Homework 2
Kernel SVM and Perceptron

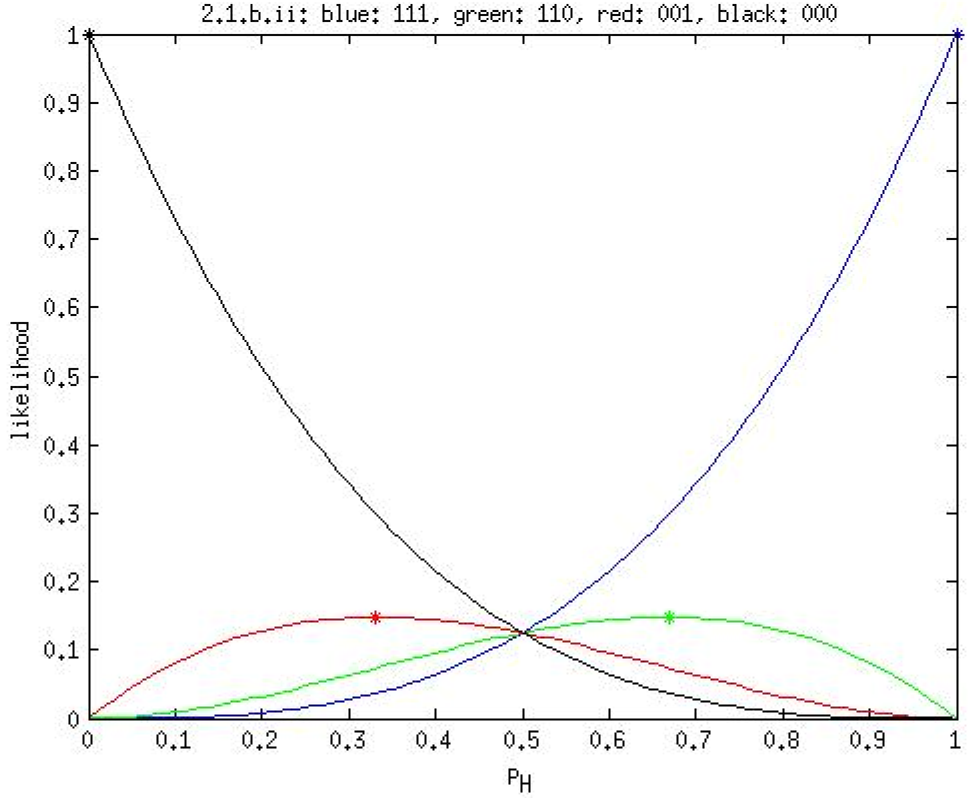
Dana Van Aken

Problem 2: Understanding the Likelihood Function, Bit by Bit

1. (a) $H_i \sim Bernoulli(p_H)$, $\qquad N_H \sim Binomial(N_{bits}, p_H)$.

   (b) i. L$(H_1, ..., H_{Nbits}; p_H) = \prod_{i=1}^{N_{bits}} P(H_i; p_H) = \prod_{i=1}^{N_H} p_H \prod_{i=1}^{N_{bits}-N_H}(1-p_H) = p_H^{N_H}(1-p_H)^{N_{bits}-N_H}$.

   ii. Log-likelihood vs. p_H



These estimates do make sense given the data. It is obvious that the p_H that maximizes the likelihood of getting 3 heads in a row (sequence [111]) is when $p_H = 1$. The intuition is the same for getting all tails (sequence [000]). The likelihood of getting sequence [110] out of 3 coin flips is maximized when the probability of getting heads is 2/3. Similarly, the maximum likelihood of flipping 2 tails out of 3 coin flips is when the probability of getting heads is equal to 1 out of 3 flips.

   iii. [000]: $\int_0^1 p_H^0(1-p_H)^3 dp_H = \int_0^1 (1-p_H)^3 dp_H = (-1)\frac{1}{4}(1-p_H)^4\Big|_0^1 = \frac{-1}{4}\big((1-1)^4-(1-0)^4\big) = \frac{1}{4}$

