Lecture 4

Linear Regression as Predictor

Logistic Regression as Classifier

Suthep "Jogie" Madarasmi, Ph.D.

Applications of Gradient Descent

1. Linear Regression
2. Logistic Regression
3. Regularized Linear and Logistic Regression
4. Neural Networks Learning. Multi-layer Perceptrons.

Linear Regression in ML

- Use Linear models to fit features **x** to predict output value y.
- Use the fitting to predict future values of y using **x**.
- Example: Software Cost based on various parameters such as number of input screens, number of reports, number of users, …

Logistic Regression (LR): Class A or Class B?

Based on linear regression, but converted to a Yes/No decision for classification by the sigmoid function. The linear function's output is converted into a probability distribution y ∈ [0, 1]

$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}; \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

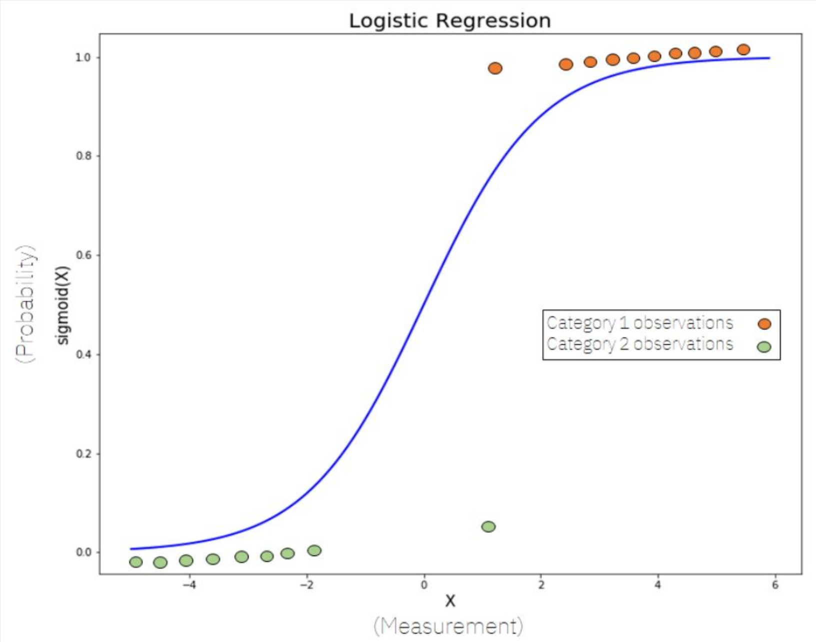
For Linear Regression:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \text{ ; where } x_0 = 1$$

$$= \sum_{i=0}^{n} \theta_i x_i = \theta^T x \text{ ; for linear regression}$$

For Logistic Regression:

$$h_\theta(x) = g\left(\theta^T x\right) \text{ ; } g(z) = \frac{1}{1+e^{-z}} \text{ ; } g \text{ is the sigmoid function.}$$



Linear Regression Fit Metric: R^2

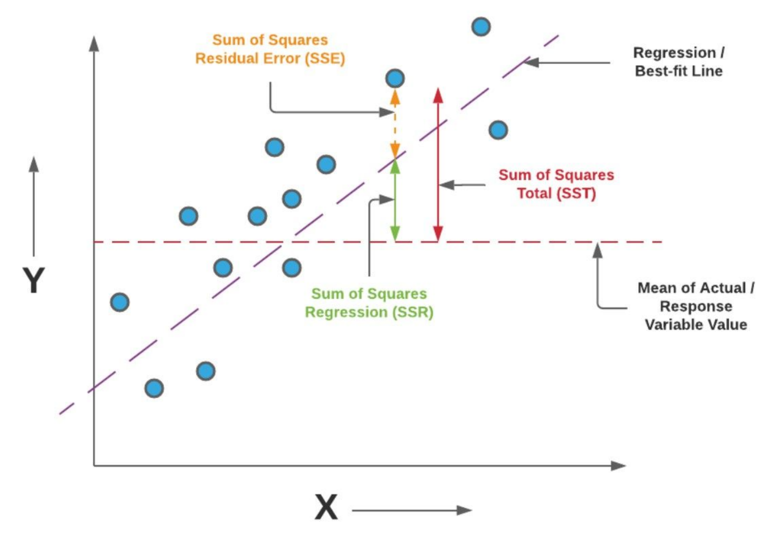
R-squared gives you the percentage variation in y explained by x-variables in a linear model:

$$y = f(x_1, x_2, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

$$R^2 = 1 - \frac{MSE}{Variance}$$

- The range is 0 to 1
- 0% to 100% of the variation in y can be explained by the x-variables.
- Higher the value $\Rightarrow$ better the linear fit and predictor.
- A unit-invariant metric that scales the MSE by the variance of y
- R2 can be negative when the chosen model does not follow the trend of the data, so fits worse than a horizontal line; ie., the average.

R^2



$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

$$= n \cdot Error_{av}$$

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2$$

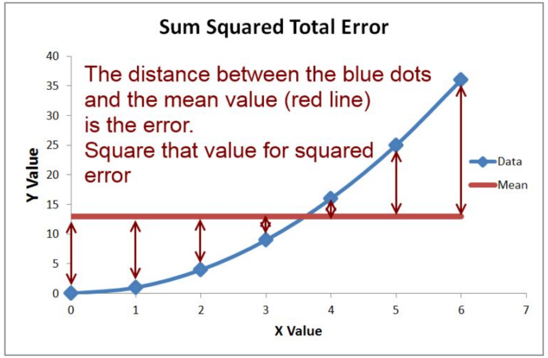
$$= n \cdot \sigma^2$$

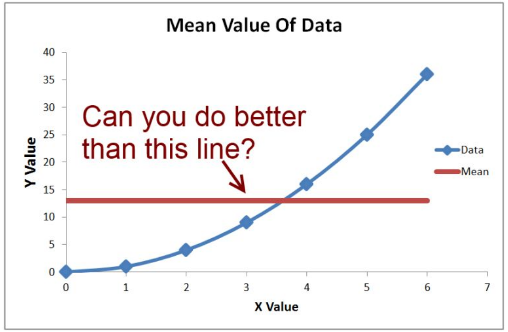
$$\sigma^2 = \sum_{i=1}^{n} \frac{(y_i - \bar{y})^2}{n}$$

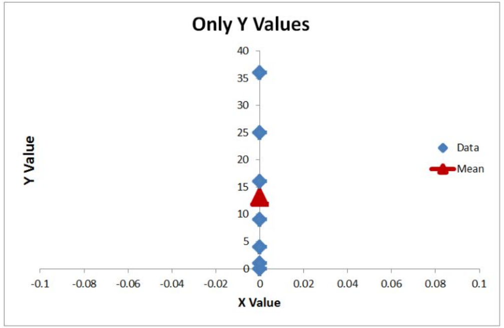
$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

