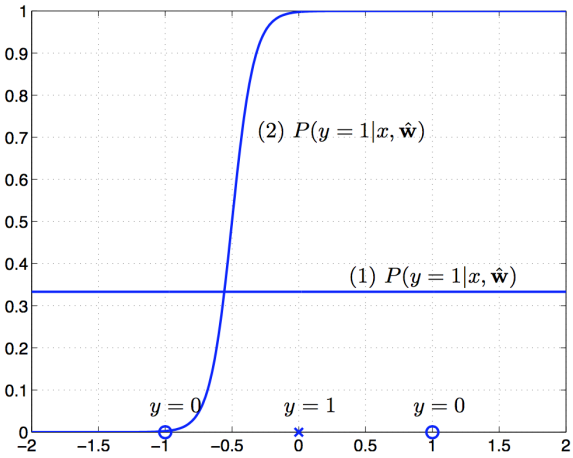
# 6.867: Exercises (Week 4)

Sept 29, 2016

## Contents

**Solution:** Don't look at the solutions until you have tried your absolute hardest to solve the problems.

# 1 Logistic regression: basic intuition with one-dimensional data




Consider a simple one dimensional logistic regression model

$$P(y = 1 \mid x, w) = \sigma(w_0 + w_1 x)$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ is the logistic function.

The figure above shows two possible conditional distributions $P(y = 1 \mid x, w)$, viewed as a function of $x$, that we can get by changing the parameters $w$.

Assume we have a data set $\mathcal{D} = \{(-1, 0), (0, 1), (1, 0)\}$.

1. Please indicate the number of classification errors for each conditional given $\mathcal{D}$.

   Conditional (1) makes ( ) classification errors

   Conditional (2) makes ( ) classification errors

2. Which of these two hypotheses assigns a higher likelihood to the data?

3. If your loss function for predictions was

$$L(g, a) = \begin{cases} 0 & \text{if } g = a \\ 1 & \text{if } g = 1 \text{ and } a = 0 \\ 10 & \text{if } g = 0 \text{ and } a = 1 \end{cases}$$

   - What output would you predict for $x = -1$ when conditional (1) is the result of your learning?

- What output would you predict for $x = -1$ when conditional (2) is the result of your learning?

**Solution:** 1. Conditional (1) makes (1) classification errors, Conditional (2) makes (1) classification errors.

2. (1). Because:

$$P_1 = 0.67 \times 0.33 \times 0.67 = 0.07, P_2 = 1 \times 1 \times 0.01 = 0.01, P_1 > P_2$$

3. When $P(y = 1|x, w) > 1/11$, predict 1, otherwise, predict 0. See week 1 Exercise, question 6 for the detail.

## 2 Logistic Regression: maximum likelihood

Bishop 4.14

Show that for a linearly separable data set, the maximum likelihood solution for the logistic regression model is obtained by finding a vector $w$ whose decision boundary $w^T\phi(x) = 0$ separates the classes and then taking the magnitude of $w$ to infinity.

**Solution:**

Using Bishop's notation, the data set is a set of pairs $(\phi_n, t_n)$ $(n = 1, \ldots N)$, with the feature vector $\phi_n = \phi(x^{(n)})$, and the target value for that feature vector, $t_n$.

If the data set is linearly separable, any decision boundary separating the two classes will have the property

$$w^T\phi_n \geqslant 0 \qquad \text{if} \qquad t_n = 1,$$
$$w^T\phi_n < 0 \qquad \text{otherwise.}$$

The likelihood function can be written

$$p(\mathbf{t} \mid w) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n} \qquad (4.89)$$

where $\mathbf{t} = (t_1, \ldots, t_N)^T$ and $y_n = \sigma(w^T\phi_n)$. We can define an error function as the negative log of the likelihood:

$$Err(w) = -\ln p(\mathbf{t} \mid w) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} \qquad (4.90)$$

Moreover, from (4.90) we see that the negative log-likelihood will be minimized (i.e., the likelihood maximized) when $y_n = \sigma(w^T\phi_n) = t_n$ for all $n$. This will be the case when the sigmoid function ($\sigma(\cdot)$) is saturated, which occurs when its argument, $w^T\phi$, goes to $\pm\infty$, i.e., when the magnitude of $w$ goes to infinity.

# 3 More on Logistic regression

## 3.1 Regular guys

We are interested in regularizing the terms separately in logistic regression.

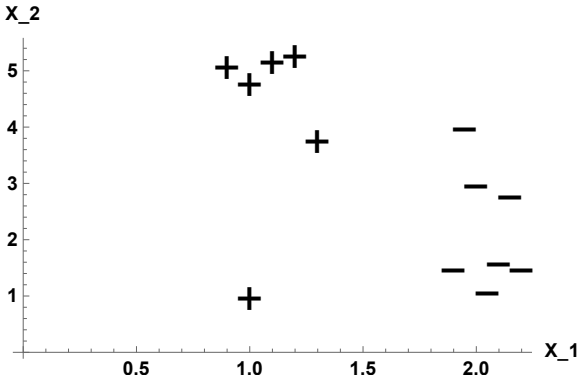
(a) Consider the data in the figure below where we fit the model

$$P(y = 1 \mid x, w) = \text{Sigmoid}(w_0 + w_1 x_1 + w_2 x_2)$$

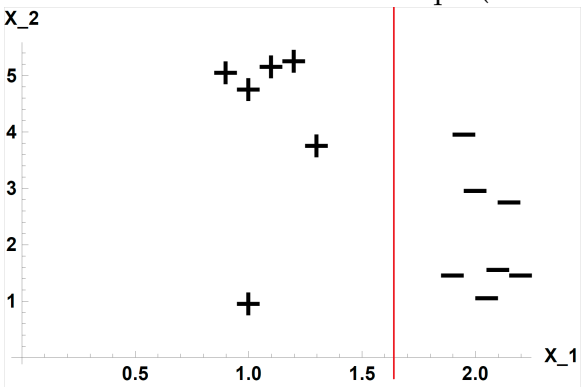
Suppose we fit the model by maximum likelihood, that is, we minimize

$$J(w) = -\log \Pr(D_{\text{train}}; w)$$

Sketch a possible decision boundary corresponding to $w^*$.




**Solution:** The solution is not unique (belows are the same). Here is one possible solution.




(b) Is your decision boundary unique?

**Solution:** No. Weights will want to go to infinity to maximize likelihood but there's room for rotation and translation.

(c) How many classification errors does it make on the training set?

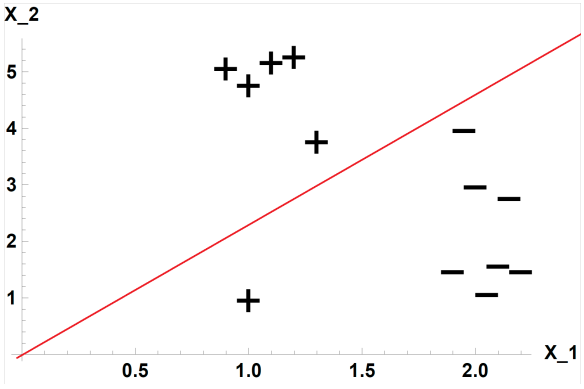
**Solution:** 0

(d) Now suppose we regularize only the $w_0$ parameter; that is, we minimize

$$J(w) = -\log \Pr(D_{\text{train}}; w) + \lambda w_0^2$$

with $\lambda$ approaching $\infty$. Sketch a possible decision boundary corresponding to $w^*$.

**Solution:**




(e) How many classification errors does it make on the training set?

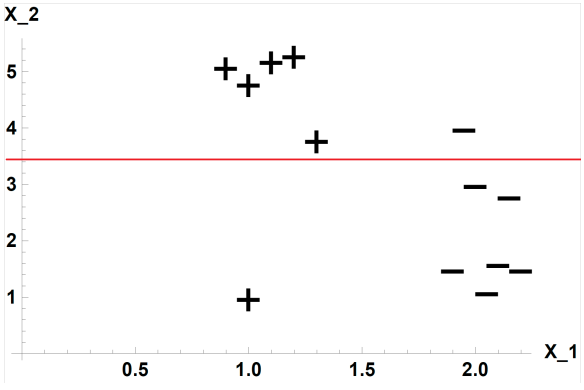
**Solution:** 1

(f) Now suppose we regularize only the $w_1$ parameter; that is, we minimize

$$J(w) = -\log \Pr(D_{\text{train}}; w) + \lambda w_1^2$$

with $\lambda$ approaching $\infty$. Sketch a possible decision boundary corresponding to $w^*$.

**Solution:**



(g) How many classification errors does it make on the training set?

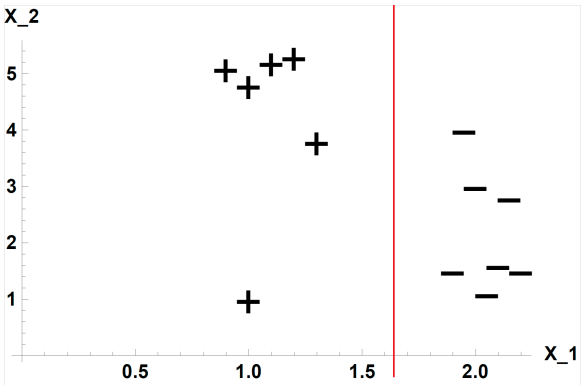
**Solution:** 2

(h) Now suppose we regularize only the $w_2$ parameter; that is, we minimize

$$J(w) = -\log \Pr(D_{\text{train}}; w) + \lambda w_2^2$$

with $\lambda$ approaching $\infty$.

**Solution:**




(i) How many classification errors does it make on the training set?

