A Data-Based Perspective on Reducing Mortality Rate in the Global Adult Population

Summary

Open-source U.N. data is used to correlate the mortality rate and certain key health risk factors of the world's adult population. The objective is to conduct a concise analysis of real-world data and show the importance of increasing the per capita healthcare spending to effectively reduce mortality rate. The complex, yet limited, data sets are analyzed using polynomial regression fitting with regularization.

The key findings are listed below:

- Strong correlation is found between country-specific health expenditures per capita below US$2500 and the mortality rates by cardiovascular diseases and diabetes, and mortality rates by chronic respiratory conditions.
- Little-to-no correlation exists between health expenditure per capita and mortality rate by cancer. This implies that even countries with high per capita health expenditures are unable to reduce the mortality by cancer significantly.
- Mortality rate by cardiovascular diseases and diabetes is relatively high in countries with low health expenditure, while it is comparable to the mortality rate by cancer in countries with high health expenditure. This implies that countries with low health expenditure should address cardiovascular diseases and diabetes by moderately increasing their health expenditure to reduce overall mortality rate in their adult population.
- Overall mortality can also be reduced by lowering mortality by chronic respiratory conditions through higher health expenditures, but to a lesser extent.

Introduction

The three major categories of medical conditions, cancer, cardiovascular diseases and diabetes, and chronic respiratory conditions, account for about half of the overall global mortality rate. Closer examination of these conditions and their effects on mortality rate, which is age standardized for the ages 30 – 70 per 100 000 population, is conducted. All data for this analysis is country-specific and extracted from the United Nations (U.N.) online data website for 187 countries [1]. Three key health risk factors are also included in the analysis and provided by the U.N. data set:

- percentage of adults aged above 20 years who are obese,
- percentage of defined population with raised blood pressure (systolic blood pressure >= 140 or diastolic blood pressure >= 90),

- percentage of defined population with raised fasting blood glucose (>= 126 mg/dl) or on medication for raised blood glucose.

The objective for this analysis is to quantify the extent each of these factors has on mortality rate and propose proper resource allocation to control them [2-3]. This is especially important for developing countries with limited financial resources. The per capita total expenditure on health at an average exchange rate (US$) is included in the analysis because it affects mortality rate significantly. As some of the above mentioned datasets are only available for the year 2008, this analysis is limited to that year.

Pre-observations

The age standardized mortality rate by all causes per 100 000 population is plotted for each country as a function of four factors in Figure 1: (1) per capita total expenditure on health in US$, (2) percentage of obese adults, (3) percentage of defined population with raised fasting blood glucose or on medication for raised blood glucose, and (4) percentage of defined population with raised blood pressure.

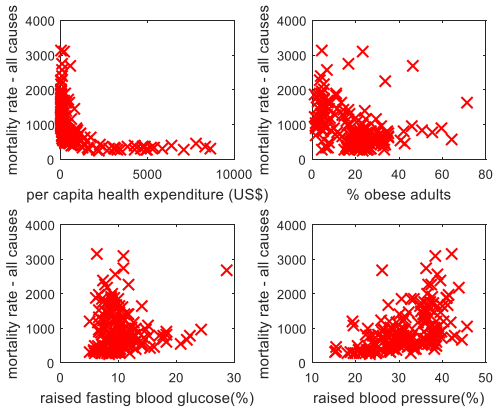

Figure 1: Age standardized mortality rate by all causes plotted, for all countries, against the per capita health expenditure and the three key risk factors.

It is evident from the top left figure of Figure 1 that mortality rate decreases rapidly with an increase in per capita total expenditure on health. Mortality rate is independent of per capita total expenditure on health beyond expenditure of approximately US$2500. Contrary to common perception, there is no readily identifiable trend between mortality rate and obesity, raised fasting blood glucose, and raised blood pressure [2-3]. However, some weak correlation appears to exist between mortality rate and raised blood pressure. As cardiovascular diseases and diabetes are more often linked to these risk factors than for the other conditions, graphs of mortality rate by cardiovascular diseases and diabetes are plotted as a function of these risk factors in Figure 2.

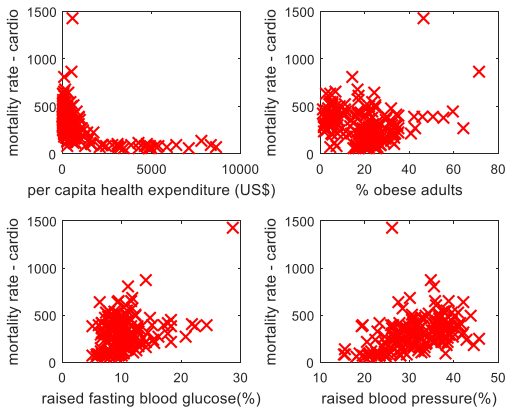

Figure 2: Age standardized mortality rate by cardiovascular diseases and diabetes plotted, for all countries, against the per capita health expenditure and the key risk factors.

Figure 2 indicates that increasing per capita health expenditure is very effective in reducing mortality rate by cardiovascular diseases and diabetes. The plots of mortality rate by cardiovascular diseases and diabetes versus obesity, raised fasting blood glucose, and raised blood pressure show varying, but weaker degrees of correlation. Therefore, an analysis is performed to understand the main causes of mortality rate so that proportional health expenditures can be targeted at these diseases. Figure 3 shows the mortality rate by all causes, cancer, cardiovascular diseases and diabetes, and chronic respiratory conditions as a function of per capita expenditure on health.