   [111]: $\int_0^1 p_H^3(1-p_H)^0 dp_H = \int_0^1 p_H^3 dp_H = \frac{1}{4}p_H^4\Big|_0^1 = \frac{1}{4}(1^4-0^4) = \frac{1}{4}$

[110]: $\int\limits_0^1 p_H^2(1-p_H)^1 dp_H = \int\limits_0^1 (p_H^2 - p_H^3)dp_H = (\frac{1}{3}p_H^3 - \frac{1}{4}p_H^4)\big|_0^1 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}$

[001]: $\int\limits_0^1 p_H^1(1-p_H)^2 dp_H = \int\limits_0^1 p_H(1-p_H^2)dp_H = \int\limits_0^1 p_H(1-2p_H+p_H^2)dp_H$

$= \int\limits_0^1 (p_H - 2p_H^2 + p_H^3)dp_H = (\frac{1}{2}p_H^2 - \frac{2}{3}p_H^3 + \frac{1}{4}p_H^4)\big|_0^1 = \frac{1}{2} - \frac{2}{3} + \frac{1}{4} = \frac{1}{12}$

You can tell that this is not a valid probability distribution over p_H because the total sum of the area under these curves is not equal to 1. The reason that it's invalid is because these 4 sequences are only a subset of the total possible sequences, (for example, we are missing [101], [100], etc.).

(c) We choose p_H that maximizes the probability of the observed sequence. We find this value of p_H by maximizing the likelihood function.

$\hat{p}_H = \underset{p_H}{\arg\max} P(H_1, ..., H_{N_{bits}}; p_H) = \underset{p_H}{\arg\max}\big(p_H^{N_H}(1-p_H)^{N_{bits}-N_H}\big)$

There are different techniques that can be used to find the value of p_H that maximizes the likelihood function (e.g. taking the derivative, gradient descent). To find this maximizing value, we can take the derivative of this likelihood function, set it equal to 0, and solve for p_H since a closed-form solution exists for this particular likelihood function.

Instead of maximizing the likelihood function, we will instead maximize the log-likelihood function which reaches its maximum value at the same points as the original function, (this is because the logaritm function is monotonically increasing).

$\hat{p}_H = \underset{p_H}{\arg\max}\ log\big(P(H_1, ..., H_{N_{bits}}; p_H)\big) = \underset{p_H}{\arg\max}\ log\big(\big(p_H^{N_H}(1-p_H)^{N_{bits}-N_H}\big)\big)$

$= \underset{p_H}{\arg\max}\ log(p_H^{N_H}) + log(1-p_H)^{N_{bits}-N_H} = \underset{p_H}{\arg\max}\ N_H log(p_H) + (N_{bits} - N_H) log(1-p_H)$

Now take the derivative of the log-likelihood function:

$\frac{d}{dp_H}\big(N_H log(p_H) + (N_{bits} - N_H) log(1-p_H)\big) = \frac{N_H}{p_H} - \frac{N_{bits}-N_H}{1-p_H}$

Set it equal to 0 and solve for p_H:

$\frac{N_H}{p_H} - \frac{N_{bits}-N_H}{1-p_H} = 0$

$\frac{N_H}{p_H} = \frac{N_{bits}-N_H}{1-p_H}$

$\frac{N_H(1-p_H)}{p_H} = N_{bits} - N_H$

$\frac{1-p_H}{p_H} = \frac{N_{bits}-N_H}{N_H}$

$\frac{1}{p_H} - 1 = \frac{N_{bits}}{N_H} - 1$

$p_H = \frac{N_H}{N_{bits}}$

This result shows that the maximizing value of p_H is the number of heads (or 1-bits) divided by the total number of coin tosses (or bits).