**Solution:** 0

### 3.2 Advertising!

We consider here a logistic regression model for classifying potential web ads. The class labels indicate whether the ad will generate a lot of clicks ($y = 1$) or few clicks ($y = -1$). The probabilities over the labels, given the ad $x$, are assigned according to

$$P(y = 1 \mid x, \underline{\theta}) = g(\,\underline{\theta}^T \underline{\phi}(x)\,)$$

where $g(z) = (1 + e^{-z})^{-1}$ is the logistic (sigmoid) function. The feature vectors simply indicate whether a word $w$ appears in the word $x$:

$$\phi_w(x) = \begin{cases} 1, & \text{if } x \text{ contains word } w \\ 0, & \text{otherwise} \end{cases}$$

There are only two words we are interested in so that $w \in \{\text{weird}, \text{trick}\}$. The ads are first turned into all lowercase letters before evaluating the corresponding feature vectors.

In this problem, we do not add an extra "1" to the feature vectors; they have only 2 components and therefore the weight vectors will only have two components.

We would like to train the logistic regression model based on past ads $x_1, \ldots, x_n$ and labels $y_1, \ldots, y_n$ (from focus group ratings) by maximizing the regularized log-likelihood of the labels:

$$\sum_{t=1}^{n} \log P(y_t \mid x_t, \underline{\theta}) - \frac{\lambda}{2}\|\underline{\theta}\|^2 = \sum_{t=1}^{n} \log g(\, y_t \underline{\theta}^T \underline{\phi}(x_t)\,) - \frac{\lambda}{2}\|\underline{\theta}\|^2$$

The problem is a bit hard to solve well, however, since we only had enough money to acquire three labeled ads:

$$\begin{aligned} \underline{\phi}(x_1) &= [1,1]^T \quad y_1 = 1 \\ \underline{\phi}(x_2) &= [1,0]^T \quad y_2 = -1 \\ \underline{\phi}(x_3) &= [0,0]^T \quad y_3 = 1 \end{aligned}$$

(a) Does it matter how the third ad is labeled?

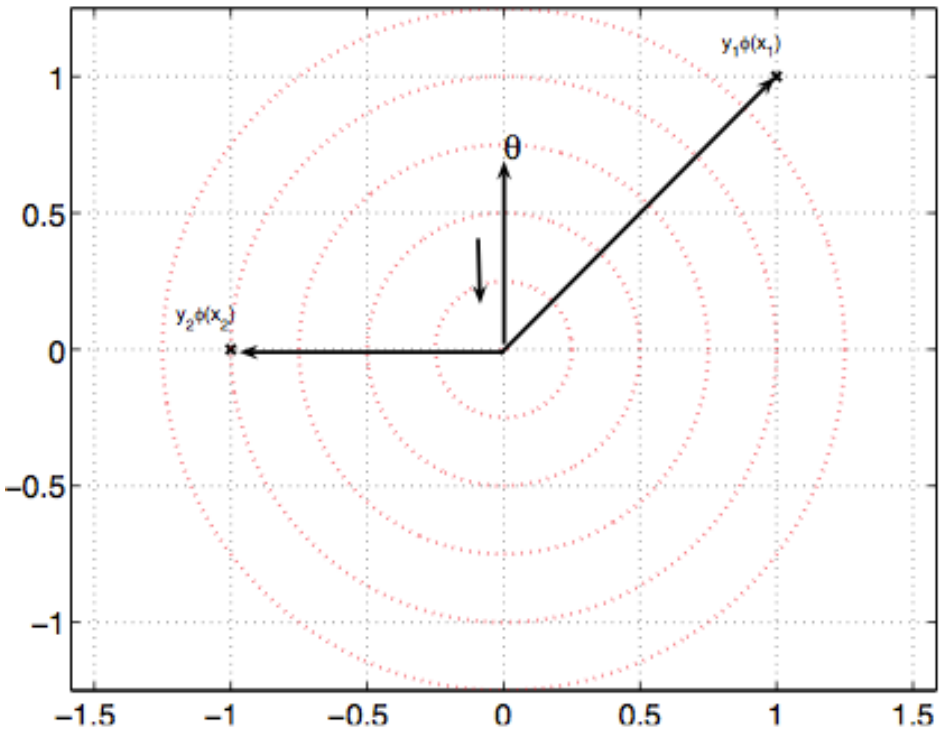
(b) What would be the value of the resulting training log-likelihood be if we set $\lambda = 0$?

(c) The logistic regression model associates class probabilities with each point. Does the effect of the regularization penalty on these probabilities depend on the norms $\|\underline{\phi}(x_t)\|$?

(d) For large $\lambda$ (strong regularization), the log-likelihood terms will behave as linear functions of $\underline{\theta}$ (see Figure 4).

$$\log g(\, y_t \underline{\theta}^T \underline{\phi}(x_t)\,) \approx \frac{1}{2} y_t \underline{\theta}^T \underline{\phi}(x_t)$$

In this regime (large $\lambda$), draw how $\hat{\underline{\theta}}$ behaves as a function of $\lambda$. In other words, draw $\hat{\underline{\theta}}$ (at any scale) and its direction of change with increasing $\lambda$. We will classify correctly only one of the training examples. Why?



Points $y_1\underline{\phi}(x_1)$ and $y_2\underline{\phi}(x_2)$ along with the contours of the regularization term $\|\underline{\theta}\|$. The solution $\hat{\underline{\theta}}$ is scaled down towards zero by the increasing $\lambda$

(e) For this particular case, but for general $\lambda > 0$, will the resulting classification decisions (predicted labels) for new exams depend on the value of $\lambda$? (Assuming the loss is symmetric, that is $y = 1$ if $P(y = 1|x, \underline{\theta}) > 0.5$)

**Solution:**

(a) No. $y_3\underline{\theta}^t\underline{\phi}(x_3)$ is always zero.

(b) log(1/2). Exams one and two would be classified correctly with probability one (no log-loss). We are, however, forced to assign probability 1/2 to each possible label for the third exam.

(c) Yes, the magnitude of the argument to the sigmoid depends on $\|\underline{\phi}(x_t)\|$.

(d) Because the solution $\hat{\underline{\theta}}$ is affected by the norms of the feature vectors and $\underline{\phi}(x_1)$ has the larger norm. See Figure 4 where $\hat{\underline{\theta}}$ is a scaled down version of $y_1\underline{\phi}(x_1) + y_2\underline{\phi}(x_2)$.

(e) Yes. The ratio $\frac{\theta_1}{\underline{\theta}_2}$ changes when $\lambda$ changes.

### 3.3 Being greedy

Here we will look at methods for selecting input features for a logistic regression model

$$P(y = 1 \mid \mathbf{x}, w) = g(w_0 + w_1x_1 + w_2x_2)$$

The available training examples are very simple, involving only binary valued inputs:

| Number of copies | $x_1$ | $x_2$ | $y$ |
|---|---|---|---|
| 10 | 1 | 1 | 1 |
| 10 | 0 | 1 | 0 |
| 10 | 1 | 0 | 0 |
| 10 | 0 | 0 | 1 |

So, for example, there are 10 copies of $\mathbf{x} = [1, 1]^T$ in the training set, all labeled $y = 1$. The correct label is actually a deterministic function of the two features: $y = 1$ if $x_1 = x_2$ and zero otherwise.

We define greedy selection in this context as follows: we start with no features (train only with $w_0$) and successively try to add new features provided that each addition strictly improves the training log-likelihood. We use no other stopping criterion.

1. Could greedy selection add either $x_1$ or $x_2$ in this case? Answer Y or N.
2. What is the classification error of the training examples that we could achieve by including both $x_1$ and $x_2$ in the logistic regression model?
3. Suppose we define another possible feature to include, a function of $x_1$ and $x_2$. Which of the following features, if any, would permit us to correctly classify all the training examples when used in combination with $x_1$ and $x_2$ in the logistic regression model:

- $x_1 - x_2$
- $x_1 x_2$
- $x_2^2$

4. Could the greedy selection method choose this feature as the first feature to add when the available features are $x_1$, $x_2$ and your choice of the new feature? Answer Y or N.

**Solution:**

1. No. Because each single feature does not review any information about outcome $y$.

2. 0.25. One of the best models we can have is like: $P(y = 0|\mathbf{x}, w) = exp(-(x_1 + x_2 - 0.5))^{-1}$, which will predict

| $x_1$ | $x_2$ | $y$ | Correct if? |
|---|---|---|---|
| 1 | 1 | 0 | wrong |
| 0 | 1 | 0 | correct |
| 1 | 0 | 0 | correct |
| 0 | 0 | 1 | correct |

3. Only $x_1 x_2$ can perfectly classify all training examples. One classification boundary could be: $y = 1$ if $10x_1x_2 - x_1 - x_2 + 0.5 > 0$.

4. Yes. The greedy algorithm works as:

- Starts with no features
- Includes $x_1$. The training error is 0.5
- Includes $x_2$. The training error reduces to 0.25
- Includes $x_1 x_2$. The training error reduces to 0

## 4 Logistic regression vs LDA/QDA

(Murphy 4.20)

Suppose we train the following binary classifiers via maximum likelihood.

(a) GaussI: A generative classifier, where the class conditional densities are Gaussian, with both covariance matrices set to I (identity matrix), i.e., $p(\mathbf{x} \mid y = c) = \mathcal{N}(\mathbf{x} \mid \mu_c, I)$. We assume $p(y)$ is uniform.

(b) GaussX: as for GaussI, but the convariance matrices are unconstrained, i.e., $p(\mathbf{x} \mid y = c) = \mathcal{N}(\mathbf{x} \mid \mu_c, \Sigma_c)$.

(c) LinLog: A logistic regression model with linear features.

(d) QuadLog: A logistic regression model, using linear and quadratic features (i.e., polynomial basis function expansion of degree 2).