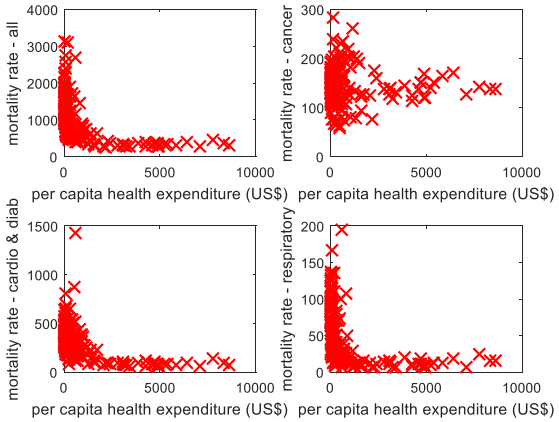

Figure 3: Age standardized mortality rate by all causes and by each of the three major categories of medical conditions (cancer, cardiovascular diseases and diabetes, and chronic respiratory conditions), plotted as a function of per capita total expenditure on health.

There are several key observations in Figure 3. Firstly, mortality rate by cardiovascular diseases and diabetes and by chronic respiratory conditions decrease rapidly by increased expenditure on health, especially when the country's per capita total expenditure on health is less than US$2500. Secondly, mortality rate by cancer shows no strong dependence on per capita total expenditure on health and is practically constant at about 150 per 100,000 population in

countries with high expenditure on health. Finally, the mortality rate by cardiovascular diseases and diabetes in countries with low health expenditure is higher than mortality rate by the other two causes, but it is as low as mortality rate by cancer in countries with high health expenditure. This data strongly suggests that increasing health expenditure to target cardiovascular diseases and diabetes can very effectively reduce the overall mortality rate in a given country.

In order to quantify the reduction in mortality rate, a model is developed to predict 1/(mortality rate) for each condition. In other words, the population's cancer, cardiovascular disease and diabetes, and chronic respiratory conditions survival rates are described as a function of per capita health expenditure. The model is only developed for countries with less than US$2500 per capita total expenditure on health. This limit is set for three reasons: (1) countries with limited financial resources can consider not spending more than US$2500 per capita health expenditure because the marginal return per additional health expenditure is low; (2) only 23 out of 187 countries, or 12 %, spend more than US$2500, which is probably because of economic reasons; and (3) the relatively large statistical variations from country to country for this small number of countries with expenditures of more than US$2500 results in overfitting in this regime.

Figure 4 shows 1/(mortality rate) by cancer, cardiovascular diseases and diabetes, and chronic respiratory conditions as a function of per capita health expenditure up to US$2500. 1/(mortality rate) by chronic respiratory conditions is about an order of magnitude higher than 1/(mortality rate) by the other two causes, which implies that mortality rates by due to cancer and cardiovascular diseases and diabetes are more prevalent due to the inverse relationship. A simple visual analysis of the scatter plots does not yield any obvious trends or differences in trends so polynomial regression models are developed and fitted to the three datasets.

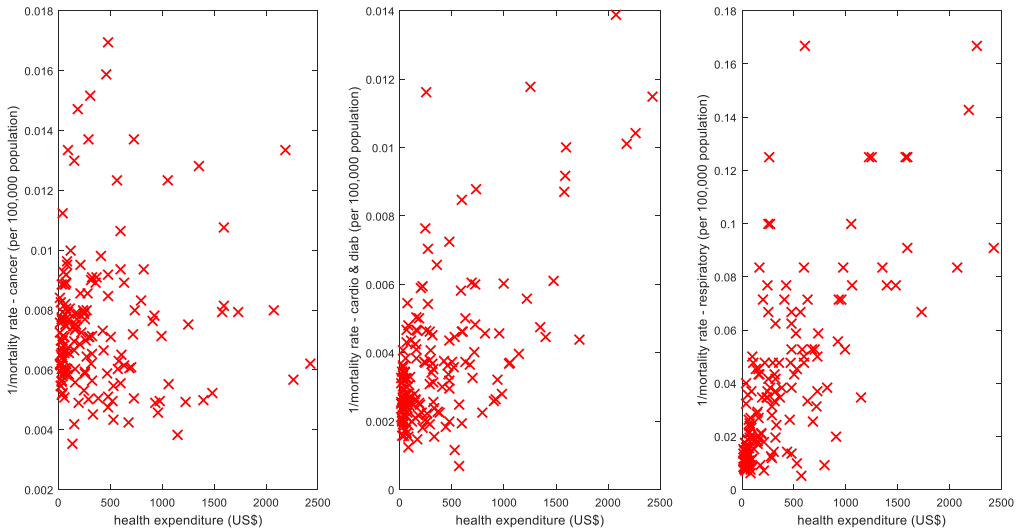

Figure 4: 1/(age standardized mortality rate) by cancer, cardiovascular diseases and diabetes, and chronic respiratory conditions plotted as a function of per capita total expenditure on health.