2. (a) $L(O_1, ..., O_{N_{bits}}; p_H) = P(O_1, ..., O_{N_{bits}}; p_H) = \prod_{i=1}^{N_{bits}} P(O_i; p_H)$

$= \prod_{i=1}^{N_{bits}} \sum_{t=0}^{1} P(O_i, H_i = t; p_H) = \prod_{i=1}^{N_{bits}} \sum_{t=0}^{1} P(O_i|H_i = t; p_H)P(H_i = t; p_H)$

$= \prod_{i=1}^{N_{bits}} P(O_i|H_i = 0; p_H)P(H_i = 0; p_H) + P(O_i|H_i = 1; p_H)P(H_i = 1; p_H)$

$= \prod_{i=1}^{N_{bits}} P(O_i|H_i = 0; p_H)(1 - p_H) + P(O_i|H_i = 1; p_H)p_H$

Given: $O_i = H_i + \epsilon_i$, $\epsilon i \sim N(0, \sigma^2)$

$H_i = t: \quad O_i = t + \epsilon_i \qquad$ Solve for $\epsilon_i: \qquad \epsilon_i = O_i - t$

$t = 0: \qquad N(O_i; 0, \sigma^2) \qquad t = 1: \qquad N(O_i - 1; 0, \sigma^2)$

$L(O_1, ..., O_{N_{bits}}; p_H) = \prod_{i=1}^{N_{bits}} = N(O_i; 0, \sigma^2)(1 - p_H) + N(O_i - 1; 0, \sigma^2)p_H$

Let $\alpha_i = N(O_i; 0, \sigma^2)$ and $\beta_i = N(O_i - 1; 0, \sigma^2)$

$L(O_1, ..., O_{N_{bits}}; p_H) = \prod_{i=1}^{N_{bits}} \alpha_i(1 - p_H) + \beta_i p_H$

$$L(O_1, ..., O_{N_{bits}}; p_H) = \prod_{i=1}^{N_{bits}} p_H(\beta_i - \alpha_i) + \alpha_i$$

$$LL(O_1, ..., O_{N_{bits}}; p_H) = \sum_{i=1}^{N_{bits}} log(p_H(\beta_i - \alpha_i) + \alpha_i)$$

Check concavity: check 2nd derivative

$\frac{dLL}{dp_H} = \sum_{i=1}^{N_{bits}} \frac{\beta_i - \alpha_i}{p_H(\beta_i - \alpha_i) + \alpha_i}$

$\frac{d^2LL}{dp_H^2} = \sum_{i=1}^{N_{bits}} -\left(\frac{\beta_i - \alpha_i}{p_H(\beta_i - \alpha_i) + \alpha_i}\right)^2$

The expression inside the parenthesis is always squared so it's guaranteed to be positive. The whole expression is then multiplied by -1 and we take the summation of all of these expressions. This shows that the second derivative is *always* negative for all values of p_H.

Concavity is important because it means there exists a value of some variable that minimizes/maximizes the function we are optimizing. These mins and maxs can be local or (hopefully) global.

(b) $\hat{p}_H = \arg\max_{p_H} \sum_{i=1}^{N_{bits}} log(p_H(\beta_i - \alpha_i) + \alpha_i)$