After training we compute the performance of each model M on the training set as follows:

$$L(M) = \frac{1}{n}\sum_{i=1}^{n} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}, \hat{\theta}, M)$$

(Note that this is the conditional log-likelihood $p(y \mid \boldsymbol{x}, \hat{\theta}, M)$ and not joint log-likelihood $p(y, \boldsymbol{x} \mid \hat{\theta}, M)$. We now want to compare the performance of each model. We will write $L(M) \leqslant L(M')$ if model M must have lower or equal log likelihood on the training set than $M'$, for any training set (in other words, M is worse than $M'$, at least as far as training set logprob is concerned). For each of the following model pairs, state whether $L(M) \leqslant L(M'), L(M) \geqslant L(M')$, or whether no such statement can be made (i.e., M might sometimes be better than $M'$ and sometimes worse); also, for each question, briefly (1-2 sentences) explain why.

(a) GaussI, LinLog

(b) GaussX, QuadLog

(c) LinLog, QuadLog

(d) GaussI, QuadLog

(e) Now suppose we measure performance in terms of the average misclassification rate on the training set:

$$R(M) = \frac{1}{n}\sum_{i=1}^{n} I(y^{(i)} \neq \hat{y}(\boldsymbol{x}^{(i)}))$$

where $\hat{y}(x^{(i)})$ is the predicted $y$ for $x^{(i)}$. Is it true in general that $L(M) > L(M')$ implies that $R(M) < R(M')$? Explain why or why not.

**Solution:**

(a) GaussI $\leqslant$ LinLog. Both have logistic (sigmoid) posteriors $p(y \mid \boldsymbol{x}, \boldsymbol{w}) = \sigma(y\boldsymbol{w}^T\boldsymbol{x})$, but LinLog is the logistic model which is trained to maximize $p(y \mid \boldsymbol{x}, \boldsymbol{w})$. (GaussI may have high joint $p(y, \boldsymbol{x})$, but this does not necessarily mean $p(y \mid \boldsymbol{x})$ is high; LinLog can achieve the maximum of $p(y \mid \boldsymbol{x})$, so will necessarily do at least as well as GaussI.)

(b) GaussX $\leqslant$ QuadLog. Both have logistic posteriors with quadratic features, but QuadLog is the model of this class maximizing the average log probabilities.

(c) LinLog $\leqslant$ QuadLog. Logistic regression models with linear features are a subclass of logistic regression models with quadratic functions. The maximum from the superclass is at least as high as the maximum from the subclass.

(d) GaussI $\leqslant$ QuadLog. Follows from above inequalities.

(e) Although one might expect that higher log likelihood results in better classification performance, in general, having higher average $\log p(y \mid \boldsymbol{x})$ does not necessarily translate to higher or lower classification error. For example, consider linearly separable data. We have L(LinLog) $>$ L(GaussI), since maximum likelihood logistic regression will set

the weights to infinity, to maximize the probability of the correct labels (hence $p(y^{(i)} \mid \boldsymbol{x}^{(i)}, \hat{\boldsymbol{w}}) = 1$ for all $i$). However, we have $R(\text{LinLog}) = R(\text{GaussI})$, since the data is linearly separable. (The GaussI model may or may not set $\sigma$ very small, resulting in possibly very large class conditional pdfs; however, the posterior over $y$ is a discrete pmf, and can never exceed 1.)

As another example, suppose the true label is always 1 (as opposed to 0), but model $M$ always predicts $p(y = 1 \mid \boldsymbol{x}, M) = 0.49$. It will always misclassify, but it is at least close to the decision boundary. By contrast, there might be another model $M'$ that predicts $p(y = 1 \mid \boldsymbol{x}, M') = 1$ on even-numbered inputs, and $p(y = 1 \mid \boldsymbol{x}, M') = 0$ on odd-numbered inputs. Clearly $R(M') = 0.5 < R(M) = 1$, but $L(M') = -\infty < L(M) = \log(0.49)$.

## 5 Softmax

Another possible approach to classification is to use a generalized version of the logistic model. Let $x = [x_1, x_2, \ldots, x_d]$ be an input vector, and suppose we would like to classify into $k$ classes; that is, the output $y$ can take a value in $1, \ldots, k$. The softmax generalization of the logistic model uses $k(d+1)$ parameters $\theta = (\theta_{ij}), i = 1, \ldots, k, j = 0, \ldots, d$, which define the following $k$ intermediate values:

$$\begin{aligned} z_1 &= \theta_{10} + \sum_j \theta_{1j} x_j \\ &\ldots \\ z_i &= \theta_{i0} + \sum_j \theta_{ij} x_j \\ &\ldots \\ z_k &= \theta_{k0} + \sum_j \theta_{kj} x_j \end{aligned}$$

The classification probabilities under the softmax model are:

$$\Pr(y = i \mid x; \theta) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad .$$

1. Show that when $k = 2$ the softmax model reduces to the logistic model. That is, show how both give rise to the same classification probabilities $\Pr(y \mid x)$. Do this by constructing an explicit transformation between the parameters: for any given set of $2(d+1)$ softmax parameters, show an equivalent set of $(d+1)$ logistic parameters.

**Solution:** The posterior of a logistic model with weights $\theta'$

$$P(Y = 1 | x; \theta) = \frac{1}{1 + e^{-z'}}$$

where $z' = \theta'_0 + \sum_j \theta'_j x_j$. The posterior of the softmax model when $k = 2$,

$$P(Y = 1|x; \theta) = \frac{e^{z_1}}{e^{z_1} + e^{z_2}}$$

equating the two

$$\begin{aligned}
\frac{1}{1 + e^{-z'}} &= \frac{e^{z_1}}{e^{z_1} + e^{z_2}} \\
e^{z_1} + e^{z_2} &= e^{z_1} + e^{z_1} e^{-z'} = e^{z_1} + e^{z_1 - z'} \\
e^{z_1 - z'} &= e^{z_2} \\
z' &= z_1 - z_2
\end{aligned}$$

If $\theta'_j = \theta_{1j} - \theta_{2j}$ for each $j$, the softmax model reduces to the logistic model.

2. Which of the decision regions from question 2 can represent decision boundaries for a softmax model?

**Solution:** Only linear decision boundaries are possible.

3. Show that the softmax model, for any $k$, can always be represented by a Gaussian mixture model. What type of Gaussian mixture models are equivalent to softmax models?

**Solution:** Consider a softmax model with $k$ classes and weights $\theta_{ij}$ and denote $\theta_i$ as a $d$-element vector with components $(\theta_i)_j = \theta_{ij}$ for $1 \leqslant j \leqslant d$, then the softmax posterior is given by

$$P(y|\mathbf{x}) = \frac{e^{\theta_y^T \mathbf{x} + \theta_{y0}}}{\sum_i e^{\theta_i^T \mathbf{x} + \theta_{i0}}}$$

We would like to find a Gaussian mixture model $(\pi_i, \mu_i, \Sigma_i)_{i=1..k}$ that yields the same posterior. The posterior of a $k$-component Gaussian mixture model (GMM) is given by

$$\begin{aligned}
P(y|\mathbf{x}; (\pi_i, \mu_i, \Sigma_i)_{i=1..k}) &= \frac{\pi_y |\Sigma_y|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x} - \mu_y)^T \Sigma_y^{-1} (\mathbf{x} - \mu_y)}}{\sum_i \pi_i |\Sigma_i|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)}} \\
&= \frac{\pi_y |\Sigma_y|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}^T \Sigma_y^{-1} \mathbf{x} - \mathbf{x}^T \Sigma_y^{-1} \mu_y - \mu_y^T \Sigma_y^{-1} \mathbf{x} + \mu_y^T \Sigma_y^{-1} \mu_y)}}{\sum_i \pi_i |\Sigma_i|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}^T \Sigma_i^{-1} \mathbf{x} - \mathbf{x}^T \Sigma_i^{-1} \mu_i - \mu_i^T \Sigma_i^{-1} \mathbf{x} + \mu_i^T \Sigma_i^{-1} \mu_i)}}
\end{aligned}$$

Since covariance matrices are symmetric

$$= \frac{\pi_y |\Sigma_y|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}^T \Sigma_y^{-1} \mathbf{x} - 2\mu_y^T \Sigma_y^{-1} \mathbf{x} + \mu_y^T \Sigma_y^{-1} \mu_y)}}{\sum_i \pi_i |\Sigma_i|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}^T \Sigma_i^{-1} \mathbf{x} - 2\mu_i^T \Sigma_i^{-1} \mathbf{x} + \mu_i^T \Sigma_i^{-1} \mu_i)}}$$

The exponents are quadratic to $\mathbf{x}$. But in the softmax posterior the exponents are linear to $\mathbf{x}$. In order to make them equal, we need to choose identical covariance matrices to cancel out quadratic terms as follows

$$\begin{aligned}
&= \frac{\pi_y |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}^T\Sigma^{-1}\mathbf{x} - 2\mu_y^T\Sigma^{-1}\mathbf{x} + \mu_y^T\Sigma^{-1}\mu_y)}}{\sum_i \pi_i |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}^T\Sigma^{-1}\mathbf{x} - 2\mu_i^T\Sigma^{-1}\mathbf{x} + \mu_i^T\Sigma^{-1}\mu_i)}} \\
&= \frac{\pi_y e^{-\frac{1}{2}(-2\mu_y^T\Sigma^{-1}\mathbf{x} + \mu_y^T\Sigma^{-1}\mu_y)}}{\sum_i \pi_i e^{-\frac{1}{2}(-2\mu_i^T\Sigma^{-1}\mathbf{x} + \mu_i^T\Sigma^{-1}\mu_i)}} \\
&= \frac{e^{\mu_y^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_y^T\Sigma^{-1}\mu_y + \log\pi_y}}{\sum_i e^{\mu_i^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i + \log\pi_i}}
\end{aligned}$$