Polynomial regression model of 1/mortality rate by cancer

Modeling is performed with MATLAB and the code used is provided in the appendix of this report. A 4th-degree polynomial regression fit is performed on the data. This is typically sufficient to describe a relatively complex relationship between the predicted variable and the predictor variable, as seen here. The function describing the predicted 1/mortality rate is as follows:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

$h_\theta(x)$ is the predicted 1/(mortality rate by cancer), θs are the fitted parameters, and x is the per capita total expenditure on health.
In addition, regularization is also applied to prevent overfitting by the high order polynomial fit. The cost function of the model which the algorithm, a more memory-efficient nonlinear programming solver similar to *fminunc* in MATLAB, minimizes is:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

$h_\theta(x^{(i)})$ is the predicted 1/(mortality rate by cancer) for the i^{th} example of per capita total expenditure on health, $y^{(i)}$ is the i^{th} example of observed 1/(mortality rate by cancer), m is the number of observed examples (i.e. number of countries), and λ is the regularization parameter. The dataset is divided into a training dataset with 100 countries, a cross-validation dataset with 32 countries, and a test dataset with 32 countries. Models with different values of the regularization parameter, λ, are obtained by fitting the training dataset. A good fit minimizes the cost function which minimizes the error between the predicted and the observed 1/(mortality rate). λ is the regularization parameter which prevents overfitting by limiting the magnitude of the fitted parameters: θ_j where j = 1, 2, 3 or 4. The model with the lowest error (i.e. cost function without the regularization term) calculated from the cross-validation dataset is then selected from the models with different λ to determine a suitable λ.

Figure 5 shows both the training and cross-validation errors for the models with different values of λ (0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10). The cross validation error is practically constant at the various values of λ. Figure 6 shows polynomial regression fits with two extreme values of λ, zero and ten, using the data from all the countries with less than US$2500 per capita expenditure on health.

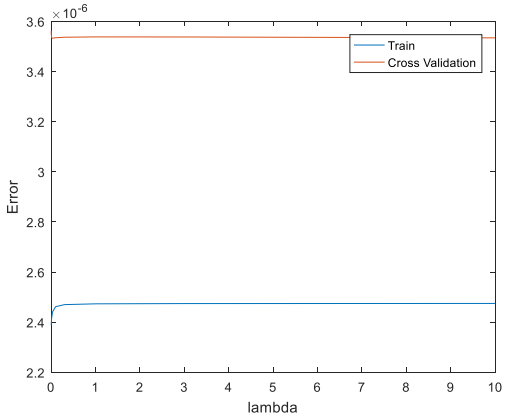

Figure 5: Errors of models with different values of regularization parameter, λ, calculated from training dataset (blue line) and cross validation dataset (red line) for 1/(mortality rate by cancer).

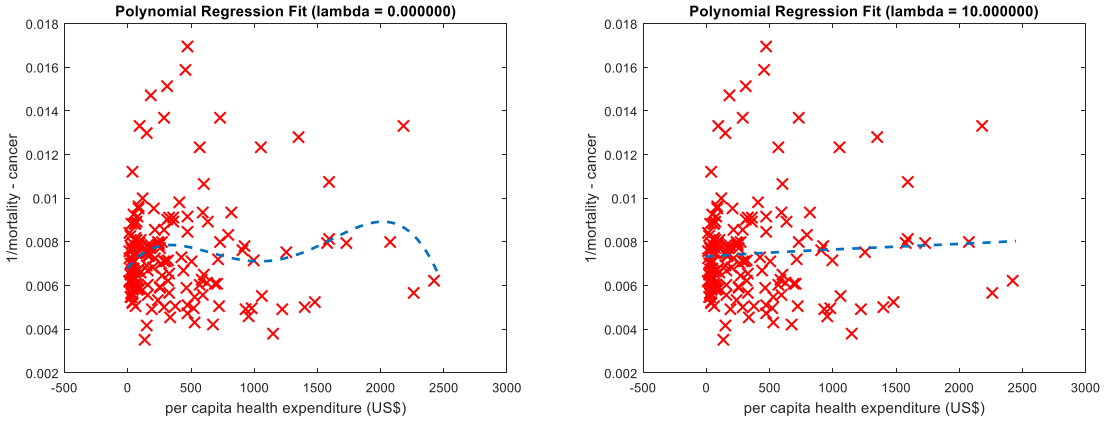

Figure 6: Polynomial regression fits (blue dotted line) for 1/(mortality rate by cancer) as a function of per capita expenditure on health with regularization parameter, λ = 0 (left) and λ = 10 (right).

There is high variance in the regression fit with λ = 0 at higher values of health expenditure due to the small number of countries available in this regime. Additional data is not available, which limits the predictiveness of the model. The regression fit with a non-zero regularization parameter prevents overfitting. The fitting parameters, θ_j, with j = 0, 1, 2, 3 and 4, are 0.0075, $1.9*10^{-4}$, $-8.1*10^{-5}$, $3.0*10^{-5}$, and $5.1*10^{-6}$, respectively, when λ = 10. The non-zero λ fit implies that the 1/(mortality rate by cancer) is practically independent of per capita expenditure on health.

Polynomial regression model of 1/mortality rate by cardiovascular diseases and diabetes

A similar analysis is performed to model the 1/(mortality rate by cardiovascular diseases and diabetes) as a function of per capita expenditure on health. Figure 7 shows both the training and cross-validation errors for the models with different values of λ. Figure 7 shows that the cross-validation error is lowest at $\lambda = 0$. Hence, polynomial regression fit is performed at $\lambda = 0$ using all the countries with less than US$2500 per capita expenditure on health, as shown in Figure 8.