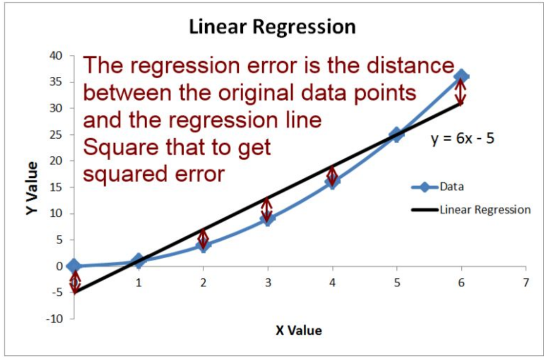
R^2: The Coefficient of Determinant

http://www.fairlynerdy.com/what-is-r-squared/









$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}}$$

Sum Over All The Data Points

Square The Result

$$SS_{Total} = \sum (y_i - \bar{y})^2$$

Sum Squared Total Error

Each Data Point

Mean Value

Sum Over All The Data Points

Square The Result

$$SS_{Regression} = \sum (y_i - y_{Regression})^2$$

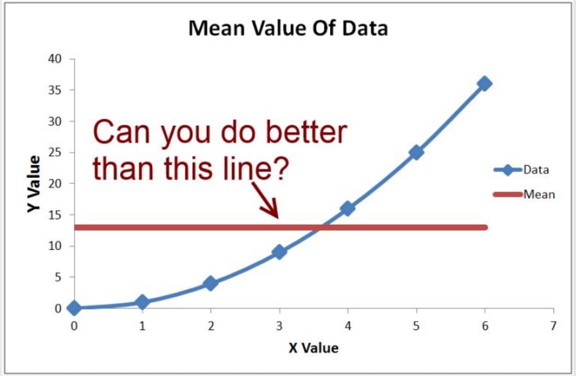
Sum Squared Regression Error

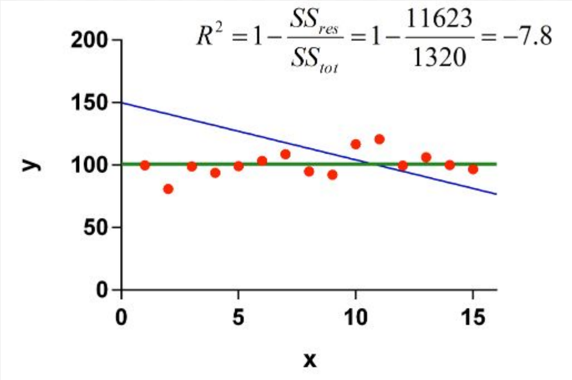
Each Data Point

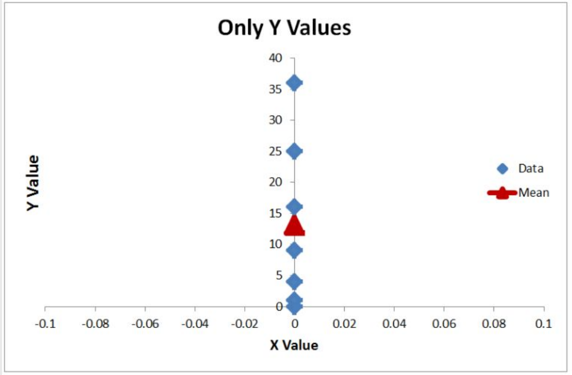
Regression Value

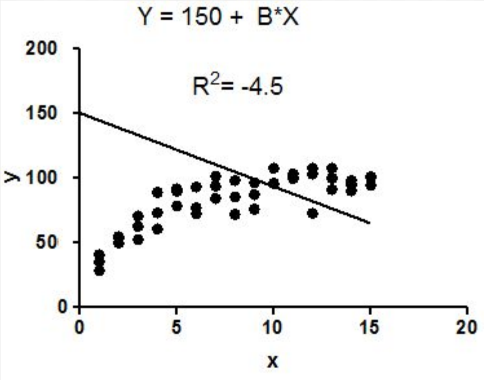
R^2 can be negative when the fit is poor

R2 is negative when regression line is worse than the average line.

