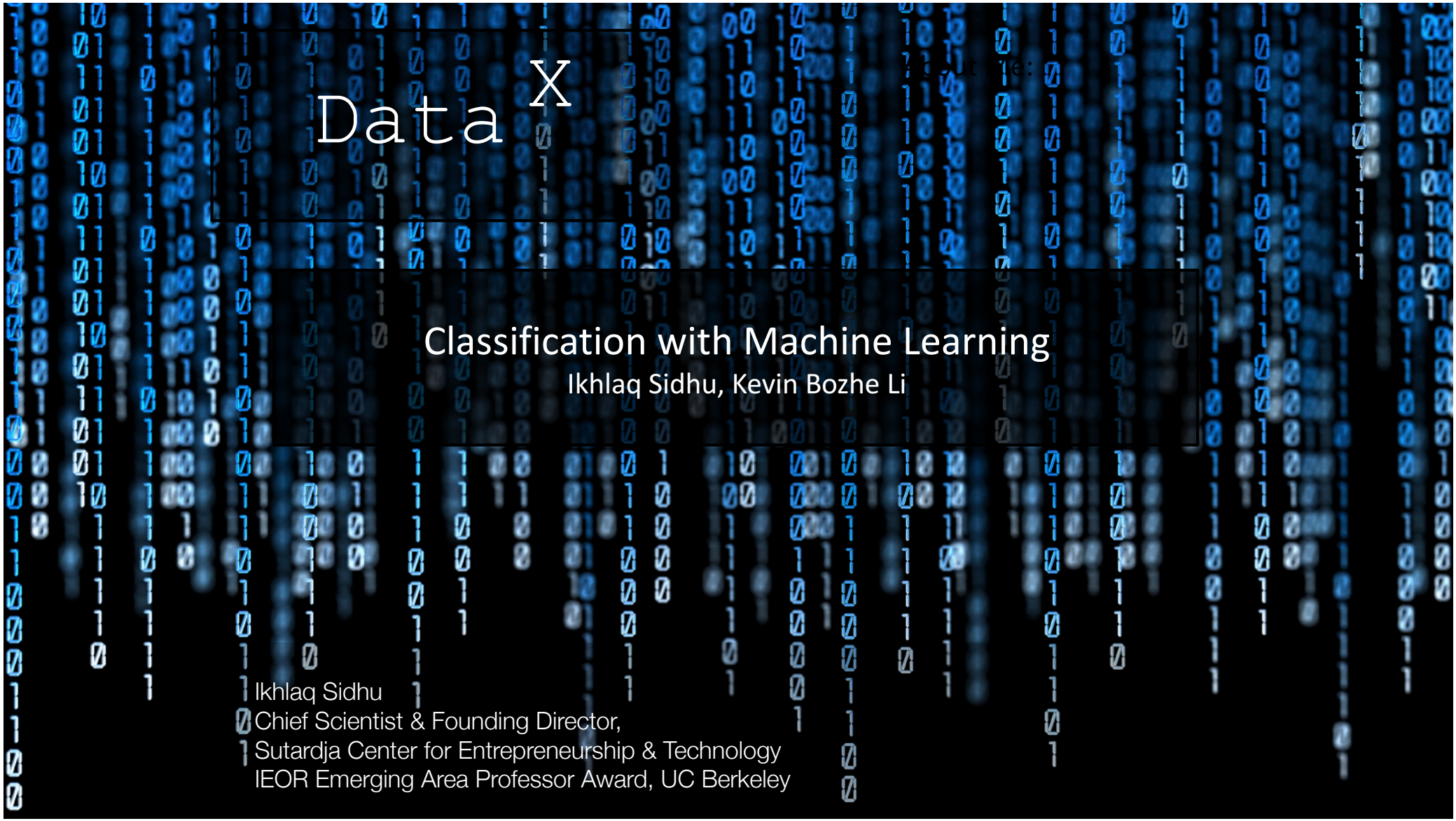
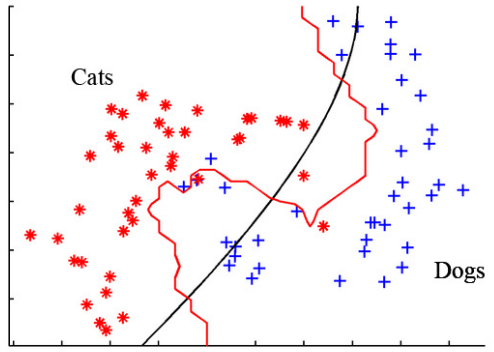
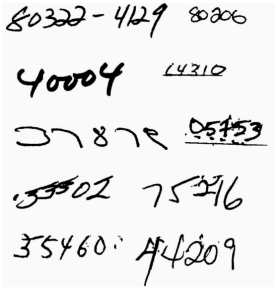

Data X
Classification with Machine Learning
Ikhlaq Sidhu, Kevin Bozhe Li
Ikhlaq Sidhu
Chief Scientist & Founding Director,
Sutardja Center for Entrepreneurship & Technology
IEOR Emerging Area Professor Award, UC Berkeley

Classification

Examples

- Radar: WWII
- Spam Detection
- Image Classification: Cats VS Dogs
- Image Classification: Recognizing Digits





Our Goal: To classify items.

We start with this: (X,Y): (x1,y1), (x2, y2) .. (xn,yn)

An input
xi

Model: f(x,W)

That is leaned

An output
y1

One hot vector style vector:
y1 = [0 0 0 1 0 0]

Y(i,1) = 1 if picture is dog, else -1*

Y(i,2) = 1 if picture is cat, else -1*

- xi can be a vector for each data element, like
  x3 = [12 15 22] = [height, weight, color]

- For a picture: **x_1 = [32 x 32 x 3]:** (matrix) array of numbers

y_i = [+1 , -1, .. -1]
or
y_i = [+1 , 0, .. 0]
* Sometimes 0 vs 1 instead of -1 vs 1

In practice, the vector [0 0 1 0 0 ..] can Be translated to simply '2', meaning category 2.