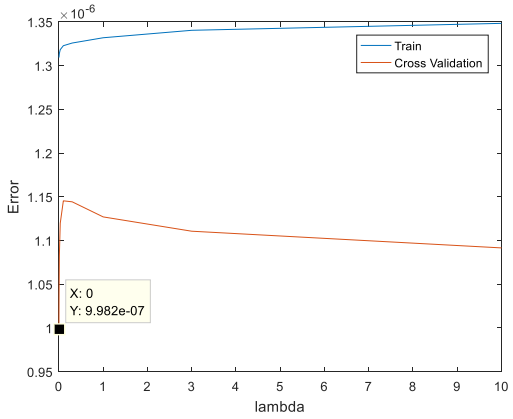

Figure 7: Errors of models with different values of regularization parameter, λ, calculated from training dataset (blue line) and cross validation dataset (red line) for 1/(mortality rate by cardiovascular disease).

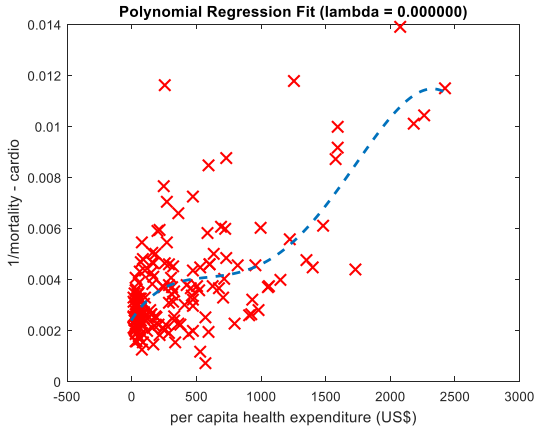

Figure 8: Polynomial regression fit (blue dotted line) for 1/(mortality rate by cardiovascular disease) as a function of per capita expenditure on health with regularization parameter, $\lambda = 0$.

The fitting parameters in Figure 8, θ_j, with j = 0, 1, 2, 3 and 4 are 0.0038, 0.0039, -0.0141, 0.0231 and -0.0113, respectively. The regression fit shows that there is an increase in 1/(mortality rate) with health expenditure. Distribution of the data causes the fit to be non-linear, which may need to be investigated in greater detail. Above ~ US$2000, the 1/(mortality rate) decreases slightly with increasing health expenditure. This may be an artifact from the small number of countries with larger health expenditure. Alternatively, it may indicate that higher expenditures are less effective in combating cardiovascular diseases and diabetes.

Polynomial regression model of 1/mortality rate by chronic respiratory condition

1/(mortality rate by chronic respiratory conditions) as a function of per capita expenditure on health is also modeled. Figure 9 shows the training and cross-validation errors for the models with different values of λ. As the cross-validation error is lowest at λ = 0.3, polynomial regression fit is performed at this λ value using all the countries with less than US$2500 per capita expenditure on health (Figure 10).

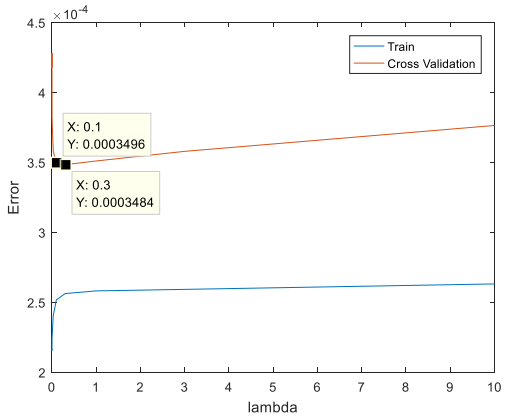

Figure 9: Errors of models with different values of regularization parameter, λ, calculated from training dataset (blue line) and cross validation dataset (red line) for 1/(mortality rate by chronic respiratory condition).

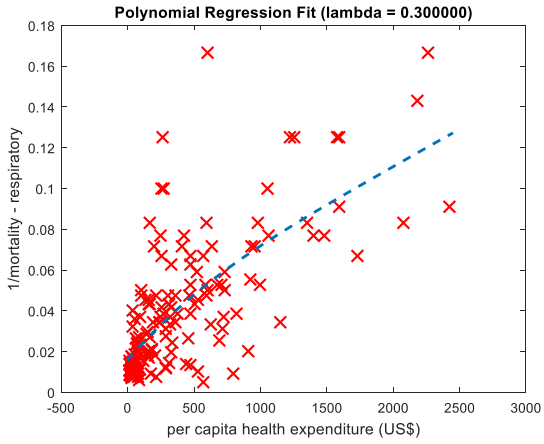

Figure 10: Polynomial regression fit (blue dotted line) for 1/(mortality rate by chronic respiratory conditions) as a function of per capita expenditure on health with regularization parameter, λ = 0.3.

The fitting parameters in Figure 10, θ_j, with j = 0, 1, 2, 3 and 4 are 0.0388, 0.0362, -0.0246, 0.0165 and -0.0045, respectively. The fit indicates that increasing spending in countries

with low health expenditure can also be very instrumental to increase the survival rates specific to chronic respiratory conditions, and thus lowering the overall mortality rate.

