Section 04: Solutions

Solution:

- Problem 1: Give students 3-4 mins per problem to work through Prob 1.1 and 1.2.
- Problem 2: Gradient Descent parts a-d & slide walkthrough
- Problem 3: Part 3.1 (optional review), Part 3.2 a-b (conceptual review)

1. Biased Test Error

Is the test error unbiased for these programs? If not, how can we fix the code so it is?

1.1. Program 1

```
1  # Given dataset of 1000-by-50 feature
2  # matrix X, and 1000-by-1 labels vector
3
4  mu = np.mean(X, axis=0)
5  X = X - mu
6
7  idx = np.random.permutation(1000)
8  TRAIN = idx[0:900]
9  TEST = idx[900::]
10
11 ytrain = y[TRAIN]
12 Xtrain = X[TRAIN, :]
13
14 # solve for argmin_w ||Xtrain*w - ytrain||_2
15 w = np.linalg.solve(np.dot(Xtrain.T, Xtrain), np.dot(Xtrain.T, ytrain))
16
17 b = np.mean(ytrain)
18
19 ytest = y[TEST]
20 Xtest = X[TEST, :]
21
22 train_error = np.dot(np.dot(Xtrain, w)+b - ytrain,
23                 np.dot(Xtrain, w)+b - ytrain ) / len(TRAIN)
24 test_error = np.dot(np.dot(Xtest, w)+b - ytest,
25                 np.dot(Xtest, w)+b - ytest ) / len(TEST)
26
27 print('Train error = ', train_error)
28 print('Test error = ', test_error)
```

Solution:

The error is at the beginning of the program on lines 4 and 5. Notice how μ is a function of both the train and test data. By de-meaning the entire dataset before splitting, we are intertwining the train and test data. The correct procedure is:

- Split into train and test
- Compute the mean of the train data, μ_{train}
- De-mean both the train and test data with μ_{train}

1.2. Program 2

```
# We are given: 1) dataset X with n=1000 samples and 50 features and 2) a vector y of length 1000 with labels.
# Consider the following code to train a model, using cross validation to perform hyperparameter tuning.

def fit(Xin, Yin, _lambda):
    w = np.linalg.solve(np.dot(Xin.T, Xin) + _lambda * np.eye(Xin.shape[1]), np.dot(Xin.T, Yin))
    b = np.mean(Yin) - np.dot(w, mu)
    return w, b

def predict(w, b, Xin):
    return np.dot(Xin, w) + b

idx = np.random.permutation(1000)
TRAIN = idx[0:800]
VAL = idx[800:900]
TEST = idx[900::]

ytrain = y[TRAIN]
Xtrain = X[TRAIN, :]
yval = y[VAL]
Xval = X[VAL, :]

# demean data
mu = np.mean(Xtrain, axis=0)
Xtrain = Xtrain - mu
Xval = Xval - mu

# use validation set to pick the best hyper-parameter to use
lambdas = [10 ** -5, 10 ** -4, 10 ** -3, 10 ** -2]
err = np.zeros(len(lambdas))

for idx, _lambda in enumerate(lambdas):
    w, b = fit(Xtrain, ytrain, _lambda)
    yval_hat = predict(w, b, Xval)
    err[idx] = np.mean((yval_hat - yval)**2)

lambda_best = lambdas[np.argmin(err)]

Xtot = np.concatenate((Xtrain, Xval), axis=0)
ytot = np.concatenate((ytrain, yval), axis=0)

w, b = fit(Xtot, ytot, lambda_best)

ytest = y[TEST]
Xtest = X[TEST, :]

# demean data
Xtest = Xtest - mu

ytot_hat = predict(w, b, Xtot)
train_error = np.mean((ytot_hat - ytot) **2)
ytest_hat = predict(w, b, Xtest)
test_error = np.mean((ytest_hat - ytest) **2)

print('Train error = ', train_error)
print('Test error = ', test_error)
```