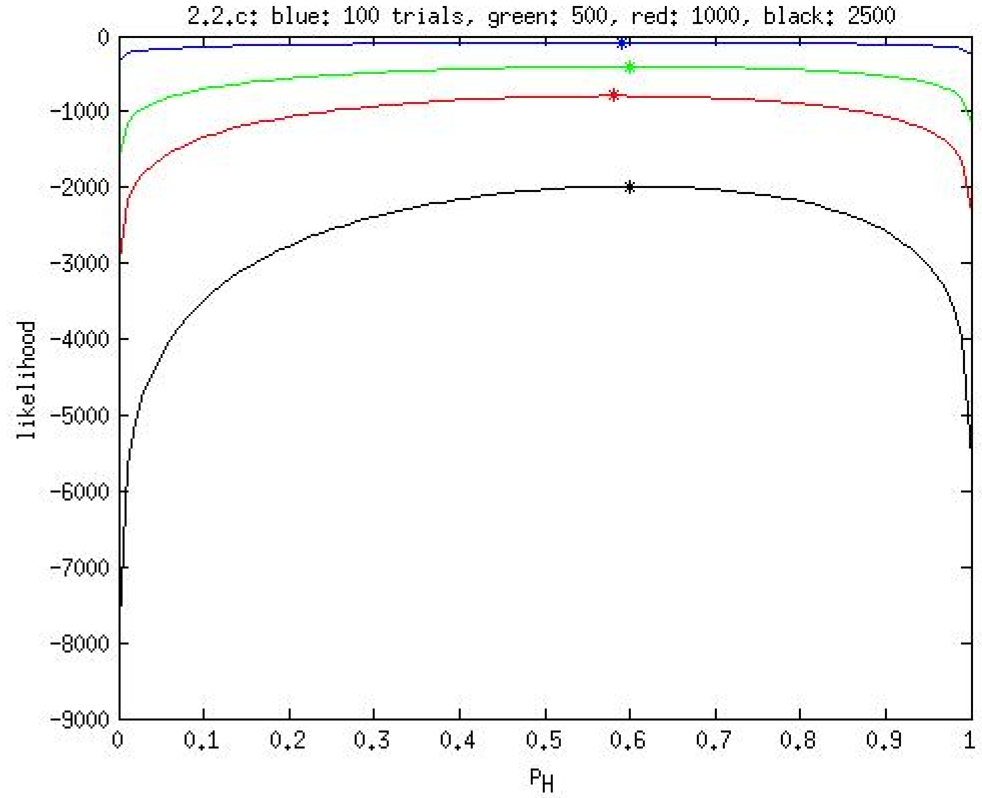
There is no closed-form solution to this equation. We can use iterative convergence methods, like gradient descent, to obtain the MLE.

(c) Log-likelihood vs p_H

When we increase σ^2, the noise deviates more from the mean which means we get more bit errors. This causes the curves on the graph to flatten out since other p_Hs become more likely and the likelihood of $\hat{p}_H$ decreases.

(d) $\hat{p}_H$, $\bar{p}_H$ vs Trial Size (see graph 2.2.d)

$\bar{p}_H$ is the ratio of the observed number of 1-bits to the total number of bits. As σ^2 approaches 0, $\hat{p}_H$ is approximately equal to $\bar{p}_H$ because the noise does not deviate far from 0 (the mean) so the bits are unlikely to be corrupted. When the variance is 0.1, $\bar{p}_H$ and $\hat{p}_h$ still perform similarly, however, $\hat{p}_H$ performs slightly better for trial sizes greater than 500 (this can be seen on graph 2.2.d). As we increase σ^2 further, the accuracy of $\hat{p}_H$ compared with $\bar{p}_H$ increases as well *as long as the trial size is sufficiently large.*