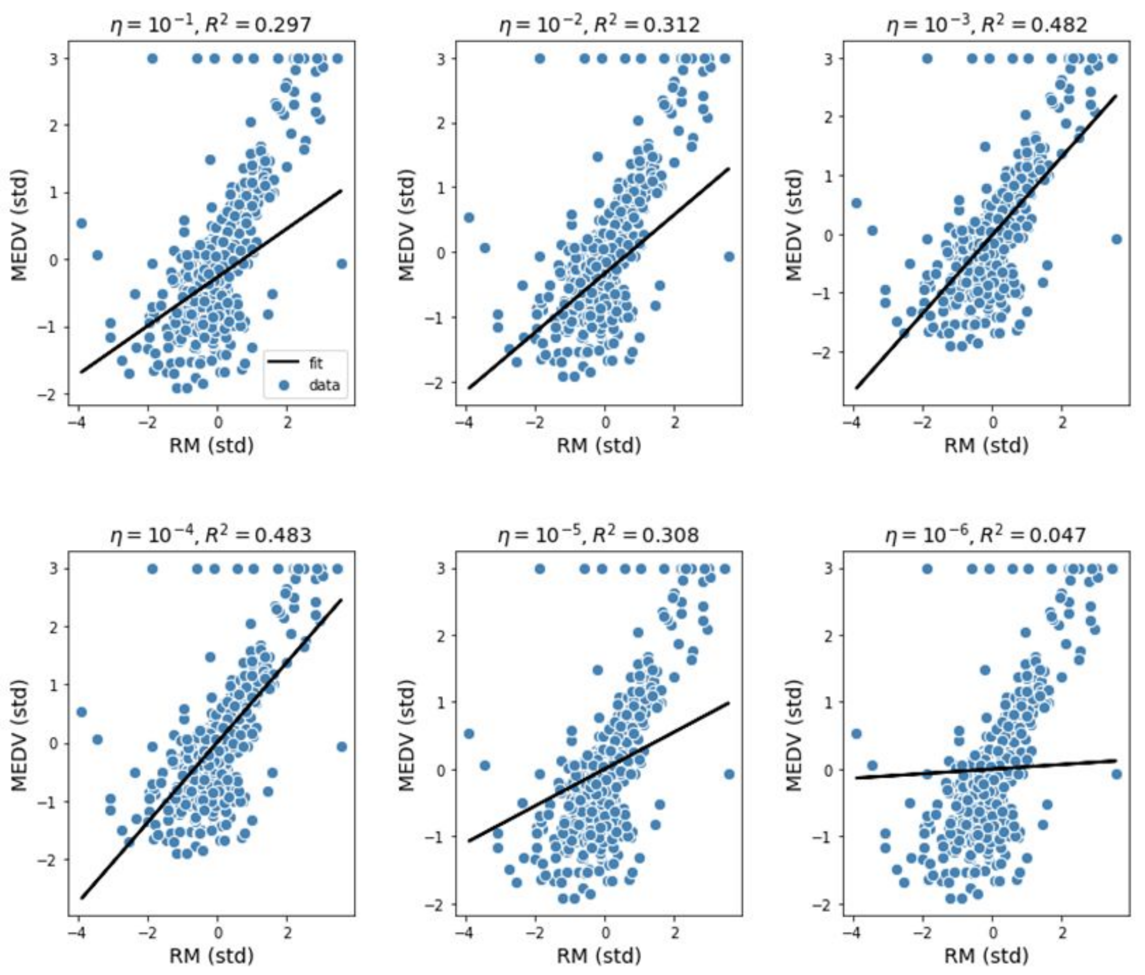




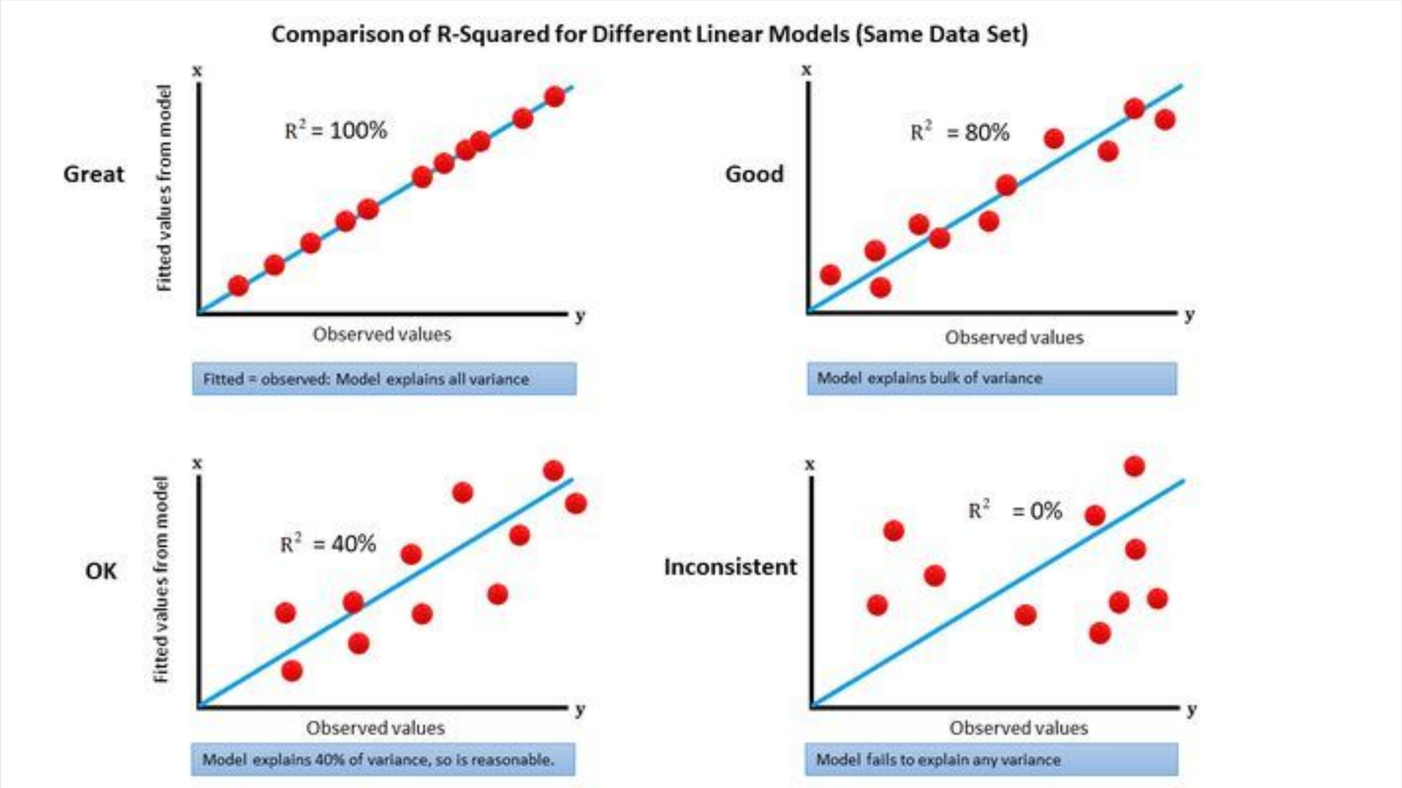




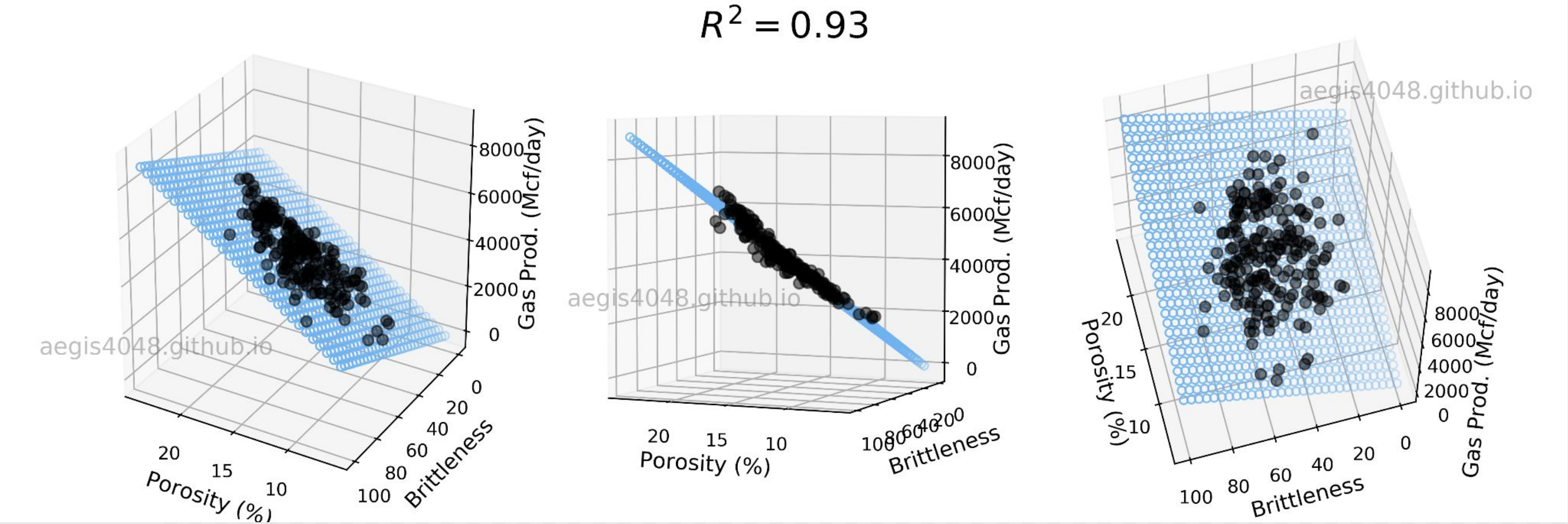
Good Fit R^2 :
0.4 to 1.



For a good fit, R^2 ranges 0.4 to 1.



Regression in 3D. $y = f(x_1, x_2)$ is the best fit plane



$y = f(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2;$

y - daily gasoline production in a well; x1 - well porosity %; x2 - well brittleness

Example call to Regression Library

Linear Regression Curve Fitting.ipynb

File Edit View Insert Runtime Tools Help Last saved at 10:19 PM

+ Code + Text

```
[ ] import numpy as np
    import matplotlib as plt
    import math
    import scipy
```

```
from sklearn.linear_model import LinearRegression
correct_ab = [1, 2]
```

Scikit Dataset: Diabetes

The Diabetes dataset has 10 features $\mathbf{x}$: $(x_1, x_2, \ldots, x_{10})$ with 442 samples.

The output y in:

$$y = f(\mathbf{x}) = f(x_1, x_2, \ldots, x_{10})$$

is the quantitative measure of disease progression.

AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
59	2	32.1	101	157	93.2	38	4	4.8598	87	151
48	1	21.6	87	183	103.2	70	3	3.8918	69	75
72	2	30.5	93	156	93.6	41	4	4.6728	85	141
24	1	25.3	84	198	131.4	40	5	4.8903	89	206
50	1	23	101	192	125.4	52	4	4.2905	80	135
23	1	22.6	89	139	64.8	61	2	4.1897	68	97
36	2	22	90	160	99.6	50	3	3.9512	82	138
66	2	26.2	114	255	185	56	4.55	4.2485	92	63
60	2	32.1	83	179	119.4	42	4	4.4773	94	110
29	1	30	85	180	93.4	43	4	5.3845	88	310

Diabetes Dataset Features 1..10

1. age — age in years
2. sex — gender
3. bmi — body mass index
4. bp — average blood pressure
5. s1 — tc, total serum cholesterol
6. s2 — ldl, low-density lipoproteins
7. s3 — hdl, high-density lipoproteins
8. s4 — tch, total cholesterol / HDL
9. s5 — ltg, possibly log of serum triglycerides level
10. s6 — glu, blood sugar level

Diabetes Progression Prediction Example

colab.research.google.com/drive/19_STVshzPU75tHqIx1jbwTEfV4b5XEu9#scrollTo=se5pnBk32ySN

Regression On Diabetes Data.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[17]
```python
#copied mainly from https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.htm
# Code source: Jaques Grobler
# License: BSD 3 clause

import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

[20]
```python
# Load the diabetes dataset
# To see what is returned by load_diabetes:
#     https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html
```

Linear Regression to show independent features

See:
https://towardsdatascience.com/linear-regressions-with-scikitlearn-a5d54efe898f

Is there a linear relationship between sepal lengths and sepal widths?

