1. Recall the definition of the expected risk $R(f)$ of a predictor f. While this cannot be computed in general note that here we defined $P_{\mathcal{X}\times\mathcal{Y}}$. Which function f^* is an obvious Bayes predictor? Make sure to explain why the risk $R(f^*)$ is minimum at f^*.

Answer:

$f^* = a_0 + a_1 x + a_2 x^2$ is an obvious Bayes predictor

In this case, the values of x and y are linked as $y = g(x) = a_0 + a_1 x + a_2 x^2$, which fixes the joint distribution $P_{X\times Y}$

Hence, for all $x \in [0,1]$. $l(\hat{y}, y) = \frac{1}{2}(f^*(x) - g(x)) = 0$
$R(f^*) = E[l(f^*(x), g(x)] = 0$
i.e. $R(f^*)$ is minimum at f^*

2. Using $\mathcal{H}_2$ as your hypothesis class, which function $f^*_{\mathcal{H}_2}$ is a risk minimizer in $\mathcal{H}_2$? Recall the definition of the approximation error. What is the approximation error achieved by $f^*_{\mathcal{H}_2}$?

Answer:

Using $\mathcal{H}_2$ as a hypothesis set, $f^*_{\mathcal{H}_2} = a_0 + a_1 x + a_2 x^2$ is a risk minimizer in $\mathcal{H}_2$.

This can be explained by the fact that the true function $g(x)$ fits into the hypothesis class. The approximation is actually zero achieved by $f^*_{\mathcal{H}_2}$

i.e. $R(f^*_{\mathcal{H}_2}) - R(f^*) = 0$

3. Considering now $\mathcal{H}_d$, with $d > 2$. Justify an inequality between $R(f^*_{\mathcal{H}_2})$ and $R(f^*_{\mathcal{H}_d})$. Which function $f^*_{\mathcal{H}_d}$ is a risk minimizer in $\mathcal{H}_d$? What is the approximation error achieved by $f^*_{\mathcal{H}_d}$?

Answer:

When $d>2$, the hypothesis set tends to be larger, which introduces more chances for the hypothesis set to cover the Bayesian predictor.

Hence, $R(f^*_{H_2}) \geq R(f^*_{Hd}) \geq 0$.

It's known that $R(f^*_{H_2}) = 0$. thus $R(f^*_{Hd}) = 0$

We can deduce that: $f^*_{Hd} = a_0 + a_1 x + a_2 x^2$

The approximation error: $R(f^*_{Hd}) - R(f^*) = 0$

4. For this question we assume $a_0 = 0$. Considering $\mathcal{H} = \{f : x \to b_1 x; b_1 \in \mathbb{R}\}$, which function $f^*_{\mathcal{H}}$ is a risk minimizer in $\mathcal{H}$? What is the approximation error achieved by $f^*_{\mathcal{H}}$? In particular what is the approximation error achieved if furthermore $a_2 = 0$ in the definition of true underlying relation $g(x)$ above?

Answer:

$$b_1^* \in \operatorname{argmin}_{b_1} E[l(f^*_H(x), g(x)]$$

$$b_1^* \in \operatorname{argmin}_{b_1} \frac{1}{2}\int_0^1 (b_1 x - a_1 x - a_2 x^2)^2 dx$$

$$b_1^* \in \operatorname{argmin}_{b_1} \frac{1}{2}\int_0^1 (b_1 - a_1)^2 x^2 - 2(a_2 b_1 - a_1 a_2) x^3 + a_2^2 x^4 dx$$

$$b_1^* \in \operatorname{argmin}_{b_1} \frac{1}{2}\left[\frac{1}{3}(b_1 - a_1)^2 x^3 - \frac{1}{2}(a_2 b_1 - a_1 a_2) x^4 + \frac{1}{5} a_2^2 x^5 \Big|_0^1\right]$$

$$b_1^* \in \operatorname{argmin}_{b_1} \frac{1}{2}\left[\frac{1}{3}(b_1^2 - 2a_1 b_1 + a_1^2) - \frac{1}{2}(a_2 b_1 - a_1 a_2) + \frac{1}{5} a_2^2\right]$$

let $h(b) = \frac{1}{3} b_1^2 - \frac{2}{3} a_1 b_1 - \frac{1}{2} a_2 b_1 + \text{(*)}$ $\quad \frac{\partial h(b)}{\partial b} = \frac{2}{3} b_1 - \frac{2}{3} a_1 - \frac{1}{2} a_2 = 0$

Hence: $b_1^* = a_1 + \frac{3}{4} a_2 \quad f^*_H = b_1^* x = (a_1 + \frac{3}{4} a_2) x$

$$R(f^*_H) - R(f^*) = \int_0^1 \frac{1}{2}\left(\frac{3}{4} a_2 x - a_2 x^2\right)^2 dx = \frac{1}{160} a_2^2$$

Furthermore, if $a_2 = 0$. $f^*_H = a_1 x$,

the approximation error $R(f^*_H) - R(f^*) = 0$.

