1) a) $\phi_1(x) = \{pretty: 1, bad: 1\}$ (-1)

$\phi_2(x) = \{good: 1, plot: 1\}$ (+1)

$\phi_3(x) = \{not: 1, good: 1\}$ (-1)

$\phi_4(x) = \{pretty: 1, scenery: 1\}$ (+1)

Run stochastic gradient descent with feature vectors above, using hinge loss.

$$\nabla_w \text{Losshinge}(w, x, y) = \begin{cases} -\phi(x)y & \text{if } w \cdot \phi(x) y < 1 \\ -0 & \text{if } w \cdot \phi(x) y \geq 1 \end{cases}$$

Using a gradient descent method $w \leftarrow w - \eta \nabla_w \text{Loss}(w, x, y)$ and hinge loss while starting @ w=0 forall features, we find:

1. "Scenery, plot" features have only increasing positive weights, max out @ $\boxed{w=1.}$

2. "good, pretty" Features have $\boxed{0 \text{ weight}}$, did not change through iteration. This due to being in one good, one bad review, cancel out so weight doesn't change for gradient hinge loss.

3. "$\boxed{\text{Not, bad}}$" features have negative increasing weights based on iteration η value, max out @ $\boxed{w=-1.}$

-Done via comperter code trial, verified w/ mental check.

1) b)

Dataset: $\left[(\text{Good}, 1)^{1}, (\text{Bad}, -1)^{2}, (\text{Not Good}, -1)^{3}, (\text{Not Bad}, 1)^{4}\right]$

Features: Use word features only for first part

1. Good
2. Bad
3. Not

$\longrightarrow \phi_1 = \begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} \quad \phi_2 = \begin{bmatrix} 0 \\ \gamma \\ 0 \end{bmatrix} \quad \phi_3 = \begin{bmatrix} x \\ 0 \\ z \end{bmatrix} \quad \phi_4 = \begin{bmatrix} 0 \\ \gamma \\ z \end{bmatrix}$

Any linear classifier where x, γ, z are numbers

Tests:

1) $W_1 x > 0 \quad [y=1]$
2) $W_2 \gamma < 0 \quad [y=-1]$
3) $W_1 x + W_3 z < 0 \quad [y=-1]$
4) $W_2 \gamma + W_3 z > 0 \quad [y=1]$

from eqns 1 and 3:

$W_3 z$ must be negative to satisfy inequality since $W_1 x$ is positive from eqn (1)

from eqns 2 and 4:

$W_3 z$ must be positive since $W_2 \gamma$ is negative to satisfy eqn 2.

Thus: If $W_3 z$ is negative and $-W_3 z > W_1 x$, then tests 1, 2, 3 can be satisfied, but 4 cannot.

If $W_3 z$ is positive and $W_3 z > W_2 \gamma$, then test 1, 2, and 4 are satisfied, not test 3.

If $W_2 \gamma < W_3 z < W_1 x$, neither test 3 or 4 is satisfied.

Thus, no linear classifier works to get error 0. ✓

2) $\sigma(z) = (1+e^{-z})^{-1}$ $\sigma(w \cdot \phi(x))$ - use nonlinear predictor

a) Write out expression for loss:

$$Loss_{squared} = (\sigma(w \cdot \phi(x)) - y)^2$$

$$\boxed{Loss_{squared} = \left[(1+e^{-w \cdot \phi(x)})^{-1} - y\right]^2}$$

b) $\nabla_w Loss_{squared} = \nabla_w(\sigma(w \cdot \phi(x)) - y)^2$

① $\nabla = 2(\sigma(w \cdot \phi(x)) - y)(\sigma'(w \cdot \phi(x)))$

$$\frac{d}{dw}\sigma(w \cdot \phi(x)) = -(1+e^{-w \cdot \phi(x)})^{-2}(-\phi(x)e^{-w\phi(x)})$$