```
X_var = irisview["sepal length (cm)"]
X_var = X_var.values.reshape(-1 , 1) #numpy.ndarray, ndim 2, shape (150 , 1)
Y_var = irisview["sepal width (cm)"]
Y_var = Y_var.values.reshape(-1 , 1) #numpy.ndarray, ndim 2, shape (150 , 1)
x_train, x_test, y_train, y_test = train_test_split(X_var, Y_var, random_state = 0) #75/25 train-test split
```

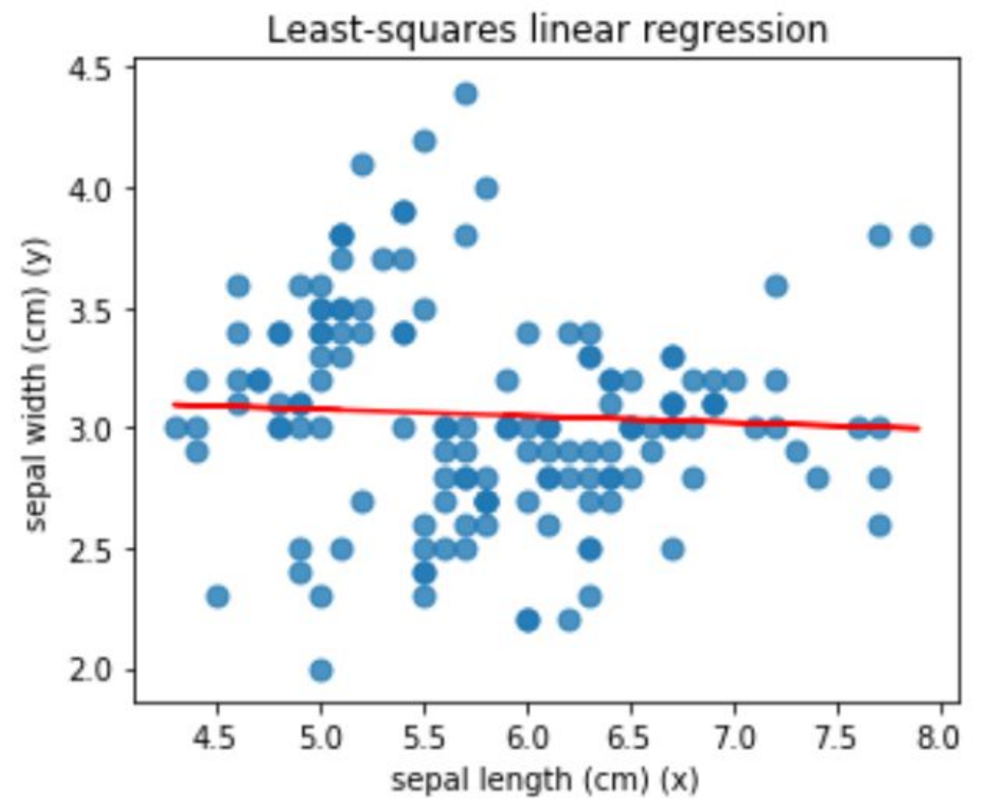
```
linreg = LinearRegression().fit(x_train, y_train)
# create the linreg object in python + train on train data
```

Linear Regression to show independent features

