3 min to Spreed

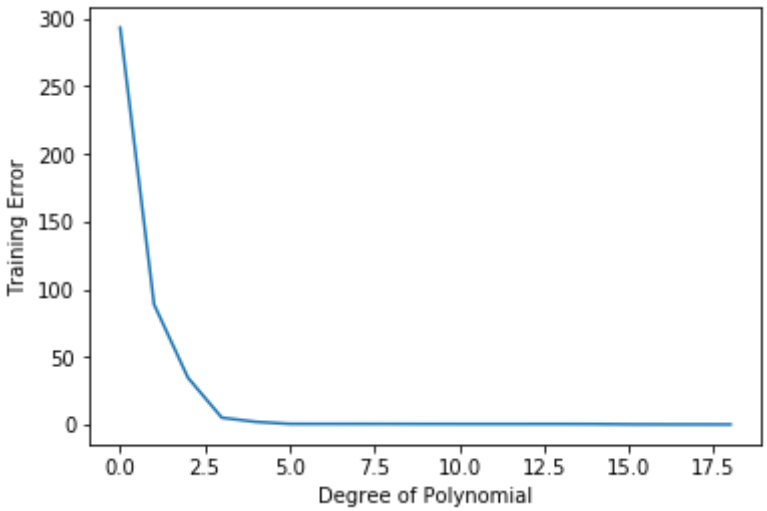
In [61]:
```
import numpy as np
import matplotlib.pyplot as plt
import os
import math
import scipy.io as spio
import pandas as pd
import itertools
from scipy.special import comb



plot_col = ['r', 'g', 'b', 'k', 'm']
plot_mark = ['o', '^', 'v', 'D', 'x', '+']

# Plots the rows in 'ymat' on the y-axis vs. 'xvec' on the x-axis
# with labels 'ylabels'
# and saves figure as pdf to 'dirname/filename'
def plotmatnsave(ymat, xvec, ylabels, dirname, filename):
    no_lines = len(ymat)
    fig = plt.figure(0)

    if len(ylabels) > 1:
        for i in range(no_lines):
            xs = np.array(xvec)
            ys = np.array(ymat[i])
            plt.plot(xs, ys, color = plot_col[i % len(plot_col)], lw=1,
label=ylabels[i])

        lgd = plt.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc=3, nco
l=3, mode="expand", borderaxespad=0.)

    savepath = os.path.join(dirname, filename)
    plt.xlabel('$x$', labelpad=10)
    plt.ylabel('$f(x)$', labelpad=10)
    plt.savefig(savepath, bbox_extra_artists=(lgd,), bbox_inches='tight'
)
    plt.close()

# Sets the labels
labels = ['$e^x$', '1st order', '2nd order', '3rd order', '4th order']

# TODO: Given x values in "x_vec", save the respective function values e
^x,
# and its first to fourth degree Taylor approximations
# as rows in the matrix "y_mat"

#scipy.interpolate.approximate_taylor_polynomial(np.exp(x), 0, 0)
xvex = np.arange(-20,8, 1)
y_mat = [[np.exp(x) for x in xvex],
        [1 + x for x in xvex],
       [1 + x + (x**2)/2 for x in xvex],
       [1 + x + (x**2)/2 + (x**3)/6 for x in xvex],
       [1 + x + (x**2)/2 + (x**3)/6 + (x**4)/24 for x in xvex]]
```

3 min to Spreed

```
# Define filename, invoke plotmatnsave
filename = 'approx_plot_2.pdf'
plotmatnsave(y_mat, xvex, labels, '.', filename)
```

In [ ]:

In [135]:
```
# There is numpy.linalg.lstsq, which you should use outside of this clas
ss
def lstsq(A, b):
    return np.linalg.solve(A.T @ A, A.T @ b)

def fit(x, y, d):
    X = np.array([[x_t**i for i in range(d)] for x_t in x])
    return X, lstsq(X, y)

data = spio.loadmat('1D_poly.mat', squeeze_me=True)
x_train = np.array(data['x_train'])
y_train = np.array(data['y_train']).T

n = 20  # max degree

err = np.zeros(n - 1)

for i in range(n - 1):
    X, w = fit(x_train, y_train, i)
    err[i] = np.sum(np.square(y_train - X.dot(w))) / len(y_train)

plt.plot(err)
plt.xlabel('Degree of Polynomial')
plt.ylabel('Training Error')
plt.show()
```



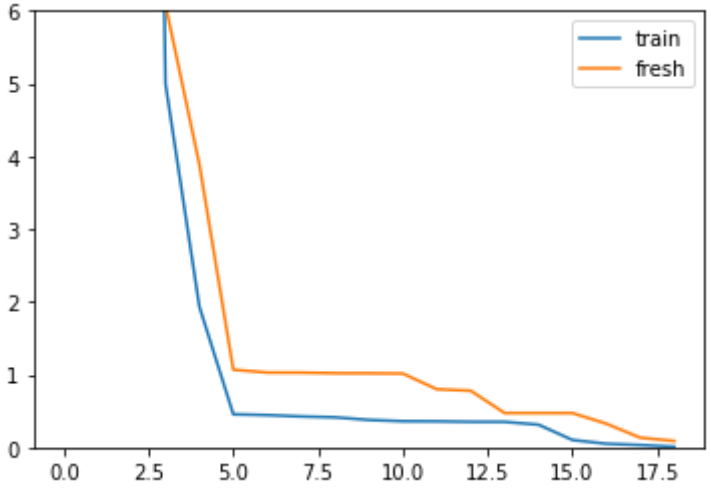
5.c

3 min to Spreed

Average training error decreases as a function of D. Increasing the degree of the polynomial function used to fit the training data decreases the average error. Once the degree hits d-1, 1 - the number of data points used by our model, our model predicts the training values perfectly/exactly. We will not be able to fit a polynomial of degree n with the standad matrix inversion method as this will have a non-trivial null space.

5.d

In [138]:
```
data = spio.loadmat('1D_poly.mat', squeeze_me=True)
x_train = np.array(data['x_train'])
y_train = np.array(data['y_train']).T
y_fresh = np.array(data['y_fresh']).T

n = 20  # max degree
err_train = np.zeros(n - 1)
err_fresh = np.zeros(n - 1)

for i in range(n - 1):
    X, w = fit(x_train, y_train, i)
    err_train[i] = np.sum(np.square(y_train - X.dot(w))) / len(y_train)

for i in range(n - 1):
    X, w = fit(x_train, y_fresh, i)
    err_fresh[i] = np.sum(np.square(y_fresh - X.dot(w))) / len(y_fresh)

plt.figure()
plt.ylim([0, 6])
plt.plot(err_train, label='train')
plt.plot(err_fresh, label='fresh')
plt.legend()
plt.show()
```



3 min to Spreed

This plot shows how our model predictions for fresh are less accurate than they were for the training data. This is due to the fact that as the degree of the polynomial used to fit the data increases, the predictions become more complex and more tailored specifically to the training values. There is a tradeoff between reducing the error we get in the training data and the generalizability of the model to other data that is not part of the training set.

5.e

"Peach lady" should use the model which best minimizes errors for the fresh data. This occurs when a 5 degree polynomial is used to fit the data. Beyond this degree the model becomes overfitted to the training data. The complexity serves to hamper the effectiveness of our model for fresh data.

5.F 5.G

In [ ]:

3 min to Spreed

In [57]:
```
data = spio.loadmat('polynomial_regression_samples.mat', squeeze_me=True
)
data_x = data['x']
data_y = data['y']
Kc = 4  # 4-fold cross validation
KD = 6  # max D = 6


def gen(D,k):
    " Creates the powers of all multivariate conditions with k variables
 and max degree n.\n",
    " Code from karakfa from\n",
    " https://stackoverflow.com/questions/37711817/generate-all-possible
-outcomes-of-k-balls-in-n-bins-sum-of-multinomial-catego\n",
    " \"\"\"\n"
    if(k==1):
        return [[D]]
    if(D==0):
        return [[0]*k]
    return [ g2 for x in range(D+1) for g2 in [ u+[D-x] for u in gen(x,k
-1) ] ]

def multivariate(data_x, D):
    if D == 0:
        return [1]
    X = []
    for i in range(len(data_x)):
        X.append(multivariate_row(data_x[i], D))
    return X


def multivariate_row(row, D):
    data_row = [1]
    for x in row:
        data_row.append(x)
    data_row = np.array(data_row)

    powers = gen(D, len(data_row))
    result = []
    for j in range(len(powers)):
        result.append(np.prod(data_row**powers[j]))
    return result

#multivariate_features = multivariate(data_x, 2)
D_multivariate_features = []
for k in range(KD):
    D_multivariate_features.append(multivariate(data_x, k))
```

3 min to Spreed

In [146]:
```
data = spio.loadmat('polynomial_regression_samples.mat', squeeze_me=True
)
data_x = data['x']
data_y = data['y']



def lstsqMULTIVARIATE(A, b, lambda_):
    return np.linalg.solve((A.T @ A) + lambda_*np.identity(len(A[0])), A
.T @ b)

X_train_1 = D_multivariate_features[k][:250]
X_test_1 = D_multivariate_features[k][250:]
Y_train_1 = data_y[:250]
Y_test_1 = data_y[250:]
#lambda_ = 0.1
#w = lstsqMULTIVARIATE(np.vstack(X_train_1), Y_train_1, lambda_)
#np.array(X_train_1).dot(w)
#D_multivariate_features[2][:50]
```

3 min to Spreed

In [148]:
```
data = spio.loadmat('polynomial_regression_samples.mat', squeeze_me=True
)
data_x = data['x']
data_y = data['y']
Kc = 4  # 4-fold cross validation
KD = 6  # max D = 6
LAMBDA = [0, 0.05, 0.1, 0.15, 0.2]

def lstsqMULTIVARIATE(A, b, lambda_):
    return np.linalg.solve((A.T @ A) + lambda_*np.eye(len(A[0])), A.T @
b)

#def fit(D, lambda_):

#    err_train = np.zeros(Kc)
#    err_test = np.zeros(Kc)
#    for k in range(Kc):
#
#        X_train = D_multivariate_features[k][k*250:250*(k+1)]
#        X_test = np.append(D_multivariate_features[k][(k+1)*250:], (D_m
ultivariate_features[k][:250*k]))
#        Y_train = data_y[k*250:250*(k+1)]
#        Y_test = np.append(data_y[(k+1)*250:], (D_multivariate_features
[k][:250*k]))
#
#       w = lstsqMULTIVARIATE(np.vstack(X_train), Y_train, lambda_)
#        print('hi mark')
#        err_train[k] = np.sum(np.square((Y_train - np.array(X_train).do
t(w)))) / len(Y_train)
#        err_test[k] = np.sum(np.square((Y_test - np.array(X_test).dot
(w)))) / len(Y_test)
#    return np.sum(err_train) / len(err_train), np.sum(err_test) / len(e
rr_test)


def fit(D, lambda_):
```

3 min to Spreed

```
    err_train = np.zeros(Kc)
    err_test = np.zeros(Kc)
    for k in range(Kc):
        np.random.shuffle(D_multivariate_features[k])
        np.random.shuffle(data_y)

        X_train = D_multivariate_features[k][250:]
        X_test = D_multivariate_features[k][:250]

        Y_train = data_y[250:]
        Y_test = data_y[:250]

        w = lstsqMULTIVARIATE(np.vstack(X_train), Y_train, lambda_)
        err_train[k] = np.sum(np.square((Y_train - np.array(X_train).dot
(w)))) / len(Y_train)
        err_test[k] = np.sum(np.square((Y_test - np.array(X_test).dot(w
)))) / len(Y_test)
    return np.sum(err_train) / len(err_train), np.sum(err_test) / len(er
r_test)




np.set_printoptions(precision=11)
Etrain = np.zeros((KD, len(LAMBDA)))
Evalid = np.zeros((KD, len(LAMBDA)))
for D in range(KD):
    print(D)
    for i in range(len(LAMBDA)):
        Etrain[D, i], Evalid[D, i] = fit(D + 1, LAMBDA[i])

print('Average train error:', Etrain, sep='\n')
print('Average valid error:', Evalid, sep='\n')

# YOUR CODE to find best D and i
```

3 min to Spreed

```
0
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
1
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
2
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
```

3 min to Spreed

```
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
3
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
4
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
5
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
```

3 min to Spreed

```
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
hi mark
Average train error:
[[ 0.07182671679  0.07188244367  0.07187541048  0.07184186813
   0.07184415236]
 [ 0.071753036    0.07183388202  0.0718987976   0.07190746852
   0.07193515026]
 [ 0.07180763744  0.07180836585  0.07183598188  0.07190196048
   0.07182514924]
 [ 0.07175686784  0.07186490528  0.07184289915  0.07189173132
   0.07184330527]
 [ 0.0717986333   0.07186792507  0.07181433832  0.07184588244
   0.07182584588]
 [ 0.07177699539  0.07185205664  0.07185248816  0.07187020781
   0.07185754449]]
Average valid error:
[[ 0.0592034313   0.06190881825  0.06555698559  0.08045415212
   0.07802876645]
 [ 0.08709529159  0.082431425    0.05608399486  0.05306716989
   0.04105100448]
 [ 0.06841046751  0.09179487428  0.08164899329  0.0546876341   0.086535
72942]
 [ 0.08877346328  0.06895134396  0.07905601196  0.05876199631
   0.07874661822]
 [ 0.06894647952  0.06831009089  0.08806512541  0.07703037398
   0.08551068358]
 [ 0.07683447185  0.07544122359  0.07545801487  0.0674781477   0.073897
53221]]
```

Best D for 5.f for lambda = 0.1 appears to be D = 4. This is the case as the average error for the test/valid is smallest here.

In [ ]:

In [ ]:

3 min to Spreed

HW02 - CS 189 Nicholas Lotto

1. a. Worked on HW with: Elimane Oketchoman, Prashanth Ganeth, Daniel Mockaitis

b. I certify that all solutions are entirely in my own words & that I have not looked at another students solutions. I have credited all external sources in this write up.

Nick L (26069160)

noise"

above y gun

Q2. Se[illegible] 2 [illegible] [illegible] [illegible] side

2.a.

$X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^{n \times 2}$ =

$$\min \| \vec{y} - X\vec{w} \|_2^2$$

$$s.t \quad \|w\|_2^2 \leq \beta^2$$

lagrangian unconstrained form

$$\min \| \vec{y} - X\vec{w} \|_2^2 + \lambda \|w\|_2^2$$

b as we increse β our weight, used to penalize, is increased as well. With the lower weight our lagrasian multiplier should be smaller.

- Big B -> lower weights -> less λ req to penalize.

Conclusion: β inverse proportal to λ

c.

$w^T x$: $\hat{y} = w_1(x_1) + w_2 x_2 + \dots + w_n x_n$ for $Xw = y$

adding error to feature vector x

$$\tilde{y} = w_1(x_0 + \varepsilon_1) + \dots + w_n(x_n + \varepsilon_n)$$

$$\tilde{y} - \hat{y} = w_1 \varepsilon_1 + \dots + w_n \varepsilon_n = w^T \varepsilon$$

upper bound

"how much noise" could increase above $\hat{y}$ gun

2.d.

$$\hat{w}_r = (X^T X + \lambda I)^{-1} X^T y$$

$\lim_{\lambda \to \infty} \hat{w}_r$ as $\lambda \to \infty$

- the weights estimate become super small. -> to zero.
- our weights are guaranteed to be @ most 1/2 ∴ as $\lambda \to \infty$ our weights go to zero.

$$\arg\min_w \|y - Xw\|_2^2 + \lambda \|w\|_2^2$$

$$(y - Xw)^T (y - Xw) + \lambda w^T w$$

$$(y^T - w^T X^T)(y - Xw) + \lambda w^T w$$

$$y^T y - y^T Xw - w^T X^T y + w^T X^T Xw + \lambda w^T w$$

$$y^T y - 2w^T X^T y + w^T X^T Xw + \lambda w^T w$$

take $\nabla(\quad)/\delta w$

$$= 0 \quad -2y^T X + 2X^T Xw + 2\lambda w = 0$$

$$w(2X^T X + 2\lambda) = 2y^T X$$

$$w = (X^T X + \lambda I)^{-1} y^T X$$

2.e.

Part I. $X^TX + \lambda I$

SVD of $X^TX + \lambda I$

$A = U\Sigma V^T$ let $X = U\Sigma V^T$

$$= V\begin{bmatrix}\Sigma_r & 0\\ 0 & 0\end{bmatrix}U^T U\begin{bmatrix}\Sigma_r & 0\\ 0 & 0\end{bmatrix}V^T + \lambda I$$

$$= V\begin{bmatrix}\Sigma_r^2 & 0\\ 0 & 0\end{bmatrix}V^T + \lambda I$$

$\updownarrow$ -> V is orthogonal $\therefore \begin{cases} V^T = V^{-1} \\ V^TV = I \end{cases}$

$$= V\begin{bmatrix}\Sigma_r^2 & 0\\ 0 & 0\end{bmatrix}V^T + V(\lambda I)V^T$$

$$= V\left(\begin{bmatrix}\Sigma_r^2 & 0\\ 0 & 0\end{bmatrix}V^T + \lambda I\right)V^T$$

$$= V\begin{bmatrix}\Sigma_r^2 + \lambda & 0\\ 0 & \lambda I\end{bmatrix}V^T$$

[• eigenvalues are along the diagonal
$\Sigma_n^2 + \lambda, \ldots, \lambda$ are eigenvalues

thus the eigenvalues for $X^TX + \lambda I$ are

$\sigma_1^2 + \lambda, \ldots, \sigma_d^2 + \lambda, \lambda$ for remaining eigenvalues that do not correspond to eigenvectors for the X^TX matrix.

Part II. regularizer term makes X^TX Full rank. Therefore $X^TX + \lambda I$ Now has good numerical inversion operations.

2+. IF X^TX had small eigenvalues than $(X^TX)^{-1}$ will cause sensitivity issues. For OLS this will result in a sensitive solution, where small changes in the features matrix result in large / drastic changes to the weight estimate.

$\lambda_1 = 100$ is a better option for the eigenvalues OF X. That is because the eigenvalues of $X^TX + \lambda I$ will then be larger & thus $(X^TX + \lambda I)^{-1}$ will have singular values which can be at (most / least) $\frac{1}{\lambda}$, Not sensitive / or highly impactful to the resulting weight

189
ote 2.

of X. That is because the eigenvalues of $X^TX + \lambda I$ will then be larger & thus $(X^TX + \lambda I)^{-1}$ will have singular values which can be at (most / ~~least~~) $\frac{1}{\lambda}$, Not sensitive / or highly dependent to the resulting weight estimate.

ES 189
Note 2.

$$X^TX = V \begin{bmatrix} 1000 & & \\ & \ddots & \\ & & \sqrt{0.02} \end{bmatrix} V^T \quad \begin{bmatrix} \text{some} \\ \text{small eigenvalues} \end{bmatrix}$$

"adding λI"

$$X^TX + \lambda I = V \begin{bmatrix} \sqrt{1000+\lambda} & 0 & 0 \\ 0 & 1+\lambda & 0 \\ 0 & 0 & \sqrt{0.01+\lambda} \end{bmatrix} V^T$$

- if $(X^TX)^{-1}$ is big then the estimate is very sensitive "little errors result in big errors" this occurs if X^TX has small eigenvalues. Adding a larger λ (piped w/ respect to the eigenvectors) helps elevate this sensitivity nature of $(X^TX + \lambda I)^{-1}$

2.h.

$\lim_{\lambda \to 0}$

- $X^TX + \lambda I$ is full rank & invertible.
- The singular values become $\frac{1}{\sigma^2 + \lambda}$ & $\frac{1}{\lambda}$, many values are @ most $1/\lambda$ ∴ as $\lim_{\lambda \to 0}$ than our singular values approach zero. This effectively reverts the effect of ridge regression by making our hyperparam approach zero.
- λ is no longer > 0 ∴ we no longer have a unique solution for when the feature matrix & strong samples make up an underdetermined system.

"From g"

$$\lim_{\lambda \to 0} \vec{w}(x) = \lim_{\lambda \to 0} (X^TX)^{-1}X^Ty + (\lambda I)^{-1}X^Ty$$

$$\Rightarrow \text{approaches OLS} = (X^TX)^{-1}X^Ty +$$

2.1. $\hat{w} = \arg\min_w \frac{1}{2}\|y - Xw\|_2^2 + \lambda\|\Gamma w\|_2^2$

$= \min_w \left(\frac{1}{2}(y-Xw)^T(y-Xw) + \lambda(\Gamma w)^T(\Gamma w) \right)$

$\frac{1}{2}(y^T - w^T X^T)(y - Xw) + \lambda w^T \Gamma^T \Gamma w$

$\frac{1}{2}(y^T y - y^T Xw - w^T X^T y + w^T X^T X w) + \lambda w^T \Gamma^T \Gamma w$

$w^T X^T y$

are grad resp/w 0

$0 = \frac{\partial}{\partial w} = \frac{1}{2}\cdot 0 - \frac{1}{2}\cdot 2 y^T X \cdot 1 + \frac{1}{2}\cdot 2 \cdot X^T X w + \lambda \cdot 2 \Gamma^T \Gamma w = 0$

$-y^T X + X^T X w + 2\lambda \Gamma^T \Gamma w = 0$ ✓

$F(w) = w^T X^T X w \qquad F(w+\Delta) = (w+\Delta)^T X^T X (w+\Delta)$

$(w^T + \Delta^T) X^T X (w + \Delta)$

$\frac{\partial F(w)}{\partial \Delta} = \frac{F(x+\Delta) - F(x)}{\Delta}$

$((w^T X^T X) + \Delta^T X^T X)(w + \Delta) \quad w^T (X X^T \Delta)$

gradient is derivative of transpose

$w^T X^T X w + w^T X^T X \Delta + \Delta^T X^T X w$

$= F(w) + w^T (X^T X + X X^T)\Delta + \Delta^T X^T X \Delta$

$F(w) + ((X^T X^T + X^T X) w)^T \Delta + 0$

$\therefore$ transpose of derivative is gradient

$\partial F_w = 2 X^T X$

✓ $(X^T X + 2\lambda \Gamma^T \Gamma) w = y^T X$

$\hat{w} = (X^T X + 2\lambda \Gamma^T \Gamma)^{-1} y^T X$

closed form general solution.

(CONT.)

Eg. Show $(X^TX)^{-1}X^Ty = \vec{w}$ has ∞ # of solutions.

If a system is underdetermined with $n < d$ then the matrix X (where X is the features matrix $\in \mathbb{R}^{n \times s}$) has a nontrivial null space. Thus it is also never invertible.

$$X\vec{w} = \vec{y}$$

$$\vec{w} = w_0 + X^T\alpha \qquad w_0 \in \text{nullspace}(X) \quad \alpha \in \mathbb{R}$$

- the vectors spanning the null space can have any linear combination of the basis vectors for the null space added to them & still get a solution

$$\vec{w} = w_0 + X^T\alpha$$

any linear combination of basis of null space gives us a solution as well ∴ we have ∞ solutions to

$$X\vec{w} = y$$

$$\vec{w} = (X^TX)^{-1}X^Ty$$

① theorem from Math 54

2 g $(X^TX+\lambda I)X^Ty=\vec{w}$ has unique solutions for $\lambda>0$

parts)

PSD. Positive semi definite

① If λ is invertible $Ax=b$ has unique sol

② a square matrix A^{-1} is invertible iff determ $\neq 0$

① From part 2.e we determined that $(X^TX+\lambda I)$ is invertible with singular values of @ least $\frac{1}{\lambda}$. In continuation from part e:

$$(X^TX+\lambda I)^{-1}=\left(V\begin{bmatrix}\Sigma r+\lambda & 0\\ 0 & \lambda\end{bmatrix}V^T\right)^{-1}$$

$$=(V^T)^{-1}\begin{bmatrix}(\Sigma r+\lambda)^{-1} & 0\\ 0 & (\lambda)^{-1}\end{bmatrix}V^{-1}$$

Singular values of @ least $1/\lambda$ means that for $\lambda>0$ None of our singular values for $(X^TX+\lambda I)^{-1}$ are zero.

② $(X^TX+\lambda I)$ is a square matrix, due to the previously stated. None of its eigenvalues for singular values are zero $\therefore$ the det, which equals the product of the eigenvalues $\neq 0$. Thus $(X^TX+\lambda I)$ is invertible.

"A is invertible for $Ax=b$ has unique sol"

$Ax=b$ $A^{-1}Ax=A^{-1}b$ $x=A^{-1}b$ unique solution

Do tonight

3h for new parts

3.F

Stars & Bars

l - Dimension: bins • we have $l+1$ bins

d - degree: balls

$l=2 \quad D=3 \qquad \binom{5}{2}$

degree 0

$1, x_{i1}, x_{i2}, x_{i1}^2, x_{i2}^2, x_{i1}x_{i2}, x_{i1}^2 x_{i2}, x_{i1}x_{i2}^2, x_{i1}^3, x_{i2}^3$

x_{i2}, x_{i1}

$l=2 \quad D=2 \qquad \binom{4}{2}$

$1, x_{i1}, x_{i2}, x_{i1}x_{i2}, x_{i2}x_{i1}, x_{i1}^2, x_{i2}^2$

Want: # of ways to Distribute coeff. [to add up l variables (with exponents) that sum to D] $\binom{l+D-1}{l-1}$

$l=2, D=2$ categories [For ex.

$x_{i1}^0 x_{i2}^0, x_{id}^2 \,|\, x_{i1}^1 x_{i2}^0, x_{id}^1 \,|\, x_{i1}^0 x_{i2}^1 x_{id}^1 \,|\, x_{i1}^2 x_{i2}^0 x_{id}^0 \,|\, \ldots$

we need the amt for the terms where the sum of the exponents is D, $\therefore$ we add an extra dummy variable to fulfill this in each of our categories

$\therefore$ the updated problem is

[# ways to add up $l+2$ variables with exponents that sum to degree D. $= \binom{l+D}{l}$

3.g. N sample points $\{x_i\}_{i=1}^{N}$ $\quad x_{i,1} = x_{i,2} = \ldots = x_{i,d} = x_i$

want $F_d \in \mathbb{R}^{N \times \binom{D+d}{d}}$ $\quad$ distinct features $l \in \{1, 2, \ldots, N\}$

Show F_d linearly dependent cols $N > 2$. Compare w/ (e)

$$F = \begin{bmatrix} p_D(x_1) \cdots p_D(x_N) \end{bmatrix}^T = \begin{bmatrix} p_D(x_1) \\ \\ p_D(x_N) \end{bmatrix} = \text{(lets look at hecce expand in 2.F w/ } d=2,\ D=3 \text{ & } N>2$$

lets set $N=2$ for this ex.

$N \times 10 \quad (N \times \binom{D+d}{d})$

$$= \begin{bmatrix} 1 & x_1 & x_1 & x_1^2 & x_1^2 & x_1^2 & x_1^3 & x_1^3 & x_1^3 & x_1^3 \\ 1 & x_2 & x_2 & x_2^2 & x_2^2 & x_2^2 & x_2^3 & x_2^3 & x_2^3 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & x_N & & & & & & & & \end{bmatrix}$$

- $\exists$ $k+1$ identical columns for degree k terms in $p_D(x_i)$
- $\therefore$ $\exists$ linearly dependent columns for a Feature Matrix w/ $N \geq 2$ & $d \geq 2$.
- " the det is zero if F column vectors are linearly dependent" $\therefore$ we know that F has a determ of zero & we know that F is not full rank as @ least two of our input points are equal (i.e. stated in part e), causing @ least one of our eigenvalues for the Feature Matrix to be zero.

compare w/ e.

3.h. Set of Sample point x_i s.t F_ℓ is Full column rank (all column vectors are independent)

(confirm @ new page)

(IDea: / concept for univariate case)

~~Sample method: obtain / sample points from sample until have n distinct points. By (e) we know F full rank unless two input data points are equal.~~

~~$$\{x_i : i \in 1, \dots, n\} \quad s.t \quad x_i \neq x_j \quad j = 1, \dots, n$$~~

~~• want our method to be functional for arbitrary D, l~~

Multivariate case: F_ℓ Full column Rank: independent columns / l-Dimension

let $x_{i,1}, x_{i,2}, \dots, x_{i,\ell}$ be distinct Scalars.

• For each $j = 1, \dots, \ell$ w/in each Sample point i select a distinct scalar (as opposed to having one distinct scalar α_i for each.

let each $\boxed{x_{i,j} = \alpha_i^{(j-1)}}$

$$x_{i,1} = \alpha_i^0$$
$$x_{i,2} = \alpha_i^1$$
$$\vdots$$
$$x_{i,\ell} = \alpha_i^{\ell-1}$$

4.a. i) e^x

@ $x_0 = 0$

1st: $F(x) \approx 1 + x$

2nd: $F(x) \approx 1 + x + \frac{x^2}{2!}$

3rd: $F(x) \approx 1 + x + x^2/2! + x^3/3!$

4th: $F(x) \approx 1 + x + x^2/2! + x^3/3! + x^4/4!$

ii) $\sin(x)$

1st: $0 + x$

2nd: $0 + x - \dots$

3rd: $0 + x - x^3/3!$

4th: $0 + x - x^3/3!$

b. $\|F - \phi_M\|_\infty$ For $M = 2$ $\quad I := [-4, 3]$

$:= \sup_{x \in I} |F(x) - \phi_M(x)| \leq \frac{T \cdot |x - x_0|^{M+2}}{(M+1)!}$

M^{th} derivative

$|F^{(M)}(x)| \leq T \quad x \in (x_0 - s, x_0 + s)$

2nd degree Taylor approximation

$\|F - \phi_2\|_\infty := \sup_{x \in I} |F(x) - \phi_2(x)| \leq \frac{T \cdot |x - x_0|^3}{3!}$

$\Lambda! \sim \sqrt{2\pi\Lambda}\left(\frac{\Lambda}{e}\right)^e$

$\geq \frac{|F^{(2)}(x)| \cdot |x - x_0|^3}{3!} = \frac{|e^3|\,|x - 0|^3}{3!}$

[largest value w/in $[-4, 3]$, the interval, for the second deriv of the function from def$^\Delta$ $T \geq |F^{(M)}(x)|$]

$= \frac{e^3 |x|^3}{3!}$

4.b Compute $\|f - \phi_M\|_\infty$ for arbitrary non zero integers M.

$I = [-4, 3]$ $\quad T \geq |f^{M}(x)|$ $\quad f^{M+1}(x) = f'(x) = e^x$

$$\leq \frac{T\,|x - x_0|^{M+1}}{(M+1)!} \leq$$

- on domain $[-4,3]$, largest M^{th} derivative is w/ $x = 3$ for e^x.

$$\leq \frac{e^3\,|x|^{(M+2)}}{(M+1)!}$$

$$n! \sim \sqrt{2\pi n}^{-1} \cdot \left(\frac{n}{e}\right)^n \qquad \left(\frac{M+1}{e}\right)^{M+1}$$

on interval $I = [-4, 3]$

$$\therefore \leq \frac{e^3\,|x|^{M+2}}{\sqrt{2\pi M + 2\pi}^{-1} \cdot \left(\frac{M+1}{e}\right)^{(M+2)}} = \frac{e^3\,|x|^{M+1}}{\sqrt{2\pi(M+1)}} \cdot \left(\frac{e}{M+1}\right)^{M+1}$$

By wolfram alpha.

- $\lim_{M \to \infty} \|f - \phi_M\|_\infty \leq \lim_{M \to \infty} \left(\frac{e^3\,|x|^{M+1}}{\sqrt{2\pi M + 2\pi}^{-1} \left(\frac{M+1}{e}\right)^{M+1}} \right) = 0$ & logic

- as the order of our taylors approx approaches ∞ the errors approach zero.

- Outside of $I: [-4, 3]$: the approximation error grows. From $[-20, -4]$ the approx is diverse from the true function & from $[3, 8]$ the true value diverges from the taylor approximation.

$I = [-4, 3]$, $\varepsilon << 1$

X or 3, constant?

4.C.

1. Upper bound: $\frac{e^3 |x|^{(M+1)}}{(M+1)!} = \frac{e^3 |x|^{M+1}}{\sqrt{2\pi M + 2\pi}} \cdot \left(\frac{e}{M+1}\right)^{M+1}$ on approx error.

Show IF $D > O(\log(1/\varepsilon))$ $\Rightarrow$ $|F(x) - \phi_D(x)| \leq \varepsilon$

For degree D, UB approx error is: $\frac{e^3}{\sqrt{2\pi D + 2\pi}} \cdot \left(\frac{|x|e}{D+1}\right)^{(D+2)} \leq \varepsilon$ (Bounded by ε)

$\sqrt{3} \cdot \sqrt{3} = \sqrt{3 \cdot 3}$ = constant factors not relevant so can drop: $\sqrt{2\pi}$, $\frac{x}{(3)}$, $\sqrt{2\pi(D+1)} =$

$$\frac{1}{\varepsilon} \leq \frac{\sqrt{2\pi D + 2\pi}}{e^3} \cdot \left(\frac{D+2}{|x|e}\right)^{D+2}$$

$$\log(1/\varepsilon) \leq \log\left(\sqrt{D+2} \cdot e^{-3} \cdot (D+1/|x|e)^{D+2}\right)$$

$$\lg(1/\varepsilon) \leq \log\left(\sqrt{D+2}\right) - 3 \cdot \log(e) + (D+1)\log\left(\frac{D+2}{|x|e}\right)$$

(constant)

$$\leq \frac{1}{2}\log(D+1) + (D+1)\log(D+1) - (D+1)\lg(|x|e)$$

$$- (D+1)\log(|x|) - (D+1)\log(e)$$

$$\lg(\tfrac{1}{\varepsilon}) \leq \tfrac{1}{2}(\log(D+1)) + (D+1)\log(D+2) - (D+1)\lg(|x|) - (D+1)$$

$$- (D+1)(1 + \log(|x|))$$

$$\log(1/\varepsilon) \leq 1/2\log(D+1) + (D+1)\log(D+2) - (D+2)$$

(constant)

$$\log(1/\varepsilon) \leq \underbrace{\frac{D+1}{2} \cdot \log(D+1) - (D+2)}$$

(*) Argument

$\varepsilon << 1$ $\rightsquigarrow$ $\log(1/\varepsilon) \leq D \cdot \log(D)$ ∴

(?) come back

...

4.C.

ii) Upper bound: $\frac{1 \cdot |x|^{D+1}}{\sqrt{2\pi(D+2}} \left(\frac{e}{D+1}\right)^{D+1}$

$$\varepsilon \geq \left(2\pi(D+1)\right)^{-1/2} \left(\frac{|x|e}{D+1}\right)^{D+1}$$

$$\frac{1}{\varepsilon} \leq \left(2\pi(D+1)\right)^{1/2} \left(\frac{D+1}{|x|e}\right)^{D+1}$$

$$\log(1/\varepsilon) \leq \frac{1}{2}\log(D+1) + (D+1)\log(D+1) - (D+1)(\log|x|e) - (D+1)$$

$$\log(1/\varepsilon) \leq \frac{D+1}{2}\log(D+1) - (D+1)$$

$$\rightsquigarrow \underline{\log(1/\varepsilon) \leq D\log(D)} \quad (\Phi) \text{ argument}$$

Φ Argument for part i) & ii) (Small Sufficiently)

- given that $D > \log(1/\varepsilon)$ $\therefore$ $D > f(\text{Big } *)$ as $\varepsilon \ll 1$

thus we know that $\log(D) > 1$ & we

$\therefore$ know $D \cdot \log(D) > \log(1/\varepsilon)$.

- we can now reverse the arguments shown above for (i) & (ii) to show this implies that $|F(x) - \phi_D(x)| \leq \varepsilon$. (applicable for both $f(x)=0$ & $f(x)=\sin(x)$ (i) (ii))

U.D. $\forall \varepsilon > 0 \quad \exists \quad D \geq 1 \quad s.t \quad \|F - p_D^{g(x)}\|_\infty < \varepsilon$

w/ respect to interval I (any closed I)

$|f^M(x)| \leq T \quad M \geq 1 \quad x \in I$ "any Function" Univariate polynomial

$$g(x) \leq \frac{1}{(M+1)!} T (x - x_0)^{M+1} < \varepsilon$$

$$\lim_{M \to \infty} g(x) = T \cdot \lim_{M \to \infty} \frac{(x - x_0)^{M+1}}{(M+1)!} = 0$$

- as M approaches infinity (we get a base high degree polynomial) that we can use to approximate any F on a closed interval.
- $(M+1)! > (X - X_0)^{M+1}$ as $M \to \infty \therefore \lim_{M \to \infty} \frac{1}{M} = 0$

4.e

I) Multivariate Taylor Series expansion in Matrix form

$F(v) = F(x,y) \qquad v = [x,y]^T \qquad v_0 = [x_0, y_0]^T$

- $F(v) = F(v_0) + \nabla F(v)^T \Delta v + \frac{1}{2} \Delta v^T \nabla^2 F(v) \Delta v +$ Higher order terms
- $\Delta v = \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$
- $\nabla^2 F(v) = H(F(x)) =$ Hessian of $F(x) = \begin{bmatrix} F_{xx} & F_{xy} \\ F_{yx} & F_{yy} \end{bmatrix}$
- $\nabla F(v) = \begin{bmatrix} F_x \\ F_y \end{bmatrix}^T$

II) $F(v) = e^x y^2 \qquad v = [x,y]^T \qquad @ \; v_0$ Multivariate 2nd Taylor

$$F(v) \approx e^{x_0} y_0^2 + \begin{bmatrix} e^{x_0} y_0^2 \\ 2y_0 e^{x_0} \end{bmatrix}^T \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} x - x_0 & y - y_0 \end{bmatrix} \begin{bmatrix} e^{x_0} y_0^2 & 2y_0 e^{x_0} \\ 2y_0 e^{x_0} & 2e^{x_0} \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$

$$\nabla F(v) = \begin{bmatrix} e^x y^2 \\ 2y e^x \end{bmatrix} \qquad \nabla^2 F(v) = \begin{bmatrix} e^x y^2 & 2y e^x \\ 2y e^x & 2e^x \end{bmatrix}$$

4.F. $F(v) = e^x y^2$ $\quad v_0 = 0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\quad v - v_0 = v$ "Δvela"

$\vec{v}(t) = \vec{0} + t(\vec{v} - \vec{0})$

$g(t) = F(v(t))$ Scalar to a scalar

$\vec{v}(t) = t\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} tx \\ ty \end{bmatrix} = [tx \;\; ty]^T$

$F([tx \;\; ty]^T)$ $\quad \boxed{F(v(t)) = e^{tx}(yt)^2}$

$\frac{1}{2}[tx \;\; ty]\begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} tx \\ ty \end{bmatrix}$ (2×1, 2×2, 2×1)

$g(t) = \begin{cases} F(v(t)) = 0 + t\begin{bmatrix} 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} + t^2\left(\frac{1}{2}[x \;\; y]\begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}\right) \\ = t \cdot \frac{1}{2} \cdot 2y^2 = t \cdot y^2 \\ \boxed{g(t) = t^2 \cdot y^2} \end{cases}$

$[x \;\; y]\begin{bmatrix} 0 \\ 2y \end{bmatrix} = [2y^2]$ (1×2, 2×1)

$g(t) = g(t_0) + g'(t_0)(t - t_0) + \frac{1}{2}g''(t_0)(t - t_0)^2$

$g(t_0) = e^{t_0 x}(y t_0)^2$

$g'(t_0) = (y t_0)^2 x e^{t_0 x} + 2y^2 t_0 e^{t_0 x}$

$g''(t_0) = 2t_0 y^2 x e^{t_0 x} + y^2 t_0^2 x^2 e^{t_0 x} + 2y^2 e^{t_0 x} + 2y^2 x t_0 e^{t_0 x}$

$g(t) =$ as $g(t_0) = F(v(t_0))$

$g'(t_0) = \nabla F'(v(t_0))$

$g''(t_0) = \nabla^2 F(v(t_0))$

$t_0 = 0 \;\therefore\; \rightarrow g(t) = g(0) + g'(0) \cdot t + \frac{1}{2}g''(0) \cdot t^2$ ∴

5. Jane & her big ass peaches

a. n Jane peaches $x_1 \dots x_n$ (Mass) predictors

- x_j for $j = 1, \dots, n$ are our predictors.
- y_i represents our response

- we can use (x_i, y_i) as our Jane data

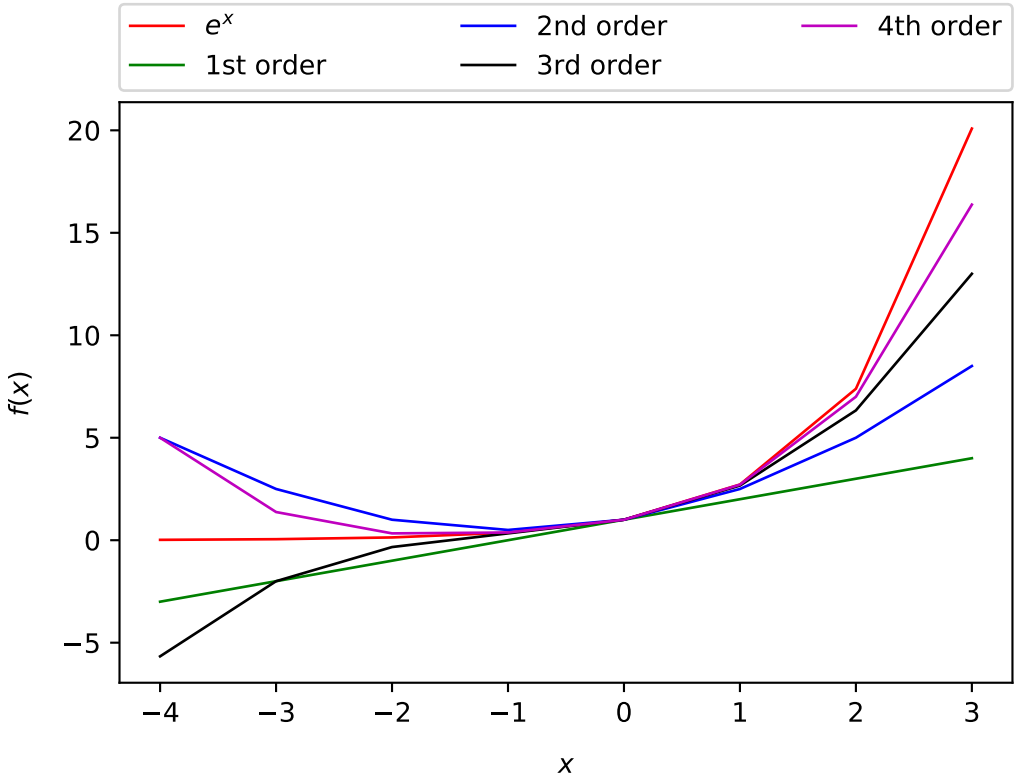
$$\hat{y}_i \approx w_0 + w_1 x_i + w_2 x_i^2 + \dots + w_D x_i^D$$

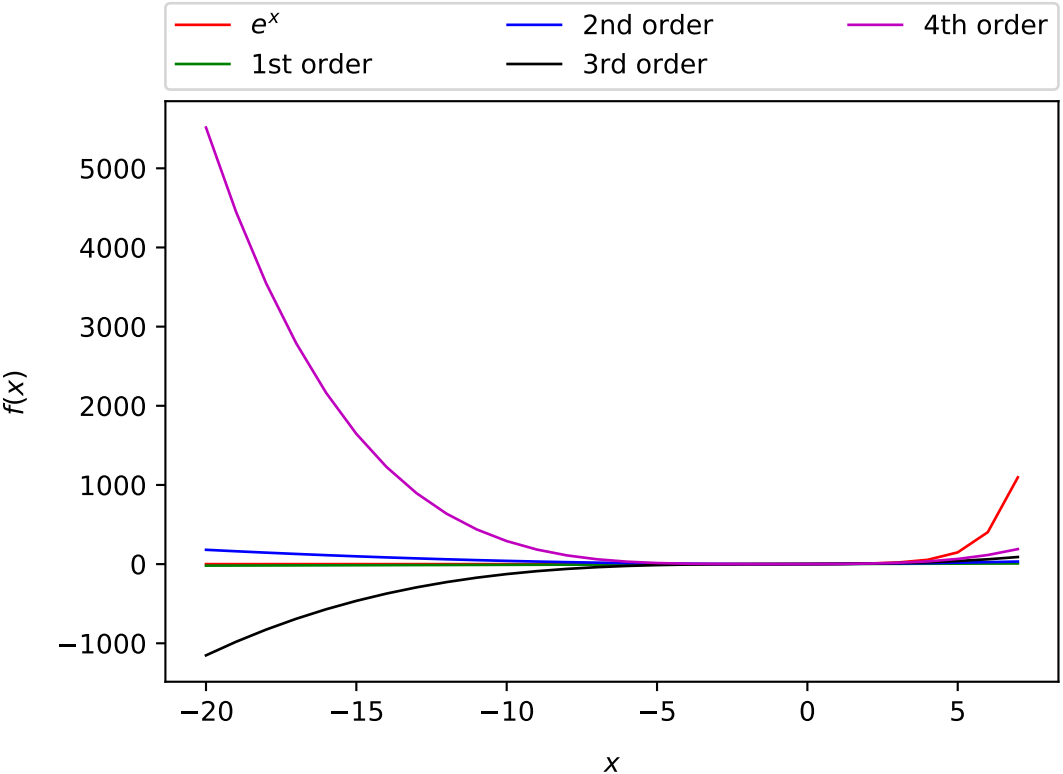
want $\vec{w} = \arg\min_{w} \|Xw - y\|_2^2$

let $X = \begin{bmatrix} 1 & x_i & x_i & \cdots & x_i^D \\ \vdots & & \vdots & & \vdots \\ \cdot & & \cdot & & \cdot \end{bmatrix}$ $i = 1, \dots, n$ be the feature Matrix

$$X \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \vec{\hat{w}} = (X^T X)^{-1} X^T \vec{y}$$

$\vec{w}$ $\vec{y}$


e^x
1st order
2nd order
3rd order
4th order
20
15
10
5
0
−5
$f(x)$
−4
−3
−2
−1
0
1
2
3
x


e^x
1st order
2nd order
3rd order
4th order
5000
4000
3000
2000
1000
0
−1000
f(x)
−20
−15
−10
−5
0
5
x

HW02 - CS 189

1.
Who else did you work with on this homework? In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

I worked on this homework with Ehimare Okoyomon, Prashanth Ganeth, and Daniel Mockaitis. We worked by getting together throughout the week and communicating on facebook.

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up}

Nicholas Lorio, 26089160

Question 8.

What implication does the squared nature of the bias found through bias-variance decomposition have on its behavior (as N approaches infinity, f(x) in h(x, D), etc). More specifically, if f is in the set of hypothesis functions, why does bias decrease as N increases?

To be clear, bias will only decrease as the sample size increases **if** the samples are representative and without inherent biases themselves. Bias arises from omitted variable bias (non representative data), data collection bias, data cleaning bias, etcetera. High bias is related to under-fitting of the, it represents the ability of the model function to approximate the data. Therefore, a higher sample size N (with good/unbiased data points) will decrease the bias of the model. This however is only true given that our function estimate is within the true value function. E.G. that we're not trying to use a linear model to fit a quadratic relationship.

Is hypothesis testing extrapolation or interpolation?

Hypothesis testing is a theoretical method of measuring the effectiveness of a hypothesis function h in relation to our real function f. We are comparing the data amongst known true values, as such it is more akin to interpolation than extrapolation. For real world data, the model is rarely known and the noise model may be entirely wrong. As such we cannot calculate bias and variance. We can however test algorithms over synthetic data.

Source Note 5
http://www.eecs189.org/static/notes/n5.pdf

“4. If f is in the set of hypothesis functions, bias will decrease with more data. If f is not in the set of hypothesis functions, then there is an underfitting problem and more data won’t help.”

Sources

https://en.wikipedia.org/wiki/Taylor_series

https://math.stackexchange.com/questions/499321/why-is-the-determinant-zero-iff-the-column-vectors-are-linearly-dependent

https://drive.google.com/file/d/0BzpkjlP_hA7_anFRU0FjQklBTWs/view

http://www.eecs189.org/static/notes/n3.pdf

http://www.eecs189.org/static/notes/n2.pdf

http://snasiriany.me/files/ml-book.pdf

https://math.stackexchange.com/questions/1985144/relationship-between-null-space-and-determinant

https://en.wikipedia.org/wiki/Taylor%27s_theorem#Taylor's_theorem_for_multivariate_functions

https://stackoverflow.com/questions/37711817/generate-all-possible-outcomes-of-k-balls-in-n-bins-sum-of-multinomial-catego