Homework 1

(1) Nearest Neighbors & Curse of Dimensionality

a) The input data lies in $\mathbb{R}^1$; i.e, $d = 1$. For a new test point, we need there to be at least one data point in its vicinity; i.e within 0.01 units. Since the total space is $[0, 1]$, and we need one data point ± 0.01 away from any test point — i.e at least one point in a 0.02 wide interval — then the total number of data points required is $\frac{1}{0.02} = 50$. However, in order to take into account the end of the interval (the literal "edge case"), we need an extra point.

Eg for $\varepsilon = 0.1$

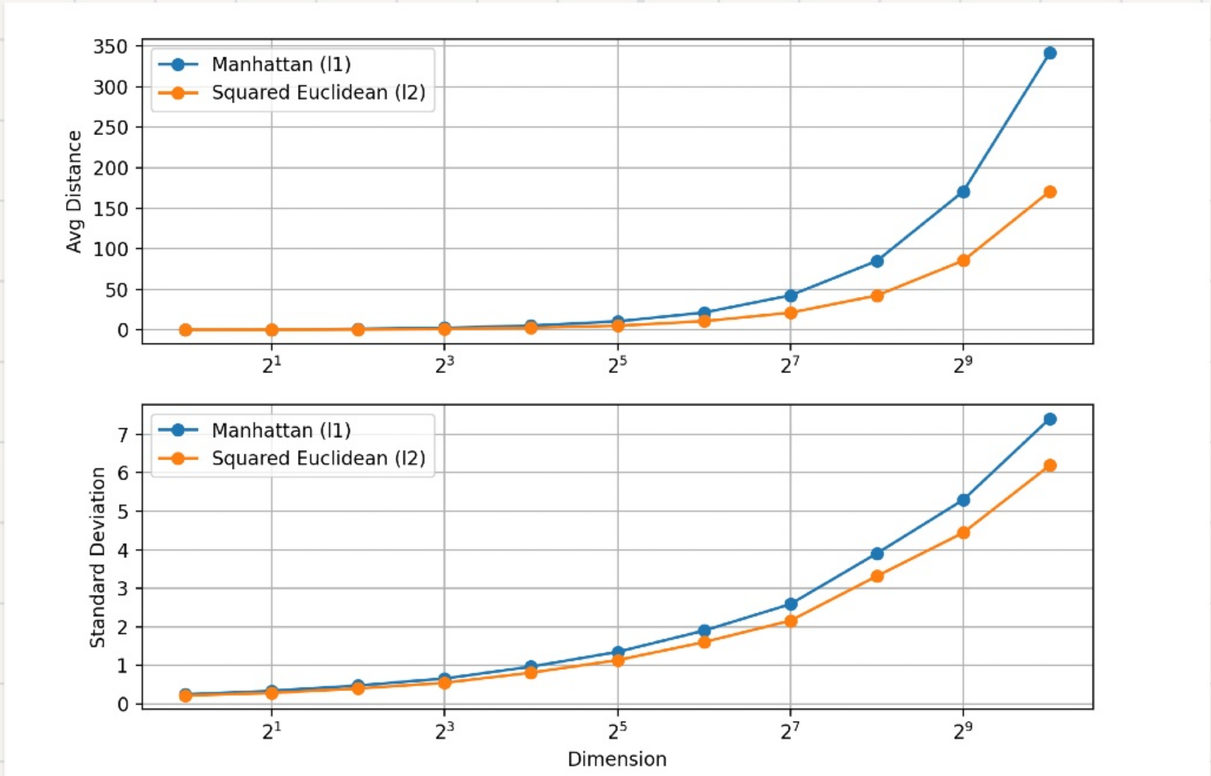
$n = \frac{1}{0.1} = 10$

Homework 1

(1) Nearest Neighbors & Curse of Dimensionality

b) The number of data points required to guarantee a maximum spacing of ε (in this case $\varepsilon = 0.01$) is proportional to $\frac{1}{\varepsilon^d} = \frac{1}{0.01^d}$. Thus, achieving this in low dimensions is relatively easy, since not many data points are required. However, in high-dimensions, where a single data point contains many (eg. 10) features, the number of data points required to satisfy this minimum spacing grows exponentially.