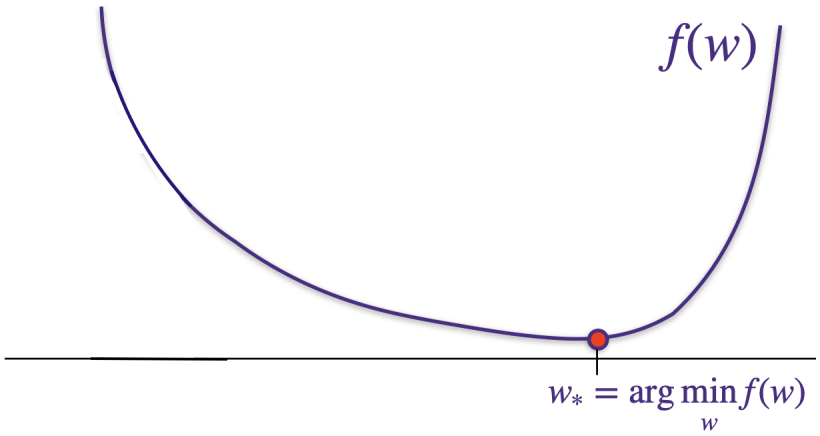
Solution:

We are adding the validation data back into training (creating `Xtot`), and then retraining the whole model on this data. However, optimal value of λ **depends** on size of the training dataset, so by adding more data we are using sub-optimal value in final `fit` call. In general, models get better the more data you give them, but only add the validation set back in if you are confident the hyperparameter doesn't depend on the number of elements, and that you aren't allowing your model access to the test set. Note: it is not incorrect to add validation data back into training. It's a trade-off between having more training data and having a reliable estimate of test performance. See offering au24 slide 39 for reference.

2. Gradient Descent

Like we've seen in lecture, gradient descent is an important algorithm commonly used to train machine learning models, particularly useful for when there is no closed form solution for the minimum of a loss function. Here, we'll go through short introduction to the algorithm.

Consider some function $f(w)$, which has some w_* for which $w_* = \arg\min_w f(w)$:

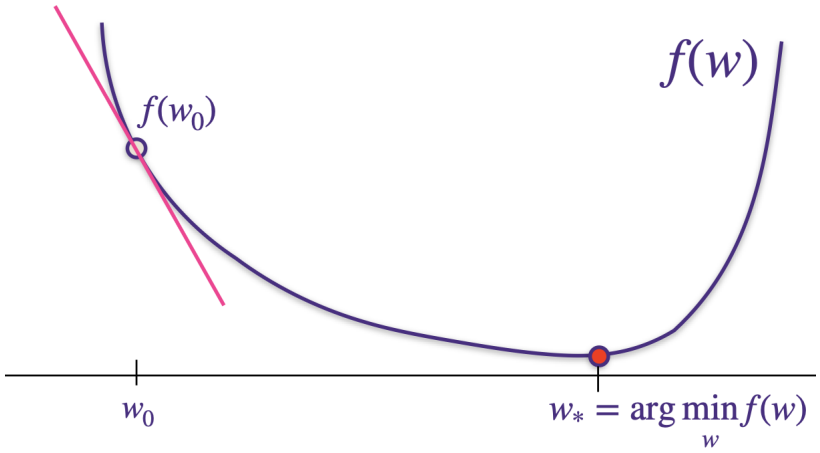


Let w_0 be some initial guess for the minimum of $f(w)$. Gradient descent will allow us to improve this solution.

(a) For some w that is very close to w_0, give the Taylor series approximation for $f(w)$ starting at $f(w_0)$.

Solution:

For w very close to w_0, we see that $f(w) \approx f(w_0) + (w - w_0)\left(\frac{df(w)}{dw}\Big|_{w=w_0}\right)$.