Conclusion

The analysis provided here suggests that increasing health care spending very effectively reduces the mortality rate by various medical conditions within a specific country. The benefits of gradually increasing health care spending per capita is supported by the data, up to US$2500. Spending more than this appears to cause, if at all, only a minor reduction in mortality rates. The data however are limited in this regime. The reduction in mortality is primarily linked to a reduction in mortality rates by cardiovascular diseases and diabetes, and to a lesser degree to a reduction in mortality rates by chronic respiratory conditions. Interestingly, increased health expenditure appears not to impact mortality rate by cancer. This analysis can serve as a guide for governments from countries with limited financial resources to allocate their resources more wisely in an effort to reduce the mortality rate. Countries with low health expenditure budgets should perhaps focus first on combating cardiovascular diseases and diabetes, and then on chronic respiratory conditions. Somewhat surprising is the fact that no correlation exists between the health expenditure and cancer mortality. This warrants further investigation because cancer is, along with cardiovascular diseases and diabetes, one of the leading causes of deaths globally. The absence of any correlation could imply that current budget allocations do not effectively fight the root-causes of cancer, or that finding a cure is inherently very difficult and pay-offs from increased spending may not be immediately quantifiable.

References

[1] http://data.un.org/Explorer.aspx?d=WHO
[2] https://www.nhlbi.nih.gov/health/health-topics/topics/obe/risks
[3] http://news.heart.org/high-blood-pressure-causing-deaths-despite-drop-heart-disease-stroke-deaths/

Appendix

The code used below is a modified version of code from the Machine Learning Coursera class by Prof. Andrew Ng (Stanford University).

1. Main MATLAB file (main.m)

```
% initialization
clear ; close all ; clc
```

```
% load UN data
data = xlsread('mortality.xlsx');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Plot data for the features: expenditure, obesity, high resting
% glucose level and high blood pressure

X = data(:,5:8); % 4 features that are suspected to influence y
y = data(:,3); % mortality rate of 187 countries

figure;
subplot(2,2,1);
plot(X(:,1), y, 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('per capita health expenditure (US$)');
ylabel('mortality rate - cardio');

subplot(2,2,2);
plot(X(:,2), y, 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('% obese adults');
ylabel('mortality rate - cardio');

subplot(2,2,3);
plot(X(:,3), y, 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('raised fasting blood glucose(%)');
ylabel('mortality rate - cardio');

subplot(2,2,4);
plot(X(:,4), y, 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('raised blood pressure(%)');
ylabel('mortality rate - cardio');

fprintf('Program paused. Press enter to continue.\n');
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

% Plot data for the specific mortality causes vs expenditure: all, cancer,
% cardiovascular & diabetes, respiratory

X = data(:,5); % per capita expenditure on health
y = data(:,1:4); % mortality rate due to different causes

figure;
subplot(2,2,1);
plot(X, y(:,1), 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('per capita health expenditure (US$)');
ylabel('mortality rate - all');

subplot(2,2,2);
plot(X, y(:,2), 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('per capita health expenditure (US$)');
ylabel('mortality rate - cancer');
```

```
subplot(2,2,3);
plot(X, y(:,3), 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('per capita health expenditure (US$)');
ylabel('mortality rate - cardio & diab');

subplot(2,2,4);
plot(X, y(:,4), 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('per capita health expenditure (US$)');
ylabel('mortality rate - respiratory');

fprintf('Program paused. Press enter to continue.\n');
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Examine only the countries with health expenditure < 2500
I = find(data(:,5) < 2500); % column matrix of indices
m = size(I); % number of countries with < $2500 health expenditure ~ 164

% Plot data for the 1/mortality rate (for cancer, cardiovascular disease,
respiratory condition)
% vs expenditure on health

X1 = data(I,5); % per capita expenditure on health
y1 = 1./data(I,2:4);
% col1: 1/mortality due to cancer
% col2: 1/mortality due to cardiovascular diseases and diabetes
% col3: 1/mortality due to respiratory condition

figure;
subplot(1,3,1);
plot(X1, y1(:,1), 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('health expenditure (US$)');
ylabel('1/mortality rate - cancer (per 100,000 population)');

subplot(1,3,2);
plot(X1, y1(:,2), 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('health expenditure (US$)');
ylabel('1/mortality rate - cardio & diab (per 100,000 population)');

subplot(1,3,3);
plot(X1, y1(:,3), 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('health expenditure (US$)');
ylabel('1/mortality rate - respiratory (per 100,000 population)');

fprintf('Program paused. Press enter to continue.\n');
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% order 1 to 164 randomly without repeat in a vector
```

```
order = randperm(164); % random selection of examples to be in training or
cross validation or test set
X = X1(order(1:100),1); % 100 eg for training set
y = y1(order(1:100),1:3);
Xval = X1(order(101:132),1); % 32 eg for cross validation set
yval = y1(order(101:132),1:3);
Xtest = X1(order(133:164),1); % 32 eg for test set
ytest = y1(order(133:164),1:3);
m = size(X,1); % m = no. of training egs


%% =========== Feature Mapping for Polynomial Regression =============
%  Polynomial regression. Use polyFeatures.m to map each example into its
powers

p = 4; % 4th order of power

% Map X onto Polynomial Features and Normalize
X_poly = polyFeatures(X, p);
[X_poly, mu, sigma] = featureNormalize(X_poly);  % Normalize
X_poly = [ones(m, 1), X_poly];                   % Add Ones

% Map X_poly_test and normalize (using mu and sigma)
X_poly_test = polyFeatures(Xtest, p);
X_poly_test = bsxfun(@minus, X_poly_test, mu);
X_poly_test = bsxfun(@rdivide, X_poly_test, sigma);
X_poly_test = [ones(size(X_poly_test, 1), 1), X_poly_test];         % Add
Ones

% Map X_poly_val and normalize (using mu and sigma)
X_poly_val = polyFeatures(Xval, p);
X_poly_val = bsxfun(@minus, X_poly_val, mu);
X_poly_val = bsxfun(@rdivide, X_poly_val, sigma);
X_poly_val = [ones(size(X_poly_val, 1), 1), X_poly_val];           % Add Ones


%% =========== Polynomial Regression =============

lambda = 0.3;
[theta] = trainLinearReg(X_poly, y(:,3), lambda);

% Plot training data and fit
figure;
plot(X, y(:,3), 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
plotFit(min(X), max(X), mu, sigma, theta, p);
xlabel('per capita health expenditure (US$)');
ylabel('1/mortality - respiratory');
title (sprintf('Polynomial Regression Fit (lambda = %f)', lambda));
xlim([-500 3000]);

fprintf('Program paused. Press enter to continue.\n');
pause;

%% =========== Validation for Selecting Lambda =============
```

```
%  Implement validationCurve to test various values of
%  lambda on a validation set. Then use this to select the
%  "best" lambda value.
%

[lambda_vec, error_train, error_val] = ...
    validationCurve(X_poly, y(:,3), X_poly_val, yval(:,3));

figure;
plot(lambda_vec, error_train, lambda_vec, error_val);
legend('Train', 'Cross Validation');
xlabel('lambda');
ylabel('Error');

fprintf('lambda\t\tTrain Error\tValidation Error\n');
for i = 1:length(lambda_vec)
    fprintf(' %f\t%f\t%f\n', ...
            lambda_vec(i), error_train(i), error_val(i));
end

fprintf('Program paused. Press enter to continue.\n');
pause;
```

