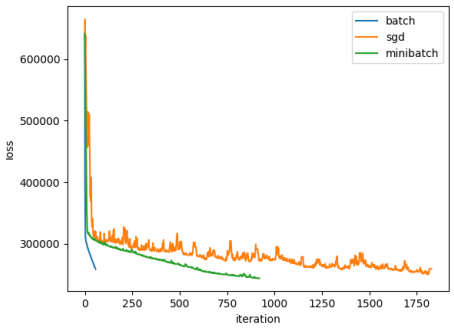
1 (a)

$\nabla_b f(w,b) = C\sum_{i=1}^{n} \frac{\partial L(x_i,y_i)}{\partial b}$, where $\frac{\partial L(x_i,y_i)}{\partial b} = \begin{cases} 0 & \text{if } y_i(x_i \cdot w + b) \geq 1 \\ -y_i & \text{otherwise} \end{cases}$

1 (b)

Code: 4_1_b.py



run name	time (s)	# iters	# epochs
batch	0.44	57	57
sgd	8.02	1827	0.28
minibatch	4.02	920	2.87

When the batch size increases, we see these effects:

1. The training curve becomes smoother.

2. The training time decreases.

3. The number of iterations decreases.

4. The dataset is traversed for more rounds.

The batch GD was the fastest, because the data were vectorized and could fit in memory, and the loss was calculated for the fewest times per data point. If these factors could be leveled, we should see sgd being the fastest, which corresponds to the fewest number of epochs required.

2 (a)

I(D) = 100 (1 - .6 ^ 2 - .4 ^ 2) = 48

Wine: G = 48 - 50 (1 - .6 ^ 2 - .4 ^ 2) - 50 (1 - .6 ^ 2 - .4 ^ 2) = 0

Running: G = 48 - 30 (1 - (2/3) ^ 2 - (1/3) ^ 2) - 70 (1 - (4/7) ^ 2 - (3/7) ^ 2) ~ 0.38

Pizza: G = 48 - 80 (1 - (5/8) ^ 2 – (3/8) ^ 2) - 20 (1 - (1/2) ^ 2 - (1/2) ^ 2) = 0.5

We choose pizza at root for its max G.

2 (b)

The split at root should use a_1, such that the gain G is maximized. The two subtrees will have the same height, because we use all attributes, but not all branches have the same number of levels, because we stop branching when the population is too small, to avoid overfitting.

3 (a)

The 1st term on the RHS:

$$2\mathrm{cost}_w(\hat{S},T) = 2\sum_{t_{ij}\in\hat{S}} |\mathrm{S_{ij}}|\mathrm{d}^2(\mathrm{t_{ij}},\mathrm{T}) = 2\sum_{\mathrm{x}\in\mathrm{S}} \mathrm{d}^2(\mathrm{t_{ij}},\mathrm{T})$$

where the last equation reads: For each x in S, x corresponds to an S_i, which corresponds to a set of t_{ij}. We sum up the squared distance from t_{ij} to T.

The 2nd term on the RHS:

$$2\sum_{i=1}^{l} cost(S_i,T_i) = 2\sum_{i=1}^{l}\sum_{x\in S_i} d^2(x,T_i) = 2\sum_{\mathrm{x}\in\mathrm{S}} \mathrm{d}^2(\mathrm{x},\mathrm{t_{ij}})$$

where the last equation reads: For each x in S, x corresponds to an S_i, which corresponds to a set of t_{ij}. We sum up the squared distance from x to t_{ij}.

The suggested inequality gives

$$2d^2(t_{ij},T) + 2d^2(x,t_{ij}) \geq \left(d(t_{ij},T) + d(x,t_{ij})\right)^2$$

Because $\mathrm{d}(x,t_{ij})$ is the distance from x to its associated t_{ij}, the triangular inequality gives

$$\mathrm{d}(t_{ij},T) + \mathrm{d}(x,t_{ij}) \geq d(x,T)$$

Stringing all the above gives the desired inequality.