② $\sigma' = \dfrac{\phi(x)e^{-w\phi(x)}}{(1+e^{-w\phi(x)})^2}$

$$\boxed{\begin{aligned} &\nabla_w Loss_{squared} = 2(p-y)p' \\ &\text{where: } p = \sigma(w \cdot \phi(x)) \\ &\qquad\quad p' = \frac{-\phi(x)e^{-w\phi(x)}}{(1+e^{-w \cdot \phi(x)})^2} \end{aligned}}$$

$$= -2e^{-\phi w}\phi(x)(1 - p^{-1} y)p^3$$

$$\boxed{2e^{-\phi(x)w}\phi(x)(p-y)p^2}$$

in terms of p

2c) ∇_w Loss Squared $= -2(p-y)p'$

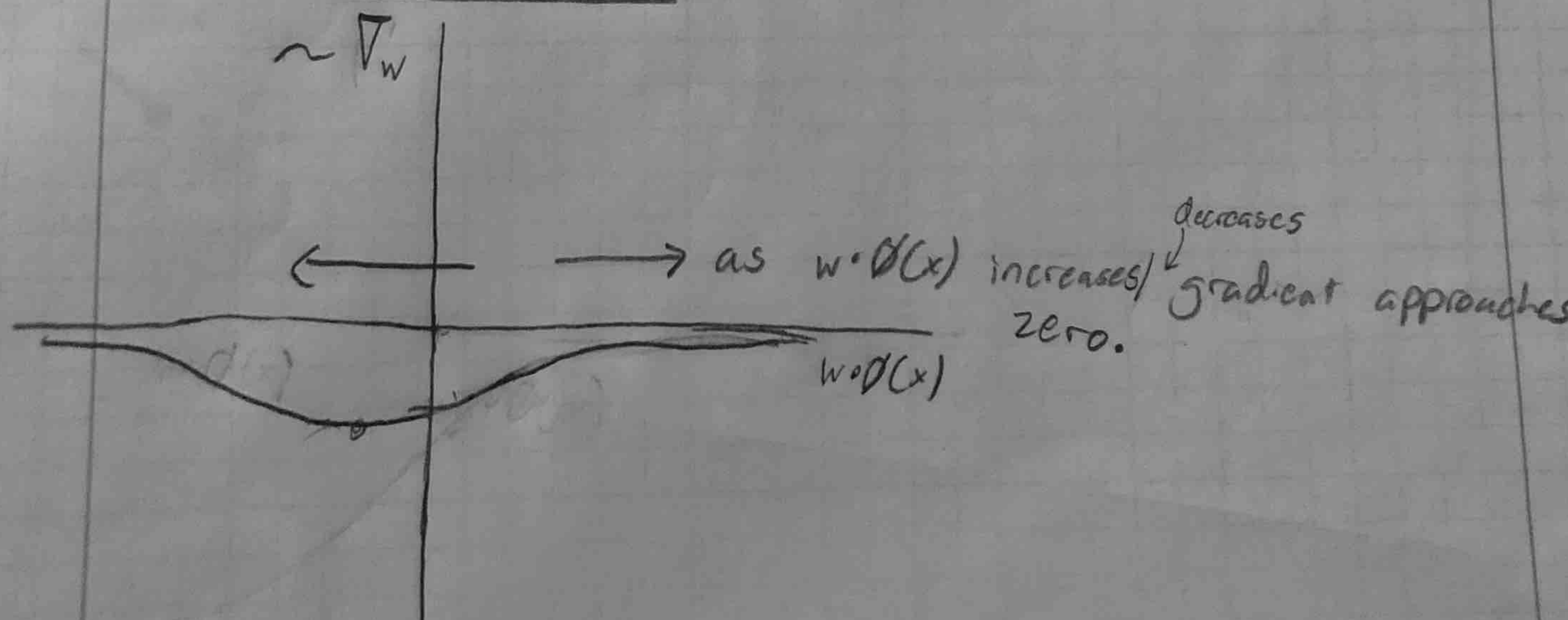
$p = \sigma(w \cdot \phi(x))$

$p' = \dfrac{\phi(x) e^{-w\phi(x)}}{(1+e^{-w\phi(x)})^2}$

- As $w \cdot \phi(x) >> 0$, and $w \cdot \phi(x) << 0$, p' approaches 0, thus creating a gradient that is very close to zero. These situations could cause a vanishing gradient problem, and will cause very slow convergent in a gradient descent method.