2. The function, polyFeatures.m, maps predictor variable of each example into its polynomial features

```
function [X_poly] = polyFeatures(X, p)
%POLYFEATURES Maps X (1D vector) into the p-th power
%   [X_poly] = POLYFEATURES(X, p) takes a data matrix X (size m x 1) and
%   maps each example into its polynomial features where
%   X_poly(i, :) = [X(i) X(i).^2 X(i).^3 ...  X(i).^p];
%

X_poly = zeros(numel(X), p);
size(X_poly)

% Given a vector X, return a matrix X_poly where the p-th column of X
% contains the values of X to the p-th power.

for i = 1:p
    X_poly(:,i) = X(:,1).^i;
end

end
```

3. The function, featureNormalize.m, normalizes each predictor variable so that the mean is zero and standard deviation is one.

```
function [X_norm, mu, sigma] = featureNormalize(X)
%FEATURENORMALIZE Normalizes the features in X
%   FEATURENORMALIZE(X) returns a normalized version of X where
%   the mean value of each feature is 0 and the standard deviation
%   is 1. This is often a good preprocessing step to do when
```

```
%   working with learning algorithms.

mu = mean(X);
X_norm = bsxfun(@minus, X, mu);

sigma = std(X_norm);
X_norm = bsxfun(@rdivide, X_norm, sigma);

end
```

4. The function, trainLinearReg.m, trains linear regression using the dataset of predictor variables, dependent variables and regularization parameter.

```
function [theta] = trainLinearReg(X, y, lambda)
%TRAINLINEARREG Trains linear regression given a dataset (X, y) and a
%regularization parameter lambda
%   [theta] = TRAINLINEARREG (X, y, lambda) trains linear regression using
%   the dataset (X, y) and regularization parameter lambda. Returns the
%   trained parameters theta.
%

% Initialize Theta
initial_theta = zeros(size(X, 2), 1);

% Create "short hand" for the cost function to be minimized
costFunction = @(t) linearRegCostFunction(X, y, t, lambda);

% Now, costFunction is a function that takes in only one argument
options = optimset('MaxIter', 200, 'GradObj', 'on');

% Minimize using fmincg
theta = fmincg(costFunction, initial_theta, options);

end
```