```
coefficient:[[-0.02743947]]
intercept:[3.21331108]
train R squared:0.0030042585383249776
test R squared:0.026506510271823158
```

A very low R shows that the 2 features are not correlated.

You need not separate your data into training and testing for this purpose.



Quiz 5. Problem 1.

1. 2 hrs. Using the Regression on the diabetes data example:

   1.1. *5 points.* Is age highly correlated with total cholesterol / HDL?

   1.2. *5 points.* Is blood pressure highly correlated with total cholesterol / HDL?

   1.3. *15 points (5 each).* Linear fit results for y = ax + b where x is the blood sugar level:

      i. Linear fit coefficients and intercept of the training data

      ii. What is the R^2 for the training data? What is the R^2 for the prediction of y based on blood sugar level for the test data?

      iii. Show a scatter plot of the train set (x, y) as blue circles and predicted (x, y) as green circles. Also show the best fit line in red.

Quiz 5. Problem 2.

2. 1.5 hrs. Use the data provided in the shared file gasoline_use.txt with 80% training data:

   2.1. *10 points*. Show the equation found by fitting the training data:

   $$y = f(x_1, x_2, x_3, x_4) = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4$$

   2.2. *5 points*. What is the R^2 for the prediction of y for the training data?

   2.3. *5 points*. What would happen to gasoline consumption if taxes are increased by $3.00? Use the training data.

Logistic Regression

Logistic Regression (LogR): Class A or Class B?

Based on linear regression, but converted to a Yes/No decision for classification by the sigmoid function. The linear function's output is converted into a probability distribution y ∈ [0, 1]

$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}; \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

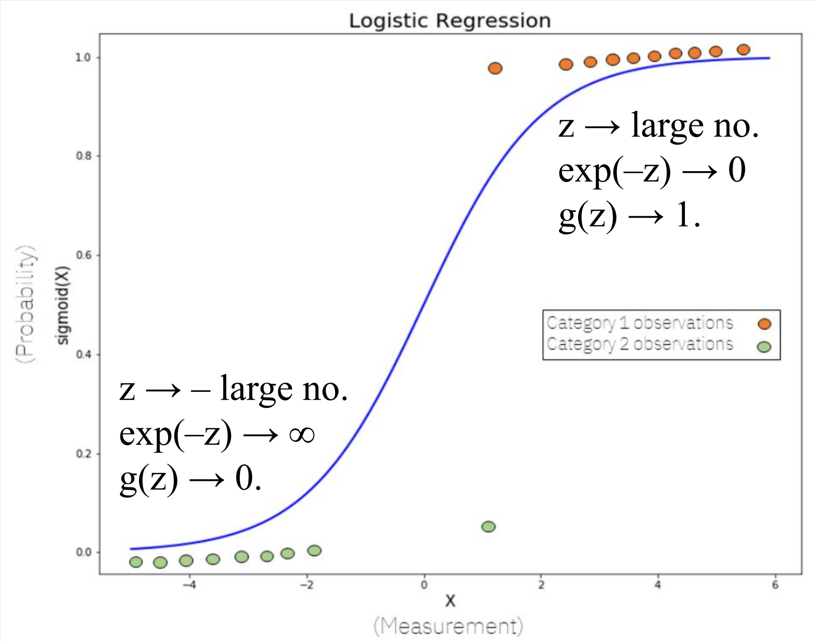
For Linear Regression:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \text{ ; where } x_0 = 1$$

$$= \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

For Logistic Regression, the linear combination undergoes a sigmoidal:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}; \quad g(z) = \frac{1}{1 + e^{-z}}$$



Logistic Regression: Yes/No, A or B only

- Bot or not Bot?
- Tumor is Malignant or Benign?
- Male or Female?

It helps to normalize data for Regression

For regression it helps to normalize the data by subtracting the mean and dividing by standard deviation.

Example, male = 1, female = 2. Actual Feature:

[1, 2, 2, 1, 1, 2, 2, 1, 2]

- Center data around mean by subtract 1.56:

  [-0.556 0.444 0.444 -0.556 -0.556 0.444 0.444 -0.556 0.444]

- Normalize by dividing by SD = 0.53
- Note that the SD of the New feature is 1.
- Remember the mean 1.56 and SD 0.53 used to transform your training data, so it can be applied to the test data.

	Value	(Value - Mean)	(Value - Mean)/SD
	1	-0.56	-1.05
	2	0.44	0.84
	2	0.44	0.84
	1	-0.56	-1.05
	1	-0.56	-1.05
	2	0.44	0.84
	2	0.44	0.84
	1	-0.56	-1.05
	2	0.44	0.84
Mean:	**1.56**	**0.00**	**0.00**
SD:	**0.53**	**0.53**	**1.00**

Logistic Regression

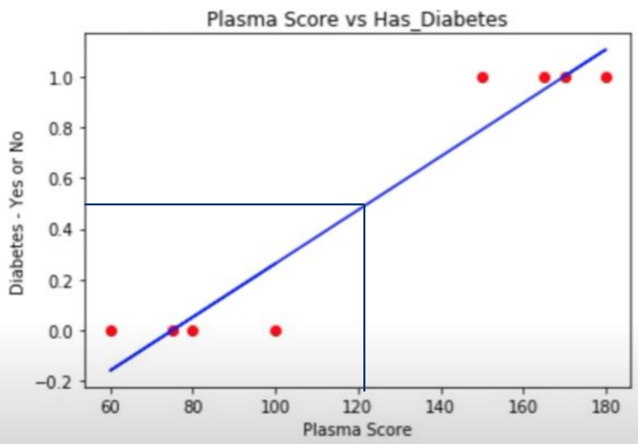
A classification algorithm used to predict a binary outcome (0 or 1) based on input features.

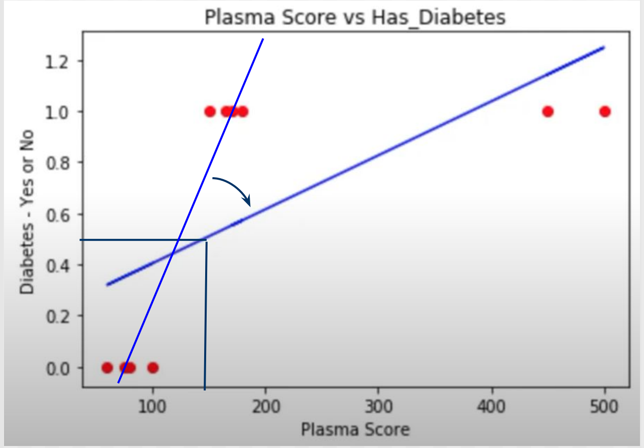
Learning Algorithm: Maximum Likelihood Estimation (MLE), a statistical method to estimate the parameters of a probability distribution based on a set of observed data. Here, MLE is used to estimate the coefficients of the input variables used to predict the probability of the binary outcome.