c)



Homework 1

(1) Nearest Neighbors & Curse of Dimensionality

d) Let the squared Euclidean distance in d dimensions be $R = Z_1 + \cdots + Z_d$, with each $Z_i = (X_i - Y_i)^2$. Given that

$$E[Z_i] = \frac{1}{3} \qquad Var[Z_i] = \frac{20}{23}$$

i) $E[R] = E[Z_1 + \cdots + Z_d] = E[Z_1] + \cdots + E[Z_d] = \left(\frac{1}{3}\right) d = \frac{d}{3}$

ii) $Var[R] = Var[Z_1 + \cdots + Z_d] = Var[Z_1] + \cdots + Var[Z_d] = \frac{20}{23} d$

Homework 1

(1) Nearest Neighbors & Curse of Dimensionality

e) $\mathbb{P}(|Z - \mathbb{E}[Z]| \geq a) \leq \frac{\text{Var}[Z]}{a^2}$ for any random variable Z.

i) $|R - E[R]| \geq r$

ii) $\mathbb{P}(E) = \mathbb{P}\left(|R - E[R]| \geq r\right) \leq \frac{\text{Var}[R]}{r^2}$ $\quad \therefore \mathbb{P}(E) \leq \frac{\text{Var}[R]}{r^2}$

iii) $r = cd \Rightarrow \mathbb{P}(E) \leq \frac{\text{Var}[R]}{c^2 d^2} = \frac{20d}{23} \cdot \frac{1}{c^2 d^2} = \left(\frac{20}{23c^2}\right)\frac{1}{d}$

So $\lim_{d \to \infty} \mathbb{P}(E) = \lim_{d \to \infty} \left(\frac{20}{23c^2} \cdot \frac{1}{d}\right) = 0$

So, at high dimensions, the distance between two randomly drawn points tends to be close to the average separation of all the points in the space. This implies that at high dimensions, points are equidistant from each other, with this distance being the mean.

Homework 1

N input vectors
p dimensions
↓

(2) Learning an Embedding for Nearest Neighbors

- Want to learn a matrix $A \in \mathbb{R}^{d \times p}$ such that the embedded data matrix $XA \in \mathbb{R}^{N \times p}$ is well-classified under the standard kNN algorithm

a) i) We must apply A to $(x_1 - x_2)$ and then take the inner product, so $d_A(x_1, x_2)^2 = \|(x_1 - x_2)A\|_2^2 = (x_1 - x_2)AA^T(x_1 - x_2)^T$. where $x_1, x_2 \in \mathbb{R}^{1 \times d}$ and $A \in \mathbb{R}^{d \times p}$.

ii) a) For any real matrix B, $M = BB^T$ is always positive semi-definite, meaning for any vector v, $v^T M v \geq 0$. It follows that

$d_A(x_1, x_2)^2 = (x_1 - x_2)AA^T(x_1 - x_2)^T \geq 0$, so

$d_A(x_1, x_2) = \sqrt{(x_1 - x_2)AA^T(x_1 - x_2)^T} \geq 0$ since $\sqrt{x} \geq 0 \quad \forall x \geq 0$.

Homework 1

(2) Learning an Embedding for Nearest Neighbors

ii) b) $d_A(x_1,x_2) = \|(x_1-x_2)A\|_2 = \|-(x_1-x_2)A\|_2 = \|(x_2-x_1)A\|_2 = d_A(x_2,x_1)$

because $\|-v\|_2 = \|v\|_2$ for any vector v.

c) Let $u = x_3-x_2$, $v = x_2-x_1$. We want to show that

$$d_A(x_1,x_3) \leq d_A(x_1,x_2) + d_A(x_2,x_3)$$

· Note that $u+v = x_3-x_1$. Therefore we may write

$$d_A(x_3,x_1)^2 = \|(x_3-x_1)A\|_2^2 = \|(u+v)A\|_2^2 = \langle (u+v)A, (u+v)A \rangle$$

$$\langle (u+v)A, (u+v)A \rangle = \|uA\|_2^2 + \|vA\|_2^2 + 2\langle uA, vA \rangle$$