5. The function, fmincg.m, minimizes a continuous differentiable multivariate function. It is used to minimize the cost function of the model.

```
function [X, fX, i] = fmincg(f, X, options, P1, P2, P3, P4, P5)
% Minimize a continuous differentialble multivariate function. Starting point
% is given by "X" (D by 1), and the function named in the string "f", must
% return a function value and a vector of partial derivatives. The Polack-
% Ribiere flavour of conjugate gradients is used to compute search
directions, and a line search using quadratic and cubic polynomial
%approximations and the Wolfe-Powell stopping criteria is used together with
%the slope ratio method for guessing initial step sizes. Additionally a bunch
%of checks are made to make sure that exploration is taking place and that
%extrapolation will not be unboundedly large. The "length" gives the length
%of the run: if it is positive, it gives the maximum number of line searches,
%if negative its
% absolute gives the maximum allowed number of function evaluations. You can
% (optionally) give "length" a second component, which will indicate the
```

```
% reduction in function value to be expected in the first line-search
%(defaults to 1.0). The function returns when either its length is up, or if
% no further progress can be made (ie, we are at a minimum, or so close that
%due to numerical problems, we cannot get any closer). If the function
%terminates within a few iterations, it could be an indication that the
%function value and derivatives are not consistent (ie, there may be a bug in
%the implementation of your "f" function). The function returns the found
% solution "X", a vector of function values "fX" indicating the progress made
% and "i" the number of iterations (line searches or function evaluations,
% depending on the sign of "length") used.
%
% Usage: [X, fX, i] = fmincg(f, X, options, P1, P2, P3, P4, P5)
%
% See also: checkgrad
%
% Copyright (C) 2001 and 2002 by Carl Edward Rasmussen. Date 2002-02-13
%
%
% (C) Copyright 1999, 2000 & 2001, Carl Edward Rasmussen
%
% Permission is granted for anyone to copy, use, or modify these
% programs and accompanying documents for purposes of research or
% education, provided this copyright notice is retained, and note is
% made of any changes that have been made.
%
% These programs and documents are distributed without any warranty,
% express or implied.  As the programs were written for research
% purposes only, they have not been tested to the degree that would be
% advisable in any important application.  All use of these programs is
% entirely at the user's own risk.
%
% [ml-class] Changes Made:
% 1) Function name and argument specifications
% 2) Output display
%

% Read options
if exist('options', 'var') && ~isempty(options) && isfield(options,
'MaxIter')
    length = options.MaxIter;
else
    length = 100;
end


RHO = 0.01;                            % a bunch of constants for line
searches
SIG = 0.5;       % RHO and SIG are the constants in the Wolfe-Powell
conditions
INT = 0.1;    % don't reevaluate within 0.1 of the limit of the current
bracket
EXT = 3.0;                    % extrapolate maximum 3 times the current
bracket
MAX = 20;                         % max 20 function evaluations per line
search
```

```
RATIO = 100;                                          % maximum allowed slope
ratio

argstr = ['feval(f, X'];                            % compose string used to call
function
for i = 1:(nargin - 3)
  argstr = [argstr, ',P', int2str(i)];
end
argstr = [argstr, ')'];

if max(size(length)) == 2, red=length(2); length=length(1); else red=1; end
S=['Iteration '];

i = 0;                                                % zero the run length
counter
ls_failed = 0;                                  % no previous line search has
failed
fX = [];
[f1 df1] = eval(argstr);                            % get function value and
gradient
i = i + (length<0);                                                % count
epochs?!
s = -df1;                                           % search direction is
steepest
d1 = -s'*s;                                                     % this is the
slope
z1 = red/(1-d1);                                     % initial step is
red/(|s|+1)

while i < abs(length)                                             % while not
finished
  i = i + (length>0);                                             % count
iterations?!

  X0 = X; f0 = f1; df0 = df1;                       % make a copy of current
values
  X = X + z1*s;                                                  % begin line
search
  [f2 df2] = eval(argstr);
  i = i + (length<0);                                               % count
epochs?!
  d2 = df2'*s;
  f3 = f1; d3 = d1; z3 = -z1;                % initialize point 3 equal to point
1
  if length>0, M = MAX; else M = min(MAX, -length-i); end
  success = 0; limit = -1;                         % initialize quanteties
  while 1
    while ((f2 > f1+z1*RHO*d1) || (d2 > -SIG*d1)) && (M > 0)
      limit = z1;                                               % tighten the
bracket
      if f2 > f1
        z2 = z3 - (0.5*d3*z3*z3)/(d3*z3+f2-f3);                 % quadratic
fit
      else
        A = 6*(f2-f3)/z3+3*(d2+d3);                                % cubic
fit
```

```
        B = 3*(f3-f2)-z3*(d3+2*d2);
        z2 = (sqrt(B*B-A*d2*z3*z3)-B)/A;       % numerical error possible -
ok!
      end
      if isnan(z2) || isinf(z2)
        z2 = z3/2;                  % if we had a numerical problem then
bisect
      end
      z2 = max(min(z2, INT*z3),(1-INT)*z3);  % don't accept too close to
limits
      z1 = z1 + z2;                                             % update the
step
      X = X + z2*s;
      [f2 df2] = eval(argstr);
      M = M - 1; i = i + (length<0);                            % count
epochs?!
      d2 = df2'*s;
      z3 = z3-z2;                    % z3 is now relative to the location of
z2
    end
    if f2 > f1+z1*RHO*d1 || d2 > -SIG*d1
      break;                                                % this is a
failure
    elseif d2 > SIG*d1
      success = 1; break;                                             %
success
    elseif M == 0
      break;                                                          %
failure
    end
    A = 6*(f2-f3)/z3+3*(d2+d3);                          % make cubic
extrapolation
    B = 3*(f3-f2)-z3*(d3+2*d2);
    z2 = -d2*z3*z3/(B+sqrt(B*B-A*d2*z3*z3));        % num. error possible -
ok!
    if ~isreal(z2) || isnan(z2) || isinf(z2) || z2 < 0 % num prob or wrong
sign?
      if limit < -0.5                                 % if we have no upper
limit
        z2 = z1 * (EXT-1);                 % the extrapolate the maximum
amount
      else
        z2 = (limit-z1)/2;                                    % otherwise
bisect
      end
    elseif (limit > -0.5) && (z2+z1 > limit)          % extraplation beyond
max?
      z2 = (limit-z1)/2;                                                %
bisect
    elseif (limit < -0.5) && (z2+z1 > z1*EXT)       % extrapolation beyond
limit
      z2 = z1*(EXT-1.0);                           % set to extrapolation
limit
    elseif z2 < -z3*INT
      z2 = -z3*INT;
    elseif (limit > -0.5) && (z2 < (limit-z1)*(1.0-INT))  % too close to
limit?
```

```
      z2 = (limit-z1)*(1.0-INT);
    end
    f3 = f2; d3 = d2; z3 = -z2;                    % set point 3 equal to point
2
    z1 = z1 + z2; X = X + z2*s;                      % update current
estimates
    [f2 df2] = eval(argstr);
    M = M - 1; i = i + (length<0);                                % count
epochs?!
    d2 = df2'*s;
  end                                                          % end of line
search

  if success                                                   % if line search
succeeded
    f1 = f2; fX = [fX' f1]';
    fprintf('%s %4i | Cost: %4.6e\r', S, i, f1);
    s = (df2'*df2-df1'*df2)/(df1'*df1)*s - df2;      % Polack-Ribiere
direction
    tmp = df1; df1 = df2; df2 = tmp;                           % swap
derivatives
    d2 = df1'*s;
    if d2 > 0                                          % new slope must be
negative
      s = -df1;                                  % otherwise use steepest
direction
      d2 = -s'*s;
    end
    z1 = z1 * min(RATIO, d1/(d2-realmin));          % slope ratio but max
RATIO
    d1 = d2;
    ls_failed = 0;                                 % this line search did not
fail
  else
    X = X0; f1 = f0; df1 = df0;  % restore point from before failed line
search
    if ls_failed || i > abs(length)          % line search failed twice in a
row
      break;                                 % or we ran out of time, so we give
up
    end
    tmp = df1; df1 = df2; df2 = tmp;                           % swap
derivatives
    s = -df1;                                                      % try
steepest
    d1 = -s'*s;
    z1 = 1/(1-d1);
    ls_failed = 1;                                      % this line search
failed
  end
  if exist('OCTAVE_VERSION')
    fflush(stdout);
  end
end
fprintf('\n');
```