(b) Now, let us choose some $\eta > 0$ that is *very small*. With this very small η, let's assume that $w_1 = w_0 - \eta\left(\frac{df(w)}{dw}\Big|_{w=w_0}\right)$. Using your approximation from part (a), give an expression for $f(w_1)$.

Solution:

$$
\begin{aligned}
w_1 &= w_0 - \eta \left(\frac{df(w)}{dw} \,\middle|\, _{w=w_0} \right) \\
f(w_1) &\approx f(w_0) + (w_1 - w_0) \left(\frac{df(w)}{dw} \,\middle|\, _{w=w_0} \right) \\
&= f(w_0) + \left(w_0 - \eta \left(\frac{df(w)}{dw} \,\middle|\, _{w=w_0} \right) - w_0 \right) \left(\frac{df(w)}{dw} \,\middle|\, _{w=w_0} \right) \\
&= f(w_0) - \eta \left(\frac{df(w)}{dw} \,\middle|\, _{w=w_0} \right)^2
\end{aligned}
$$

(c) Given your expression for $f(w_1)$ from part (b), explain why, if η is small enough and if the function approximation is a good enough approximation, we are guaranteed to move in the "right" direction closer to the minimum w_*.

Solution:

Note that in part (b), the derivative is squared and will always be a nonnegative value. Therefore, $f(w_1) < f(w_0)$.

(d) Building from your answer in part (c), write a general form for the gradient descent algorithm.

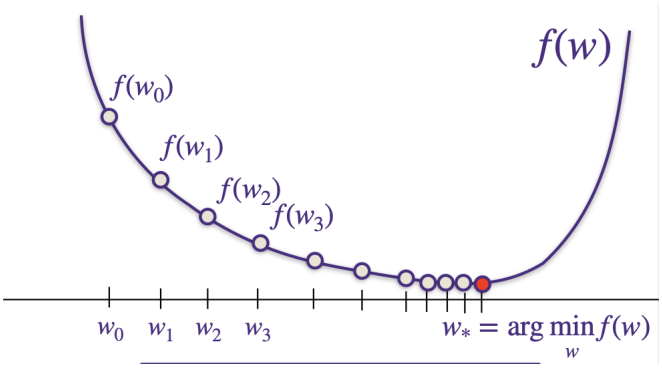
Solution:

Gradient descent is written as:

For $k = 0, 1, 2, 3, ..., w_{k+1} = w_k - \eta \left(\frac{df(w)}{dw} \,\middle|\, _{w=w_k} \right)$.

Note that as $k \to \infty$, $\left(\frac{df(w)}{dw} \,\middle|\, _{w=w_k} \right) \to 0$.

We visualize as:



3. Generalized Least Squares Regression

In class, we've seen linear regression and ridge regression. Here, we consider a problem that generalizes both of these. As a reminder, in linear regression, we seek a model that captures a linear relationship between input data and output data. The general case we consider imposes additional structure on the model.

Consider an experiment in which you have n data points $x_i \in \mathbb{R}^d$ and corresponding n observations y_i. We wish to come up with a model $\omega \in \mathbb{R}^d$ that satisfies the following properties: first, the error $\sum_{i=1}^n (x_i^\top \omega - y_i)^2$ should be small; second, we don't want small changes in training data resulting in large changes in solution; third, we want to put different weights in controlling the magnitude of different coordinates of ω. We therefore define

$$\widehat{\omega}_{\text{general}} = \arg\min_{\omega} \sum_{i=1}^{n} (y_i - x_i^\top \omega)^2 + \lambda \sum_{i=1}^{d} D_{ii} \omega_i^2.$$

Here, D is a diagonal matrix, with positive entries on the diagonal. Observe that when D is the identity matrix, we recover ridge regression, and when $\lambda = 0$, we recover least squares regression. Different weights on D_{ii} cause the magnitudes of ω_i to be controlled differently.

3.1. Closed form in the general case

Deduce the closed form solution for $\widehat{\omega}_{\text{general}}$. You should be comfortable with proofs in the "coordinate" form as well as the "matrix" form.

