HW 1

Karl Hiner

September 8, 2023

1 Linear Regression

1.1 a

$$\hat{w} = (X^T X)^{-1} X^T Y \tag{1}$$

$$y_i = w^T x_i + \varepsilon_i, \tag{2}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, $w \in \mathbb{R}^d$, and $\{X^i, Y^i\}$ is the i-th data point, with $1 \leq i \leq m$.

Using the normal equation (Eqn. 1), and the model (Eqn. 2), derive the expectation $\mathbb{E}[\hat{w}]$. Note that here X is fixed, and only Y is random.

$$\begin{aligned}
\mathbb{E}[\hat{w}] &= \mathbb{E}\big[(X^T X)^{-1} X^T Y\big] && \text{Eqn. 1} \\
&= \mathbb{E}\big[(X^T X)^{-1} X^T (Xw + \varepsilon)\big] && \text{Substitute } Y \\
&= \mathbb{E}\big[(X^T X)^{-1} X^T Xw + (X^T X)^{-1} X^T \varepsilon\big] && \text{Distribute} \\
&= \mathbb{E}\big[w + (X^T X)^{-1} X^T \varepsilon\big] && \text{Simplify} \\
&= w + (X^T X)^{-1} X^T \mathbb{E}[\varepsilon] && \text{Linearity of expectation} \\
&= w && \text{Since } \mathbb{E}[\varepsilon] = 0
\end{aligned}$$

1.2 b

Similarly, derive the variance $\mathrm{Var}[\hat{w}]$.

$$\begin{aligned}
\mathrm{Var}[\hat{w}] &= \mathrm{Var}\big[(X^T X)^{-1} X^T (Xw + \varepsilon)\big] && \text{Eqn. 1, Substitute } Y \\
&= \mathrm{Var}\big[(X^T X)^{-1} X^T Xw + (X^T X)^{-1} X^T \varepsilon\big] && \text{Distribute} \\
&= \mathrm{Var}\big[w + (X^T X)^{-1} X^T \varepsilon\big] && \text{Simplify} \\
&= \mathrm{Var}\big[(X^T X)^{-1} X^T \varepsilon\big] && \text{Since } w \text{ is constant} \\
&= (X^T X)^{-1} X^T \mathrm{Var}[\varepsilon] X (X^T X)^{-1} && \mathrm{Var}[\boldsymbol{b}^T \boldsymbol{X}] = \boldsymbol{b}^T \mathrm{Var}[\boldsymbol{X}] \boldsymbol{b} \\
&= (X^T X)^{-1} X^T (\sigma^2 I_m) X (X^T X)^{-1} && \mathrm{Var}[\varepsilon] \triangleq \sigma^2 \\
&= \sigma^2 (X^T X)^{-1} X^T X (X^T X)^{-1} && \text{Commute } \sigma^2 \\
&= \sigma^2 (X^T X)^{-1} && \text{Simplify}
\end{aligned}$$

1.3 c

Under the white noise assumption above, does $\hat{w}$ follow a Gaussian distribution with mean and variance in (a) and (b), respectively? Why or why not?

Answer: Yes, $\hat{w}$ follows a Gaussian distribution with mean and variance in (a) and (b), respectively. This is because $\hat{w}$ is a linear combination of the random variables ε_i, which are Gaussian by assumption. Since $\hat{w}$ is a linear combination of Gaussian random variables, it is itself Gaussian, with $\hat{w} \sim \mathcal{N}(w, \sigma^2(X^T X)^{-1})$.

1.4 d: Weighted linear regression

Suppose we keep the independence assumption but remove the same variance assumption. In other words, data points would be still sampled independently, but now they may have different variance σ_i. Thus, the variance (the covariance matrix) of ε would still be diagonal, but with different values:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m^2 \end{bmatrix}$$

Derive the estimator $\hat{w}$ (similar to the normal equations) for this problem using matrix-vector notations with Σ.