5. Show that the empirical risk minimizer (ERM) $\hat{\boldsymbol{b}}$ is given by the following minimization $\hat{\boldsymbol{b}} = \arg\min_{b} \|Xb - \boldsymbol{y}\|_2^2$.

Answer:

$$\hat{b} = \arg\min_{b} \|Xb - y\|_2^2 = \arg\min_{b} (Xb-y)^T (Xb-y)$$

$$= \arg\min_{b} \begin{pmatrix} b_0 + b_1 x_1 + \cdots + b_d x_1^d - y_0 \\ b_0 + b_1 x_2 + \cdots + b_d x_2^d - y_1 \\ \vdots \\ b_0 + b_1 x_N + \cdots + b_d x_N^d - y_d \end{pmatrix}^T \begin{pmatrix} b_0 + b_1 x_1 + \cdots + b_d x_1^d - y_0 \\ b_0 + b_1 x_2 + \cdots + b_d x_2^d - y_1 \\ \vdots \\ b_0 + b_1 x_N + \cdots + b_d x_N^d - y_d \end{pmatrix}$$

$$= \arg\min_{b} \sum_{i=1}^{N} (b_0 + b_1 x_i + \cdots + b_d x_i^d - y_{N-1})^2$$

$$= \arg\min_{b} \sum_{i=1}^{N} (Xb-y)_i^2$$

where $(Xb-y)_i$ denotes the ith row of $(Xb-y)$

$$= \arg\min \sum_{i=1}^{N} (f_b(x_i) - y_{i-1})^2$$

Consider N is a constant, we can rewrite this equation as:

$$\hat{b} = \arg\min \sum_{i=1}^{N} \frac{1}{N} (f_b(x_i) - y_{i-1})^2 \quad (*)$$

$(*)$ can be regarded as the empirical minimizer given the loss function: $l(x) = (f(x)-y)^2$

Hence, the empirical risk minimizer (ERM) is given by the following minimization $\hat{b} = \arg\min_{b} \|Xb-y\|_2^2$" is proved.

6. If $N > d$ and X is full rank, show that $\hat{\boldsymbol{b}} = (X^\top X)^{-1} X^\top \boldsymbol{y}$. (Hint: you should take the gradients of the loss above with respect to $\boldsymbol{b}$ [1]). Why do we need to use the conditions $N > d$ and X full rank?

Answer:

According to the previous proof in 5.
we get that $\hat{b} = \underset{b}{\operatorname{argmin}} \| Xb - y \|_2^2$

$$= \underset{b}{\operatorname{argmin}} (Xb-y)^T (Xb-y)$$

$$= \underset{b}{\operatorname{argmin}} (b^T X^T X b - y^T X b - b^T X y + y^T y)$$

let $f(b) = b^T X^T X b - y^T X b - b^T X^T y + y^T y$

$$\frac{\partial f(b)}{\partial b} = (X^T X + X^T X) b - X^T y - X^T y = 2X^T X b - 2X^T y = 0$$

let $g(b) = 2X^T X b - 2X^T y$; $\frac{\partial g(b)}{\partial b} = 2X^T X \geq 0$

Hence $X^T X b = X^T y$

With $N > d$ and X being full rank, we can obtain that $X^T X$ is invertible. thus: $\hat{b} = (X^T X)^{-1} X^T y$.

To state why we need to use the condition:

$$\begin{cases} N > d \\ X \text{ being full rank} \end{cases} \Rightarrow \begin{cases} \text{rank}(X) = d \\ X \text{ is invertible} \end{cases} \Rightarrow X^T X \text{ is invertible}$$

Once these two restrictions are disobeyed:
then $\ker(X) \neq \{0\}$. then $\ker(X^T X) \neq \{0\}$.
$X^T X$ isn't invertible anymore.