- When $\sigma(w \cdot \phi(x)) = y$, ∇_w Loss squared will equal zero. However, when this happens, the loss has been minimized, so this is a trivial case when considering vanishing gradient, as the minimum has already been reached for the loss squared equation

Gradient Plotted:



$2(p-1)(p(1-p))\phi(x)$

$p^2-p-p^3-p^2$ $3p^2-p$

$p(-p-$

2d) $2(p-1)p^2\phi e^{-\phi(x)w} \rightarrow 2(p-1)(p)(1-p)$

∇

p^3-p^2

$3p^2-p$

$(3p-1)p=0$

$\underline{\underline{p=\frac{1}{3}}}$ @ min value by taking another gradient

Plugging $p=\frac{1}{3}$ back into gradient eqn:

$\boxed{\frac{8}{27}\|\phi(x)\|}$ ← found w/ graphing calculator too, verified for gradient

- This value is either a local max or min, not convex function, not absolute max or min, 2e) D (x, y) w/ $\vec{w}$ results $\|\sigma(w, \phi(x)) - y\|_2^2 = 0$

Write an expression for y' in terms of w.

$$\boxed{y' = \sigma^{-1}(y)}$$

Justify:

$$\sigma(w, \phi(x)) - y = 0$$

$$\sigma(w, \phi(x)) = y \leftarrow y = \sigma(y')$$

$$\cancel{\sigma}(w, \phi(x)) = \cancel{\sigma}(y')$$

$$w \circ \phi(x) = y'$$

thus, when linear operator is used:

$$\|w \circ \phi - y'\|_0^2 \leftarrow$$

$$W' \circ \phi - w \circ \phi(x)$$

$\|0\|_2^2 \leftarrow$ zero-squared loss achieved through $y' = \sigma^{-1}(y)$

3d)

1) No screen fantasy-adventure in recent memory has the showmanship of clones' last 45 minutes.

Predict: -1 Actual: 1

Reason: Word "no" has vernegative weight, caused wrong prediction, overweighed positive words

2) This is the dumbest, shitchiest movie on record about an aspiring writer's coming of age.

Reason: dumbest, sketchiest had no weights assigned despite being negative words.

3) As giddy and whimsical and relevant today as it was 270 years ago.

Reason: Words "as, was" are very negatively weighted, possible overfitting regarding these words.

4) It's impossible to even categorize this as a smutty guilty pleasure.

Reason: Uses double negative, thinks "pleasure" means positive but phrasing makes negative

5) At least it's a fairly impressive debute from the director, charles stone iii.

Reason: Word "impessive" used, but sarcastically, doesn't pick up on sarcasm.

3f) Use n-grams to be feature extractor.

n=7 gave me smallest error, produced test error of 27% and training error of 0%.
Error on par w/ word feature since n-gram features can see words next to each other.

Review: "This film was not good"

- The word feature extractor would weight the word good positively, say this is positive review.
- The n-grams w/ length n=7 would take feature "notgood", would weight that negatively as the words 'not' and 'good' next to each other could be weighted negatively.

4a) $\phi(x_1) = [1,0]$
$\phi(x_2) = [1,2]$
$\phi(x_3) = [3,0]$
$\phi(x_4) = [2,2]$

$1 \to \mu_1 = [2,3]$ $\mu_2 = [2,-1]$

Assignment:

	d_1	d_2	closest:
$\phi(x_1) \to$	① $\sqrt{10}^2$	② $\sqrt{2}^2$	μ_2
$\phi(x_2) \to$	① $\sqrt{2}^2$	② $\sqrt{10}^2$	μ_1
$\phi(x_3) \to$	① $\sqrt{10}^2$	② $\sqrt{2}^2$	μ_2
$\phi(x_4) \to$	① 1	② 9	μ_1

New Centers:

$$\mu_1 = \left[\frac{1+2}{2}, \frac{2+2}{2}\right] = [1.5, 2]$$
$$\mu_2 = \left[\frac{1+3}{2}, \frac{0+0}{2}\right] = [2, 0]$$

Assignment:

	d_1	d_2	closest:
$\phi(x_1)$	$\sqrt{4.25}$	$\sqrt{1}$	μ_2
$\phi(x_2)$	$\sqrt{0.25}$	$\sqrt{5}$	μ_1
$\phi(x_3)$	$\sqrt{6.25}$	$\sqrt{1}$	μ_2
$\phi(x_4)$	$\sqrt{.25}$	2	μ_1

centers are same as before: $\therefore$ Convergence

$$\boxed{\begin{array}{ll} \mu_1 = [1.5, 2] & \phi(x_1), \phi(x_3) \to \mu_2 \\ \mu_2 = [2, 0] & \phi(x_4), \phi(x_2) \to \mu_1 \end{array}}$$

Loss = 3

4a) $\mu_1 = [0, 1]$ $\mu_2 = [3, 2]$

Assignment:	d_1	d_2	Assignment
$\phi_1(x_1)$	$\sqrt{2}$	$\sqrt{8}$	μ_1
$\phi(x_2)$	$\sqrt{2}$	$\sqrt{4}$	μ_1
$\phi(x_3)$	$\sqrt{4}$	$\sqrt{10}$	μ_2
$\phi(x_4)$	$\sqrt{5}$	$\sqrt{2}$	μ_2

New Centers:

$$\mu_1 \rightarrow \left[\frac{1+1}{2}, \frac{0+2}{2}\right] = \boxed{[1, 1]}$$

$$\mu_2 \rightarrow \left[\frac{3+2}{2}, \frac{0+2}{2}\right] = \boxed{[2.5, 1]}$$

Assignment:	d_1	d_2	Assignment
$\phi(x_1)$	1	$\sqrt{3.25}$	μ_1
$\phi(x_2)$	1	$\sqrt{3.25}$	μ_1
$\phi(x_3)$	$\sqrt{5}$	$\sqrt{1.25}$	μ_2
$\phi(x_4)$	$\sqrt{2}$	$\sqrt{1.25}$	μ_2

Centers same as before $\rightarrow$ convergence:

$\mu_1 = [1, 1]$ $\phi(x_1), \phi(x_2) \rightarrow \mu_1$

$\mu_2 = [2.5, 1]$ $\phi(x_3), \phi(x_4) \rightarrow \mu_2$

Loss = 3.5

4c) <u>Purpose</u>: Create k-means algorithm using prior knowledge about pts that belong to same cluster.

<u>Algorithm:</u>

1) Initialize centroids within set of known cluster group points, to be @ centroids of known groups

   I.E. From given problem example

   μ_1 is centroid of $\phi(x_1), \phi(x_5)$

   μ_2 is centroid of $\phi(x_2), \phi(x_3), \phi(x_4)$

2) If not enough centroids generated in part 1 (i.e. $k=3$ in in problem ex. w/ more than 5 data pts available) to get K value, generate random centroids from any unconstrainted points.

3) Assign points to centers if pts are not found in prior cluster assignment based on squared reconstruction loss. Double check that assigned points. Assign based on minimizing loss for points not priorly constrained to cluster.

4) Recalculate centers of clusters using assignments stored previously, all prior constraints still must be satisfied.