To make the linear coefficients of $\mathbf{x}$ equal to the weights in the softmax posterior, we need to set the means and covariances such that

$$\mu_i^T\Sigma^{-1} = \theta_i \quad \text{for all } i$$

where $\Sigma$ must be invertible. To satisfy this condition, let $\Sigma = I$ and $\mu_i = \theta_i$. The GMM posterior of becomes

$$P(y|\mathbf{x}; (\pi_i, \mu_i = \theta_i, \Sigma_i = I)_{i=1..k}) = \frac{e^{\theta_y^T\mathbf{x} - \frac{1}{2}\theta_y^T\theta_y + \log\pi_y}}{\sum_i e^{\theta_i^T\mathbf{x} - \frac{1}{2}\theta_i^T\theta_i + \log\pi_i}}$$

Note that if setting $\log\pi_i - \frac{1}{2}\theta_i^T\theta = \theta_{i0}$, we would obtain exactly the same representation as the softmax posterior. However, this might result in negative priors or priors that do not sum up to one. One way to solve this is to multiply both the numerator and denominator by a constant $Z$

$$P(y|\mathbf{x}; (\pi_i, \mu_i = \theta_i, \Sigma_i = I)_{i=1..k}) = \frac{Z\pi_y e^{\theta_y^T\mathbf{x} - \frac{1}{2}\theta_y^T\theta_y}}{\sum_i Z\pi_i e^{\theta_i^T\mathbf{x} - \frac{1}{2}\theta_i^T\theta_i}}$$

Then by setting $\pi_i = \frac{e^{\frac{1}{2}\theta_i^T\theta + \theta_{i0}}}{Z}$ and $Z = \sum_i e^{\frac{1}{2}\theta_i^T\theta + \theta_{i0}}$ we convert the softmax weights to a Gaussian mixture model where

$$\begin{aligned}
\pi_i &= \frac{e^{\frac{1}{2}\theta_i^T\theta + \theta_{i0}}}{\sum_i e^{\frac{1}{2}\theta_i^T\theta + \theta_{i0}}} \\
\mu_i &= \theta_i \\
\Sigma_i &= I
\end{aligned}$$

Remember that the convariance matrix $\Sigma$ must be 1) invertible 2) identical for all Gaussian components. Therefore, the softmax model can only be reduced to a special case of Gaussian mixture models.

4. A stochastic gradient ascent learning rule for softmax is given by:

$$\theta_{ij} \leftarrow \theta_{ij} + \alpha \sum_t \frac{\partial}{\partial \theta_{ij}} \log \Pr(y^t \mid x^t; \theta) \ ,$$

where $(x^t, y^t)$ are the training examples. We would like to rewrite this rule as a delta rule. In a delta rule the update is specified as a function of the difference between the target and the prediction. In our case, our target for each example will actually be a vector $y^t = (y_1^t, \ldots, y_k^t)$ where $y_i^t = 1$ if $y^t = i$ and 0 otherwise.

Our prediction will be a corresponding vector of probabilities:

$$\hat{y}^t = (\Pr(y = 1 \mid x^t; \theta), \ldots, \Pr(y = k \mid x^t; \theta))$$

Calculate the derivative above, and rewrite the update rule as a function of $y - \hat{y}$.

**Solution:** Sometimes it is easier to calculate derivatives in log-scale.

$$\log P(y = i) = z_i - \log \sum_l e^{z_l}$$

$$\frac{\partial z_i}{\partial \theta_{ij}} = x_j$$

Two cases

$$\frac{\partial \log P(y = i)}{\partial \theta_{ij}} = 1 \cdot x_j - \frac{e^{z_i}}{\sum_l e^{z_l}} x_j = y_i x_j - \hat{y}_i x_j$$

$$\frac{\partial \log P(y = k \neq i)}{\partial \theta_{ij}} = 0 \cdot x_j - \frac{e^{z_i}}{\sum_l e^{z_l}} x_j = y_i x_j - \hat{y}_i x_j$$

Combining them in the vector form

$$\frac{\partial \log P(y^t | \mathbf{x}^t)}{\partial \theta_{ij}} = y_i^t x_j - \hat{y}_i^t x_j = (y^t - \hat{y}^t)^T \mathbf{x}^t$$

Therefore, the update rule is

$$\theta \leftarrow \theta + \alpha \sum_t (y^t - \hat{y}^t)^T \mathbf{x}^t$$

## 6 SVMs: basic intuition with one-dimensional data

Assume that our training data is four 1-dimensional points, as follows:

| index | x | y |
|---|---|---|
| 1 | -2 | -1 |
| 2 | -0.1 | -1 |
| 3 | 0.1 | 1 |
| 4 | 1 | 1 |

(a) Find the values of all the $\alpha_i$ that would be found by the (linear) SVM training algorithm. You should be able to do this without going through the Lagrangian minimization procedure. Think about the conditions for the optimization directly.

**Solution:** Obviously, the support vectors are $x_2$ and $x_3$. We saw during the derivation of SVM's that $w(x_3 - x_2) = 2$, that is, $w(0.2) = 2$ and so the weight is $w = 10$. Since $w = 0.1\alpha_2 + 0.1\alpha_3$ and $\alpha_2 = \alpha_3$, we have that $w = 0.2\alpha_2$ and so

$$\alpha_2 = \alpha_3 = 50$$

(b) What would the offset be for these values of $\alpha_i$?

**Solution:** We know the margin of the support vectors must be 1, so $y_3(wx_3 + b) = 1$ so:

$$b = 0$$

(c) What if the value of C were set to 1? What would happen to the values of $\alpha_i$ and the offset? Explain.

**Solution:**

If C=1, then in this case all the points will become support vectors. Once the separator is not halfway between points 2 and 3, then one point is within the margin and has alpha =1, so the sum of the alphas on one side is > 1. Therefore, the sum of the alphas on the other side must be > 1 (since they have to be equal on both sides) and so you can see that all the points are sv's and that two of them have alpha = 1 and the other two are less than one and equal.

The detailed values need more calculation to discover.

```
>> s.Alpha  (values of y*alpha)
ans =
 0.1556
 1.0000
 -1.0000
 -0.1556
>> s.Bias
ans =
 -0.3333
```

# 7 SVMs: Where's the hyperplane?

(Bishop 7.3) Show that, irrespective of the dimensionality of the data space, a data set consisting of just two data points, one from each class, is sufficient to determine the location of the maximum-margin hyperplane.

**Solution:**

Given a data set of two data points, $\mathbf{x}^{(1)} \in C_-(y^{(1)} = -1)$ and $\mathbf{x}^{(2)} \in C_+(y^{(2)} = +1)$, the maximum margin hyperplane is determined by solving

$$\arg\min_{w,b} \frac{1}{2}\|\mathbf{w}\|^2$$

subject to the constraints

$$y^{(1)}\{\mathbf{w}^T\mathbf{x}^{(1)} + b\} - 1 \geqslant 0 \quad (1)$$

$$y^{(2)}\{\mathbf{w}^T\mathbf{x}^{(2)} + b\} - 1 \geqslant 0 \quad (2)$$

We do this by introducing Lagrange multipliers $\alpha_1$ and $\alpha_2$, and solving

$$\arg\min_{w,b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \alpha_1(-\{\mathbf{w}^T\mathbf{x}^{(1)} + b\} - 1) - \alpha_2(\{\mathbf{w}^T\mathbf{x}^{(2)} + b\} - 1) \right\} \quad (3)$$

Taking the derivative of this w.r.t. $\mathbf{w}$ and $b$ and setting the results to zero, we obtain

$$\begin{aligned} 0 &= \mathbf{w} + \alpha_1\mathbf{x}^{(1)} - \alpha_2\mathbf{x}^{(2)} \quad (4) \\ 0 &= -\alpha_1 + \alpha_2 \quad (5) \end{aligned}$$

We can now express the Lagrangian (3) as a function of $\alpha_2$ by substituting $\mathbf{w} = \alpha_1(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})$ from (4, 5) and $\alpha_1 = \alpha_2$ from (5).

$$\begin{aligned} L(\alpha_2) &= \frac{1}{2}\|\alpha_2(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})\|^2 + \alpha_2(\alpha_2(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})^T\mathbf{x}^{(1)} + b + 1) - \alpha_2(\alpha_2(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})^T\mathbf{x}^{(2)} + b - 1) \\ &= \frac{1}{2}\alpha_2^2\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\|^2 - \alpha_2^2\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\|^2 + 2\alpha_2 \\ &= -\frac{1}{2}\alpha_2^2\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\|^2 + 2\alpha_2 \quad (6) \end{aligned}$$

Differentiating $L(\alpha_2)$ w.r.t $\alpha_2$ and setting equal to 0 gives:

$$\begin{aligned} 0 &= -\alpha_2\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\|^2 + 2 \\ \alpha_2 &= \frac{2}{\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\|^2} \quad (7) \end{aligned}$$

Since $\alpha_1 = \alpha_2$, we can now use (4) to determine $\mathbf{w}$. We get $b$ by substituting $\mathbf{w}$ into (1) or (2) and solving for $b$ (with equality). Or, better still, averaging the $b$ values from both of them.