3 (b)

$\sum_{i=1}^{l} cost(S_i,T_i)$

$= \sum_{i=1}^{l} cost(S_i,T_i^*)\alpha$ (T_i is found by ALG. T_i^* means the optimal for S_i.)

$\leq \sum_{i=1}^{l} cost(S_i,T)\alpha$ (T_i^* is the optimal.)

$= \alpha \cdot cost(S,T^*)$ (Group all x in each S_i.)

3(c)

First inequality:

$\mathrm{cost}_w(\hat{S},T)$

$\leq \alpha \cdot cost_w(\hat{S},\hat{T})$ (T is found by ALG. $\hat{T}$ means the optimal for $\hat{S}$.)

$\leq \alpha \cdot cost_w(\hat{S},T^*)$ ($\hat{T}$ is optimal for $\hat{S}$, but T^* is not.)

Second inequality follows similarly from (a), by replacing T with T^*.

Therefore, by (a), (b) and the above,

$$\text{cost}(S,T)$$

$$\leq 2\text{cost}_w\left(\hat{S},T\right) + 2\sum_{i=1}^{l} cost(S_i,T_i)$$

$$\leq 4\alpha\left(\sum_{i=1}^{l} cost(S_i,T_i) + cost(S,T^*)\right) + 2\sum_{i=1}^{l} cost(S_i,T_i)$$

$$\leq (4\alpha(\alpha+1)+2\alpha)\text{cost}(S,T^*)$$

$$= (4\alpha^2+6\alpha)\text{cost}(S,T^*)$$

4 (a)

$\mathrm{P}[\tilde{\mathrm{F}}[\mathrm{i}] \leq F[i] + \epsilon t]$

$= \mathrm{P}\left[min_j\{c_{j,h_j(i)}\} \leq F[i] + \epsilon \mathrm{t}\right]$ (by definition)

$= \mathrm{P}\left[\exists \mathrm{j} \ni c_{j,h_j(i)} \leq F[i] + \epsilon \mathrm{t}\right]$ (set logic)

$= 1 - \mathrm{P}\left[\forall j, c_{j,h_j(i)} > F[i] + \epsilon \mathrm{t}\right]$ (complement)

$= 1 - \prod_j P\left[c_{j,h_j(i)} > F[i] + \epsilon t\right]$ (independence of hash functions)

Find $P[\cdot]$:

$P\left[c_{j,h_j(i)} > F[i] + \epsilon t\right]$

$= P\left[c_{j,h_j(i)} - F[i] > \epsilon t\right]$

$< \frac{\mathrm{E}\left[c_{j,h_j(i)} - F[i]\right]}{\epsilon \mathrm{t}}$ (Markov's inequality. The random variable > 0 by property 1.)

$\leq \frac{t - F[i]}{et}$ (Property 2)

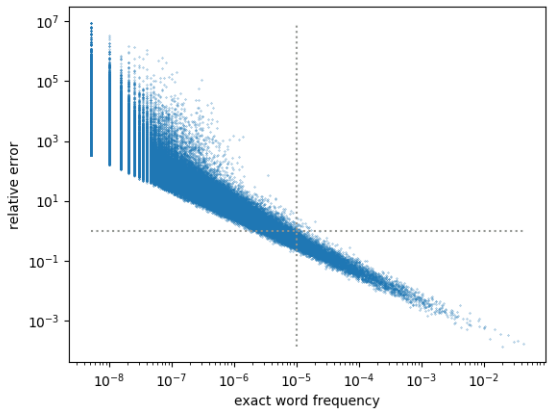
$\leq \frac{1}{e}$ (F[i] >= 0)

Back to the main derivation:

$\mathrm{P}[\tilde{\mathrm{F}}[\mathrm{i}] \leq F[i] + \epsilon t] > 1 - \left(\frac{1}{e}\right)^{log\left(\frac{1}{\delta}\right)} = 1 - \delta$

4 (b)

Code: 4_4_b.py and 4_4_b_plot.py



The relative error tends to be < 1, if the exact frequency is > 1e-5. This is consistent with the parameters.