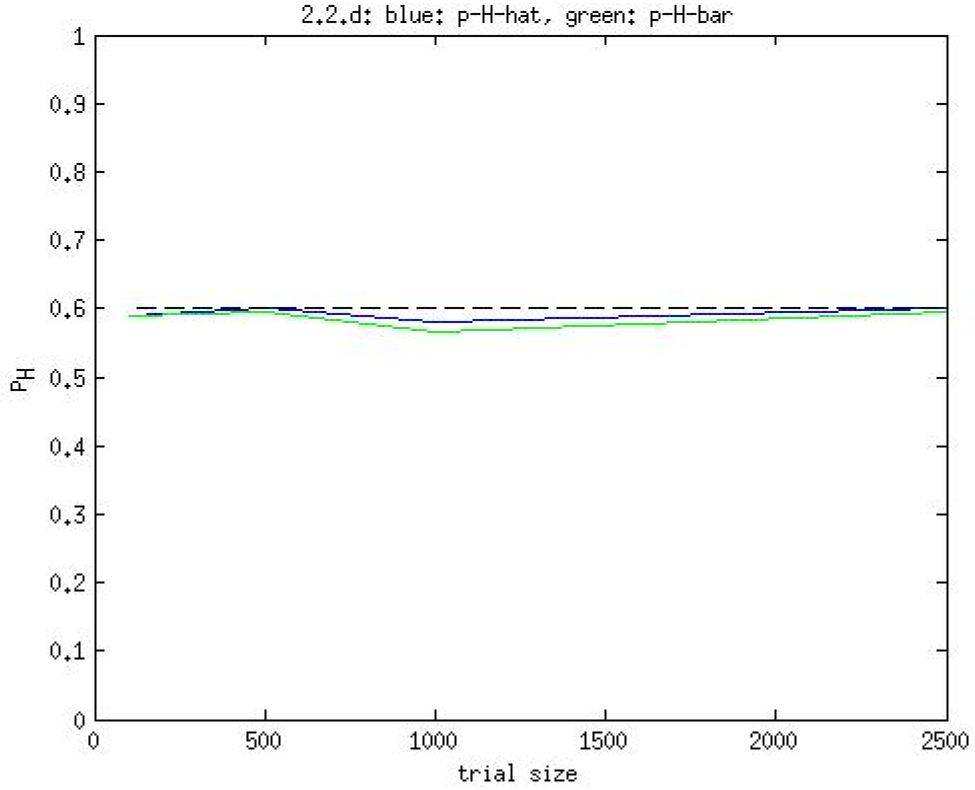
3. (a) $LL(O_1, ..., O_{N_{bits}}; p_H) = \sum_{i=1}^{N_{bits}} log(p_H(\beta_i - \alpha_i) + \alpha_i)$

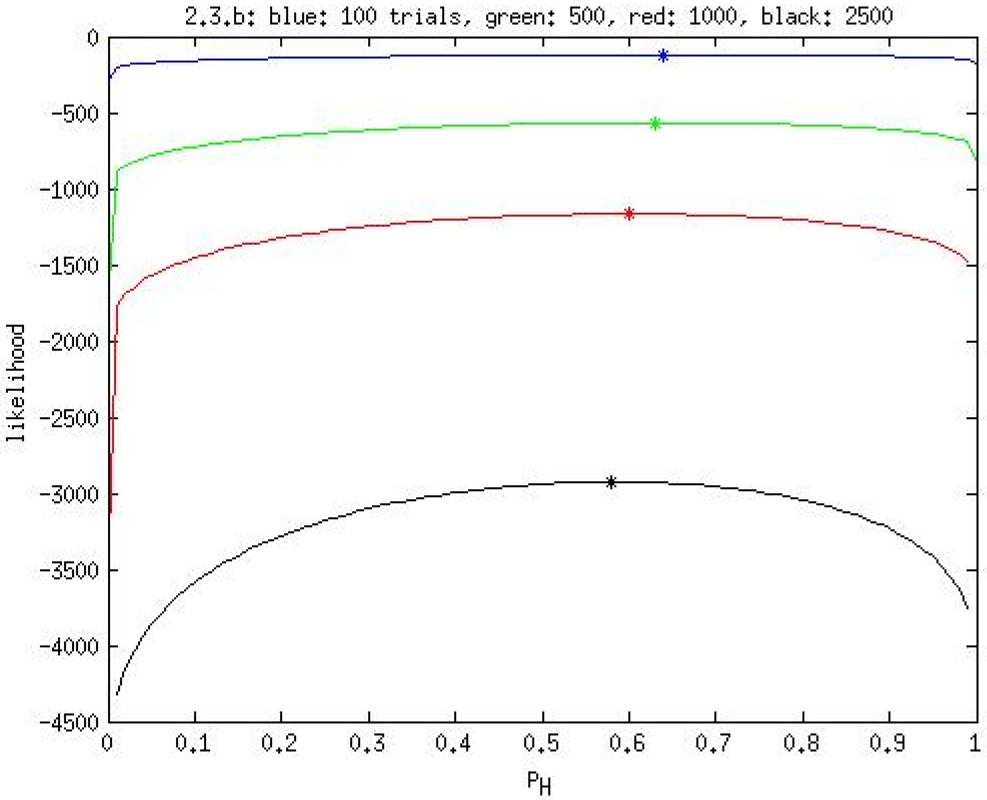
   Where: $\alpha_i \sim N(O_i; 0, \sigma_i^2)$ $\beta_i \sim N(O_i - 1; 0, \sigma_i^2)$

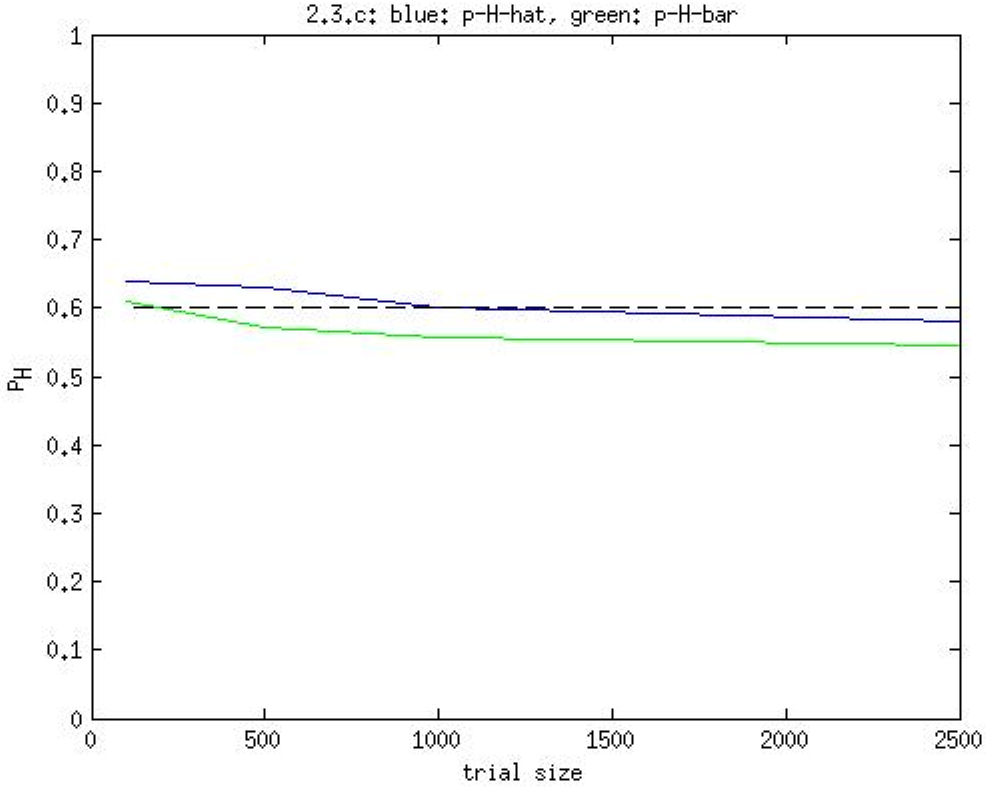
   Changing the model to include a sigma that grows with consecutive bits is independent of p_H and does not change the first or second derivative with respect to p_H (see part 2a). Therefore, the log-likelihood function is still concave.

   (b) Log-likelihood vs p_H (see graph 2.3.b)

   (c) $\hat{p}_H$, $\bar{p}_H$ vs Trial Size (see graph 2.3.c)
   $\bar{p}_H$ does better until the trial size is at least 500. After this point, $\hat{p}_H$ performs better but also starts moving away from p_H^* as the trial size increases past around 1500.


2.2.d: blue: p-H-hat, green: p-H-bar
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
p_H
0
500
1000
1500
2000
2500
trial size


2.3.b: blue: 100 trials, green: 500, red: 1000, black: 2500
0
-500
-1000
-1500
-2000
-2500
-3000
-3500
-4000
-4500
likelihood
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
p_H


2.3.c: blue: p-H-hat, green: p-H-bar
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
p_H
0
500
1000
1500
2000
2500
trial size