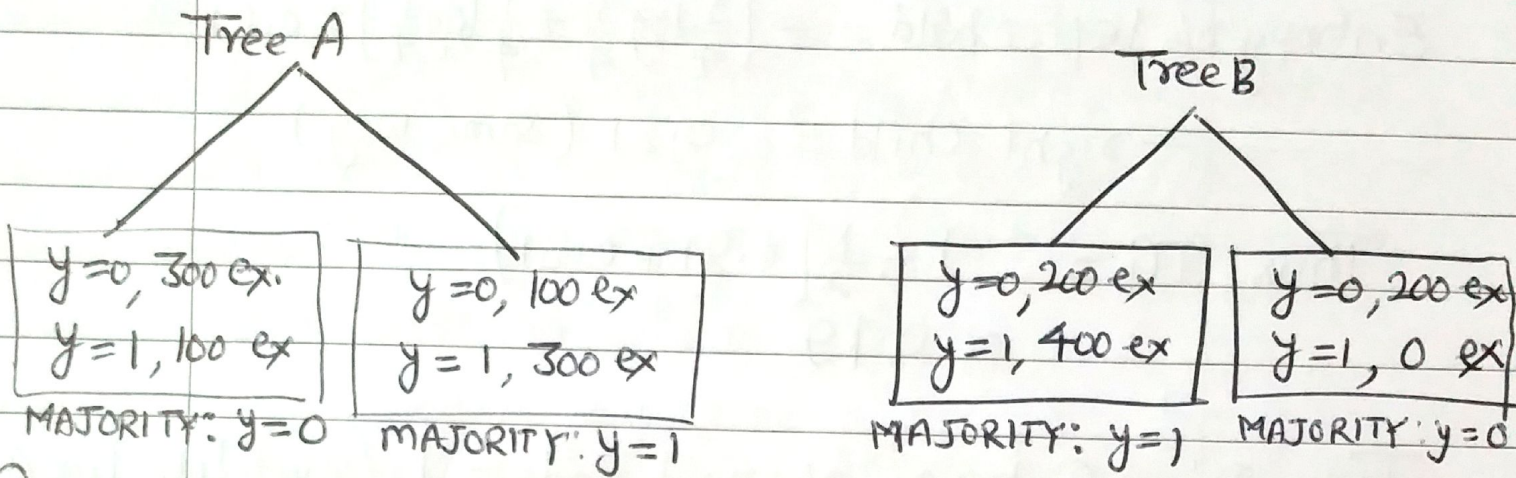
Homework 1 (Problem 1)



(1) Tree A: 200 misclassifications (100 ex. with $y=1$ are misclassified in left branch, and 100 ex. with $y=0$ are misclassified in right branch)

$$\text{Tree A misclassification rate} = \frac{200}{800} = 0.25$$

Tree B: 200 misclassifications (200 ex. with $y=0$ are misclassified in left branch, and none of the examples are misclassified in the right branch)

$$\text{Tree B misclassification rate} = \frac{200}{800} = 0.25$$

So the misclassification rate is equal for both.

(2) Tree A: Entropy of root node $= -\left[\frac{400}{800}\log\frac{400}{800} + \frac{400}{800}\log\frac{400}{800}\right]$

$= 1$

Entropy of left child $= -\left[\frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4}\right] = 0.81$

——— right child = 0.81 (same way)

Thus IG $= 1 - \frac{1}{2}[0.81 + 0.81]$

≈ 0.19

Tree B: Entropy of root node = 1 (just like Tree A)

Entropy of left child $= -\left[\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}\right] \approx 0.92$

——— right child = 0 (pure node, no ex. with $y=1$)

Thus IG $= 1 - \left[\frac{3}{4} \times 0.92 + \frac{1}{4} \times 0\right] \approx 0.31$

Thus Tree B has a higher IG.

(3) Indeed, the answers are different. As per entropy/IG criterion, Tree B is preferred.

It makes sense because IG based criterion is sensitive to node probabilities as opposed to misclassificate rate which is only based on a "hard" classification as 0/1.

Problem 2

In the limit of infinite data, and since we have a noise-free setting, each input location has a training example with its correct label.

Since the training set is infinite, a new test input will fall exactly on one of the training inputs and will get the same label (since we are using a one-nearest neighbor).