During training, the algorithm iteratively adjusts the coefficients of the input variables to maximize the likelihood of observing the binary outcomes of the input variables.

Why Linear Regression Fails at Classification?

plasmascore	diabetes
100	0
80	0
75	0
60	0
150	1
165	1
180	1
170	1





Assume plasma score > 120 are diabetic.

Assume h(x) > 0.5 is "Yes Diabetic"; otherwise "No"

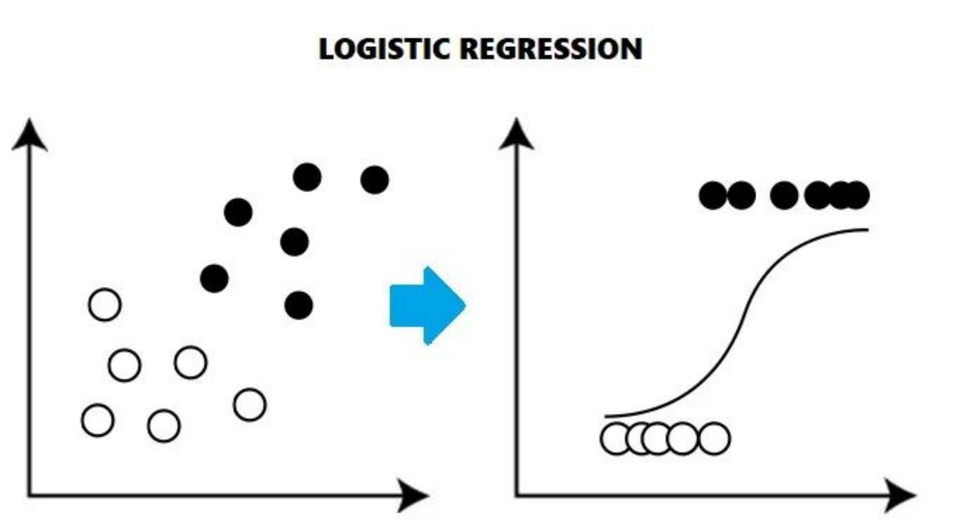
If we add 2 points (450, 1) and (500, 1), the linear fit changes.

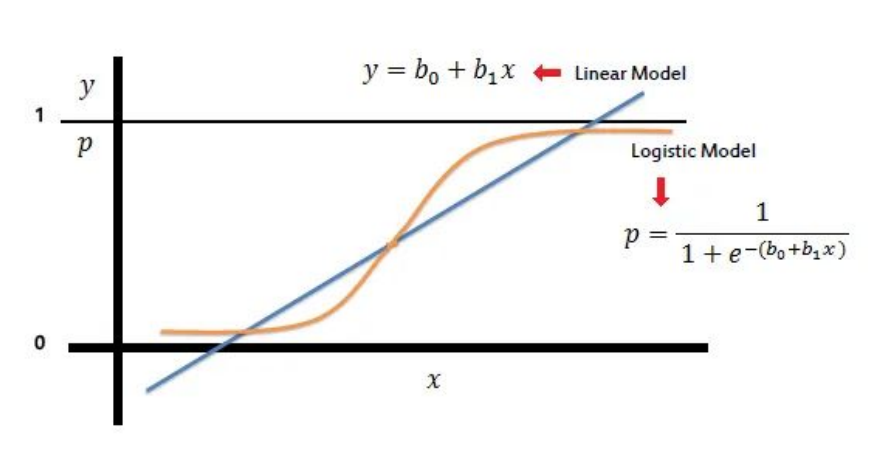
Assume h(x) > 0.5 is "Yes Diabetic"; otherwise "No".

Now at y = 0.5, the x value has increased to 150.

Linear regression is unstable to be a Classifier because you will have to keep adjusting thresholds.

Linear vs. Logistic Regression





Logistic Regression. Class: Obese or Not?

$$x = \sum_{i=0}^{n} a_i x_i;\ where\ x_0 = 1.$$
$$= a_0 + a_1 x_1$$
$$y = sigmoid(x) = \frac{1}{1+e^{-x}}$$

For Obesity classification:

- x_1 is weight
- x is obesity measure
- y is obesity likelihood $\propto$ P(x)

1. Collect dataset of patients diagnosed as obese and their weight.
2. Train our model to fit the S shape line. Parameters after Max Likelihood:

   x = 0.08333 * x_1 -7 where a_0 = -7, a_1 = 0.08333.

3. Use model to make some predictions.

- Patient 1 is 60 Kgs. x = 0.8333 * 60 - 7 = -2.

  y(x) = P(x) = Probability of Obese(x) = 1/(1+exp(-(-2))) = 0.119

- Patient 2 is 120 kgs. x = -3. y = P(x) = 0.95.

Logistic Regression. Class: Obese or Not?

$$x = \sum_{i=0}^{n} a_i x_i;\ where\, x_0 = 1.$$

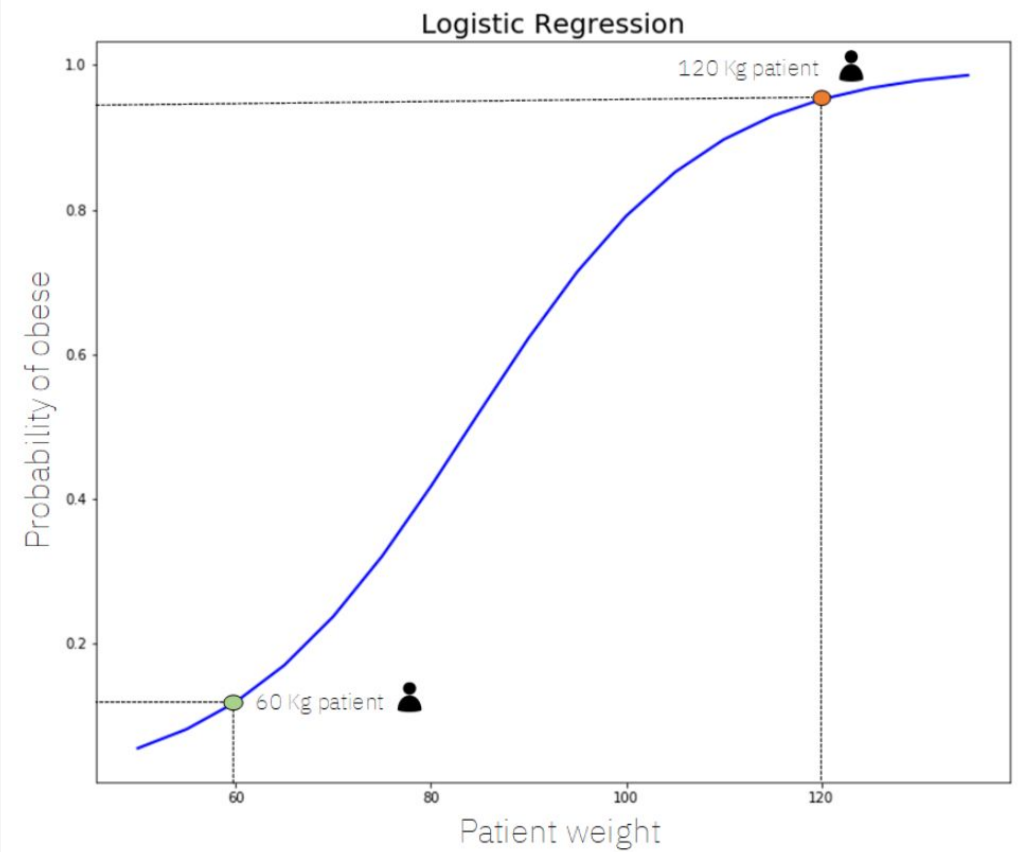
$$= a_0 + a_1 x_1$$

$$y = sigmoid(x) = \frac{1}{1+e^{-x}}$$

$x = 0.08333 * x_1 - 7$

$P(x_1 = 60) = 0.119$

$P(x_1 = 120) = 0.95.$

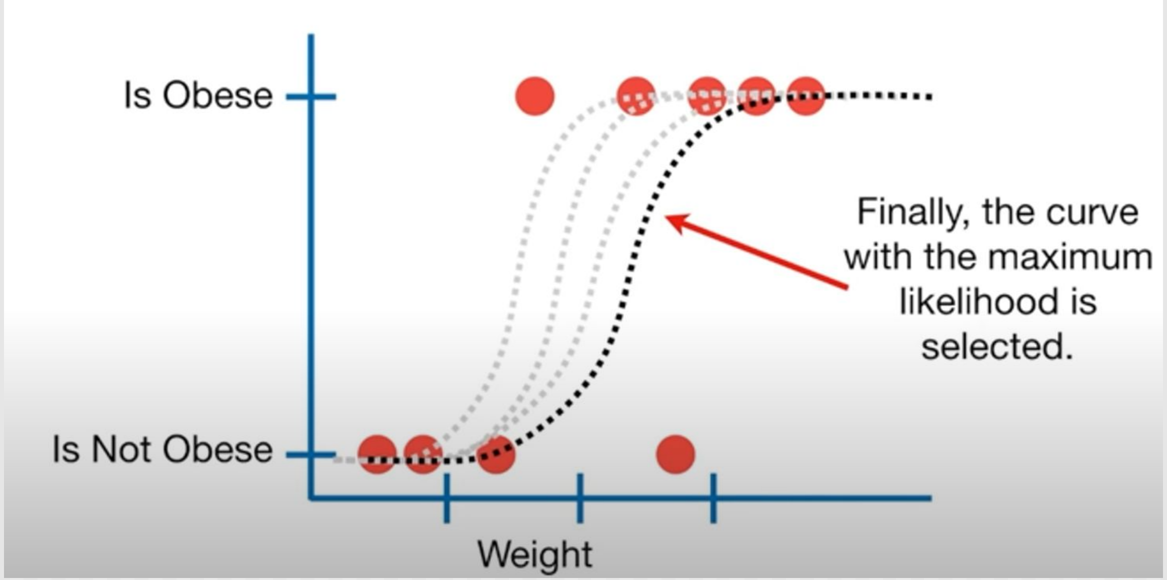


Max Likelihood curve is fit from training data

Find the best values for a_0 and a_1 that will provide maximum likelihood fit to the training data.

$$x = a_0 + a_1 x_1$$

$$y = sigmoid(x) = \frac{1}{1+e^{-x}}$$



Logistic Regression Probability Function

$$P(y=1|\theta,x) = g(z) = \frac{1}{1+e^{-\theta^T x}}$$

$$P(y=0|\theta,x) = 1 - g(z) = 1 - \frac{1}{1+e^{-\theta^T x}}$$

$$P(y|\theta,x) = \left(\frac{1}{1+e^{-\theta^T x}}\right)^y \times \left(1 - \left(\frac{1}{1+e^{-\theta^T x}}\right)\right)^{1-y}$$

Negative Log of Probability is Cost

$$P(y|\theta, x) = \left(\frac{1}{1+e^{-\theta^T x}}\right)^y \times \left(1 - \left(\frac{1}{1+e^{\theta^T x}}\right)\right)^{1-y}$$

In linear regression, we use mean squared error (MSE) as the cost function. But in logistic regression, if the above cost function uses MSE there will be many local minima, so gradient descent on MSE will fail.

Use the negative log of probability to represent the cost function to minimize. It is guaranteed to be convex for all input values, containing only one minimum, allowing us to run the gradient descent algorithm.

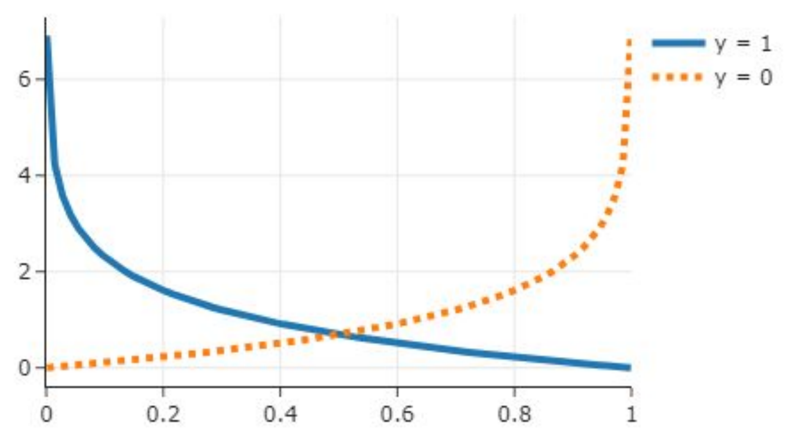
$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{, if } y = 1 \\ -log(1 - h_\theta(x)) & \text{, if } y = 0 \end{cases}$$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Maximizing probability is equivalent to minimizing the negative log cost function.

$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & , \text{if } y = 1 \\ -log(1 - h_\theta(x)) & , \text{if } y = 0 \end{cases}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$



$$\ell(p, y) = \begin{cases} -\log(p) & \text{if } y \text{ is } 1 \\ -\log(1-p) & \text{if } y \text{ is } 0 \end{cases}$$
$$= -y\log(p) - (1-y)\log(1-p)$$

Cost function has only 1 minimum

$$L(\theta_0, \theta_1, \mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_i -y_i \log(\sigma(\theta_0 + \theta_1 x_i)) - (1 - y_i)\log(1 - \sigma(\theta_0 + \theta_1 x_i))$$