Solution:

We first give the proof using "matrix" notation. The objective function can be expressed as

$$\begin{aligned} f(\omega) &= \|X\omega - y\|_2^2 + \lambda \omega^\top D \omega \\ &= (X\omega - y)^\top (X\omega - y) + \lambda \omega^\top D \omega \\ &= (X\omega)^\top X\omega - (X\omega)^\top y - y^\top X\omega + y^\top y + \lambda \omega^\top D \omega \\ &= \omega^\top X^\top X \omega - 2\omega^\top X^\top y + y^\top y + \lambda \omega^\top D \omega \\ &= \omega^\top (X^\top X + \lambda D)\omega - 2\omega^\top X^\top y + y^\top y \end{aligned}$$

The gradient of f is

$$\begin{aligned} \nabla f(\omega) &= \nabla_\omega (\omega^\top (X^\top X + \lambda D)\omega - 2\omega^\top X^\top y + y^\top y) \\ &= \nabla_\omega (\omega^\top (X^\top X + \lambda D)\omega) - 2\nabla_\omega (\omega^\top X^\top y) + \nabla_\omega (y^\top y) \\ &= 2(X^\top X + \lambda D)\omega - 2X^\top y \end{aligned}$$

Here note that $X^\top X + \lambda D$ is a symmetric matrix, which explains the factor 2 in the gradient term. Setting the gradient $\nabla f(\omega)$ to zero, we can conclude that

$$(X^\top X + \lambda D)\widehat{\omega}_{\text{general}} = X^\top y$$

If $X^\top X + \lambda D$ is full rank then we can get a unique solution:

$$\widehat{\omega}_{\text{general}} = (X^\top X + \lambda D)^{-1} X^\top y$$

Since D is already given to be a diagonal matrix with strictly positive entries on the diagonal, any strictly positive λ will make the matrix $X^\top X + \lambda D$ invertible.

Solution:

We now give a solution in the "coordinate" form. The objective, when written in coordinate form, is $f(\omega) = \sum_{i=1}^{n}(y_i - x_i^\top \omega)^2 + \lambda \sum_{i=1}^{d} D_{ii}\omega_i^2$. As in the previous proof, we first simplify it as follows and then set it zero:

$$\begin{aligned}
\nabla_\omega \left[\sum_{i=1}^{n}(y_i - x_i^\top \omega)^2 + \lambda \sum_{i=1}^{d} D_{ii}\omega_i^2\right] &= \nabla_\omega \sum_{i=1}^{n}(y_i - x_i^\top \omega)^2 + \nabla_\omega \lambda \sum_{i=1}^{d} D_{ii}\omega_i^2 \\
&= \sum_{i=1}^{n} \nabla_\omega (y_i - x_i^\top \omega)^2 + 2\lambda D\omega \\
&= -\sum_{i=1}^{n} 2x_i(y_i - x_i^\top \omega) + 2\lambda D\omega \\
&= -\sum_{i=1}^{n} 2x_i y_i + \sum_{i=1}^{n} 2x_i x_i^\top \omega + 2\lambda D\omega \\
&= -2\sum_{i=1}^{n} x_i y_i + 2\left(\sum_{i=1}^{n} x_i x_i^\top + \lambda D\right)\omega \\
&= 0 \ \ \text{(set it to be 0)}
\end{aligned}$$

$$\widehat{\omega}_{\text{general}} = \left(\sum_{i=1}^{n} x_i x_i^\top + \lambda D\right)^{-1} \left(\sum_{i=1}^{n} x_i y_i\right)$$

Note that, as expected, this exactly matches the answer we got from the previous approach (because x_i's are all the rows of X, and therefore $\sum_i x_i y_i = X^\top y$, and $\sum_i x_i x_i^\top = X^\top X$).

3.2. Special cases: linear regression and ridge regression

(a) In the simple least squares case ($\lambda = 0$ above), what happens to the resulting $\widehat{\omega}$ if we double all the values of y_i?