Answer:

We want to minimize

$$\arg\min_w \sum_{i=1}^{m} \frac{1}{\sigma_i^2}(y_i - w^T x_i)^2.$$

In matrix-vector notation, this is equivalent to

$$\arg\min_w (Y - Xw)^T \Sigma^{-1} (Y - Xw).$$

Taking the derivative with respect to w and setting it to zero:

$$\begin{aligned} \frac{\partial}{\partial w}\big((Y - Xw)^T \Sigma^{-1}(Y - Xw)\big) &= 0 \\ \frac{\partial}{\partial w}\big(w^T X^T \Sigma^{-1} X w - 2w^T X^T \Sigma^{-1} Y + Y^T \Sigma^{-1} Y\big) &= 0 \\ -2X^T \Sigma^{-1} Y + 2X^T \Sigma^{-1} X w &= 0 \\ (X^T \Sigma^{-1} X) w &= X^T \Sigma^{-1} Y \end{aligned}$$

Thus, the weighted least squares estimator $\hat{w}$ is:

$$\hat{w} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} Y$$

2 Ridge Regression

For linear regression, it is often assumed that $y_i = w^T x_i + \varepsilon$, where $w, x \in \mathbb{R}^d$ by absorbing the constant term (bias) in an affine hypothesis into w, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable. Given m i.i.d. samples $z_i = (x_i, y_i), 1 \leq i \leq m$, we define $Y = (y_1, \cdots, y_m)^T$ and

$X = (x_1, \cdots, x_m)^T$. Thus, we have $Y \sim \mathcal{N}(Xw, \sigma^2 I_m)$. Show that the ridge regression estimate is the mean of the posterior distribution under a Gaussian prior $w \sim \mathcal{N}(0, \tau^2 I)$. Find the explicit relation between the regularization parameter λ in the ridge regression estimate of the parameter w, and the variances σ^2, τ^2.

Answer:

The ridge regression estimate is defined as

$$\begin{aligned}\hat{w}^{\text{Ridge}} &\triangleq \arg\min_w \sum_{i=1}^{m} (w^T x_i - y_i)^2 + \lambda \|w\|^2 \\ &= \arg\min_w \|Xw - Y\|^2 + \lambda \|w\|^2.\end{aligned}$$

Taking the derivative with respect to w and setting it to zero results in the following expression for $\hat{w}^{\text{Ridge}}$ (as derived in class and in the text):

$$\hat{w}^{\text{Ridge}} = (X^T X + \lambda I)^{-1} X^T Y$$

Now, we'll show that this estimator is also the mean of the posterior distribution of w when we assume a Gaussian prior $w \sim \mathcal{N}(0, \tau^2 I)$.

The posterior distribution of w is proportional to the product of the likelihood and the prior:

$$\begin{aligned}p(w|Y) &\propto p(Y|w)p(w) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}\|Y - Xw\|^2\right) \exp\left(-\frac{1}{2\tau^2}\|w - 0\|^2\right) \\ &= \exp\left(-\frac{1}{2\sigma^2}\|Xw - Y\|^2\right) \exp\left(-\frac{1}{2\tau^2}\|w\|^2\right)\end{aligned}$$

(We neglect the normalization constant $P(Y)$ since it does not depend on w. We also neglect both of the Gaussian normalizing factors since they do not affect the location of the mode.)

We want to find the mean of this posterior distribution. Since multiplying two Gaussian PDFs results in another Gaussian PDF, the posterior is also Gaussian. The mean of a Gaussian PDF is also its mode, which is the location of its maximum. We can find this (single) maximum, $\hat{w}^{\text{Mean}}$, by finding where its derivative with respect to w is equal to zero.

$$\begin{aligned}-log(p(w|Y)) &\propto -log\left(\exp\left(-\frac{1}{2\sigma^2}\|Xw - Y\|^2\right) \exp\left(-\frac{1}{2\tau^2}\|w\|^2\right)\right) \\ &= \frac{1}{2\sigma^2}\|Xw - Y\|^2 + \frac{1}{2\tau^2}\|w\|^2 \\ 0 &= \frac{\partial}{\partial w}\left(\frac{1}{2\sigma^2}\|Xw - Y\|^2 + \frac{1}{2\tau^2}\|w\|^2\right) \\ 0 &= \frac{1}{\sigma^2}X^T(Xw - Y) + \frac{1}{\tau^2}w \\ 0 &= \frac{1}{\sigma^2}X^T X w - \frac{1}{\sigma^2}X^T Y + \frac{1}{\tau^2}w \\ \hat{w}^{\text{Mean}} &= \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}X^T X + \frac{1}{\tau^2}I\right)^{-1} X^T Y\end{aligned}$$