$$cost(h_\theta(x), y) = -y^{(i)} \times log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \times log(1 - h_\theta(x^{(i)}))$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \times log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \times log(1 - h_\theta(x^{(i)}))\right]$$

Total cost for m observed data values.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \times log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \times log(1 - h_\theta(x^{(i)}))\right]$$

Gradient Descent Update Rule:

$$\theta_j \leftarrow \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

$$\theta_j \leftarrow \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

$$\frac{d(e^x)}{dx} = e^x$$

$$\frac{d(\ln(x))}{dx} = \frac{1}{x}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Important: After each iteration, update each parameter vector together: $\theta = [\theta_0, \theta_1, ..., \theta_n]$

Derivative of the Sigmoid Function

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] = \frac{d}{dx}(1+e^{-x})^{-1} \\
&= -1 * (1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{-e^{-x}}{-(1+e^{-x})^2} \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \frac{e^{-x}+(1-1)}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \left[\frac{(1+e^{-x})}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right] \\
&= \frac{1}{1+e^{-x}} \left[1 - \frac{1}{1+e^{-x}}\right] \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}
$$

$$\sigma'(x) = \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \times log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \times log(1 - h_\theta(x^{(i)}))\right]$$

Gradient Descent Update Rule:

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

$$\frac{d(\ln(x))}{dx} = \frac{1}{x}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$\sigma'(x) = \frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x))$$

-dJ/dθ_j = y_j*(1/h(x))*h(x)*(1-h(x))*x_j + (1-y_j)* (1/(1-h(x))*(-1)*h(x)*(1-h(x))*x_j
= y_j * (1-h(x))*x_j - (1-y_j)*h(x)*x_j
= y_j * x_j -h(x)*y_j*x_j - h(x)*x_j + h(x)*y_j*x_j
= y_j*x_j - h(x)*x_j = (y_j - h(x))*x_j

dJ/dθ_j = (h(x) - y_j)*x_j

$$\begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} = \frac{1}{m} x^T (h(x) - y)$$

Cost Function in Vectorized Form:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \times log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \times log(1 - h_\theta(x^{(i)}))\right]$$

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

θ is n x 1; n - number of unknown parameters
X is m x n; m - number of data points
h is m x 1
y is m x 1
J(θ) - a scalar

Cost Function in Vectorized Form:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \times log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \times log(1 - h_\theta(x^{(i)}))\right]$$

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1 - y)^T \log(1 - h)\right)$$

θ is n x 1; n - number of unknown parameters
X is m x n; m - number of data points
h is m x 1; y is m x 1 ; $J(\theta)$ - a scalar

Gradient Descent Update Rule in Vectorized Form:

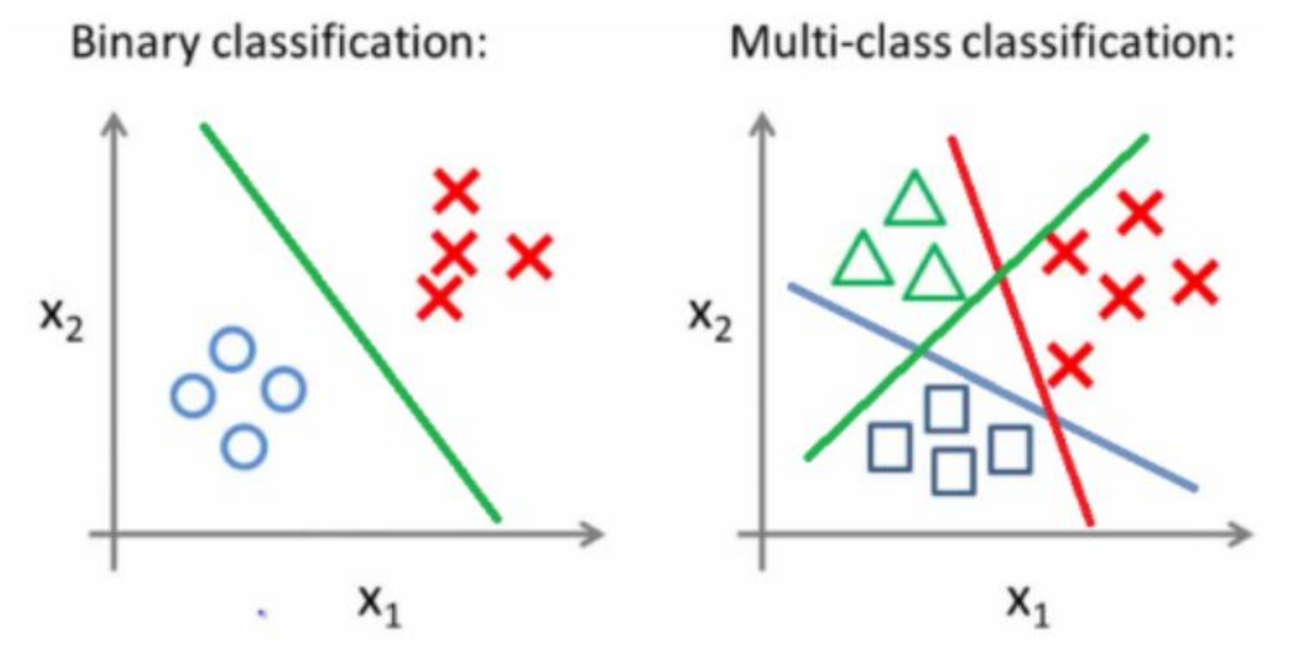
$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

$$\begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} = \frac{1}{m} x^T (h(x) - y)$$

$$\frac{\partial (J(\theta))}{\partial (\theta)} = \frac{1}{m} \mathrm{X}^{\mathrm{T}} [\ \mathrm{h}_\theta(\mathrm{x}) - \mathrm{y}]$$

Log Regression for Multi-Class Classification



Choose the class with highest probability

Logistic Regression using Call to Scipy Lib

Logistic Regression by Library Function.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[26]

