Assignment #1

Due: 11:59pm February 8, 2019

Homework 1: Linear Regression

Introduction

This homework is on different forms of linear regression and focuses on loss functions, optimizers, and regularization. Linear regression will be one of the few models that we see that has an analytical solution. These problems focus on deriving these solutions and exploring their properties.

If you find that you are having trouble with the first couple problems, we recommend going over the fundamentals of linear algebra and matrix calculus. We also encourage you to first read the Bishop textbook, particularly: Section 2.3 (Properties of Gaussian Distributions), Section 3.1 (Linear Basis Regression), and Section 3.3 (Bayesian Linear Regression). Note that our notation is slightly different but the underlying mathematics remains the same.

Please type your solutions after the corresponding problems using this LaTeX template, and start each problem on a new page. You will submit your solution PDF, your tex file, and your code to Canvas.

Problem 1 (Priors as Regularization,15pts)

In this problem we consider a model of Bayesian linear regression. Define the prior on the parameters as,

$$p(\theta) = \mathcal{N}(\theta \,|\, \mathbf{0}, \sigma_\theta^2 \mathbf{I}),$$

where σ_θ^2 is as scalar variance hyperparameter that controls the variance of the Gaussian prior. Define the likelihood as,

$$p(\mathbf{y} \,|\, \mathbf{X}, \theta) = \prod_{i=1}^{n} \mathcal{N}(y_i \,|\, \theta^\mathsf{T} \mathbf{x}_i, \sigma_n^2),$$

where σ_n^2 is another fixed scalar defining the variance.

1. Using the fact that the posterior is the product of the prior and the likelihood (up to a normalization constant), i.e.,

   $$\arg\max_\theta \ln p(\theta \,|\, \mathbf{y}, \mathbf{X}) = \arg\max_\theta \ln p(\theta) + \ln p(\mathbf{y} \,|\, \mathbf{X}, \theta).$$

   Show that maximizing the log posterior is equivalent to minimizing a regularized loss function given by $\mathcal{L}(\theta) + \lambda \mathcal{R}(\theta)$, where

   $$\mathcal{L}(\theta) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^\mathsf{T} \mathbf{x}_i)^2$$
   $$\mathcal{R}(\theta) = \frac{1}{2} \theta^\mathsf{T} \theta$$

   Do this by writing $\ln p(\theta \,|\, \mathbf{y}, \mathbf{X})$ as a function of $\mathcal{L}(\theta)$ and $\mathcal{R}(\theta)$, dropping constant terms if necessary. Conclude that maximizing this posterior is equivalent to minimizing the regularized error term given by $\mathcal{L}(\theta) + \lambda \mathcal{R}(\theta)$ for a λ expressed in terms of the problem's constants.

2. Notice that the form of the posterior is the same as the form of the ridge regression loss

   $$\mathcal{L}(\theta) = (\mathbf{y} - \mathbf{X}\theta)^\top (\mathbf{y} - \mathbf{X}\theta) + \lambda \theta^\top \theta.$$

   Compute the gradient of the loss above with respect to θ. Simplify as much as you can for full credit. Make sure to give your answer in vector form.

3. Suppose that $\lambda > 0$. Knowing that $\mathcal{L}$ is a convex function of its arguments, conclude that a global optimizer of $\mathcal{L}(\theta)$ is

   $$\theta = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} \tag{1}$$

   For this part of the problem, assume that the data has been centered, that is, pre-processed such that $\frac{1}{n} \sum_{i=1}^{n} x_{ij} = 0$.

4. What might happen if the number of weights in θ is greater than the number of data points N? How does the regularization help ensure that the inverse in the solution above can be computed?

Solution

1. $lnp(\mathbf{y} \mid \mathbf{X}, \theta) = \sum_{i=1}^{n} lnN(y_i | \theta^T \mathbf{x_i}, \sigma_n^2) = \sum_{i=1}^{n} [-\frac{1}{2} ln(2\pi\sigma_n^2) - \frac{(y_i - \theta^T \mathbf{x_i})^2}{2\sigma_n^2}]$
   $lnp(\theta) = lnN(\theta \mid \mathbf{0}\sigma_\theta^2 \mathbf{I}) = -\frac{1}{2} ln(2\pi\sigma_\theta^2) - \frac{1}{2\sigma_\theta^2} \theta^T \theta$
   $lnp(\mathbf{y} \mid \mathbf{X}, \theta) + lnp(\theta) \propto \sum_{i=1}^{n} [-\frac{(y_i - \theta^T \mathbf{x_i})^2}{2\sigma_n^2}] - \frac{1}{2\sigma_\theta^2} \theta^T \theta \propto -[\frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^T \mathbf{x_i})^2 + \frac{\sigma_n^2}{\sigma_\theta^2} \frac{1}{2} \theta^T \theta]$
   $lnp(\mathbf{y} \mid \mathbf{X}, \theta) + lnp(\theta) \propto -[L(\theta) + \frac{\sigma_n^2}{\sigma_\theta^2} R(\theta)]$
   Maximizing the negative of the bracketed expression above is equal to minimizing the bracketed expression. Thus, maximizing the log posterior is equivalent to minimizing a regularized loss function where $\lambda = \frac{\sigma_n^2}{\sigma_\theta^2}$.

2. Call $u = (\mathbf{y} - \mathbf{X}\theta)$ where $\mathbf{y}$ is a m by 1, $\mathbf{x}$ is n by m, and θ is m by 1.
   $L(\theta) = u^T u + \lambda \theta^t \theta$
   $\frac{\partial L(\theta)}{\partial \theta} = 2u^T \frac{\partial u}{\partial \theta} + 2\lambda \theta^T$
   $\rightarrow \frac{\partial u}{\partial \theta} = \frac{\partial (\mathbf{y} - \mathbf{X}\theta)}{\partial \theta} = -\mathbf{X}$
   $\frac{\partial L(\theta)}{\partial \theta} = -2(\mathbf{y} - \mathbf{X}\theta)^T \mathbf{X} + 2\lambda \theta^T$

3. Set $\frac{\partial L(\theta)}{\partial \theta} = 0$.
   $0 = -2(\mathbf{y} - \mathbf{X}\theta)^T \mathbf{X} + 2\lambda \theta^T$
   $(\mathbf{y} - \mathbf{X}\theta)^T \mathbf{X} = \lambda \theta^T$
   $\mathbf{y}^T \mathbf{X} - \theta^T \mathbf{X}^T \mathbf{X} = \lambda \theta^T$
   $\mathbf{y}^T \mathbf{X} = \theta^T (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})$
   $\mathbf{X}^T \mathbf{y} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \theta$
   $(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y} = \theta$

4. If the number of weights are greater than the number of points, then $\mathbf{X}$ will not be invertible - i.e. $\mathbf{X}$ is singular - meaning that $\mathbf{X}^T \mathbf{X}$ is also not invertible. When a matrix is singular, its inverse is ill defined and thus the optimization solved above breaks down. However, when you introduce perturbations into the covariance matrix $\mathbf{X}^T \mathbf{X}$, then we can guarantee that for any λ not equal to the eigenvalues of the covariance matrix that the matrix $(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})$ is not singular, meaning we have a proper solution for the weights.

Problem 2 (Optimizing a Kernel, 15pts)

Kernel-based regression techniques are similar to nearest-neighbor regressors: rather than fit a parametric model, they predict values for new data points by interpolating values from existing points in the training set. In this problem, we will consider a kernel-based regressor of the form:

$$f(x^*) = \frac{\sum_n K(x_n, x^*) y_n}{\sum_n K(x_n, x^*)}$$

where (x_n, y_n) are the training data points, and $K(x, x')$ is a kernel function that defines the similarity between two inputs x and x'. A popular choice of kernel is a function that decays with the distance between the two points, such as

$$K(x, x') = \exp(-||x - x'||_2^2) = \exp(-(x - x')(x - x')^T)$$

However, the squared Euclidean distance $||x - x'||_2^2$ may not always be the right choice. In this problem, we will consider optimizing over squared Mahalanobis distances

$$K(x, x') = \exp(-(x - x')W(x - x')^T)$$

where W is a symmetric D by D matrix. Intuitively, introducing the weight matrix W allows for different dimensions to matter differently when defining similarity.

1. Let $\{(x_n, y_n)\}_{n=1}^N$ be our training data set. Suppose we are interested in minimizing the squared loss. Write down the loss over the training data $\mathcal{L}(W)$ as a function of W.

2. In the following, let us assume that $D = 2$. That means that W has three parameters: W_{11}, W_{22}, and $W_{12} = W_{21}$. Expand the formula for the loss function to be a function of these three parameters.

3. Derive the gradients with respect to each of the parameters in W.

4. Consider the following data set:

```
x1 , x2 , y
  0 , 0 , 0
  0 , .5 , 0
  0 , 1 , 0
  .5 , 0 , .5
  .5 , .5 , .5
  .5 , 1 , .5
  1 , 0 , 1
  1 , .5 , 1
  1 , 1 , 1
```

   And the following kernels:

$$W_1 = \alpha \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad W_2 = \alpha \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix} \qquad W_3 = \alpha \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

   with $\alpha = 10$. Write some code to compute the loss with respect to each kernel. Which does best? Why? Does the choice of α matter?

5. **Bonus, ungraded.** Code up a gradient descent to optimize the kernel for the data set above. Start your gradient descent from W_1. Report on what you find.

Solution

1. $L(W) = \sum_{i=1}^{n}(y_i - f(x_i))^2 = \sum_{i=1}^{n}(y_i - \frac{\sum_{j=1,j\neq i}^{n} k(x_j,x_i)y_i}{\sum_{j=1,j\neq i}^{n} k(x_j,x_i)})^2 = \sum_{i=1}^{n}(y_i - \frac{\sum_{j=1,j\neq i}^{n} exp[-(x_j-x_i)W(x_j-x_i)^T]y_i}{\sum_{j=1,j\neq i}^{n} exp[-(x_j-x_i)W(x_j-x_i)^T]})^2$

2. $L(W) = \sum_{i=1}^{n}(y_i - \frac{\sum_{j=1,j\neq i}^{n} exp[-(x_j-x_i)W(x_j-x_i)^T]y_i}{\sum_{j=1,j\neq i}^{n} exp[-(x_j-x_i)W(x_j-x_i)^T]})^2 = \sum_{i=1}^{n}(y_i - \frac{\sum_{j=1,j\neq i}^{n} exp[B_j]y_i}{\sum_{j=1,j\neq i}^{n} exp[B_j]})^2$
   $\rightarrow$ where $B_j = -[(x_{j,1} - x_{i,1})^2 w_{1,1} + (x_{j,2} - x_{i,2})^2 w_{2,} + 2(x_{j,2} - x_{i,2})(x_{j,1} - x_{i,1})w_{1,2}]$

3. $\frac{\partial L(W)}{\partial w_{1,1}} = \sum_{i=1}^{n} 2(y_i - \frac{\sum_{j=1,j\neq i}^{n} exp[B_j]y_i}{\sum_{j=1,j\neq i}^{n} exp[B_j]}) \cdot (-1) \cdot \frac{\partial(\frac{\sum_{j=1,j\neq i}^{n} exp[B_j]y_i}{\sum_{j=1,j\neq i}^{n} exp[B_j]})}{\partial w_{1,1}}$
   where $\frac{\partial(\frac{\sum_{j=1,j\neq i}^{n} exp[B_j]y_i}{\sum_{j=1,j\neq i}^{n} exp[B_j]})}{\partial w_{1,1}} =$
   $= \frac{(\sum_{j=1,j\neq i}^{n} y_j exp(B_j)\frac{\partial B_j}{\partial w_{1,1}}) \cdot (\sum_{j=1,j\neq i}^{n} exp(B_j)) - (\sum_{j=1,j\neq i}^{n} y_j exp(B_j)) \cdot (\sum_{j=1,j\neq i}^{n} exp(B_j)\frac{\partial B_j}{\partial w_{1,1}})}{[\sum_{j=1,j\neq i}^{n} exp(B_j)]^2}$
   and where $\frac{\partial B_j}{\partial w_{1,1}} = (-(x_{j,1} - x_{i,1}))$

   Similarly,
   $\frac{\partial L(W)}{\partial w_{2,2}}$ follows the same pattern as $\frac{\partial L(W)}{\partial w_{1,1}}$ but where we replace $\frac{\partial B_j}{\partial w_{1,1}}$ with $\frac{\partial B_j}{\partial w_{2,2}} = (-(x_{j,2} - x_{i,2}))$

   Finally,
   $\frac{\partial L(W)}{\partial w_{1,2}}$ follows from $\frac{\partial L(W)}{\partial w_{1,1}}$ but where we replace $\frac{\partial B_j}{\partial w_{1,1}}$ with $\frac{\partial B_j}{\partial w_{1,2}} = 2(x_{j,2} - x_{i,2})(x_{j,1} - x_{i,1})$

4. Code is displayed at the end of this document. In calculating the loss for all three weight matrices, we see that W_1 strictly does best over the others. I believe this is the case because both dimensions of the data - the xs - are highly correlated with the response, so $W_!$ maintains information from both these dimensions instead of diminishing them by the value 0.1.

Problem 3 (Modeling Changes in Republicans and Sunspots, 15pts)

The objective of this problem is to learn about linear regression with basis functions by modeling the number of Republicans in the Senate. The file `data/year-sunspots-republicans.csv` contains the data you will use for this problem. It has three columns. The first one is an integer that indicates the year. The second is the number of sunspots. The third is the number of Republicans in the Senate. The data file looks like this:

```
Year,Sunspot_Count,Republican_Count
1960,112.3,36
1962,37.6,34
1964,10.2,32
1966,47.0,36
```

and you can see plots of the data in the figures below. The horizontal axis is the year, and the vertical axis is the number of Republicans and the number of sunspots, respectively.

(Data Source: http://www.realclimate.org/data/senators_sunspots.txt)

1. Implement basis function regression with ordinary least squares for years vs. number of Republicans in the Senate. Some sample Python code is provided in `linreg.py`, which implements linear regression. Plot the data and regression lines for the simple linear case, and for each of the following sets of basis functions (use basis (b) only for Republicans v. Years, skip for Sunspots v. Republicans):
   (a) $\phi_j(x) = x^j$ for $j = 1, \ldots, 5$
   (b) $\phi_j(x) = \exp \frac{-(x-\mu_j)^2}{25}$ for $\mu_j = 1960, 1965, 1970, 1975, \ldots 2010$
   (c) $\phi_j(x) = \cos(x/j)$ for $j = 1, \ldots, 5$
   (d) $\phi_j(x) = \cos(x/j)$ for $j = 1, \ldots, 25$
2. In addition to the plots, provide one or two sentences for each with numerical support, explaining whether you think it is fitting well, overfitting or underfitting. If it does not fit well, provide a sentence explaining why. A good fit should capture the most important trends in the data.
3. Next, do the same for the number of sunspots vs. number of Republicans, using data only from before 1985. What bases provide the best fit? Given the quality of the fit, would you believe that the number of sunspots controls the number of Republicans in the senate?

Solution

Attached at the end of this document are the plots and other information relevant for this solution.

1. Plots with performance measures at end.
2. In regressing Republican Count on Year (with no extra basis), we see that the line does a fair job at capturing an overall increasing trend relationship. However, we clearly see some non-linear, sinusoidal curve not captured by the linear fit. In failing to capture the non-linearity, this model underfits to the data, as seen in performance measures of R^2 and mean squared error compared to the other models.

In regressing Republican Count on Year with the (a) basis, we see the model capturing non-linearity in the data. However, we can visually confirm that the data does not hold to a polynomial fit. Thus, this model under-performs, fairing worse than the no-extra-basis model on performance measures of R^2 and mean squared error.

In regressing Republican Count on Year with the (b) basis, we see the model capturing non-linearity and the apparent increasing, sinusoidal nature of the data. In capturing this relationship, we see significant performance improvement in measures of R^2 and mean squared error. However, we see that the model-fit hugs the data very closely, passing directly through several data points with jagged movements. This model may be slightly overfitting to the data.

In regressing Republican Count on Year with the (c) basis, we see the model capturing non-linearity and the apparent increasing, sinusoidal nature of the data. However, while this model does capture the latter, we see significant reduced performance compared to the previous model. Further, visually, the line does not seem to "fit" the data well. Likely, the data is influence significantly by the data from 1980, 1982, and 1984 - the beginning of the Reagan years. This model slightly underfits.

In regressing Republican Count on Year with the (d) basis, we see clear evidence of overfitting. Importantly, the number of predictors exceeds the number of data points. As a result, we see the mode;-fit passing directly through several data points. The latter is shown in its near perfect performance measures on the training data. Most likely, this model model will underperform on new data - it is not generalizable .

Among all the bases (multiple of basis?), basis (c) seems to be the most well fit. This model is by no means ideal, but it captures the important increasing, sinusoidal nature of the data without overfitting.

3. In looking at the plots of regressing Republican count on Sunspot Count, we see clear evidence that the model with basis (d) overfits. Using basis (c) reduces the severity of the overfitness, but the extreme jagged turns in the fit warrants caution of overfitting. Basis (a) and the no-extra-basis model look to capture a slight positive relationship with the data, but the evidence of this relationship is by no means substantial as evidence by the poor performance measures and the visual fit of the data. It appears that there are some influential data points on the upper portion of the plot influencing the regression fit, but without them the fit would become flatter than it already is. Thus, I do not believe that the number of sunspots controls or indicates the number of Republicans in the senate.

Problem 4 (Administrative)

- Name: Meriton Ibrahimi
- Email: meritonibrahimi@college.harvard.edu
- Collaborators: N/A
- Approximately how long did this homework take you to complete (in hours): a couple hours (maybe 8) with the latex and formatting taking up some more time (couple hours)

cs181_hw1

February 9, 2019

In [1]:
```
# libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# implements OLS
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

0.1 Problem 2

In [2]:
```
#data
x1 = np.array([0,0,0,.5,.5,.5,1,1,1])
x2 = np.array([0,.5,1,0,.5,1,0,.5,1])
x = np.stack((x1,x2), axis=1)
y = x1.copy()
W1 = np.array([[1,0],[0,1]])
W2 = np.array([[0.1,0],[0,1]])
W3 = np.array([[.1,0],[0,0.1]])
```

In [3]:
```
# function
def loss(x, W):

    # helper
    # kernel
    def kern(xj, xi, W):
        x = (xj-xi).reshape(1,2)
        temp = np.matmul(np.matmul(x,W), x.T)
        return(np.exp(-temp)[0][0])

    loss = 0
    for i in range(9):
        top = 0
        bottom = 0
        for j in range(9):
            if i != j:
```

```
                    k = kern(x[j], x[i], W)
                    top += k*y[j]
                    bottom += k
                loss += np.power(y[i] - (top/bottom),2)
            return(loss)
```

In [4]:
```
# caclulate loss
for alpha in [1,5,10,15,20,25,50,100,500,1000]:
    print('alpha = {}'.format(alpha))
    print('W1: {:4f}'.format(loss(x,alpha*W1)))
    print('W2: {:4f}'.format(loss(x,alpha*W2)))
    print('W2: {:4f}'.format(loss(x,alpha*W3)))
    print()
```

```
alpha = 1
W1: 1.184018
W2: 1.956544
W2: 1.809430

alpha = 5
W1: 0.436880
W2: 2.130945
W2: 1.493124

alpha = 10
W1: 0.338216
W2: 2.226382
W2: 1.184018

alpha = 15
W1: 0.314893
W2: 2.178787
W2: 0.957612

alpha = 20
W1: 0.308236
W2: 2.047904
W2: 0.795466

alpha = 25
W1: 0.306324
W2: 1.909514
W2: 0.680220

alpha = 50
W1: 0.305557
W2: 1.569671
W2: 0.436880
```

```
alpha = 100
W1: 0.305556
W2: 1.501659
W2: 0.338216

alpha = 500
W1: 0.305556
W2: 1.500000
W2: 0.305557

alpha = 1000
W1: 0.305556
W2: 1.500000
W2: 0.305556
```

From the above, we see that W1 strictly provides the lower loss.

0.2 Problem 3

```
In [5]: # read in data
        dat = pd.read_csv('data/year-sunspots-republicans.csv')
        display(dat.head())
```

	Year	Sunspot_Count	Republican_Count
0	1960	112.3	36
1	1962	37.6	34
2	1964	10.2	32
3	1966	47.0	36
4	1968	105.9	43

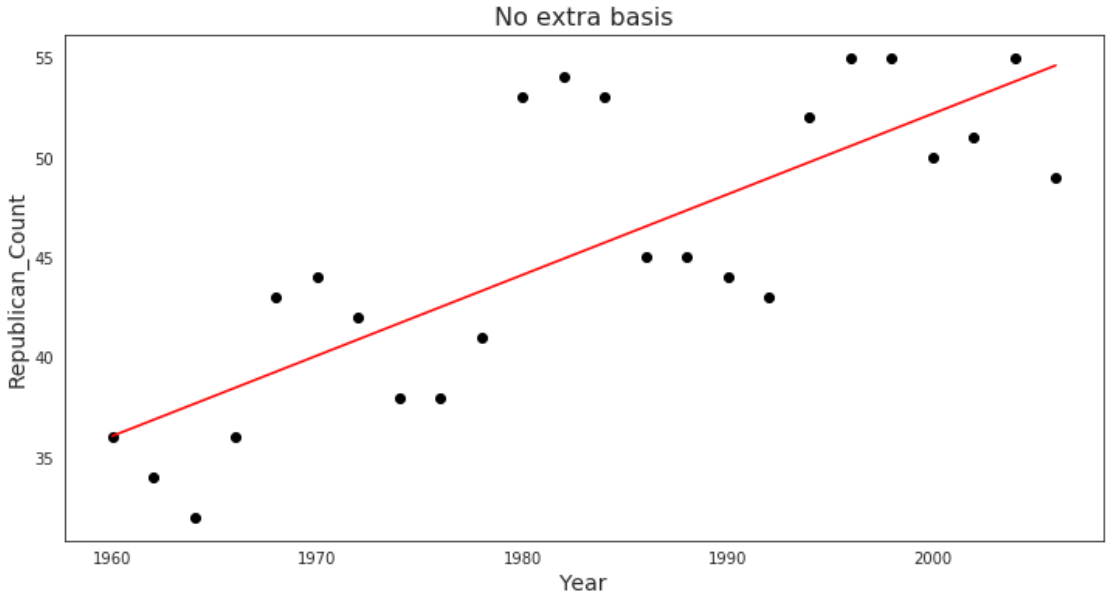
```
In [6]: # basis functions
        def b1(x,j):
            return (np.power(x,j))
        def b2(x, mu):
            return (np.exp(-np.square((x-mu))/25))
        def b3(x,j):
            return (np.cos(x/j))
```

```
In [7]: def make_plot(X, y, x_lab, y_lab, title):
            lin_reg = LinearRegression().fit(X,y)
            with sns.axes_style("white"):
                fig, ax = plt.subplots(1, 1, figsize=(12, 6))
                ax.scatter(X[x_lab], y, color = 'k')
                ax.set_xlabel(x_lab, fontsize=14)
                ax.set_ylabel(y_lab, fontsize=14)
```

```
        pred = lin_reg.predict(X)
        ax.plot(X[x_lab], pred, color='r')
        ax.set_title(title, fontsize = 16)
        print('R2 Score: {:.4f}'.format(r2_score(y, pred)))
        print('Mean Sqaured Error: {:.4f}'.format(mean_squared_error(y, pred)))
```

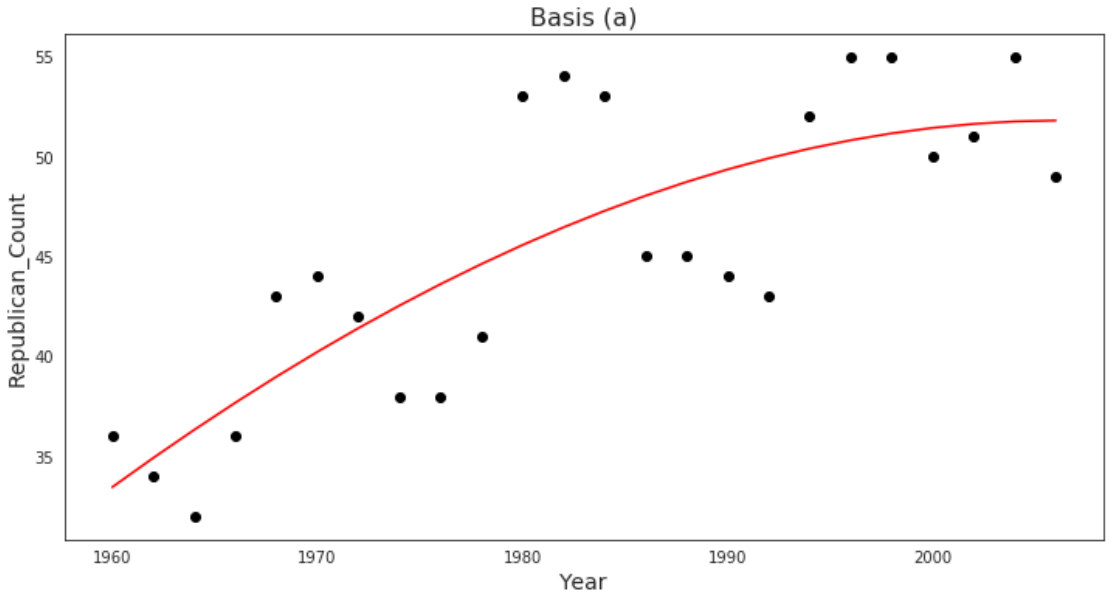
In [8]:
```
# Year on Republican_Count
make_plot(dat['Year'].to_frame(), dat['Republican_Count'],
          'Year', 'Republican_Count', 'No extra basis')
```

```
R2 Score: 0.6138
Mean Sqaured Error: 19.5871
```



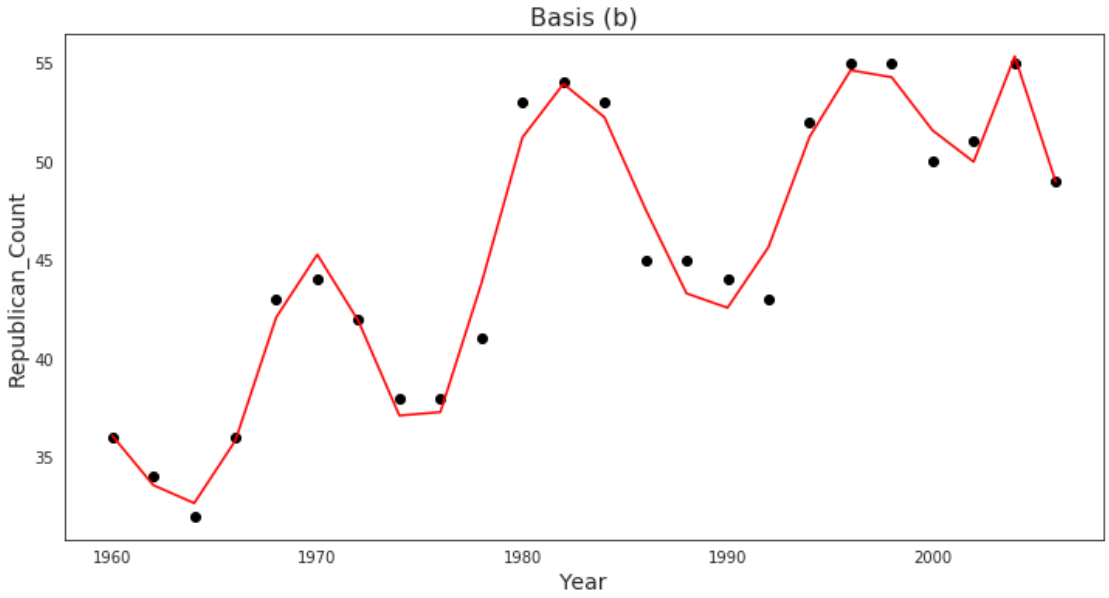
In [9]:
```
# 3.1.a
X = dat['Year'].to_frame()
y = dat['Republican_Count'].to_frame()
for i in [1,2,3,4,5]:
    s = 'x^'+str(i)
    X[s] = X['Year'].apply(b1, args = (i,))
make_plot(X, y, 'Year', 'Republican_Count', 'Basis (a)')
```

```
R2 Score: 0.6510
Mean Sqaured Error: 17.7029
```



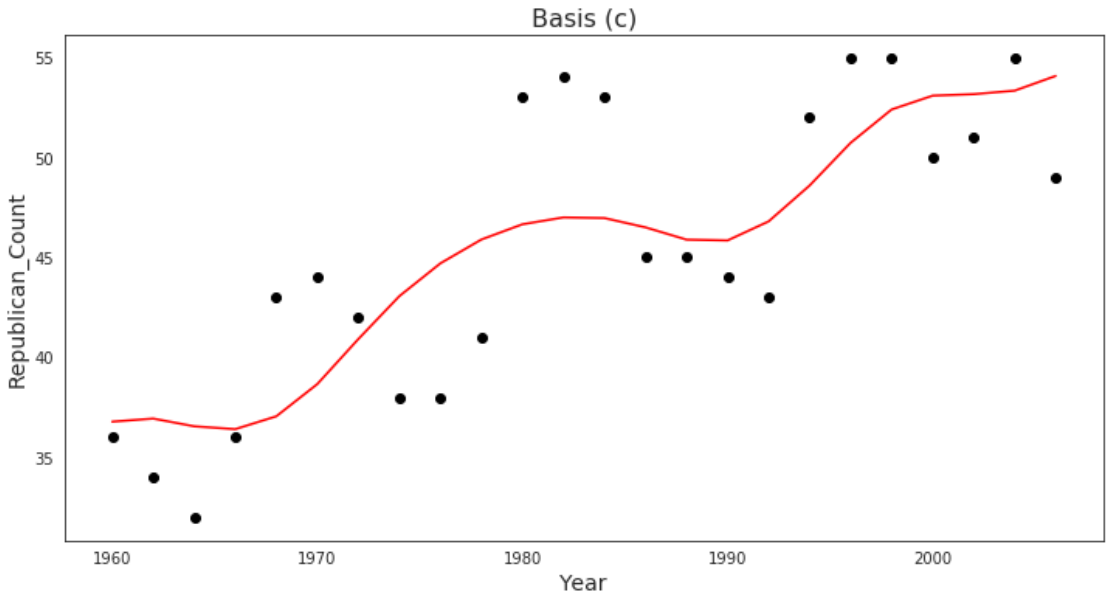

```
In [10]: # 3.1.b
         X = dat['Year'].to_frame()
         y = dat['Republican_Count'].to_frame()
         step = 5
         for i in np.arange(1960, 2010+step, step):
             s = 'x:'+str(i)
             X[s] = X['Year'].apply(b2, args = (i,))
         make_plot(X, y, 'Year', 'Republican_Count', 'Basis (b)')

R2 Score: 0.9675
Mean Sqaured Error: 1.6490
```



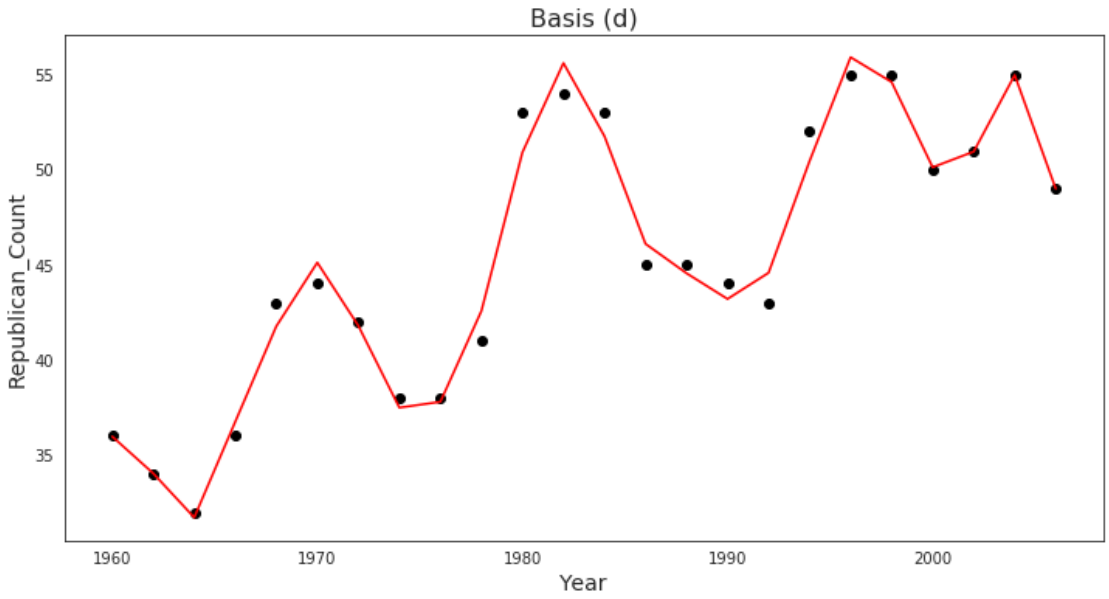

```
In [11]: # 3.1.c
         X = dat['Year'].to_frame()
         y = dat['Republican_Count'].to_frame()
         for i in [1,2,3,4,5]:
             s = 'x:'+str(i)
             X[s] = X['Year'].apply(b3, args = (i,))
         make_plot(X, y, 'Year', 'Republican_Count', 'Basis (c)')
```

```
R2 Score: 0.6586
Mean Sqaured Error: 17.3162
```



In [12]:
```
# 3.1.d
X = dat['Year'].to_frame()
y = dat['Republican_Count'].to_frame()
for i in np.arange(1,25+1):
    s = 'x:'+str(i)
    X[s] = X['Year'].apply(b3, args = (i,))
make_plot(X, y, 'Year', 'Republican_Count', 'Basis (d)')
```

```
R2 Score: 0.9814
Mean Sqaured Error: 0.9435
```



In [13]:
```
X.shape
```

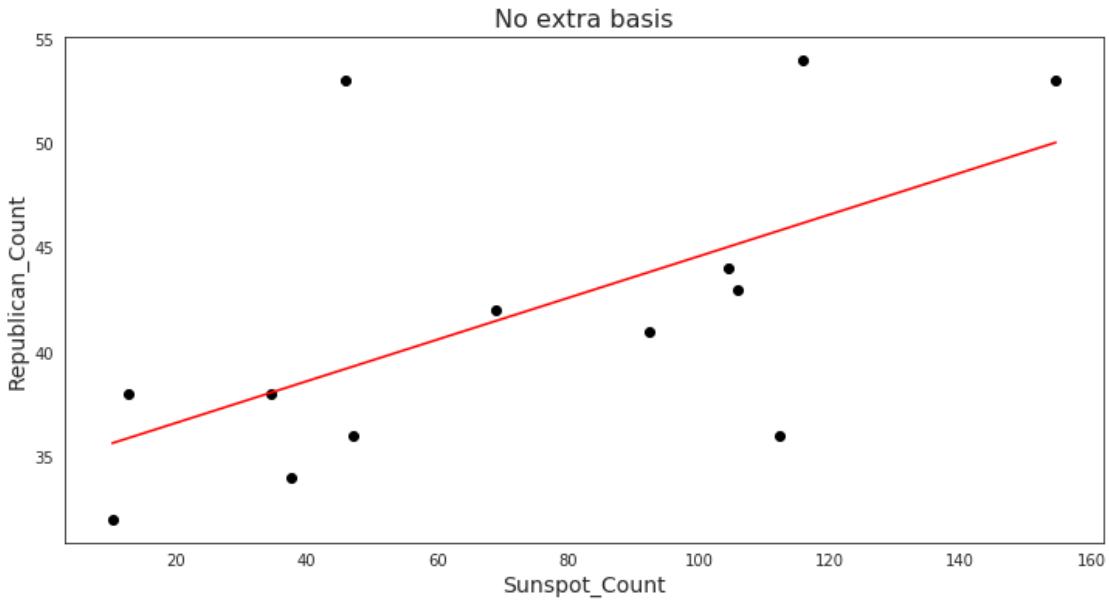
Out[13]:
```
(24, 26)
```

In [14]:
```
# filter for part(3)
dat = dat[dat['Year'] < 1985].sort_values('Sunspot_Count')
display(dat.head())
```

	Year	Sunspot_Count	Republican_Count
2	1964	10.2	32
8	1976	12.6	38
7	1974	34.5	38
1	1962	37.6	34
12	1984	45.9	53

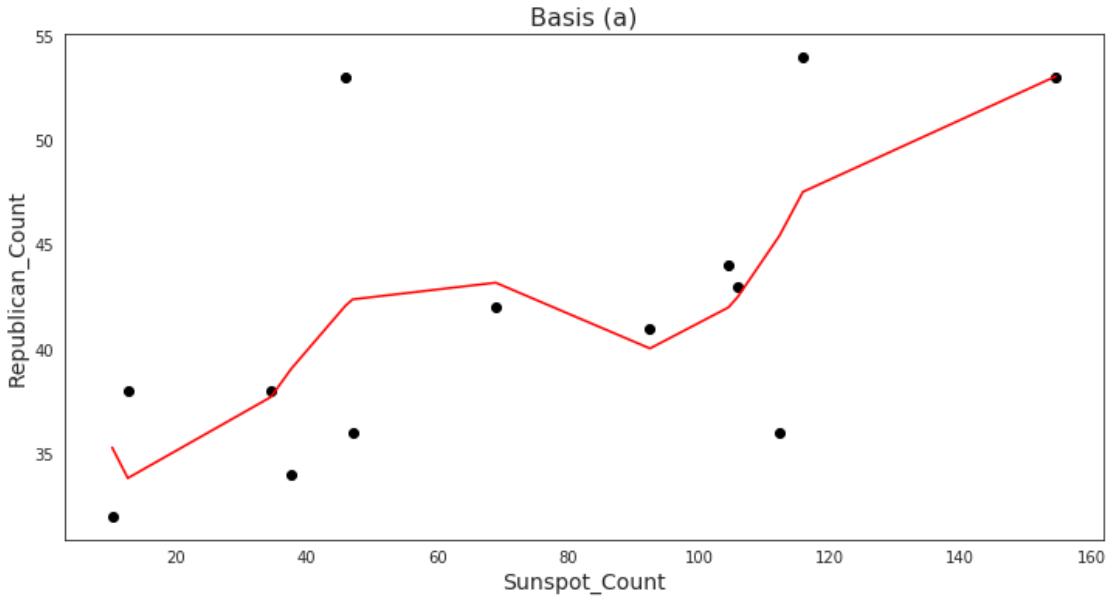
In [15]:
```
# Year on Republican_Count
make_plot(dat['Sunspot_Count'].to_frame(), dat['Republican_Count'],
          'Sunspot_Count', 'Republican_Count', 'No extra basis')
```

```
R2 Score: 0.3651
Mean Sqaured Error: 32.2192
```



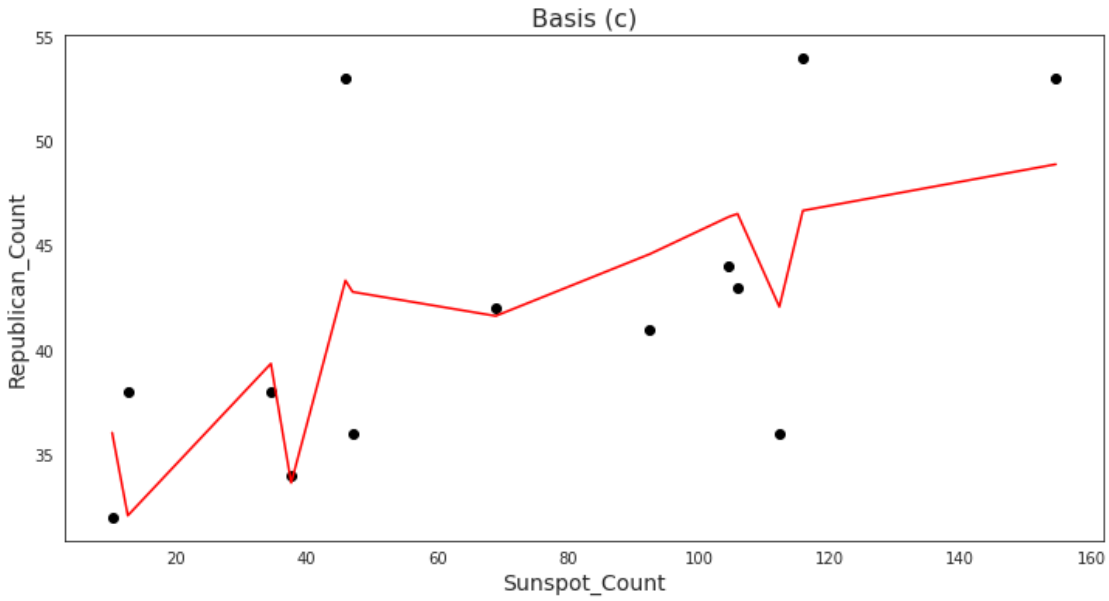
In [16]:
```
# 3.3.a
X = dat['Sunspot_Count'].to_frame()
y = dat['Republican_Count'].to_frame()
for i in [1,2,3,4,5]:
    s = 'x^'+str(i)
    X[s] = X['Sunspot_Count'].apply(b1, args = (i,))
make_plot(X, y, 'Sunspot_Count', 'Republican_Count', 'Basis (a)')
```

```
R2 Score: 0.4676
Mean Sqaured Error: 27.0175
```



In [17]:
```
# 3.3.c
X = dat['Sunspot_Count'].to_frame()
y = dat['Republican_Count'].to_frame()
for i in [1,2,3,4,5]:
    s = 'x:'+str(i)
    X[s] = X['Sunspot_Count'].apply(b3, args = (i,))
make_plot(X, y, 'Sunspot_Count', 'Republican_Count', 'Basis (c)')
```

```
R2 Score: 0.4976
Mean Sqaured Error: 25.4936
```



```
In [18]: # 3.3.d
         X = dat['Sunspot_Count'].to_frame()
         y = dat['Republican_Count'].to_frame()
         for i in np.arange(1,25+1):
             s = 'x:'+str(i)
             X[s] = X['Sunspot_Count'].apply(b3, args = (i,))
         make_plot(X, y, 'Sunspot_Count', 'Republican_Count', 'Basis (d)')
```

```
R2 Score: 1.0000
Mean Sqaured Error: 0.0000
```