Now, we can find the value of λ equates $\hat{w}^{\text{Mean}}$ and $\hat{w}^{\text{Ridge}}$:

$$\begin{aligned}
\hat{w}^{\text{Mean}} &= \hat{w}^{\text{Ridge}} \\
\frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}X^TX + \frac{1}{\tau^2}I\right)^{-1}X^TY &= (X^TX + \lambda I)^{-1}X^TY \\
\left(\frac{1}{\sigma^2}X^TX + \frac{1}{\tau^2}I\right)^{-1}\frac{1}{\sigma^2} &= (X^TX + \lambda I)^{-1} \\
\frac{1}{\sigma^2}(X^TX + \lambda I) &= \frac{1}{\sigma^2}X^TX + \frac{1}{\tau^2}I \\
\frac{\lambda}{\sigma^2}I &= \frac{1}{\tau^2}I \\
\lambda &= \frac{\sigma^2}{\tau^2}
\end{aligned}$$

Thus, $\hat{w}^{\text{Mean}} = \hat{w}^{\text{Ridge}}$ if we set $\lambda = \frac{\sigma^2}{\tau^2}$.

3 Lasso estimator

The LASSO regression problem can be shown to be the following optimization problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{i=1}^{m} (\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 \text{ subject to } \|\boldsymbol{w}\|_1 \leq \lambda,$$

where $\lambda > 0$ is a regularization parameter. Here, we develop a stochastic gradient descent (SDG) algorithm for this problem, which is useful when we have $m >> d$, where d is the dimension of the parameter space.

3.1 a

Write $\boldsymbol{w} = \boldsymbol{w}^+ - \boldsymbol{w}^-$, where $\boldsymbol{w}^+, \boldsymbol{w}^- \geq 0$ are the positive and negative parts of $\boldsymbol{w}$ respectively. Let w_j be the jth component of $\boldsymbol{w}$. When $w_j \leq 0$, $w_j^+ = 0$ and $w_j^- = -w_j$. Similarly, when $w_j \geq 0$, $w_j^+ = w_j$ and $w_j^- = 0$. Find a quadratic function, Q, of $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ such that

$$\min_{\boldsymbol{w}^+,\boldsymbol{w}^-\geq 0} \sum_{i=1}^{m} Q(\boldsymbol{w}^+, \boldsymbol{w}^-)$$

is equivalent to the above LASSO problem. Expain the equivalence.

Answer:

[Mohri et al] show in Eqn. 11.33 that the LASSO problem can be rewritten as

$$\min_{\boldsymbol{w}^+,\boldsymbol{w}^-\geq 0} \sum_{i=1}^{m} \big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T\boldsymbol{x}_i - y_i\big)^2 + \lambda \sum_{j=1}^{d} (w_j^+ + w_j^-).$$

To see why these two formulations are equivalent, consider some optimal solution $\boldsymbol{w} = \boldsymbol{w}^+ - \boldsymbol{w}^-$. Observe that if both w_j^+ and w_j^- were greater than zero for some j, we could adjust both w_j^+ and w_j^- by $-\delta_j = -\min(w_j^+, w_j^-) > 0$.

This change would not affect the difference $w_j^+ - w_j^-$, but would reduce the sum $w_j^+ + w_j^-$ by $2\delta_j$, improving the regularization term. This implies the original solution was not optimal, which is

a contradition. Thus, for any optimal solution $\boldsymbol{w}$, either $w_j^+ = 0$ or $w_j^- = 0$ for all j, and we can conclude the following quadratic function admits the same solutions as the LASSO problem:

$$Q(\boldsymbol{w}^+, \boldsymbol{w}^-) = \big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big)^2 + \lambda \sum_{j=1}^{d} (w_j^+ + w_j^-)$$

3.2 b

[Mohri et al Ex. 11.10] Derive a stochastic gradient descent algorithm for the quadratic program (with affine constraints) in part (a).