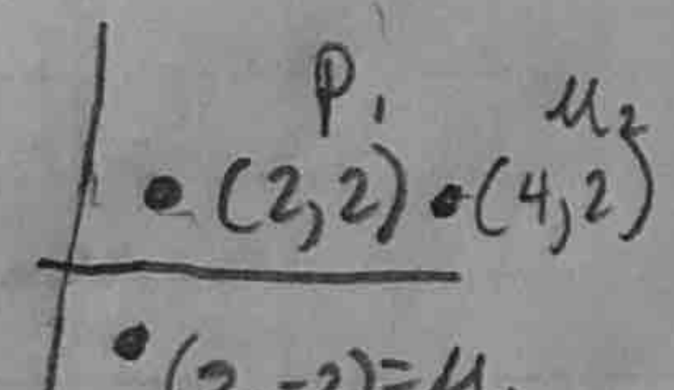
5) Repeat process until loss reaches required tolerance.

d) K-means finds a local minima, but not absolute minima for dataset. If run mulriple times w/ different random initializations, may reach a more minima value of loss.

e) If we scale all dimensions, K-means will arrive @ same clusters. This is because reconstruction loss will be altered by same $(\text{scale factor})^2$ for all points in $\mathbb{R}^d$, so centroid assignment won't change.

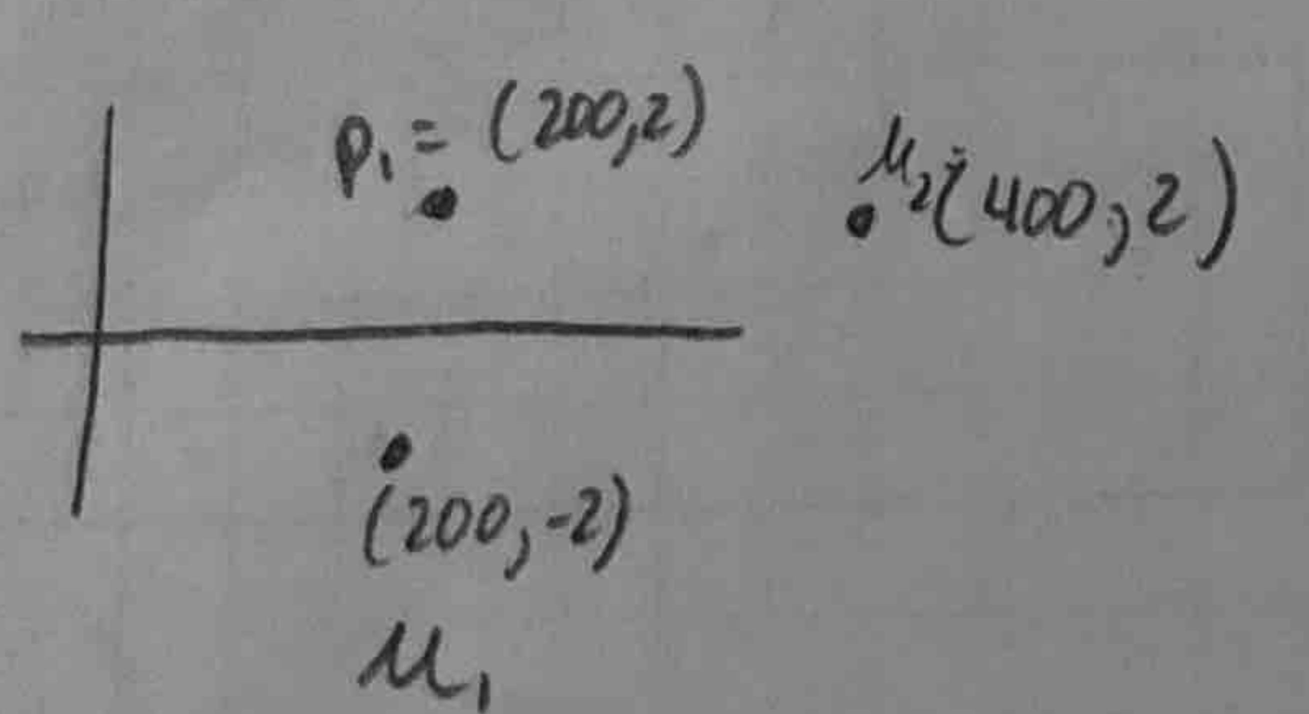
However, if only 1-d is scaled, will end up w/ different clusters. Look @ example below:

①



- In ①, p_1 in cluster w/ μ_2.

② Scale $x \cdot 100$



- Now, w/ scaled x by 100, p_1 belongs in cluster μ_1.