## 8 SVMs: primal scream

The primal *soft-margin* SVM optimization problem given by

$$\begin{aligned} \text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \\ y^{(i)}(w\cdot x^{(i)} + w_0) \geqslant 1-\xi_i, \;\; i=1,\ldots,n \\ \xi_i \geqslant 0, \;\; i=1,\ldots,n \end{aligned}$$

involves both $\mathbf{w}, w_0$ and the slack variables $\xi_i$. We can rewrite the optimization problem in terms of $\mathbf{w}, w_0$ alone. This is done by explicitly solving for the optimal values of the slack variables $\xi_i = \xi_i(\mathbf{w}, w_0)$ as functions of $\mathbf{w}, w_0$. The values of these slack variables, as functions of $\mathbf{w}, w_0$, are "loss-functions" as shown below. What functions $\xi_i(\mathbf{w}, w_0)$ determine the optimal $\xi_i$? Are all the margin constraints satisfied with these expressions for the slack variables?

The resulting minimization problem over $\mathbf{w}, w_0$ can be formally written as

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i(\mathbf{w}, w_0)$$

where the first (regularization) term biases our solution towards zero in the absence of any data and the remaining terms give rise to the loss functions, one loss function per training point, encouraging correct classification. Do we need any additional constraints? Many learning criteria can be understood and compared in the above regularization + loss form.

**Solution:**

From the geometric formulation of SVM, it is clear that the value of the slack variable is exactly the distance of the of the support vectors from the margin. Therefore, for support vectors, we have $\xi_i/\|w\| = (y_i/\|w\| - d)$ where $d = |(w^T x_i + w_0)|/\|w\|$ is the distance of a point to the decision boundary.

In order to rewrite the slack variables $\xi_i$ in the form of a loss function we start by observing that if $y \in \{-1, 1\}$ then $\xi_i = |y_i - (wx_i + w_0)| = |1 - y_i(wx_i + w_0)|$ unless $y_i(wx_i + w_0) \geqslant 1$, in which case $\xi_i = 0$. So only the positive part of the absolute value remains and we can write $\xi_i = \max\{1 - y_i(wx_i + w_0), 0\}$. From this definition it follows that $\xi_i \geqslant 0$ and $\xi_i \geqslant 1 - y_i(wx_i + w_0)$ so the two formulations are exactly equivalent and we can re-write the primal formulation of SVM as $\min_{w,w_0} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\max\{1 - y_i(wx_i + w_0), 0\}$. Note that we have a fitting term and an $L_2$ regularization penalty on the weight vector much like what we had for ridge regression or regularized logistic regression.

# 9 SVMs: modifying basic formulation

## 9.1 Lopsided

We are trying to solve a classification problem with support vector machines. In our problem there are only a few positive training examples and we are certain that they are classified correctly. We also have a large number of negative training examples, some of which may be misclassified. We'd like to modify the basic dual form of the SVM optimization problem,

$$(1)\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)^T} \mathbf{x}^{(j)} (2) \text{ subject to} \quad \alpha_i \geqslant 0, \ \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

to better solve this type of problem. We would like to ensure that we won't misclassify any of the positive examples but could misclassify some of the negative examples. We believe you have to introduce additional parameter(s) (or constants for the purpose of solving the quadratic programming problem) in order to achieve this. In your solution, please use $I_+$ to index positively labeled examples ($y_i = +1$) and $I_-$ for negative examples ($y_i = -1$). In other words, $i \in I_+$ means that $y_i = +1$, and $|I_+|$ is the number of positive examples.

1. Optimization problem:
   maximize:
   subject to:
   Your solution must be in the dual form. You can refer to (1) and (2) above.

   > **Solution:** Maximize: (1) as above
   >
   > subject to: (2) and $\alpha_i \leqslant C^-$ for $i \in I_-$ (negative examples).
   > In other words, we limit how strongly the margin constraints are enforced for the negative examples. Positive examples have no such limit and the classifier will have to satisfy the margin constraints exactly for the positive examples.

2. Which of the following alternative criteria would work for optimizing your new parameters. We have underlined any differences between the alternatives.

   (a) We train your SVM algorithm $|I_+|$ times, each time leaving out one of the positive examples, and testing the classifier on the left out example. The parameter(s) are set to minimize the resulting number of misclassified examples.

   > **Solution:** Since we are only focusing on how well the positive examples are classified, setting $C^- = 0$ would be optimal. As a result, we wouldn't enforce any classification constraints on the negative examples.

   (b) We train your SVM algorithm $|I_-|$ times, each time leaving out one of the negative examples, and testing the classifier on the left out example. The parameter(s) are set to minimize the resulting number of misclassified examples. Briefly explain why this would or would not work:

**Solution:** The optimization problem described above strictly enforces the classification constraints for the positive examples. Thus no matter how we set $C^-$ it won't be possible to misclassify any of them on the training set. However, focusing solely on the negative examples will not try to gauge how well we generalize in terms of classifying positive examples. We are simply trying to generalize well in terms of correctly classifying negative examples (by optimizing this cross-validation error) with the constraint that we still have have to classify all the positive training examples correctly.

(c) We train your SVM algorithm n times, each time leaving out one of the examples, positive or negative, and testing the classifier on the left out example. The constant is set to minimize the resulting number of misclassified examples.

**Solution:** This is the standard cross-validation error and would work here as well, since we are still trying to measure generalization error in both positives and negatives.

### 9.2 Data fusion

You are still trying to build a classifier with data that you gathered on two different days with two different instruments. You trust the labels of the data gathered with instrument 1 twice as much as the labels of the data gathered with instrument 2. You have lots of friends with different opinions about how to handle this.

We will use $(x^i, y^i), i = 1 \dots n$ to denote data from instrument 1 (more accurate) and $(u^i, v^i)$ to denote data from instrument 2. Slack variables for the instrument 1 data will be $\xi$ and for the instrument 2 data will be $\zeta$. Lagrange multipliers for the instrument 1 data will be $\alpha$ and for the instrument 2 data will be $\beta$.

- Pat suggests that you can insert a multiplier of 2 into the slack penalties for the data points gathered with instrument 1, so that the optimization problem is

$$\min_{\theta,\xi,\zeta} \frac{1}{2}\|\theta\|^2 + 2c\sum_{i=1}^{n}\xi_i + c\sum_{j=1}^{m}\zeta_j$$

subject to

$$\begin{aligned} y^i(\theta \cdot x^i + \theta_0) &\geqslant 1 - \xi_i \quad \text{for all } i \in \{1, \dots, n\} \\ v^j(\theta \cdot u^j + \theta_0) &\geqslant 1 - \zeta_j \quad \text{for all } j \in \{1, \dots, m\} \\ \xi_i &\geqslant 0 \quad \text{for all } i \in \{1, \dots, n\} \\ \zeta_j &\geqslant 0 \quad \text{for all } j \in \{1, \dots, m\} \end{aligned}$$

- Dana suggests that you can insert a multiplier of 2 into the Lagrange multipliers of data points gathered with instrument 1, so that the dual optimization problem is:

$$\max_{\alpha,\beta} \sum_{i=1}^{n} 2\alpha_i + \sum_{j=1}^{m} \beta_j - 2\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j (x^i \cdot x^j) - 2\sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_i \beta_j y^i v^j (x^i \cdot u^j) - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \beta_i \beta_j v^i v^j (u^i \cdot u^j)$$

subject to

$$\begin{aligned} c \geqslant 2\alpha_i \geqslant 0 \quad &\text{for all } i \in \{1, \ldots, n\} \\ c \geqslant \beta_j \geqslant 0 \quad &\text{for all } j \in \{1, \ldots, m\} \\ \sum_{i=1}^{n} 2\alpha_i y^i + \sum_{j=1}^{m} \beta_j v^j &= 0 \end{aligned}$$

- Robin suggests that you can duplicate the points that you gathered with instrument 1 in the data set, and then proceed as usual.

1. Are these approaches equivalent, in the sense of resulting in the same separator? For each pair, show that they are equivalent or not.

**Solution:** We begin with deriving the dual form of Pat's suggestion. Using Lagrange multipliers, the new objective function is

$$\begin{aligned} L(\theta, \theta_0, a, b, e, f) = &\frac{1}{2}\|\theta\|^2 + 2c\sum_{i=1}^{n} \xi_i + c\sum_{j=1}^{m} \zeta_j - \sum_{i=1}^{n} a_i \left[y^i(\theta \cdot x^i + \theta_0) - 1 + \xi_i\right] \\ &- \sum_{i=1}^{m} b_i \left[v^i(\theta \cdot u^i + \theta_0) - 1 + \zeta_i\right] - \sum_{i=1}^{n} e_i \xi_i - \sum_{i=1}^{m} f_i \zeta_i \end{aligned}$$

subject to

$$\begin{aligned} &a_i \geqslant 0, b_i \geqslant 0, e_i \geqslant 0, f_i \geqslant 0 \\ &y^i(\theta \cdot x^i + \theta_0) - 1 + \xi_i \geqslant 0 \\ &v^i(\theta \cdot u^i + \theta_0) - 1 + \zeta_i \geqslant 0 \\ &e_i \xi_i = 0, f_i \zeta_i = 0 \\ &a_i \left[y^i(\theta \cdot x^i + \theta_0) - 1 + \xi_i\right] = 0 \\ &b_i \left[v^i(\theta \cdot u^i + \theta_0) - 1 + \zeta_i\right] = 0 \end{aligned}$$