Answer:

In stochastic gradient descent, we randomly choose a training pair $(\boldsymbol{x}_i, y_i)$ and compute the gradient with respect to this pair.

Here is our quadratic objective function for a single training pair $(\boldsymbol{x}_i, y_i)$:

$$Q(\boldsymbol{w}^+, \boldsymbol{w}^-) = \big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big)^2 + \lambda \sum_{j=1}^{d} (w_j^+ + w_j^-)$$

Q has two parameters, $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$, so we need to compute the partial derivatives with respect to each of these parameters:

$$\begin{aligned}
\frac{\partial Q}{\partial w_j^+} &= \frac{\partial}{\partial w_j^+}\big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big)^2 + \frac{\partial}{\partial w_j^+}\lambda \sum_{j=1}^{d} (w_j^+ + w_j^-) && \text{Split} \\
&= \frac{\partial}{\partial w_j^+}\big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big)^2 + \lambda \boldsymbol{I}_d && \text{Solve regularization term} \\
&= 2\big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big)\frac{\partial}{\partial w_j^+}\big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big) + \lambda && \text{Chain rule} \\
&= 2\big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big)\frac{\partial}{\partial w_j^+}\big((\boldsymbol{w}^+)^T \boldsymbol{x}_i - (\boldsymbol{w}^-)^T \boldsymbol{x}_i\big) + \lambda && y_i \text{ is constant} \\
&= 2\big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big)\frac{\partial}{\partial w_j^+}\big((\boldsymbol{w}^+)^T \boldsymbol{x}_i\big) + \lambda && (\boldsymbol{w}^-)^T \boldsymbol{x}_i \text{ is constant} \\
&= 2\big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big)x_{i,j} + \lambda && \text{Only the } j\text{th component of } \boldsymbol{x}_i \text{ interacts with } w_j
\end{aligned}$$

Similarly, we can derive

$$\frac{\partial Q}{\partial w_j^-} = -2\big((\boldsymbol{w}^+ - \boldsymbol{w}^-)^T \boldsymbol{x}_i - y_i\big)x_{i,j} + \lambda.$$

Our update also enforce the constraint, $\boldsymbol{w}^+, \boldsymbol{w}^- \geq 0$. Here is the final SGD update rule, where μ is the learning rate:

$$w_j^+ \leftarrow \max\left(0, w_j^+ - \mu\frac{\partial Q}{\partial w_j^+}\right), w_j^- \leftarrow \max\left(0, w_j^- - \mu\frac{\partial Q}{\partial w_j^-}\right)$$

3.3 c

Suppose $X = [x_1, \cdots, x_m]^T$ is orthonormal and there exists a solution w for $Xw = Y$, where $Y = [y_1, \cdots, y_m]^T$ with no more than k non-zero elements. Can the SGD algorithm get arbitrarily close to w? Explain why or why not.

Based on the theorem of stable recovery of sparse w from Candes, Romberg, and Tao (2004), if there is a true sparse w with fewer than S entries equal to zero, where X satisfies S-restricted isometry, such that $Xw = Y$, then an estimator subject to $\arg\min_w \|w\|_1$ can exactly recover the true w.

Any orthonormal matrix X satisfies S-restricted isometry for any S. Specifically, since an othonormal matrix X preserves the length of all vectors, $\|X\boldsymbol{x}\| = \|\boldsymbol{x}\|$, then $\delta_S = 0$ for any S in

$$(1 - \delta_S)\|\boldsymbol{x}\|^2 \leq \|X\boldsymbol{x}\|^2 \leq (1 + \delta_S)\|\boldsymbol{x}\|^2.$$

With regard to the ability of SGD to recover this w, note that Q is the Lagrangian of the LASSO problem, composed of a quadratic term (with a unique global minimum) and the L1 regularization term required for sparse recovery. Thus, there exists a choice of regularization parameter λ, and a sufficiently small learning rate μ, s.t. the SGD algorithm will, with high probability, converge to the global minimum of Q, and thus the true w.