7. Write a function called `least_square_estimator` taking as input a design matrix $X \in \mathbb{R}^{N\times(d+1)}$ and the corresponding vector $\boldsymbol{y} \in \mathbb{R}^N$ returning $\hat{\boldsymbol{b}} \in \mathbb{R}^{(d+1)}$. Your function should handle any value of N and d, and in particular return an error if $N \leq d$. (Drawing x at random from the uniform distribution makes it almost certain that any design matrix X with $d \geq 1$ we generate is full rank).

ANSWER TO Q_7

```python
def least_square_estimator(X, y):
    """
    array: N*(d+1)
    array: d+1
    """
    N, d_plus_1 = X.shape
    if N > (d_plus_1-1):
        return np.linalg.inv(X.T @ X) @ X.T @ y
    else:
        raise ValueError("N must be larger than d")
```

8. Recall the definition of the empical risk $\hat{R}(\hat{f})$ on a sample $\{x_i, y_i\}_{i=1}^N$ for a prediction function $\hat{f}$. Write a function `empirical_risk` to compute the empirical risk of $f_{\boldsymbol{b}}$ taking as input a design matrix $X \in \mathbb{R}^{N\times(d+1)}$, a vector $\boldsymbol{y} \in \mathbb{R}^N$ and the vector $\boldsymbol{b} \in \mathbb{R}^{(d+1)}$ parametrizing the predictor.

ANSWER TO Q_8

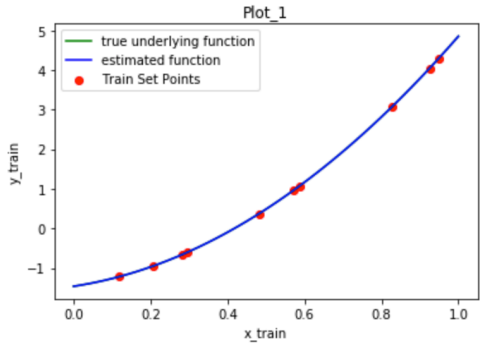
```python
def empirical_risk(X, y, b):
    N, _ = X.shape
    emp_risk = 1/(2*N) * np.linalg.norm(X @ b - y) **2
    return emp_risk
```

9. Use your code to estimate $\hat{\boldsymbol{b}}$ from `x_train, y_train` using $d = 5$. Compare $\hat{\boldsymbol{b}}$ and $\boldsymbol{a}$. Make a single plot (Plot 1) of the plan (x, y) displaying the points in the training set, values of the true underlying function $g(x)$ in $[0, 1]$ and values of the estimated function $f_{\hat{\boldsymbol{b}}}(x)$ in $[0, 1]$. Make sure to include a legend to your plot.

ANSWER TO Q_9

```
#X_train: N_train * (d+1)
#coeff: (d+1)*1
#y_train: N_train * 1
N_train, N_test = 10, 1000
true_d, esti_d = 2, 5
a = get_a(deg_true=true_d)

# Design X_train by changing true_d to esti_d
# With the esti_d, use estimated_b to represent the latent relationship between x_train & y_train
x_train, y_train = draw_sample(deg_true=true_d, a=a, N=N_train)
X_train = get_design_mat(x=x_train, deg=esti_d)
esti_b = least_square_estimator(X_train, y_train)
print(f'The generated a is {a}')
print(f'The estimated b is {esti_b}')
print('*'*100)
print('From the Plot_1, we can see that the scatter nearly perfect fits. ')

gx = get_design_mat(np.linspace(0,1),2) @ a
fb = get_design_mat(np.linspace(0,1),esti_d) @ esti_b

# Plot the True underlying functions
# Plot the Estimated function
plt.scatter(x_train, y_train, label='Train Set Points', color='red')
plt.plot(np.linspace(0,1), gx, label='true underlying function', color='green')
plt.plot(np.linspace(0,1), fb, label='estimated function', color='blue')
plt.title('Plot_1')
plt.xlabel('x_train')
plt.ylabel('y_train')
plt.legend()
plt.show()
```