By the Cauchy-Schwarz inequality, $\langle uA, vA \rangle \leq \|uA\|_2 \|vA\|_2$

$$\Rightarrow d_A(x_3,x_1)^2 \leq \|uA\|_2^2 + \|vA\|_2^2 + 2\left(\|uA\|_2 \|vA\|_2\right) = \left(\|uA\|_2 + \|vA\|_2\right)^2$$

Homework 1

(2) Learning an Embedding for Nearest Neighbors

ii) c) From previous page: $d_A(x_3, x_1)^2 \leq \left(\|uA\|_2 + \|vA\|_2\right)^2$

Taking the square root of both sides: $d_A(x_3, x_1) \leq \|uA\|_2 + \|vA\|_2$

By symmetry of norms, $d_A(x_3, x_1) = d_A(x_1, x_3)$. so

$$d_A(x_1, x_3) \leq \|uA\|_2 + \|vA\|_2 \quad \text{as required.} \ \square$$

b) i) Standard kNN has just one hyperparameter k; in this variant you must also decide how many dimensions p the embedding $A \in \mathbb{R}^{d \times p}$ should keep.

ii) The approach is parametric; the algorithm learns a matrix $A \in \mathbb{R}^{d \times p}$ as part of training. The entries of this matrix — the $d \times p$ real numbers — are the learned parameters.

Homework 1

(2) Learning an Embedding for Nearest Neighbors

b) iii) The inference procedure in the kNN algorithm involves classifying a query point based on the majority class of its neighbors. The inner optimization in this training setup mirrors that process in that, for each point, it checks how unanimous the vote is among its neighbors under a given embedding. It then chooses the label that optimizes unanimity. Basically, it chooses the label that makes kNN more "confident" in its predictions.

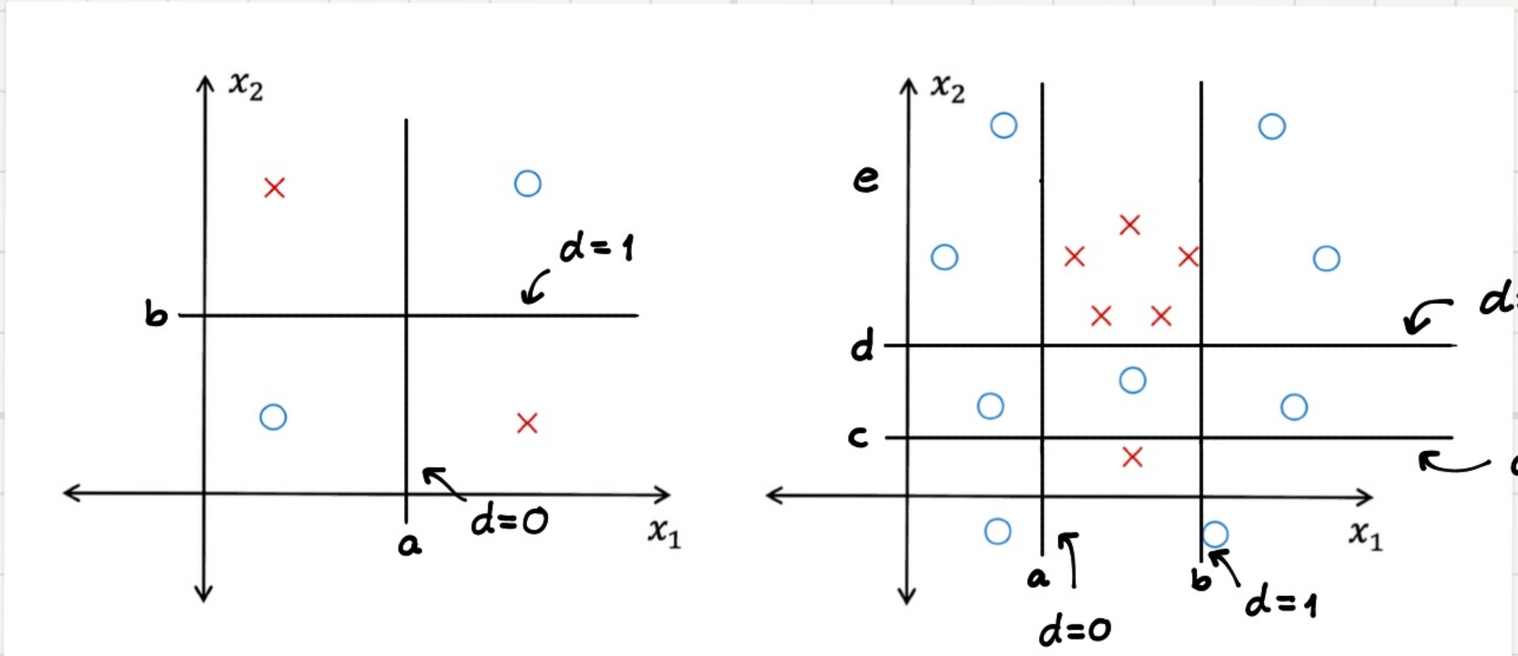
iv) This bilevel optimization problem solves the embedded nearest neighbors problem by choosing a transformation matrix A that maximizes how clustered together points of a given class are in the transformed space, thus making kNN's predictions more accurate.