Problem 3

$$f(x_*) = x_*^T w$$

$$= x_*^T (X^T X)^{-1} X^T y$$

$$= x_*^T (X^T X)^{-1} \sum_{n=1}^{N} x_n y_n$$

$$= \sum_{n=1}^{N} x_*^T \Sigma^{-1} x_n y_n$$

$$= \sum_{n=1}^{N} w_n y_n$$

Note the form of each w_n:

$$w_n = x_*^T \vec{\Sigma} x_n$$

This is a general form of an inner product similarity between the test input x_* and training input x_n, modulated by the matrix $\Sigma = X^T X$, which depends on the entire training data. In contrast, in the weighted version of KNN, the similarities only depend on the test input and the nearest neighbor (e.g. $w_n = x_*^T x_n$ or $w_n = \frac{1}{\|x_* - x_n\|}$).

Problem 4

To have a different amount of regularization on each component of w, we can modify the regularizer (l_2) from $\lambda w^T w = \sum_{d=1}^{D} \lambda w_d^2$ to $\sum_{d=1}^{D} \lambda_d w_d^2$ where λ_d denotes the regularization hyperparameter for component w_d of the weight vector w.

Note that we can also write it in terms of vector and matrix as follows:

$\sum_{d=1}^{D} \lambda_d w_d^2 = w^T \Lambda w$ where Λ is a diagonal matrix of size $D \times D$ with $\{\lambda_d\}_{d=1}^{D}$ along its diagonals.

The full objective will be:

$$\mathcal{L}(w) = \sum_{n=1}^{N} (y_n - w^T x_n)^2 + w^T \Lambda w$$

Solving this for w is almost identical to the way we have seen in the class, so I'll skip the full derivation. The solution will be

$$\hat{w} = (X^T X + \Lambda)^{-1} X^T y$$

(An aside): Probabilistic Version: Just as $N(0, \lambda^{-1} I)$ prior on w corresponds to an l_2 regularizer, changing the prior to $N(0, \Lambda^{-1})$ will correspond to the above regularizer with component-wise regularization

Problem 5

$$\mathcal{L}(W) = \text{trace}\left[(Y-XW)^T(Y-XW)\right]$$

with $W = BS$, the objective will be

$$\mathcal{L}(B,S) = \text{trace}\left[(Y-XBS)^T(Y-XBS)\right]$$

If B is known, the objective is convex in S (note that it is just like a quadratic function of S)

To get the optimal S, we take the derivative of $\mathcal{L}(S)$ with respect to S (note that since B is assumed known, I wrote $\mathcal{L}(B,S)$ as (just $\mathcal{L}(S)$) and set it to zero. Just like the linear regression case, it is possible to separate S and get a closed form expression.

$$\frac{\partial}{\partial S}\mathcal{L}(S) = 0$$

$$\Rightarrow \frac{\partial}{\partial S}\text{trace}\left[(Y-\tilde{X}S)^T(Y-\tilde{X}S)\right] = 0$$

(where I have denoted $\tilde{X} = XB$)

For the above derivative, you can either directly use the expression of derivative of trace(quadratic form) from matrix

cookbook, or expand it as

$$\text{trace}\left[(Y-\tilde{X}S)^T(Y-\tilde{X}S)\right]$$

$$= \text{trace}\left[(Y^T - S^T\tilde{X}^T)(Y-\tilde{X}S)\right]$$

multiply the terms and then take derivatives w.r.t. S.

In either case, the final expression will turn out to be (skipping the equations of derivative, since we already did it in PS-1 and also on Piazza).

$$\hat{S} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T Y \quad (\text{with } \tilde{X} = XB)$$

which is exactly in the same form as the solution of standard multi-output regression (PS-1) with X replaced by $\tilde{X} = XB$ (it is like reducing the dimensions of X from D to K by multiplying with B)

Bonus Problem (not for credit): In this case, we need to solve for both B and S. We can use alternating optimization (assume B fixed, solve for S; assume S fixed, solve for B).

Expression for S, given B will be the same as above. Expression for B, given S will be

$$\hat{B} = (X^TX)^{-1}X^TYS^T(SS^T)^{-1}$$

__/__/____

Another "shorthand" to find the solution:

As per the multioutput regression model

$$XW = Y$$
$$XBS = Y$$
$$X^TXBS = X^TY$$
$$B^TX^TXBS = B^TX^TY$$

(B^TX^TXB: square matrix now; multiplied by B^T so that we can invert in the next step)

$$S = (B^TX^TXB)^{-1}B^TX^TY$$
$$S = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^TY$$

We can do the same for B (assuming S known) starting with

$$X^TXBS = X^TY$$
$$X^TXBSS^T = X^TYS^T$$

premultiply by $(X^TX)^{-1}$ and postmultiply by $(SS^T)^{-1}$

$$B = (X^TX)^{-1}X^TYS^T(SS^T)^{-1}$$

Note that in this approach, we didn't solve the optimization problem of minimizing the trace term, but rather solved it as linear system of equations (they are equivalent).