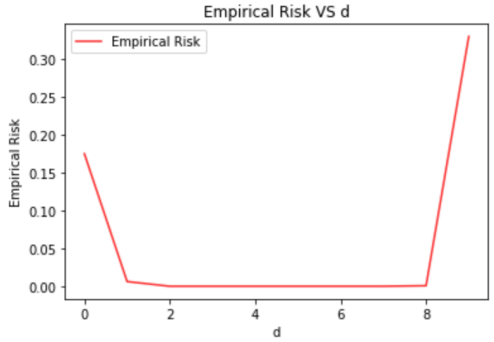
```
The generated a is [-1.46027496  1.476993    4.84103081]
The estimated b is [-1.46027496e+00  1.47699300e+00  4.84103081e+00  2.04789785e-09
  1.30665967e-09  3.17754711e-11]
****************************************************************************************************
From the Plot_1, we can see that the scatter nearly perfect fits.
```



10. Now you can adjust d. What is the minimum value for which we get a "perfect fit"? How does this result relates with your conclusions on the approximation error above?

ANSWER TO Q_10

```
N_train, N_test = 10, 1000
emp_risk = []
for d in range(10):
    true_d, esti_d = 2, d
    x_train, y_train = draw_sample(deg_true=true_d, a=a, N=N_train)
    X_train = get_design_mat(x=x_train, deg=esti_d)
    esti_b = least_square_estimator(X_train, y_train)
    emp_risk.append(empirical_risk(X_train, y_train, esti_b))

plt.plot(range(10), emp_risk, label='Empirical Risk', color='red', alpha=0.8)
plt.title('Empirical Risk VS d')
plt.xlabel('d')
plt.ylabel('Empirical Risk')
plt.legend()
plt.show()
```




```
emp_risk
```

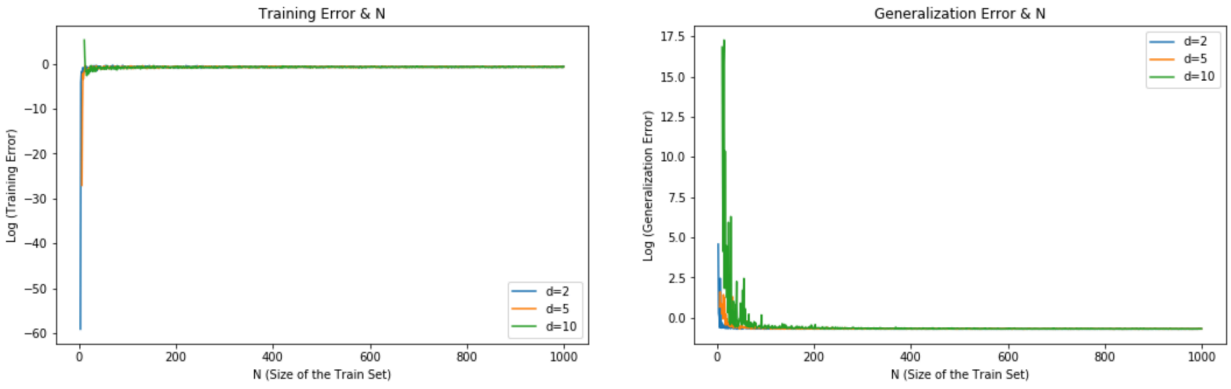
```
[1.5030719653456406,
 0.04886300621441407,
 9.68326761158792e-28,
 1.9938577708596854e-25,
 1.143015372461334e-22,
 1.0966908714390253e-19,
 1.9132537079401906e-16,
 2.9029778107946162e-11,
 0.01016300044355156,
 8.651662401906895]
```

- The Minimum value that we get a 'perfect fit' is 2, with the empirical risk=9.68326761158792e-28.
- This result is the same as the conclusion on the approximation error above.
- Once the hypothesis space H_d has a degree d that is larger than the degree of the true underlying function, we could get the approximation error 0.