v) This tells us that only d-2 features in the original d-dimensional space actually matter for classification; the remaining 2 don't help distinguish between classes. So A* flattens the data into a (d-2)-dimensional subspace without a performance cost.

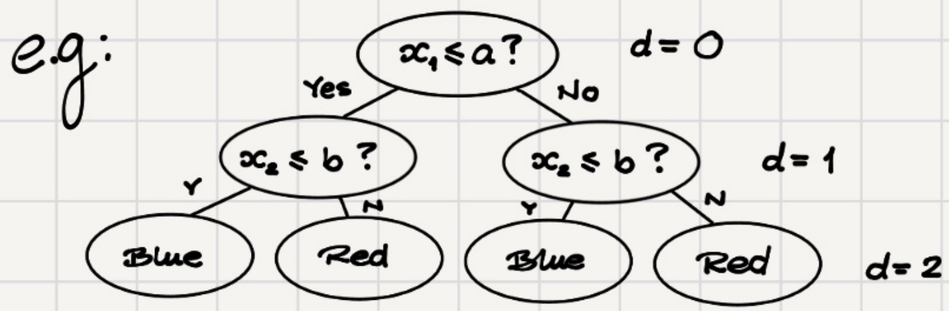
Homework 1

③ Decision Trees Warmup

a)



e.g:

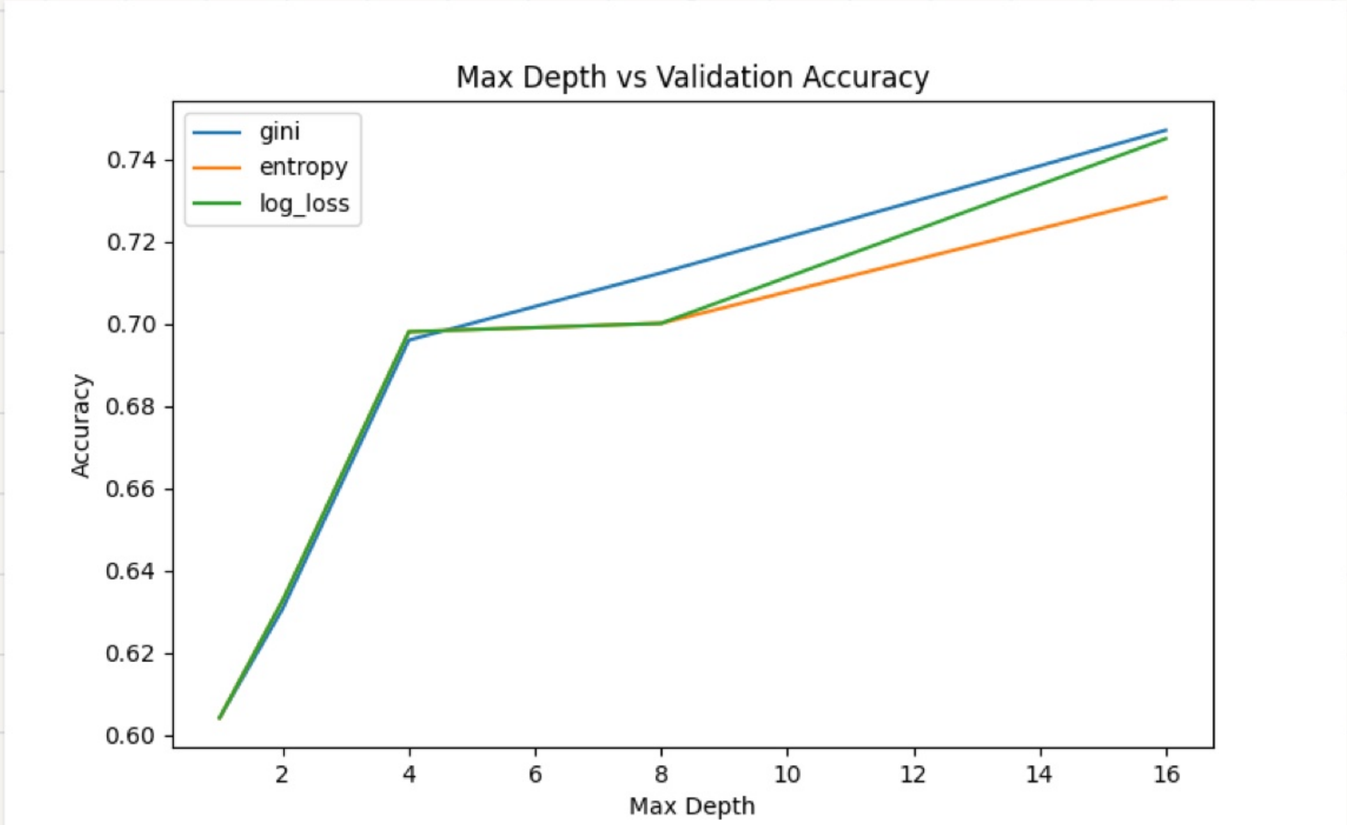


b) Total depth of tree for plot on left is 2; total depth of tree for plot on right is 4

