptrons, there was much mention of learning rates and modifying the changes in weights by the error encountered. However, I chose to follow more or less what is presented in the pseudocode in the slides. I'm aware that this is a naive approach, but during testing it was successful.

- **Number of Iterations**

  I chose to iterate until error was zero, or the iterations had reached 100. In my linear perceptron, the error would reach zero after 97 iterations, so 100 seemed a reasonable upper bound. In the kernel perceptron,

- **Memoization**

  After waiting many many minutes for my code to run, I decided to memoize the kernel inner products in a matrix. While the summations for each pass are still computationally expensive, this sped things up for the many passes required in the batch mode.