Optimizing $\theta, \theta_0, \xi_i, \zeta_i$

$$\frac{\partial L}{\partial \theta} = 0 \implies \theta = \sum_{i=1}^{n} a_i y^i x^i + \sum_{i=1}^{m} b_i \nu^i u^i \tag{9.1}$$

$$\frac{\partial L}{\partial \theta_0} = 0 \implies \sum_{i=1}^{n} a_i y^i + \sum_{i=1}^{m} b_i \nu^i = 0 \tag{9.2}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \implies a_i = 2c - e_i \tag{9.3}$$

$$\frac{\partial L}{\partial \zeta_i} = 0 \implies b_i = c - f_i \tag{9.4}$$

Since $a_i \geqslant 0, b_i \geqslant 0, e_i \geqslant 0, f_i \geqslant 0$, Eq. 9.3 and 9.4 become

$$2c \geqslant a_i \geqslant 0$$
$$c \geqslant b_i \geqslant 0$$

Use these results to eliminate $\theta, \theta_0, \xi_i, \zeta_i$ from the Lagrangian, we obtain the dual Lagrangian in the form

$$\bar{L}(a, b) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{m} b_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y^i y^j (x^i \cdot x^j) - \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j y^i \nu^j (x^i \cdot u^j)$$
$$- \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} b_i b_j \nu^i \nu^j (u^i \cdot u^j)$$

with constraints

$$2c \geqslant a_i \geqslant 0$$
$$c \geqslant b_i \geqslant 0$$
$$\sum_{i=1}^{n} a_i y^i + \sum_{i=1}^{m} b_i \nu^i = 0$$

Let $\alpha_i = \frac{a_i}{4}, \beta_i = b_i$, the dual form can be rewritten as

$$\max_{\alpha, \beta} \sum_{i=1}^{n} 4\alpha_i + \sum_{i=1}^{m} \beta_i - 8 \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j (x^i \cdot x^j) - 4 \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j y^i \nu^j (x^i \cdot u^j)$$
$$- \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \beta_i \beta_j \nu^i \nu^j (u^i \cdot u^j)$$

subject to

$$c \geqslant 2\alpha_i \geqslant 0$$
$$c \geqslant \beta_i \geqslant 0$$
$$\sum_{i=1}^{n} 4\alpha_i y^i + \sum_{i=1}^{m} \beta_i \nu^i = 0$$

It is clear that Dana's suggestion is different from Pat's. Therefore, Dana $\neq$ Pat.

If we use the same slack variable for points after duplicating, Robin's suggestion can be written as

$$\min_{\theta,\xi,\zeta} \frac{1}{2}\|\theta\|^2 + c\sum_{i=1}^{n}\xi_i + c\sum_{i=n+1}^{2n}\xi_i + c\sum_{j=1}^{m}\zeta_j$$

subject to

$$\begin{aligned} y^i(\theta\cdot x^i+\theta_0) &\geqslant 1-\xi_i \quad \text{for all } i\in\{1,\ldots,2n\}\\ v^j(\theta\cdot u^j+\theta_0) &\geqslant 1-\zeta_j \quad \text{for all } j\in\{1,\ldots,m\}\\ \xi_i &\geqslant 0 \quad \text{for all } i\in\{1,\ldots,2n\}\\ \zeta_j &\geqslant 0 \quad \text{for all } j\in\{1,\ldots,m\}\end{aligned}$$

which is essentially equal to Pat's suggestion. Therefore, Robin = Pat.

In summary, we have Pat $\neq$ Dana, Robin = Pat and Dana $\neq$ Robin.

### 9.3 Objective fusion

We asked a few students to rate their midterm exam according to whether they thought it was difficult ($y = 1$), all right ($y = 2$), or easy ($y = 3$). Each student also provided us with a few pieces of information about themselves such as other courses they had taken, the program they were in, and so on. We could use this additional information to construct a feature vector $\underline{\phi}_i$ for each student $i = 1,\ldots,n$. On the basis of the rating labels, $y_1,\ldots,y_n$ and the feature vectors, $\underline{\phi}_1,\ldots,\underline{\phi}_n$, we could learn to predict how a particular type of student would react to the exam.

We decided to divide the prediction task into two binary classification tasks

Task 1: whether $y = 1$ (binary label -1) or $y > 1$ (binary label +1)

Task 2: whether $y \leqslant 2$ (binary label -1) or $y = 3$ (binary label +1)

So, we needed two binary classifiers. Since the ratings fall on an ordinal scale, it seemed wise to couple these tasks together. We opted to use common parameters $\underline{\theta}$ for the two tasks but different thresholds $b_1$ and $b_2$ for Task 1 and 2, respectively. The corresponding estimation problem is given by

$$\begin{aligned} &\text{Minimize } \frac{1}{2}\|\underline{\theta}\|^2 \text{ with respect to } \underline{\theta}, b_1, \text{ and } b_2, \text{ subject to}\\ \text{Task 1:}\quad & -1(\underline{\theta}\cdot\underline{\phi}_i - b_1) \geqslant 1 \text{ if } y_i = 1, \quad +1(\underline{\theta}\cdot\underline{\phi}_i - b_1) \geqslant 1 \text{ if } y_i > 1\\ \text{Task 2:}\quad & -1(\underline{\theta}\cdot\underline{\phi}_i - b_2) \geqslant 1 \text{ if } y_i \leqslant 2, \quad +1(\underline{\theta}\cdot\underline{\phi}_i - b_2) \geqslant 1 \text{ if } y_i = 3\\ &\text{for all } i = 1,\ldots,n\end{aligned}$$

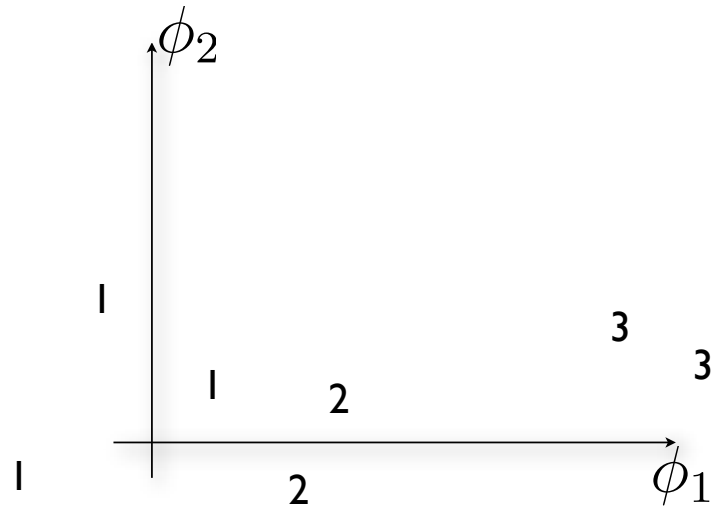
(a) Briefly explain why we would like to make sure that $b_1 \leqslant b_2$?

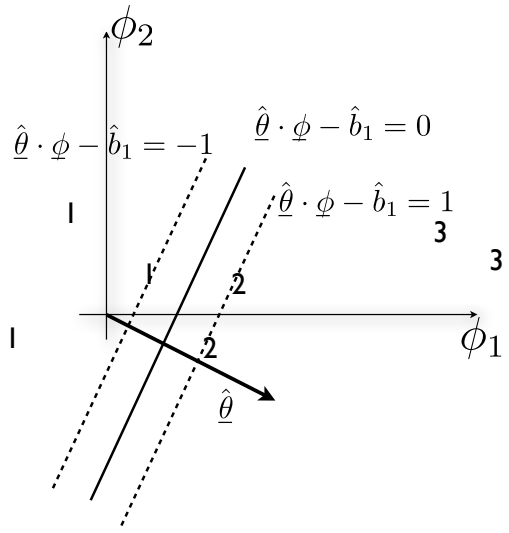
**Solution:** $b_1 \leqslant b_2$ ensures that the labels from the two binary classifiers are always consistent.

(b) If the problem is separable in the sense that the quadratic program has a solution, and all the rating labels occur at least once, are we guaranteed that the solution $\hat{\underline{\theta}}$, $\hat{b}_1$, $\hat{b}_2$ satisfies $\hat{b}_1 \leqslant \hat{b}_2$?

**Solution:** Yes.

(c) Suppose we omit Task 2 constraints altogether and only focus on Task 1 in order to solve for $\hat{\underline{\theta}}$ and $\hat{b}_1$. Draw approximately the resulting decision boundary and margin constraints in Figure 1 based on the data in the figure.