Homework 1

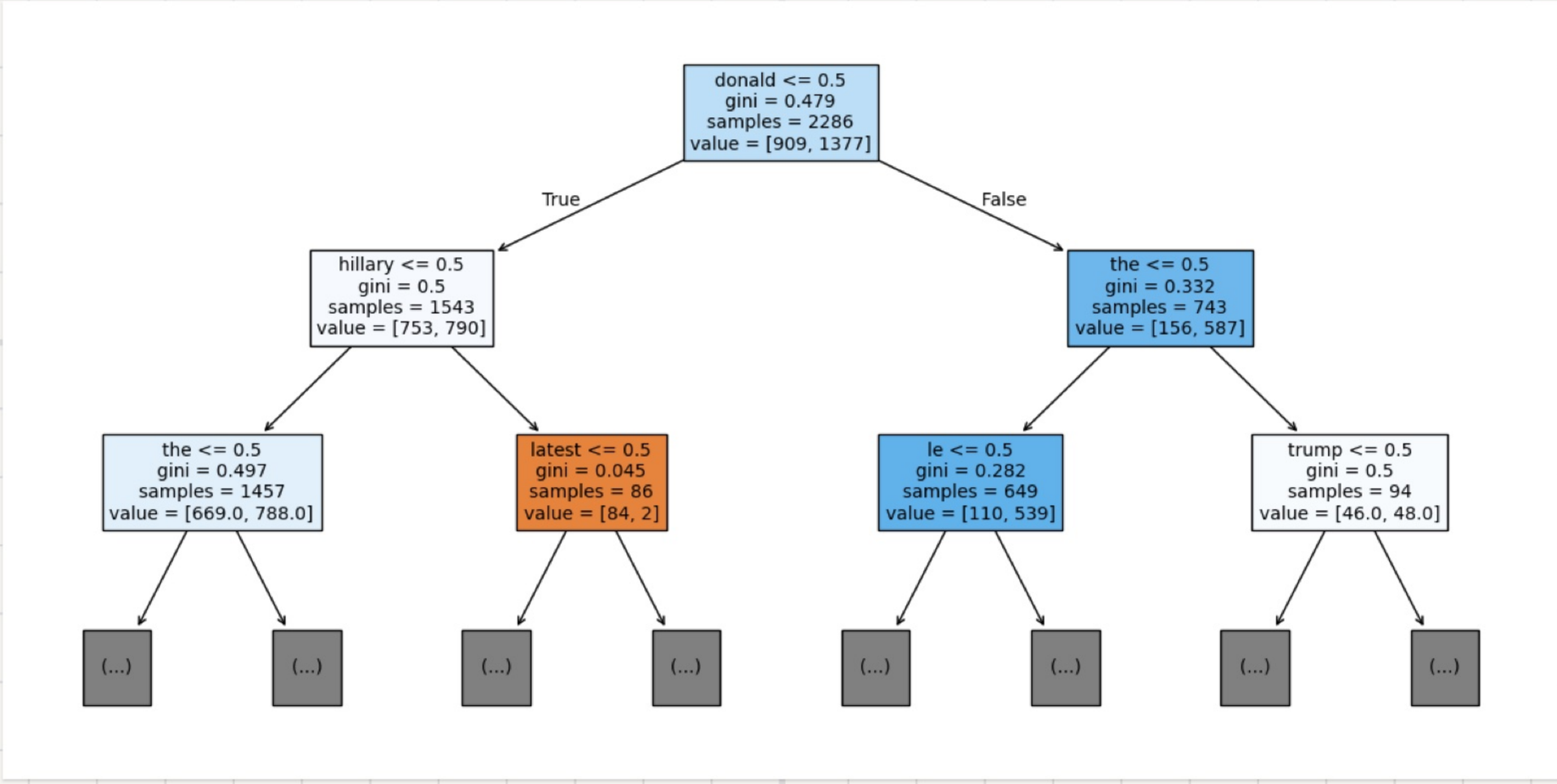
(4) b) Output from select_model():

```
Criterion: gini     | Max Depth: 1  | Validation Accuracy: 0.6041
Criterion: entropy  | Max Depth: 1  | Validation Accuracy: 0.6041
Criterion: log_loss | Max Depth: 1  | Validation Accuracy: 0.6041
Criterion: gini     | Max Depth: 2  | Validation Accuracy: 0.6306
Criterion: entropy  | Max Depth: 2  | Validation Accuracy: 0.6327
Criterion: log_loss | Max Depth: 2  | Validation Accuracy: 0.6327
Criterion: gini     | Max Depth: 4  | Validation Accuracy: 0.6959
Criterion: entropy  | Max Depth: 4  | Validation Accuracy: 0.6980
Criterion: log_loss | Max Depth: 4  | Validation Accuracy: 0.6980
Criterion: gini     | Max Depth: 8  | Validation Accuracy: 0.7102
Criterion: entropy  | Max Depth: 8  | Validation Accuracy: 0.6939
Criterion: log_loss | Max Depth: 8  | Validation Accuracy: 0.7000
Criterion: gini     | Max Depth: 16 | Validation Accuracy: 0.7510
Criterion: entropy  | Max Depth: 16 | Validation Accuracy: 0.7347
Criterion: log_loss | Max Depth: 16 | Validation Accuracy: 0.7306
```



Homework 1

(4) c) Optimal Decision Tree



Homework 1

(5) Regularized Linear Regression

a) if $w_j > 0$: $\quad w_j \leftarrow w_j - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\left(w^T x^{(i)} + b - t^{(i)}\right)x_j^{(i)} + \alpha_j + \beta_j w_j\right)$

$$b \leftarrow b - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\left(w^T x^{(i)} + b - t^{(i)}\right)\right)$$

if $w_j = 0$: $\quad w_j \leftarrow w_j - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\left(w^T x^{(i)} + b - t^{(i)}\right)x_j^{(i)}\right)$

$$b \leftarrow b - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\left(w^T x^{(i)} + b - t^{(i)}\right)\right)$$

if $w_j < 0$: $\quad w_j \leftarrow w_j - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\left(w^T x^{(i)} + b - t^{(i)}\right)x_j^{(i)} - \alpha_j + \beta_j w_j\right)$

$$b \leftarrow b - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\left(w^T x^{(i)} + b - t^{(i)}\right)\right)$$