4 Logistic Regression

Logistic regression is named after the log-odds of success (the logit of the probability) defined as below:

$$\ln\left(\frac{P[Y = 1|X = x]}{P[Y = 0|X = x]}\right),$$

where

$$P[Y = 1|X = x] = \frac{\exp(w_0 + w^T x)}{1 + \exp(w_0 + w^T x)}.$$

4.1 a

Show that log-odds of success is a linear function of X.

$$\begin{aligned}
\ln\left(\frac{P[Y = 1|X = x]}{P[Y = 0|X = x]}\right) &= \ln\left(\frac{P[Y = 1|X = x]}{1 - P[Y = 1|X = x]}\right) && \text{Since } Y \text{ has two possible values} \\
&= \ln\left(\frac{\frac{\exp(w_0 + w^T x)}{1+\exp(w_0 + w^T x)}}{1 - \frac{\exp(w_0 + w^T x)}{1+\exp(w_0 + w^T x)}}\right) && \text{Substitute for } P[Y = 1|X = x] \\
&= \ln\big(\exp(w_0 + w^T x)\big) && \text{Simplify} \\
&= w_0 + w^T x && \ln(e^x) = x
\end{aligned}$$

Since vector addition and dot product are both linear functions, $w_0 + w^T x$ is also a linear function of X.

4.2 b

Show that the logistic loss $L(z) = \log(1 + \exp(-z))$ is a convex function.

If we show that the second derivative of $L(z)$ is nonnegative for all z, then we have shown that $L(z)$ is convex.

$$L'(z) = \frac{-\exp(-z)}{1+\exp(-z)}$$
$$L''(z) = \frac{\exp(-z)}{(1+\exp(-z))^2}$$

Since $L''(z) > 0$ for all z, $L(z)$ is a convex function.

5 Programming: Recommendation System

Problem Summary: A rating by user u on movie i is approximated by

$$M_{u,i} \approx \sum_{k=1}^{d} U_{u,k} V_{i,k}. \tag{3}$$

We want to fit each element of U and V by minimizing squared reconstruction error over all training data points. That is, the objective function we minimize is given by

$$E(U,V) = \sum_{u=1}^{n}\sum_{i=1}^{m}(M_{u,i} - U_u V_v^T)^2 = \sum_{u=1}^{n}\sum_{i=1}^{m}(M_{u,i} - \sum_{k=1}^{d} U_{u,k}V_{i,k})^2, \tag{4}$$

where U_u is the uth row of U and V_i is the ith row of V.

We use gradient descent:

$$U_{v,k} \leftarrow U_{v,k} - \mu\frac{\partial E}{\partial U_{v,k}}, \quad V_{j,k} \leftarrow V_{j,k} - \mu\frac{\partial E}{\partial V_{j,k}}, \tag{5}$$

where μ is the update rate.

5.1 a

Derive the update formula in (5) by solving the partial derivatives.

Answer:

$$\begin{aligned}
\frac{\partial E}{\partial U_{v,k}} &= \frac{\partial}{\partial U_{v,k}}\sum_{u=1}^{n}\sum_{i=1}^{m}(M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'})^2 \\
&= \sum_{u=1}^{n}\sum_{i=1}^{m}\frac{\partial}{\partial U_{v,k}}(M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'})^2 \\
&= \sum_{u=1}^{n}\sum_{i=1}^{m} 2(M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'})\frac{\partial}{\partial U_{v,k}}(M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'}) && \text{Chain rule} \\
&= \sum_{u=1}^{n}\sum_{i=1}^{m} 2(M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'})(-\frac{\partial}{\partial U_{v,k}}\sum_{k'=1}^{d} U_{u,k'}V_{i,k'}) && \frac{\partial}{\partial U_{v,k}}M_{u,i} = 0 \\
&= \sum_{i=1}^{m} 2(M_{v,i} - \sum_{k'=1}^{d} U_{v,k'}V_{i,k'})(-V_{i,k}) && \frac{\partial}{\partial U_{v,k}}U_{u,k'}V_{i,k'} = 0, k' \neq k, u \neq v \\
&= -2\sum_{i=1}^{m}(M_{v,i} - \sum_{k'=1}^{d} U_{v,k'}V_{i,k'})V_{i,k}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E}{\partial V_{j,k}} &= \frac{\partial}{\partial V_{j,k}} \sum_{u=1}^{n}\sum_{i=1}^{m} (M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'})^2 \\
&= \sum_{u=1}^{n}\sum_{i=1}^{m} 2(M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'})(-\frac{\partial}{\partial V_{j,k}} \sum_{k'=1}^{d} U_{u,k'}V_{i,k'}) \qquad \text{Same first three steps} \\
&= \sum_{u=1}^{n} 2(M_{u,j} - \sum_{k'=1}^{d} U_{u,k'}V_{j,k'})(-U_{j,k}) \qquad \frac{\partial}{\partial V_{j,k}} U_{u,k'}V_{j,k'} = 0, k' \neq k, i \neq j \\
&= -2\sum_{u=1}^{n} (M_{u,j} - \sum_{k'=1}^{d} U_{u,k'}V_{j,k'})U_{j,k}
\end{aligned}$$

5.2 b

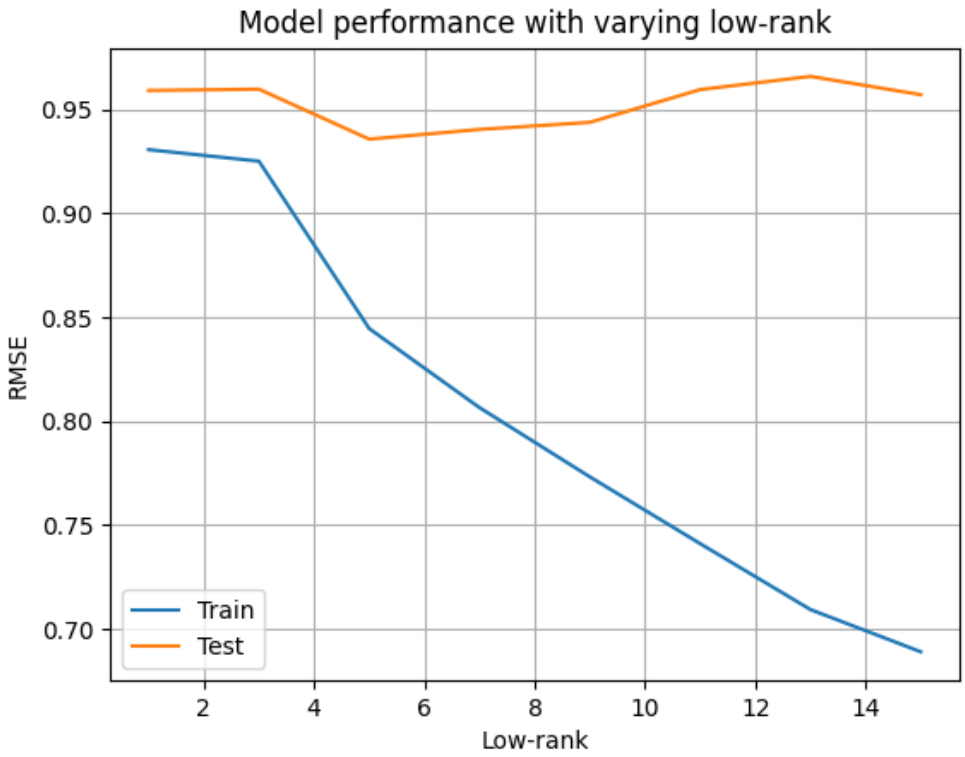
Redo part (a) using the regularized objective function below:

$$E(U,V) = \sum_{u=1}^{n}\sum_{i=1}^{m} (M_{u,i} - \sum_{k=1}^{d} U_{u,k}V_{i,k})^2 + \lambda \sum_{u,k} U_{u,k}^2 + \lambda \sum_{i,k}^{m} V_{i,k}^2$$

$$\begin{aligned}
\frac{\partial E}{\partial U_{v,k}} &= \frac{\partial}{\partial U_{v,k}} \left(\sum_{u=1}^{n}\sum_{i=1}^{m} (M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'})^2 + \lambda \sum_{u,k'} U_{u,k'}^2 + \lambda \sum_{i,k'}^{m} V_{i,k'}^2 \right) \\
&= \frac{\partial}{\partial U_{v,k}} \left(\sum_{u=1}^{n}\sum_{i=1}^{m} (M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'})^2 \right) + \frac{\partial}{\partial U_{v,k}} \lambda \sum_{u=1}^{n}\sum_{k'=1}^{d} U_{u,k'}^2 + \frac{\partial}{\partial U_{v,k}} \lambda \sum_{i,k'}^{m} V_{i,k'}^2 \\
&= -2\sum_{i=1}^{m} (M_{v,i} - \sum_{k'=1}^{d} U_{v,k'}V_{i,k'})V_{i,k} + 2\lambda U_{v,k}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E}{\partial V_{j,k}} &= \frac{\partial}{\partial V_{j,k}} \left(\sum_{u=1}^{n}\sum_{i=1}^{m} (M_{u,i} - \sum_{k'=1}^{d} U_{u,k'}V_{i,k'})^2 \right) + \frac{\partial}{\partial V_{j,k}} \lambda \sum_{u,k'} U_{u,k'}^2 + \frac{\partial}{\partial V_{j,k}} \lambda \sum_{i=1}^{m}\sum_{k'=1}^{d} V_{i,k'}^2 \\
&= -2\sum_{u=1}^{n} (M_{u,j} - \sum_{k'=1}^{d} U_{u,k'}V_{j,k'})U_{j,k} + 2\lambda V_{j,k}
\end{aligned}$$

5.3 5.4 Report



As shown in the chart above, as `lowRank` increases, the training error decreases monotonically, since the model has more capacity to fit the training data. However, the error on the test set starts to increase after `lowRank` = 5, indicating that the model is overfitting the training data. I also found that training for too many iterations resulted in higher test errors, even for small values of `lowRank`. Thus, using a stopping criteria based on the error, rather than a fixed number of steps across all values of `lowRank`, was important to reduce overfitting.

I used grid search to find the best values of μ and λ, using a `lowRank` of 5 and running each iteration to "convergence" (see discussion of error-delta stopping criteria below), or until a maximum number of 1500 gradient descent steps. I chose to fix the rank at the highest evaluated value of 5 to provide the most degrees of freedom to the model, and thus the most potential for it to (over)fit the data and provide a good signal for evaluating the regularization parameter λ. I used the provided train/test split for each run, and did not perform cross-validation. Here is the relevant code for my hyperparameter search:

```
lr = 5 # "Low-rank". Using the highest evaluated value.
max_iter = 1000

best_learn_rate = 0
best_reg_coef = 0
best_rmse = float('inf')
for learn_rate in [0.0001, 0.0002, 0.0004, 0.0005, 0.0008]:
    for reg_coef in [0.001, 0.01, 0.02, 0.04, 0.1, 0.2, 0.4]:
```

```
        U, V = run(rate_mat, lr, True, learn_rate, reg_coef, max_iter)
        rmse_val = rmse(U, V, test_mat)
        if rmse_val < best_rmse:
            best_rmse = rmse_val
            best_learn_rate = learn_rate
            best_reg_coef = reg_coef

print('Learn rate: {}, Reg coef: {}'.format(best_learn_rate, best_reg_coef))
```

The best performing values tested for (μ, λ) were $(0.0001, 0.2)$.

My model stop stops gradient descent when the error delta between iterations is less than 10^{-4}, or when reaching the maximum number of iterations, whichever comes first. I chose this error delta value empirically by manually searching for values that stop iterating nearly where the validation error (over the provided "test" set) starts to increase for various values of `lowRank`.