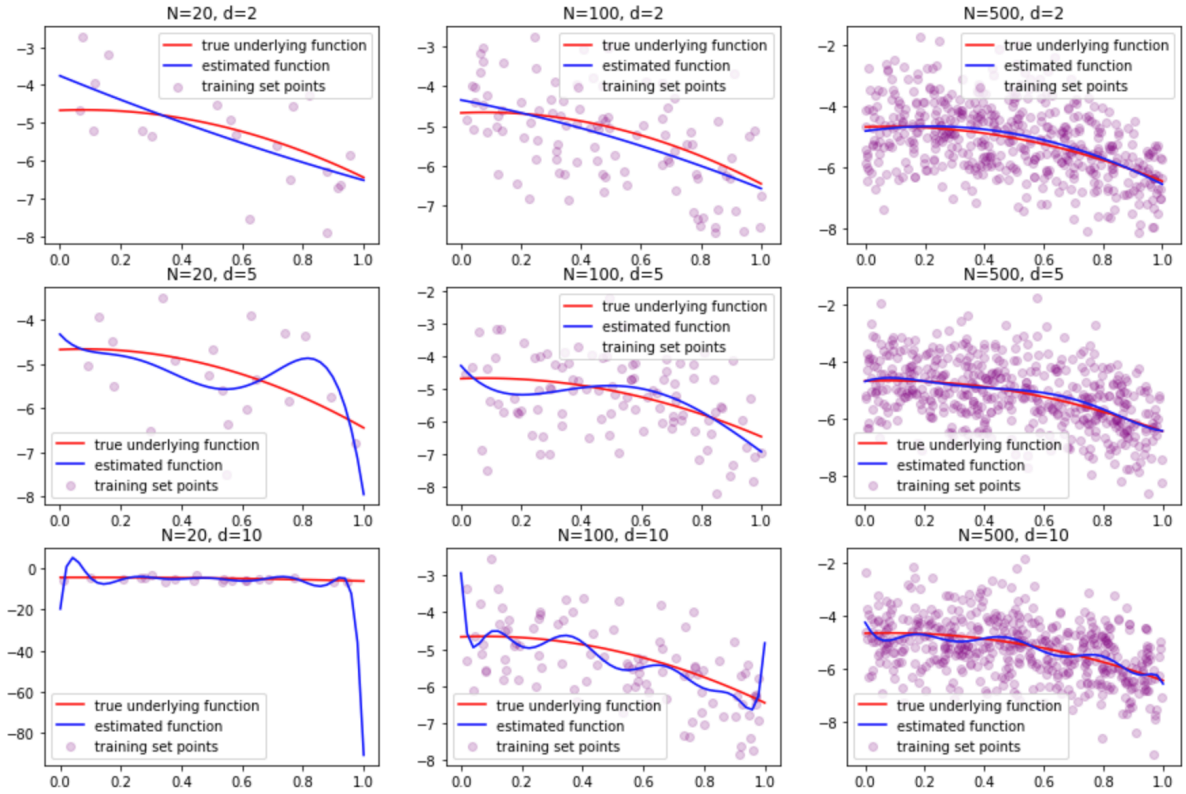
11. Plot e_t and e_g as a function of N for $d < N < 1000$ for $d = 2$, $d = 5$ and $d = 10$ (Plot 2). You may want to use a logarithmic scale in the plot. Include also plots similar to Plot 1 for 2 or 3 different values of N for each value of d.

ANSWER TO Q_11

```
x_test, y_test = draw_sample_with_noise(deg_true=true_d, a=a, N=N_test)

def get_et_eg(esti_d):
    et = []
    eg = []
    for n in range(esti_d+1, 1000):
        x_train, y_train = draw_sample_with_noise(deg_true=true_d, a=a, N=n)
        X_train = get_design_mat(x_train, deg=esti_d)
        X_test = get_design_mat(x_test, deg=esti_d)
        esti_b = least_square_estimator(X_train, y_train)
        et.append(empirical_risk(X_train, y_train, esti_b))
        eg.append(empirical_risk(X_test, y_test, esti_b))
    return et, eg


fig = plt.figure(figsize=(18,5))

ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)

for esti_d in [2, 5, 10]:
    ax1.plot(range(esti_d+1,1000), np.log(get_et_eg(esti_d)[0]), label=f'd={esti_d}')
    ax2.plot(range(esti_d+1,1000), np.log(get_et_eg(esti_d)[1]), label=f'd={esti_d}')

ax1.set_title('Training Error & N')
ax2.set_title('Generalization Error & N')
ax1.set_xlabel('N (Size of the Train Set)')
ax2.set_xlabel('N (Size of the Train Set)')
ax1.set_ylabel('Log (Training Error)')
ax2.set_ylabel('Log (Generalization Error)')
ax1.legend()
ax2.legend()

plt.show()