**Solution:** See the Figure.

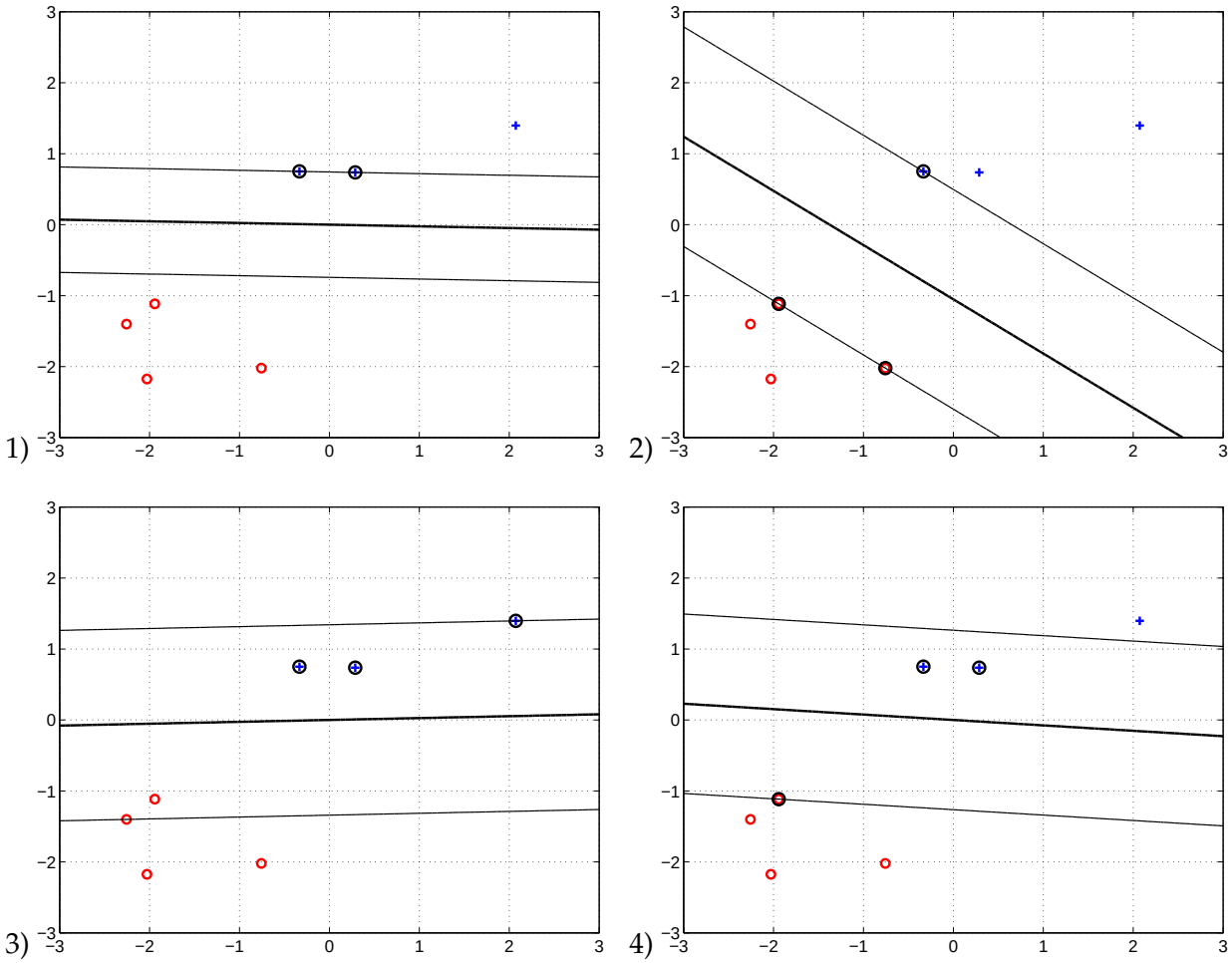
(d) How could you check if the solution $\hat{\underline{\theta}}$ based on Task 1 constraints alone also works as a solution to the combined task?

**Solution:** If we can find $b_2$ such that $\hat{\underline{\theta}}$ together with this $b_2$ satisfies Task 2 constraints, then it is the optimal solution to the combined task ($\|\hat{\underline{\theta}}\|^2$ cannot be smaller because of Task 1 constraints and Task 2 constraints are also satisfied.)

## 10 SVM-like training methods

As you may have suspected, the course staff enjoys writing endless varieties of SVM-like training methods. It is time to sort them out a bit. The figure below shows both decision boundaries and support vectors (circled) from different SVM-like training methods. In all cases, the boundaries correspond to $\hat{\underline{\theta}} \cdot \underline{x} + \hat{\theta}_0 = 0$, where $\hat{\theta}_0 = 0$ unless $\theta_0$ is included in the training method. $J_+$ and $J_-$ index positive ('x') and negative ('o') training examples, respectively. There are five methods and four figures. *For each method, list all figures that show a possible solution for that method. Note that some methods may have more than one possible figure.*

Here are plots of $\hat{\underline{\theta}} \cdot \underline{x} + \hat{\theta}_0 = 0$ for different training methods along with the support vectors. Points labeled $+1$ are in blue, points labeled $-1$ are in red. The line $\hat{\underline{\theta}} \cdot \underline{x} + \hat{\theta}_0 = 0$ is shown in bold; in addition we show the lines $\hat{\underline{\theta}} \cdot \underline{x} + \hat{\theta}_0 = -1$ and $\hat{\underline{\theta}} \cdot \underline{x} + \hat{\theta}_0 = 1$ in non-bold. Support vectors have bold circles surrounding them.




(a)

$$\min \frac{1}{2}\|\underline{\theta}\|^2 + C\sum_{t=1}^{n} \xi_t \text{ s.t. } \xi_t \geqslant 0, \quad y_t(\underline{\theta}^T \underline{x}_t + \theta_0) \geqslant 1 - \xi_t \quad t = 1, \ldots, n$$

where $C = \infty$.

**Solution:** 2. This is the hard margin two-class formulation with offset. We have to have positive and negative support vectors. Only 2 is possible.

(b)

$$\min \frac{1}{2}\|\underline{\theta}\|^2 + C\sum_{t=1}^{n} \xi_t \text{ s.t. } \xi_t \geqslant 0, \quad y_t(\underline{\theta}^T \underline{x}_t) \geqslant 1 - \xi_t, \quad t = 1, \ldots, n$$

where $C = \infty$.

**Solution:** 1. This is the hard margin two-class method without offset. The boundary has to go through origin and has to classify the examples correctly. Only 1 is possible.

(c)

$$\min \frac{1}{2}\|\underline{\theta}\|^2 + C\sum_{t=1}^{n} \xi_t \text{ s.t. } \xi_t \geqslant 0, \ \ y_t(\underline{\theta}^T \underline{x}_t) \geqslant 1 - \xi_t, \ \ t = 1, \ldots, n$$

where $C = 1$.

**Solution:** 1,4. Two-class version with slack and without offset. The boundary has to go through the origin though permits margin violations. However, any example that violates the margin constraint is necessarily a support vector. Depending on the value of C relative to the scale in the figures, only 1 and 4 are possible.

(d)

$$\min \frac{1}{2}\|\underline{\theta}\|^2 + C_+\sum_{t\in J_+} \xi_t + C_-\sum_{t\in J_-} \xi_t \text{ s.t. } \xi_t \geqslant 0, \ \ y_t(\underline{\theta}^T \underline{x}_t) \geqslant 1 - \xi_t, \ \ t = 1, \ldots, n$$

where $C_+ = 1$ and $C_- = 0$.

**Solution:** 1,3,(4). This is a soft margin 1-class method. The boundary has to go through origin. Since $C^- = 0$, the method pays only attention to the positive examples (the constraints for negative examples can be violated without cost). Slack is included so margins for positive examples can be violated depending on the value of $C^+$. 1 and 3 are possible. 4 acceptable together with 1 and 3 if one views the constraints for negative examples as "tight" due to freely setting the corresponding slack variables.

(e)

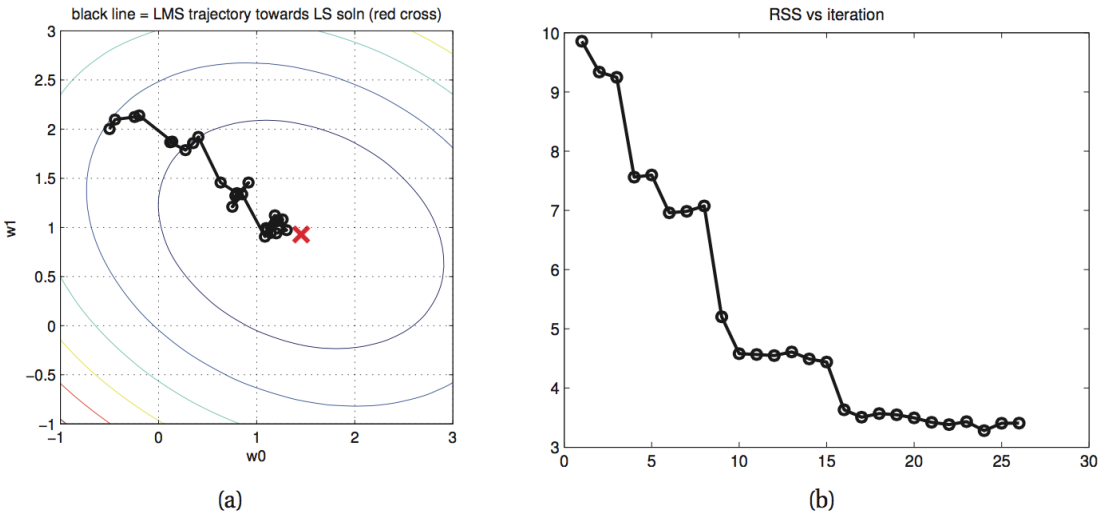
$$\min \frac{1}{2}\|\underline{\theta}\|^2 + C_+\sum_{t\in J_+} \xi_t + C_-\sum_{t\in J_-} \xi_t \text{ s.t. } \xi_t \geqslant 0, \ \ y_t(\underline{\theta}^T \underline{x}_t) \geqslant 1 - \xi_t, \ \ t = 1, \ldots, n$$

where $C_+ = \infty$ and $C_- = 0$.

**Solution:** 1. This is hard margin 1-class method. Only a is possible.

# 11 Review: Regression and model selection

## 11.1 A bumpy hill




This is figure 8.8 from Murphy's textbook. It shows the progress of a gradient descent algorithm, minimizing squared error in a regression problem. Rather than doing the "batch" gradient descent method, where each step goes down the gradient of the squared error, this algorithm randomly selects a training example and takes a step down the gradient of squared error *for that example only.*

Which of the following are true? Why?

- Gradient descent steps always decrease the objective.
- Gradient descent steps with appropriately chosen step size always decrease the objective.
- Stochastic gradient descent steps always decrease the objective.
- Stochastic gradient descent steps with appropriately chosen step size always decrease the objective.
- There is a training method that at each step, randomly choose a element from the training set and take a gradient step based only on the error for that example. This method is an instance of stochastic gradient descent (using empirical error as metric).
- There are circumstances in which stochastic gradient descent is to be preferred to exact gradient descent.
- Training by randomly choosing an element of your training set, randomly perturbing the $x$ values, and taking a gradient step based only on the error for that example is an instance of stochastic gradient descent with the same metric as exact gradient descent on the empirical error.

- Training by randomly choosing an element of your training set, randomly perturbing the $x$ values, and taking a gradient step based only on the error for that example is an instance of stochastic gradient descent with the same metric as exact gradient descent on the ridge regression error.