6. The function, linearRegCostFunction.m, computes the cost function and its gradient. It is used in fmincg.m.

```
function [J, grad] = linearRegCostFunction(X, y, theta, lambda)
%LINEARREGCOSTFUNCTION Compute cost and gradient for regularized linear
%regression with multiple variables
%   [J, grad] = LINEARREGCOSTFUNCTION(X, y, theta, lambda) computes the
%   cost of using theta as the parameter for linear regression to fit the
%   data points in X and y. Returns the cost in J and the gradient in grad


m = length(y); % number of training examples

J = 0;
grad = zeros(size(theta));

PredY = X * theta;
J = sum((PredY - y).^2) / 2 / m + lambda / 2 / m * sum((theta(2:end)).^2);

grad(1) = sum((PredY - y).* X(:,1)) / m;
for i = 2:size(grad)
    grad(i) = sum((PredY - y).* X(:,i)) / m + lambda / m * theta(i);
end

% =========================================================================

grad = grad(:);

end
```

7. The function, plotFit.m, plots a polynomial regression fit over the existing figure of the dataset.

```
function plotFit(min_x, max_x, mu, sigma, theta, p)
%PLOTFIT Plots a learned polynomial regression fit over an existing figure.
%Also works with linear regression.
%   PLOTFIT(min_x, max_x, mu, sigma, theta, p) plots the learned polynomial
%   fit with power p and feature normalization (mu, sigma).

% Hold on to the current figure
hold on;

% We plot a range slightly bigger than the min and max values to get
% an idea of how the fit will vary outside the range of the data points
x = (min_x - 15: 0.05 : max_x + 25)';

% Map the X values
X_poly = polyFeatures(x, p);
X_poly = bsxfun(@minus, X_poly, mu);
X_poly = bsxfun(@rdivide, X_poly, sigma);

% Add ones
X_poly = [ones(size(x, 1), 1) X_poly];
```

```
% Plot
plot(x, X_poly * theta, '--', 'LineWidth', 2)

% Hold off to the current figure
hold off

end
```

8. The function, validationCurve.m, plots the training and cross validation errors that is used to select a suitable value of regularization parameter.

```
function [lambda_vec, error_train, error_val] = ...
    validationCurve(X, y, Xval, yval)
%VALIDATIONCURVE Generate the train and validation errors needed to
%plot a validation curve that we can use to select lambda
%   [lambda_vec, error_train, error_val] = ...
%       VALIDATIONCURVE(X, y, Xval, yval) returns the train
%       and validation errors (in error_train, error_val)
%       for different values of lambda. You are given the training set (X,
%       y) and validation set (Xval, yval).
%

% Selected values of lambda
lambda_vec = [0 0.001 0.003 0.01 0.03 0.1 0.3 1 3 10]';

error_train = zeros(length(lambda_vec), 1);
error_val = zeros(length(lambda_vec), 1);

for i = 1:length(lambda_vec)
    lambda = lambda_vec(i);
    [theta] = trainLinearReg(X, y, lambda);

    PredY = X * theta;
    error_train(i) = sum((PredY - y).^2) / 2 / size(X,1) ;

    PredYval = Xval * theta;
    error_val(i) = sum((PredYval - yval).^2) / 2 / size(Xval, 1) ;

end

end
```