Homework 1

(5) Regularized Linear Regression

b) Since we drop the bias term and set $\lambda_1 = 0$:

$$J_{reg}^{\beta}(\omega) = \frac{1}{2N}\sum_{i=1}^{N}\left(\omega^T x^{(i)} - t^{(i)}\right)^2 + \frac{\beta}{2}\sum_{j=1}^{D}\omega_j^2 \quad (\text{where } y = \omega^T x)$$

Taking the partial derivative w.r.t ω_j: $\frac{\partial J_{reg}^{\beta}}{\partial \omega_j} = \frac{1}{N}\sum_{i=1}^{N}\left(\omega^T x^{(i)} - t^{(i)}\right)x_j^{(i)} + \beta\omega_j$

But $\omega^T x^{(i)} = \sum_{j'=1}^{D}\omega_{j'}x_{j'}^{(i)} \Rightarrow \frac{\partial J_{reg}^{\beta}}{\partial \omega_j} = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j'=1}^{D}\omega_{j'}x_{j'}^{(i)} - t^{(i)}\right)x_j^{(i)} + \beta\omega_j$

So $\frac{\partial J_{reg}^{\beta}}{\partial \omega_j} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j'=1}^{D}\omega_{j'}x_{j'}^{(i)}x_j^{(i)} - \frac{1}{N}\sum_{i=1}^{N}t^{(i)}x_j^{(i)} + \beta\omega_j$

$\Rightarrow \sum_{j'=1}^{D}\left[\frac{1}{N}\sum_{i=1}^{N}x_{j'}^{(i)}x_j^{(i)}\right]\omega_{j'} - \frac{1}{N}\sum_{i=1}^{N}t^{(i)}x_j^{(i)} + \beta\omega_j$

Homework 1

(5) Regularized Linear Regression

b) From before: $\sum_{j'=1}^{D}\left[\frac{1}{N}\sum_{i=1}^{N} x_{j'}^{(i)} x_j^{(i)}\right] w_{j'} - \frac{1}{N}\sum_{i=1}^{N} t^{(i)} x_j^{(i)} + \beta w_j$

We can rewrite βw_j as $\beta \delta_{j,j'} w_{j'}$ inside a sum over j': $\beta w_j = \sum_{j'=1}^{D}\left[\beta \delta_{j,j'}\right] w_{j'}$

Moving βw_j inside the sum over j', we have:

$$\frac{\partial J_{reg}^{\beta}}{\partial w_j} = \sum_{j'=1}^{D}\left[\frac{1}{N}\sum_{i=1}^{N} x_{j'}^{(i)} x_j^{(i)} + \beta \delta_{j,j'}\right] w_{j'} - \frac{1}{N}\sum_{i=1}^{N} t^{(i)} x_j^{(i)}$$

· Comparing with $\frac{\partial J_{reg}^{\beta}}{\partial w_j} = \sum_{j'=1}^{D} A_{jj'} w_{j'} - c_j = 0$, we can "read off" $A_{jj'}$ and c_j as

$$A_{jj'} = \frac{1}{N}\sum_{i=1}^{N} x_{j'}^{(i)} x_j^{(i)} + \beta \delta_{j,j'} \quad \text{and} \quad c_j = \frac{1}{N}\sum_{i=1}^{N} t^{(i)} x_j^{(i)}$$

Homework 1

(5) Regularized Linear Regression

c) From before: $A_{jj'} = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)} + \beta\delta_{j,j'}$ and $c_j = \frac{1}{N}\sum_{i=1}^{N} t^{(i)} x_j^{(i)}$

$$\frac{\partial J_{reg}^{\beta}}{\partial w_j} = \sum_{j'=1}^{D} A_{jj'} w_{j'} - c_j = 0 \Rightarrow \hat{A}\vec{w} = \vec{c}$$

From our expressions for $A_{jj'}$ and c_j, we can see that

$$\hat{A} = \frac{1}{N} X^T X + \beta \hat{I}, \quad \text{and} \quad \vec{c} = \frac{1}{N} X^T \vec{t}$$

We can solve for $\vec{w}$ by inverting $\hat{A}$:

$$\vec{w} = \hat{A}^{-1}\vec{c} = \left(\frac{1}{N} X^T X + \beta I\right)^{-1}\left(\frac{1}{N} X^T t\right)$$