**Solution:**

- Gradient descent steps always decrease the objective. No. Right step size should be choose.
- Gradient descent steps with appropriately chosen step size always decrease the objective. Yes if gradient is non-zero.
- Stochastic gradient descent steps always decrease the objective. No
- Stochastic gradient descent steps with appropriately chosen step size always decrease the objective. No. See Figure 8.8.
- There is a training method that at each step, randomly choose a element from the training set and take a gradient step based only on the error for that example. This method is an instance of stochastic gradient descent (using empirical error as metric). Yes. This method is an instance of stochastic gradient descent.
- There are circumstances in which stochastic gradient descent is to be preferred to exact gradient descent. Yes. (1) Get out of the local minimal. (2) Computational efficient.
- Training by randomly choosing an element of your training set, randomly perturbing the $x$ values, and taking a gradient step based only on the error for that example is an instance of stochastic gradient descent with the same metric as exact gradient descent on the empirical error. No. This should corresponds to Ridge regression.
- Training by randomly choosing an element of your training set, randomly perturbing the $x$ values, and taking a gradient step based only on the error for that example is an instance of stochastic gradient descent with the same metric as exact gradient descent on the ridge regression error. Yes. Please refer to Week 2 Exercise, Problem 13 for the proof.

## 11.2 Model selection

Dana wants to use ridge regression to fit a model to data, use a model-selection procedure to select an appropriate value of $\lambda$, and produce a good estimate of how the resulting model will perform on unseen data. There are three data sets available: $D_{\text{train}}$, $D_{\text{validate}}$, $D_{\text{test}}$, *all drawn from the same distribution*. Define

$$J(w, \lambda, D) = \sum_i (y^{(i)} - w^T x^{(i)})^2 + \lambda \|w\|_2^2$$

Write the following answers in terms of J and the data sets.

(a) Provide the definition of a function $W^*(\lambda)$ that specifies the optimal value of $w$ for given a value of $\lambda$, according to the ridge regression criterion.

**Solution:**

$$w^*(\lambda) = \arg\min_w J(w, \lambda, D_{\text{train}})$$

(b) Provide an expression for $\lambda^*$, which is the optimal value of $\lambda$. You may use the function $W^*$ in your expression.

**Solution:**

$$\lambda^* = \arg\min_\lambda J(w^*(\lambda), 0, D_{\text{validate}})$$

(c) Provide an expression for $\hat{E}$, the estimated error of the final resulting regression hypothesis on unseen data. You may use the function $W^*$ and or the value $\lambda^*$ in your definition.

**Solution:**

$$\hat{E} = J(W^*(\lambda^*), 0, D_{\text{test}})$$

### 11.3 Hip to be square

Consider a regression problem where the two dimensional input points $\mathbf{x} = [x_1, x_2]^T$ are constrained to lie within the unit square: $x_i \in [-1, 1]$, $i = 1, 2$. The training and test input points $\mathbf{x}$ are sampled uniformly at random within the unit square. The target outputs $y$ are governed by the following model

$$y \sim N(x_1^3 x_2^5 - 10 x_1 x_2 + 7 x_1^2 + 5 x_2 - 3,\ 1)$$

In other words, the outputs are normally distributed with mean given by

$$x_1^3 x_2^5 - 10 x_1 x_2 + 7 x_1^2 + 5 x_2 - 3$$

and variance 1.

We learn to predict $y$ given $\mathbf{x}$ using linear regression models with 1st through 10th order polynomial features. The models are nested in the sense that the higher order models will include all the lower order features. The estimation criterion is the mean squared error.

We first train a 1st, 2nd, 8th, and 10th order model using $n = 20$ training points, and then test the predictions on a large number of independently sampled points.

1. Select all the appropriate model(s) for each column. If you think the highest, or lowest, error would be shared among several models, be sure to list all models.

| | Lowest training error | Highest training error | Lowest test error (typically) |
|---|---|---|---|
| 1st order | ( ) | ( ) | ( ) |
| 2nd order | ( ) | ( ) | ( ) |
| 8th order | ( ) | ( ) | ( ) |
| 10th order | ( ) | ( ) | ( ) |

Briefly explain your selection in the last column, i.e., the model you would expect to have the lowest test error:

2. We now train the polynomial regression models using $n = 10^6$ (one million) training points. Again select the appropriate model(s) for each column. If you think the highest, or lowest, error would be shared among several models, be sure to list all models.

| | Lowest bias | Lowest variance | Lowest test error |
|---|---|---|---|
| 1st order | ( ) | ( ) | ( ) |
| 2nd order | ( ) | ( ) | ( ) |
| 8th order | ( ) | ( ) | ( ) |
| 10th order | ( ) | ( ) | ( ) |

3. True or False: The bias of a polynomial regression model depends on the number of training points.

4. True or false: The variance of a polynomial regression model depends on the number of training points.

**Solution:** 1.

| Lowest training error | Highest training error | Lowest test error |
|---|---|---|
| 8th/10th order | 1st order | 2nd order |

2.

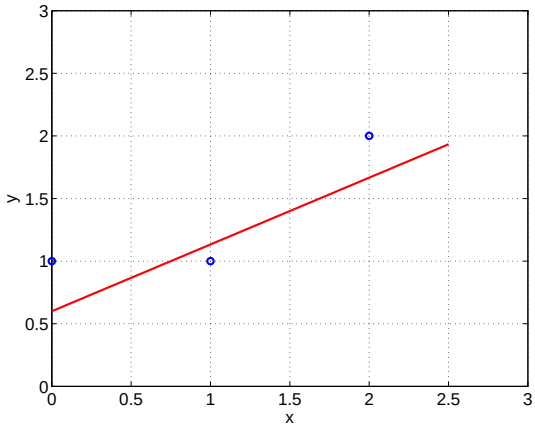
| Lowest bias | Lowest variance | Lowest test error |
|---|---|---|
| 8th/10th | 1st order | 8th order |

3. False. The bias of polynomial regression is independent of the number of training samples. More training samples only reduce the variance.

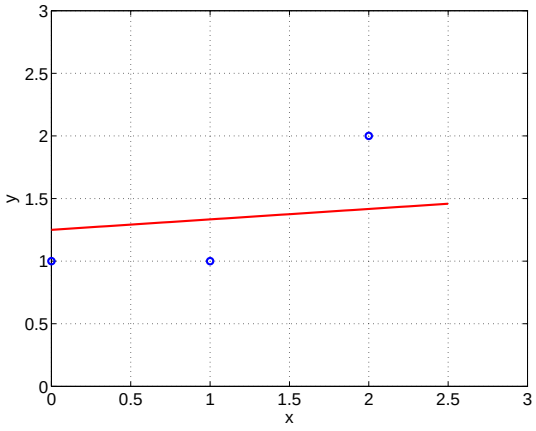
4. True. Variance converges to zero when training samples goes to infinity, when the order of select model is less than or equal to the true model.

### 11.4 Regularization in linear regression

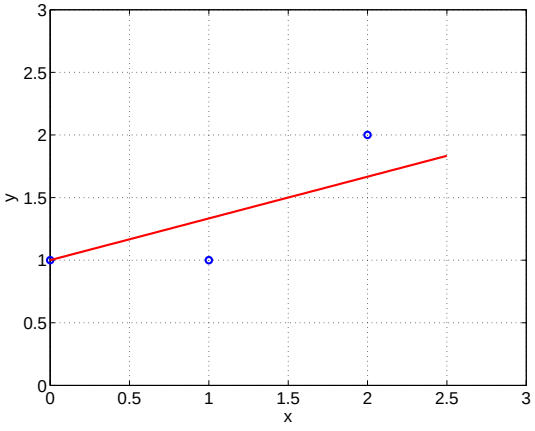
a)



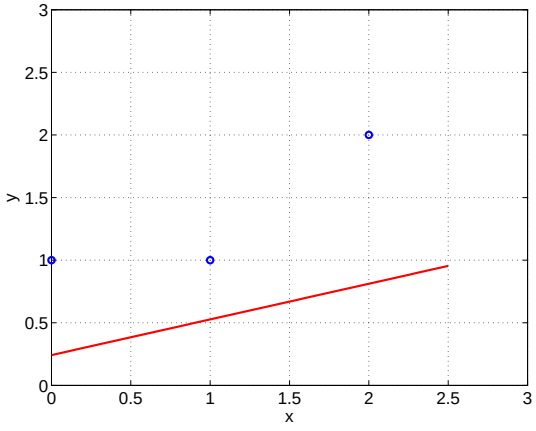

b)




c)




d)




The figure below plots linear regression results on the basis of only three data points. We used various types of regularization to obtain the plots (see below) but got confused about which plot corresponds to which regularization method. Please assign each plot to one (and only one) of the following regularization method.

1. $\sum_{t=1}^{3}(y_t - \theta x_t - \theta_0)^2 + \lambda\theta^2$ where $\lambda = 1$
2. $\sum_{t=1}^{3}(y_t - \theta x_t - \theta_0)^2 + \lambda\theta^2$ where $\lambda = 10$
3. $\sum_{t=1}^{3}(y_t - \theta x_t - \theta_0)^2 + \lambda(\theta^2 + \theta_0^2)$ where $\lambda = 1$
4. $\sum_{t=1}^{3}(y_t - \theta x_t - \theta_0)^2 + \lambda(\theta^2 + \theta_0^2)$ where $\lambda = 10$

**Solution:** 1. (c), 2. (b), 3. (a), 4. (d)

Numerically solve it:

$$1.\theta = 0.33, \theta_0 = 1$$
$$2.\theta = 0.08, \theta_0 = 1.25$$
$$3.\theta = 0.53, \theta_0 = 0.60$$
$$4.\theta = 0.28, \theta_0 = 0.24$$