```



```python
fig = plt.figure(figsize=(15,10))
iteration_times = 1
N_test = 1000

for esti_d in [2, 5, 10]:
    for N_train in [20, 100, 500]:
        x_train, y_train = draw_sample_with_noise(deg_true=true_d, a=a, N=N_train)
        X_train = get_design_mat(x=x_train, deg=esti_d)
        esti_b = least_square_estimator(X_train, y_train)

        gx = get_design_mat(np.linspace(0,1), 2) @ a
        fb = get_design_mat(np.linspace(0,1), esti_d) @ esti_b

        ax = fig.add_subplot(3,3,iteration_times)
        iteration_times += 1
        ax.scatter(x_train, y_train, label='training set points', color='purple', alpha=0.2)
        ax.plot(np.linspace(0,1), gx, label='true underlying function', color='red')
        ax.plot(np.linspace(0,1), fb, label='estimated function', color='blue')
        ax.set_title(f'N={N_train}, d={esti_d}')
        ax.legend()
```

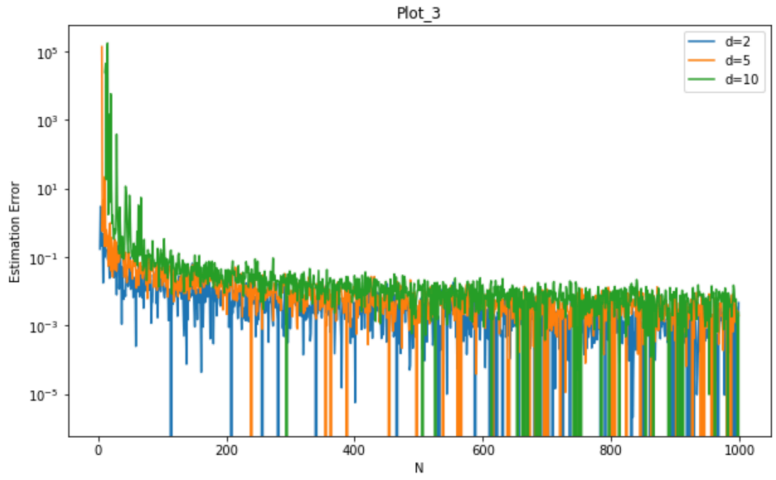


12. Recall the definition of the estimation error. Using the test set, (which we intentionally chose large so as to take advantage of the law of large numbers) give an empirical estimator of the estimation error. For the same values of N and d above plot the estimation error as a function of N (Plot 3).

ANSWER TO Q_12

Recall the definition of the estimation error. Using the test set, (which we intentionally chose large so as to take advantage of the law of large numbers) give an empirical estimator of the estimation error. For the same values of N and d above plot the estimation error as a function of N (Plot 3).

```
plt.figure(figsize=(10, 6))

x_test, y_test = draw_sample_with_noise(deg_true=true_d, a=a, N=N_test)
X_test_ = get_design_mat(x=x_test, deg=true_d)
risk = empirical_risk(X_test_, y_test, a)

def test_estimation_error(estimated_d):
    esti_err = []
    for N_train in range(estimated_d+1, 1000):
        x_train, y_train = draw_sample_with_noise(deg_true=true_d, a=a, N=N_train)
        X_train = get_design_mat(x=x_train, deg=estimated_d)
        estimated_b = least_square_estimator(X_train, y_train)

        X_test = get_design_mat(x=x_test, deg=estimated_d)


        emp_risk = empirical_risk(X_test, y_test, estimated_b)
        esti_err.append(emp_risk-risk)

    return esti_err

for esti_d in [2, 5, 10]:
    plt.plot(range(esti_d+1,1000), test_estimation_error(esti_d), label=f'd={esti_d}')
    plt.yscale('log')
plt.legend();
plt.title('Plot_3');
plt.xlabel('N');
plt.ylabel('Estimation Error');
```



13. The generalization error gives in practice an information related to the estimation error. Comment on the results of (Plot 2 and 3). What is the effect of increasing N? What is the effect of increasing d?

From Plot_2 and Plot_3:

- By increasing N, the estimation error and generalization error both decrease. With more data available, the law of large numbers can be fully taken use of, thus our estimation about the expected risk using the average behavior could be more accurate.
- By increasing d, the estimation error and generalization error both increase. With the same amount of data and an estimated function with greater degree, chances are that we may choose a much more complex function and lead to overfit easily.
- To sum up, the estimation error and generalization are highly correlated.

14. Besides from the approximation and estimation there is a last source of error we have not discussed here. Can you comment on the optimization error of the algorithm we are implementing?

- In this case, there is no optimization function.
- Since the empirical risk function here is convex, we could calculate the global minimum directly, rather than keep wandering around the true empirical risk minimizer.

Answer to Question 15: