Q1

Part A

We begin,

$$w_{t+1} = w_t - \alpha \nabla f_i(w_t)$$

$$f(w) = \frac{1}{2}(f_1 + f_2) \implies \nabla f_i(w) = w \pm 1$$

$$w_{t+1} = w_t - \alpha \nabla f_i(w_t)$$

$$f(w_{t+1}) = \frac{1}{2}\left[w_t^2 - 2w_t\alpha\nabla f_i(w_t) + \alpha^2(\nabla f_i(w_t))^2\right]$$

Over expectation,

$$E[\nabla f_i(w)] = f(w) \implies E[f(w_{t+1})|w_t] = \frac{w_t^2}{2} - \alpha w_t^2 + \frac{\alpha^2}{2}E[(\nabla f_i(w_t))^2]$$

$$E[f(w_{t+1})|w_t] = \frac{w_t^2}{2} - \alpha w_t^2 + \frac{\alpha^2}{2}E[w_t^2 \pm 2w_t + 1]$$

$$E[f(w_{t+1})|w_t] = f(w_t) - 2\alpha f(w_t) + \alpha^2 f(w_t) + \alpha^2/2$$

$$E[f(w_{t+1})|w_t] = (1-\alpha)^2 E[f(w_t)] + \frac{\alpha^2}{2}$$

Applying the recurrence over 2K iterations,

$$E[f(w_{2K})] = (1-\alpha)^{4K} f(w_0) + \alpha^2/2 \sum_{2K-1}(1-\alpha)^{2i} = \frac{(1-\alpha)^{4K}}{2} + \frac{1-(1-\alpha)^{4K}}{1-(1-\alpha)^2}\alpha^2/2$$

$$\boxed{E[f(w_{2K})] = \frac{(1-\alpha)^{4K}}{2} + \frac{1-(1-\alpha)^{4K}}{1-(1-\alpha)^2}\frac{\alpha^2}{2}}$$

Part B

We apply the following identity, which is valid for $\alpha \in (0, 1/2)$,

$$1 - \alpha \geq e^{-2\alpha} \implies E[f(w_{2K})] \geq \frac{e^{-8K\alpha}}{2} + \frac{1-e^{-8K\alpha}}{1-e^{-4\alpha}}\frac{\alpha^2}{2} \geq \frac{e^{-8K\alpha}}{2} + \frac{\alpha^2}{2}$$

$$E[f(w_{2K})] \geq \frac{e^{-8K\alpha}}{2} + \frac{\alpha^2}{2}$$

Using our result from a, we differentiate wrt α to minimize the loss,

$$\alpha = 4Ke^{-8K\alpha} \implies \frac{\alpha}{8K} + \frac{\alpha^2}{2}$$

$$E[f(w_{2K})] \geq \frac{\alpha}{8K} + \frac{\alpha^2}{2} \geq \frac{\alpha}{8K} \geq \frac{1}{16K}$$

We arrive at the desired asymptotic bound. Graphing using Mathematica, we see this is indeed a valid and correct lower bound for all α and K.

$$E[f(w_{2K})] \geq \frac{1}{16K} \implies \boxed{E[f(w_{2K})] = \Omega\left(\frac{1}{K}\right)}$$

as desired

Part C

We begin,

$$w_{t+1} = w_t - \alpha \nabla f_i(w_t)$$
$$f(w) = \frac{w^2}{2} \implies \nabla f(w) = w$$

We necessarily must run through both component losses per epoch,

$$w_{t+1} = w_t - \alpha \nabla f_i(w_t)$$
$$w_{t+1} = (1-\alpha)w_t \pm \alpha$$

Using the adjacent component loss in the next iteration,

$$w_{t+2} = (1-\alpha)((1-\alpha)w_t \pm \alpha) \mp \alpha$$
$$w_{t+2} = (1-\alpha)^2 w_t \pm \alpha^2$$
$$f(w_{t+2}) = (1-\alpha)^4 f(w_t) + \alpha^4/2 \pm \alpha^2(1-\alpha)^2$$
$$E(f(w_{t+2})) = (1-\alpha)^4 E(f(w_t)) + \alpha^4/2$$
$$E(f(w_{2K})) = (1-\alpha)^{4K} E(f(w_0)) + \alpha^4/2 \sum_{i=0}^{K-1} (1-\alpha)^{4i}$$
$$\boxed{E[f(w_{2K})] = \frac{(1-\alpha)^{4K}}{2} + \frac{1-(1-\alpha)^{4K}}{1-(1-\alpha)^4}\frac{\alpha^4}{2}}$$

Part D

Looking back at the second term, since α is bounded between 0 and 1/2, we clearly see α multiplied by the coefficient cannot exceed 1. Therefore, we can ignore this coefficient and an α term to make the inequality,

$$E[f(w_{2K})] \leq \frac{(1-\alpha)^{4K}}{2} + \frac{\alpha^3}{2}$$

Applying the exponential inequality,

$$E[f(w_{2K})] \leq \frac{e^{-4\alpha K}}{2} + \frac{\alpha^3}{2}$$

Now to remove the exponential, let's pick α such that $\alpha = \frac{3\ln(K)}{4K}$. Then, our inequality

$$E[f(w_{2K})] \leq \frac{e^{\ln(K^{-3})}}{2} + \frac{27\ln(K)^3}{128K^3} = \frac{1}{2K^3} + \frac{27\ln(K)^3}{128K^3}$$
$$E[f(w_{2K})] \leq \frac{64 + 27\ln(K)^3}{128K^3} \implies E[f(w_{2K})] = \mathcal{O}\left(\frac{\log(K)^3}{K^3}\right)$$

Equivalently,

$$\mathcal{O}\left(\frac{\log(K)^3}{K^3}\right) \implies \boxed{\mathcal{O}\left(\frac{1}{K^2}\right)}$$

as desired

Q2 - Nesterov Momentum

Part A

We begin,

$$\begin{cases} v_{t+1} = w_t - \alpha\nabla f(w_t) \\ w_{t+1} = v_{t+1} + \beta(v_{t+1} - v_t) \end{cases}$$

$$\implies v_{t+1} = v_t + \beta(v_t - v_{t-1}) - \alpha\nabla f(v_t + \beta(v_t - v_{t-1}))$$

$$\nabla f(w_t) = \gamma w_t \implies v_{t+1} = v_t + \beta(v_t - v_{t-1}) - \alpha\gamma(v_t + \beta(v_t - v_{t-1}))$$

$$v_{t+1} = v_t + \beta v_t - \beta v_{t-1} - \alpha\gamma v_t - \alpha\gamma\beta v_t + \alpha\gamma\beta v_{t-1}$$

$$v_{t+1} = (1 + \beta - \alpha\gamma - \alpha\gamma\beta)v_t + (\alpha\gamma\beta - \beta)v_{t-1}$$

In matrix form, this last equation is

$$\begin{bmatrix} v_{t+1} \\ v_t \end{bmatrix} = \begin{bmatrix} (1 + \beta - \alpha\gamma - \alpha\gamma\beta) & \alpha\gamma\beta - \beta \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_t \\ v_{t-1} \end{bmatrix}$$

$$\begin{bmatrix} v_{t+1} \\ v_t \end{bmatrix} = \begin{bmatrix} (1 + \beta)(1 - \alpha\gamma) & -\beta(1 - \alpha\gamma) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_t \\ v_{t-1} \end{bmatrix}$$

We can apply this definition repeatedly to the rightmost vector:

$$\begin{bmatrix} v_{t+1} \\ v_t \end{bmatrix} = \begin{bmatrix} (1 + \beta)(1 - \alpha\gamma) & -\beta(1 - \alpha\gamma) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} (1 + \beta)(1 - \alpha\gamma) & -\beta(1 - \alpha\gamma) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_{t-1} \\ v_{t-2} \end{bmatrix}$$

Unraveling this recurrence until we get to v_1, v_0,

$$\begin{bmatrix} v_{t+1} \\ v_t \end{bmatrix} = \begin{bmatrix} (1 + \beta)(1 - \alpha\gamma) & -\beta(1 - \alpha\gamma) \\ 1 & 0 \end{bmatrix} (\dots) \begin{bmatrix} v_1 \\ v_0 \end{bmatrix}$$

As desired, we arrive at

$$\boxed{\begin{bmatrix} v_{t+1} \\ v_t \end{bmatrix} = \begin{bmatrix} (1 + \beta)(1 - \alpha\gamma) & -\beta(1 - \alpha\gamma) \\ 1 & 0 \end{bmatrix}^t \begin{bmatrix} v_1 \\ v_0 \end{bmatrix}}$$

Part B

We begin,

$$\begin{vmatrix} (1 + \beta)(1 - \alpha\gamma) - \lambda & -\beta(1 - \alpha\gamma) \\ 1 & -\lambda \end{vmatrix} = \lambda^2 - \lambda(1 + \beta)(1 - \alpha\gamma) + \beta(1 - \alpha\gamma)$$

$$\lambda = \frac{(1 + \beta)(1 - \alpha\gamma) \pm \sqrt{(1 + \beta)^2(1 - \alpha\gamma)^2 - 4\beta(1 - \alpha\gamma)}}{2}$$

$$\lambda = \frac{(1 + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})(1 - \frac{\gamma}{L}) \pm \sqrt{(1 + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^2(1 - \frac{\gamma}{L})^2 - 4\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}(1 - \frac{\gamma}{L})}}{2}$$

$$\lambda = \frac{(1-\frac{\gamma}{L})\sqrt{\kappa} \pm \sqrt{\kappa(1-\frac{\gamma}{L})^2 - (\kappa-1)(1-\frac{\gamma}{L})}}{\sqrt{\kappa}+1}$$

$$\lambda = \frac{(1-\frac{\gamma}{L})\sqrt{\kappa} \pm \sqrt{\frac{L}{\mu}(1-\frac{\gamma}{L})^2 - (\frac{L}{\mu}-1)(1-\frac{\gamma}{L})}}{\sqrt{\kappa}+1}$$

$$\lambda = \frac{(1-\frac{\gamma}{L})\sqrt{\kappa} \pm \sqrt{(\frac{L}{\mu}-\frac{\gamma}{\mu})(1-\frac{\gamma}{L}) - (\frac{L}{\mu}-1)(1-\frac{\gamma}{L})}}{\sqrt{\kappa}+1}$$

As desired, we arrive at

$$\boxed{\lambda = \frac{(1-\frac{\gamma}{L})\sqrt{\kappa} \pm \sqrt{(1-\frac{\gamma}{L})(1-\frac{\gamma}{\mu})}}{\sqrt{\kappa}+1}}$$

Part C

The determinant of a matrix is the product of its eigenvalues, and $\lambda_1\lambda_2 = |\lambda|^2$ for complex eigenvalues (which $\mu \leq \gamma \leq L$ ensures). We find the determinant as before,

$$|\lambda|^2 = \begin{vmatrix} (1+\beta)(1-\alpha\gamma) & -\beta(1-\alpha\gamma) \\ 1 & 0 \end{vmatrix} = (1-\alpha\gamma)\beta$$

Using our definitions, we see that as desired,

$$\beta = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}, \quad \alpha = 1/L \implies |\lambda|^2 = (1-\alpha\gamma)\beta = (1-\frac{\gamma}{L})\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$$

$$\boxed{|\lambda|^2 = (1-\frac{\gamma}{L})\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}}$$

Further, to maximize λ within this range, we clearly must minimize $1-\gamma/L$.
Therefore, in the upper bound, we require $\gamma = \mu$.

$$\implies |\lambda|^2 \leq (1-\frac{\mu}{L})\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = (1-\frac{1}{\kappa})\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = \frac{\kappa-1}{\kappa}\frac{\kappa-1}{(\sqrt{\kappa}+1)^2}$$

$$\implies |\lambda| \leq \frac{\kappa-1}{\sqrt{\kappa}(\sqrt{\kappa}+1)} = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}} = 1 - \frac{1}{\sqrt{\kappa}}$$

As desired, we arrive at an upper bound,

$$\implies \boxed{|\lambda| \leq 1 - \frac{1}{\sqrt{\kappa}}}$$

Part D

In effect, we combine the results from parts a-c. We take the gradient,

$$f = \frac{1}{2}w^T A w \implies \nabla f = Aw$$

Using our scalar result from a, we see that the computation is identical where γ is A and ones with the identity matrix accordingly,

$$\begin{bmatrix} v_{t+1} \\ v_t \end{bmatrix} = \begin{bmatrix} (1+\beta)(I-\alpha A) & -\beta(I-\alpha A) \\ 1 & 0 \end{bmatrix}^t \begin{bmatrix} v_1 \\ v_0 \end{bmatrix}$$

We can simply compute the product of the eigenvalues as the determinant of M, since $|\lambda|^2 = \det(M)$.

By the block matrix determinant identity, we have:

$$|\lambda|^2 = \det(M) = \det(0 - (-\beta(I - \alpha A)) = \beta(I - \alpha A))$$

$$|\lambda|^2 = \det(\beta(I - \alpha A)) = \det(I - \frac{A}{L})\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$$

$$|\lambda|^2 = \det(\beta(I - \alpha A)) = \det(I - \frac{A}{L})\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$$

Now we must show that the expression and therefore the eigenvalues are bounded appropriately. Note that by assumption, A's eigenvalues are bounded below by μ and above by L. Therefore, the operator norm of A is bounded by the corresponding singular values $\sqrt{\lambda} = \sqrt{\mu}, \sqrt{L}$, and A follows the same bounds as γ in C by $\det(A) = |\lambda|^2$. In an equivalent argument, we can also recognize the above determinant as bounded by the eigenvalues of A, following from the eigenvalue equation form. Either way, since the eigenvalues of A and the determinant expression are bounded by μ and λ, we conclude the assumptions for part C are valid in this scenario. Then reusing our steps and results from part C, we similarly surmise,

$$|\lambda|^2 = \det(\beta(I - \alpha A)) = (1 - \frac{\gamma}{L})\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$$

where γ is again appropriately bounded by μ and L. We reuse our secondary result from C, then,

$$|\lambda| \leq 1 - \frac{1}{\sqrt{\kappa}}$$

Collecting our results, we arrive at the desired result,

$$\boxed{\begin{bmatrix} v_{t+1} \\ v_t \end{bmatrix} = \begin{bmatrix} (1+\beta)(I-\alpha A) & -\beta(I-\alpha A) \\ 1 & 0 \end{bmatrix}^t \begin{bmatrix} v_1 \\ v_0 \end{bmatrix}}$$

$$\boxed{\lambda \leq 1 - \frac{1}{\sqrt{\kappa}}}$$

Q3 - Dimension Reduction

Part A

We compute k-nearest neighbors with k=1 as follows with mostly valid python using

$$d = ||x - x_i||^2$$

```
def knn(x): #k = 1
        nearest = float('inf')
        label = 0
        # iterate over (xi, yi) dataset
        for (xi, yi) in training:
                #compute distance
                dist = sum((x[j]-xi[j])**2 for j in range(d))
                if dist < nearest:
                        nearest = dist
                        label = yi
        return label
```

We can compute euclidean distance by simply taking the square root of the squared distance, but this adds computation and since square root preserves order, doesn't affect the ordering of vectors by distances.

Part B

Subtracting elementwise, squaring, and summing costs $d + d + d - 1$. Adding the comparison check after, this costs $3d$. This must be done for every training dataset example, or $3dn$. Finally, the above routine must be run for every test example, or $3dnm$.

$$\boxed{3dmn \text{ computations} = \mathcal{O}(dmn)}$$

Part C

We repeat in mostly valid python but compute squared distances more efficiently. We compute the distance to the zero vector and compare nonzero element differences. Squared distance is computed as

$$d = ||x - x_i||^2 = \langle x, x\rangle - 2\langle x, x_i\rangle + \langle x_i, x_i\rangle$$

Since every distance is incremented by $\langle x, x\rangle$, it is irrelevant for comparison and we can drop it to improve computation. We only care about the differences in distances.

$$d' = \langle x_i, x_i\rangle - 2\langle x, x_i\rangle$$

We can optimize these dot products with sparsity. The code is below,

```
def knn(x): #k = 1
        nearest = float('inf')
        label = 0
        # iterate over (xi, yi) dataset
        for (xi, yi) in training:
                #get index and value arrays
                idxs = xi['index']
                vals = xi['vals']
                #compute dist heuristic in O(nonzero elements)
                xidotxi = sum((val)**2 for val in vals)
                xdotxi = sum(val[i]*x[i] for i in range(len(vals)))
                dist = xidotxi - 2*xdotxi
                if dist < nearest:
                        nearest = dist
                        label = yi
        return label
```

Again, the square root preserves order, so we save computation by comparing the squared distances.

Part D

The code is identical to before, except computing distances only requires iterating over the number of nonzero vector entries instead of all dimensions. For each distance analog, we compute

$$d' = \langle x_i, x_i\rangle - 2\langle x, x_i\rangle$$

For each training example, we compute the dot product but only iterate over nonzero elements. There are an average pd nonzero elements, costing $2pd - 1$. We compute the dot product with itself and the test example. $4pd - 2$. Combining them in the formula above involves scaling by 2 and subtracting scalars, for 2 ops. We add another op for the comparison check. So each training example incurs $4pd + 1$ operations. Over every training example, this costs, $4pdn + n$. So in total per example, we consume $4pdn + n$ operations. We have m examples, for

$$(4pdn + n)m \text{ computations} = \boxed{\mathcal{O}(pdmn)}$$

If $p \approx 1$, using sparse data formats has no benefit, but for very sparse $p << 1$, the savings is significant.