Solution:

As can be seen from the formula $\widehat{\omega} = (X^\top X)^{-1} X^\top y$, doubling y doubles ω as well. This makes sense intuitively as well because if the observations are scaled up, the model should also be.

(b) In the simple least squares case ($\lambda = 0$ above), what happens to the resulting $\widehat{\omega}$ if we double the data matrix $X \in \mathbb{R}^{n \times d}$?

Solution:

As can be seen from the formula $\widehat{\omega} = (X^\top X)^{-1} X^\top y$, doubling X halves ω. This also makes sense intuitively because the error we are trying to minimize is $\|X\omega - y\|_2^2$, and if the X has doubled, while y has remained unchanged, then ω must compensate for it by reducing by a factor of 2.

(c) Suppose $D = I$ (that is, it is the identity matrix). That is, this is the *ridge* regression setting. Explain why $\lambda > 0$ ensures that the solution exists and the matrix can be inverted.

Solution:

The solution is $\widehat{\omega} = (X^\top X + \lambda I)^{-1} X^\top y$. We already saw in a previous part that $X^\top X$ is always positive semidefinite, that is, its eigenvalues are at least zero. Adding λI, where $\lambda > 0$, ensures that $X^\top X + \lambda I$ is in fact positive *definite*. This helps us have a good condition number.

4. MAP as Regularization

Recall the regularization techniques that were presented in class this week and ponder their objectives:

(a) **Ridge-Regression**: $\hat{w}_{ridge} = \arg\min_w \sum_{i=1}^{n} (y_i - x_i^\top w)^2 + \lambda \|w\|_2^2$

(b) **LASSO**: $\hat{w}_{LASSO} = \arg\min_w \sum_{i=1}^{n} (y_i - x_i^\top w)^2 + \lambda \|w\|_1$

Reminder: don't ever regularize your bias term. This term doesn't add any complexity to the model (since it just shifts), so we'd like it to take on any value that best fits our training data.

The two types of regularization above can be derived from a statistical perspective in which we assume some prior belief about what the weights of our model should be and then observe data to further update the belief.

More specifically, let w denote our weights and Y, X our data(Y represents the labels and X the inputs). As before, $p(X, Y|w)$ represents the **likelihood function**. We specify our belief of what the weights should be through a **prior distribution** over $p(w)$. Using Bayes' Rule, we can write our updated belief of what the weights ought to be after observing the data as:

$$p(w|X, Y) = \frac{p(X, Y|w)p(w)}{p(X, Y)} = \frac{p(X, Y|w)p(w)}{\int_{w'} p(X, Y|w')p(w')dw'}$$

where we call $p(w|X, Y)$ the **posterior distribution** and $p(X, Y)$ the **evidence**.

What **Maximum A Posteriori Estimation(MAP)** does is compute the weights which maximize the posterior distribution, $p(w|X, Y)$. This type of estimation differs from MLE (which maximizes the likelihood function $p(X, Y|w)$) by taking into account our prior belief of what the weights are, namely $p(w)$. More specifically, the MAP estimate is:

$$\begin{aligned}
\hat{w}_{MAP} &= \arg\max_w p(w|X, Y) \\
&= \arg\max_w \frac{p(X, Y|w)p(w)}{p(X, Y)} \\
&= \arg\max_w p(X, Y|w)p(w) \\
&= \arg\max_w \log p(X, Y|w) + \log p(w)
\end{aligned}$$

where we dispose of the denominator because it doesn't depend on w. Contrast this with the MLE which is:

$$\hat{w}_{MLE} = \arg\max_w p(X, Y|w)$$

Let us now study how we can obtain the Ridge and LASSO regression objectives from this perspective:

(a) Suppose the elements of w are independently distributed according to a Laplacian distribution:

$$p(w_i) = \frac{\lambda}{4\sigma^2} \exp(-|w_i| \frac{\lambda}{2\sigma^2}).$$

Show that under this prior on w, MAP estimation of the linear measurement model recovers the LASSO objective.