```python
import numpy as np
import matplotlib as plt
import math
from scipy.optimize import minimize
```

```python
from sklearn.datasets import load_iris
```

Logistic Regression using Gradient Descent

Logistic Regression by Gradient Descent.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[8]
```
import numpy as np
import matplotlib as plt
import math
from scipy.optimize import minimize
```

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
X, y = load_iris(return_X_y=True)
```

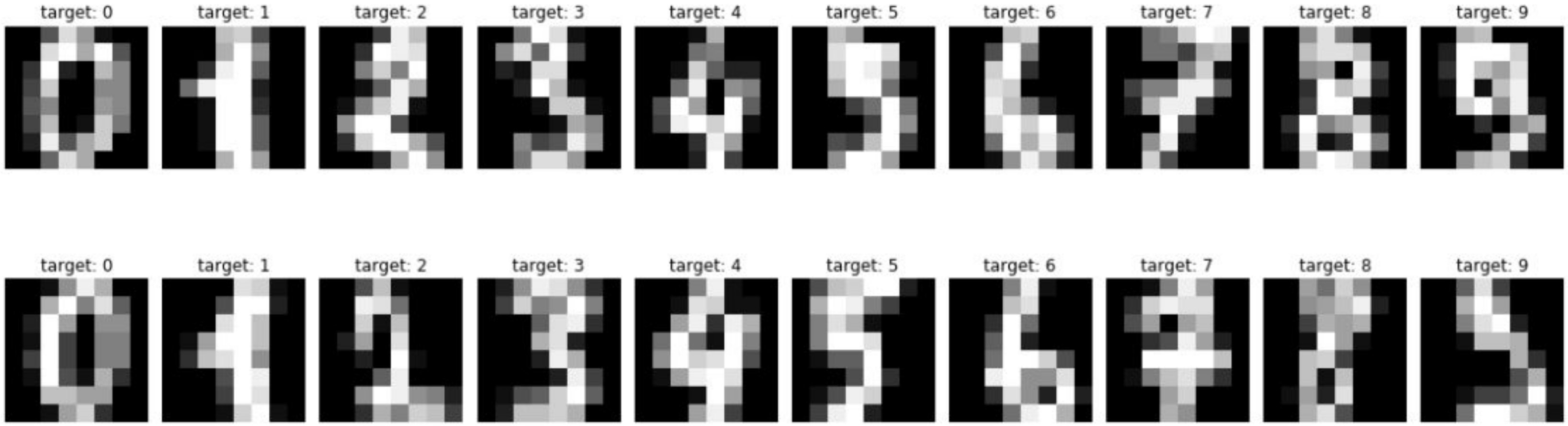
Logistic Regression Example

https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a

About plotting

```
fig, axes = plt.subplots(2, 10, figsize=(16, 6))
for i in range(20):
    axes[i//10, i %10].imshow(mnist.images[i], cmap='gray');
    axes[i//10, i %10].axis('off')
    axes[i//10, i %10].set_title(f"target: {mnist.target[i]}")

plt.tight_layout()
```



Ways to Avoid Overfitting

1. Reduce the number of features. Eg.: PCA.

2. Regularization - reduce the values of parameters θ_j

Regularization for regression

Housing:

- Features: $x_1, x_2, \ldots, x_{100}$
- Parameters: $\theta_0, \theta_1, \theta_2, \ldots, \theta_{100}$

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

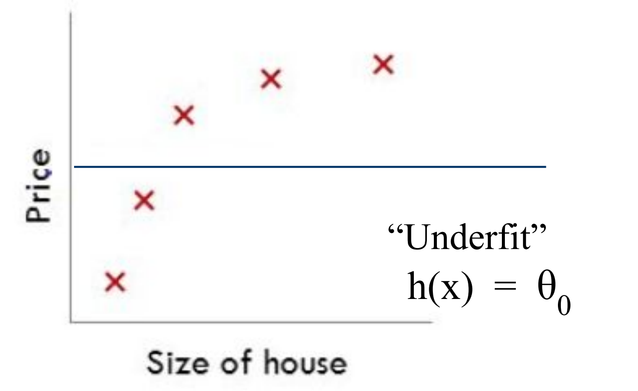
$$J(\theta) = \boxed{\frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right]} + \boxed{\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}$$

Regularized Cost Function for Linear Regression

In regularized linear regression, we choose θ to minimize

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

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



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

$$\theta_1 \to 0;\ \theta_2 \to 0;\ \theta_3 \to 0;\ \theta_4 \to 0;$$

Gradient Descent with Regularization

Regularization Term

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

start at θ_1

Regularization Parameter

Repeat until converge {

$\frac{\partial}{\partial\theta_0}J(\theta)$

$$\theta_0 := \theta_0 - \alpha\boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_0^{(i)}}$$

$\frac{\partial}{\partial\theta_j}J(\theta)$

$$\theta_j := \theta_j - \alpha\boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} + \boxed{\frac{\lambda}{m}\theta_j}} \qquad j = 1, 2, \ldots, n$$

regularization term

}

Gradient Descent with Regularization

Regularization Parameter

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

start at θ_1

Regularization Term

Repeat until converge {

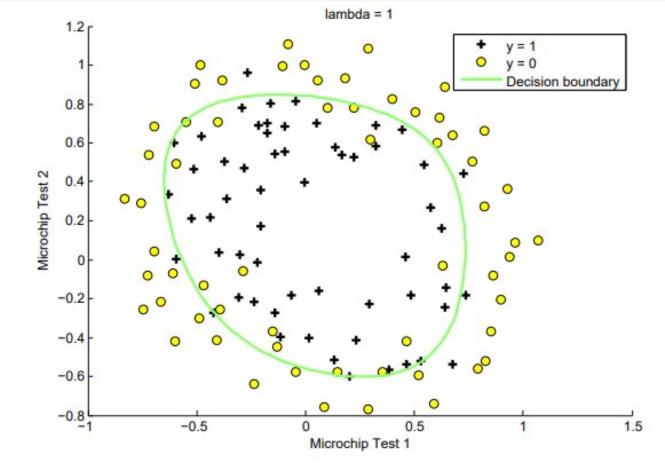
$$\theta_0 := \theta_0 - \alpha \underbrace{\boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right){x_0}^{(i)}}}_{\frac{\partial}{\partial\theta_0}J(\theta)}$$

$$\theta_j := \theta_j - \alpha \underbrace{\boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right){x_j}^{(i)} + \boxed{\frac{\lambda}{m}\theta_j}}}_{\frac{\partial}{\partial\theta_j}J(\theta)} \qquad j = 1, 2, \ldots, n$$

regularization term

}

Lambda vs. Overfitting in Regularized Log R



$\lambda = 1$ (ideal)

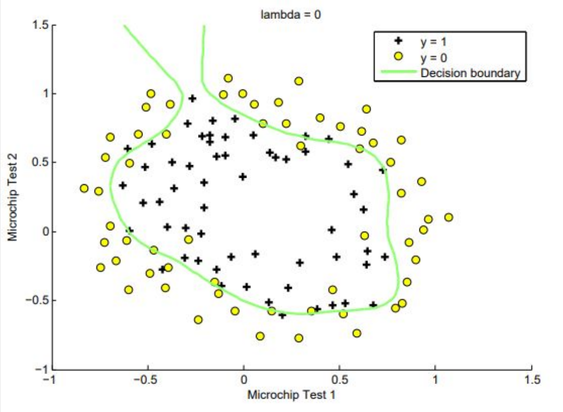


Figure 5: No regularization (Overfitting) ($\lambda = 0$)

$\lambda = 0$ (over)

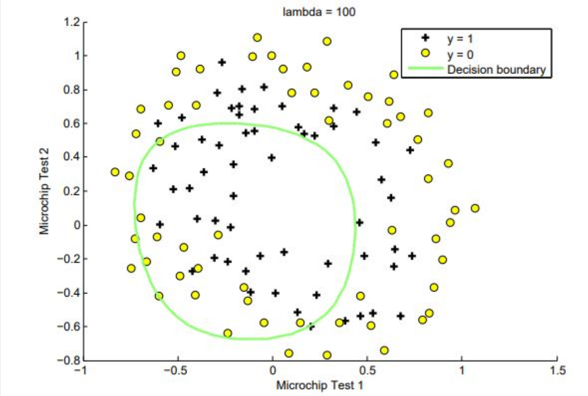


Figure 6: Too much regularization (Underfitting) ($\lambda = 100$)

$\lambda = 100$ (under)

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2.$$