Solution:

We work in the argmax space, which allows us to drop and add constants or monotonically increasing functions as necessary.

$$
\begin{aligned}
\arg\max_w p(w \mid Y, X) &= \arg\max_w \log p(w \mid Y, X) \\
&= \arg\max_w \log p(X, Y \mid w) + \log p(w) \\
&\overset{*}{=} \arg\max_w \log \left\{ (2\pi)^{-\frac{n}{2}} \left(\sigma^2\right)^{-\frac{n}{2}} \exp(-\frac{1}{2}(Y - Xw)^\top (\sigma^2 \mathbb{I})^{-1}(Y - Xw) \right\} \\
&\quad + \sum_{i=1}^{n} \log \frac{\lambda}{4\sigma^2} \exp(-|w_i| \frac{\lambda}{2\sigma^2}) \\
&= \arg\max_w -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(Y - Xw)^\top (Y - Xw) \\
&\quad + n \log \frac{\lambda}{4\sigma^2} - \frac{\lambda}{2\sigma^2} \sum_{i=1}^{n} |w_i| \\
&\overset{**}{=} \arg\max_w -\frac{1}{2\sigma^2} \left[(Y - Xw)^\top (Y - Xw) + \lambda \sum_{i=1}^{n} |w_i| \right] \\
&\overset{***}{=} \arg\max_w -\|Y - Xw\|_2^2 + \lambda \|w\|_1 \\
&= \arg\min_w \|Y - Xw\|_2^2 + \lambda \|w\|_1,
\end{aligned}
$$

where the first starred equality follows from applying the PDFs, and the second and third follow from dropping constant terms or multipliers. In other words, solving MAP with a Laplacian prior also solves the LASSO regression problem.

(b) Derive an expression for the prior on w that corresponds to the ridge regression objective. What is the significance of this result?

Solution:

Our high-level approach to this problem is to expand the terms of the objective to a form that resembles the core of a PDF, then attach the additive/multiplicative constants necessary to recover the full form of the PDF. We do this first with the likelihood term (since we know its form), and then with the prior.

$$
\begin{aligned}
\arg\min_w \|Y - Xw\|_2^2 + \lambda \|w\|_2^2 &= \arg\max_w -(Y - Xw)^\top (Y - Xw) + -\lambda w^\top w \\
&= \arg\max_w -\frac{1}{2\sigma^2}(Y - Xw)^\top (Y - Xw) - \frac{\lambda}{2\sigma^2} w^\top w \\
&= \arg\max_w -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(Y - Xw)^\top (Y - Xw) - \frac{\lambda}{2\sigma^2} w^\top w \\
&= \arg\max_w \mathcal{N}(Y; Xw, \sigma^2 \mathbb{I}) - \frac{\lambda}{2\sigma^2} w^\top w \\
&= \arg\max_w \mathcal{N}(Y; Xw, \sigma^2 \mathbb{I}) - \frac{n}{2}\log\left(2\pi\frac{\sigma^2}{\lambda}\right) - \frac{1}{2} w^\top \left(\frac{\sigma^2}{\lambda}\mathbb{I}\right)^{-1} w \\
&= \arg\max_w \mathcal{N}(Y; Xw, \sigma^2 \mathbb{I}) + \mathcal{N}\left(w; 0, \frac{\sigma^2}{\lambda}\mathbb{I}\right).
\end{aligned}
$$

In other words, our prior is that $w \sim \mathcal{N}\left(0, \frac{\sigma^2}{\lambda}\mathbb{I}\right)$.

This means that when we apply ℓ_2 regularization to our linear regression problem (i.e., ridge regression), we make the implicit assumption that our weight vector is drawn from a Gaussian prior. More generally, we can see that applying various forms of regularization correspond to different prior assumptions on w.