Our Goal: To classify items.

We have this: (X,Y)

xi → Model: f(x,W)

Actual Results:

$y_i = [y_{i,1}, y_{i,2}, .. y_{i,k}]$

$y_i = [+1 , -1, .. -1]$

And sometimes mapped to

$y_i = [+1 , 0, 1.. 0]$

Machine Learning Steps to train a classifier model

1. Choose our model: $f(x_i, W)$ = our estimate of Y

   If f(x) is a Linear model: $f(i) = f(x_i) = Wx_i = w_1x_{i,1}+w_2 x_{i,2}...$
   W is list of parameters.
   Results in a different estimated value of Yi for each sample point x_i

2. Define a loss function (L) = which is a function L(Y_atual, vs Y_estimated)

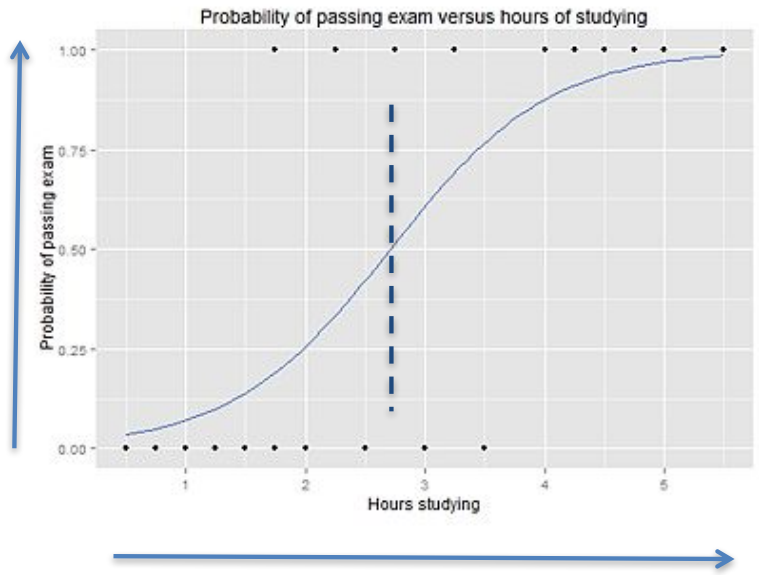
3. Optimize across the parameter space (W) to minimize the loss function to

Example Classification with Logistic Regression

Data: students study for an exam:
(x= hours studied, y = pass/not pass)

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

y (or t):
0 = fail
1 = pass



x

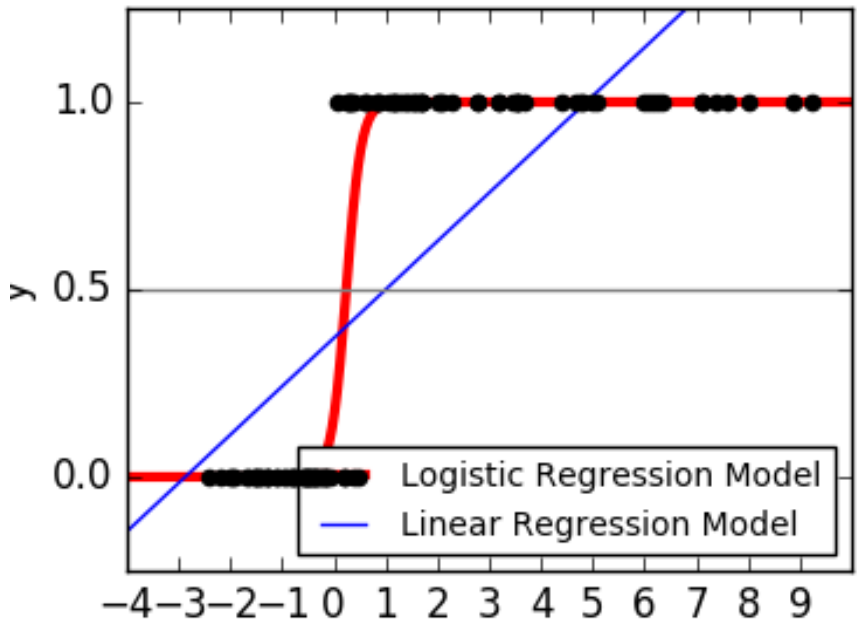
In this case output is binary (pass or not pass)

Problem:
Student studies x hours
Will the student pass

We use this curve to predict the probability that the student would pass given x hours of study

If Prob > 0.5, we classify student will pass the exam

Example: Logistic Regression
We use Logistic Regression because
a line is not a good estimator for binary results (classification)

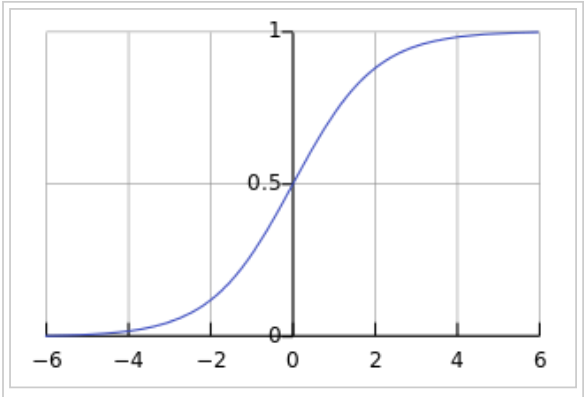


The logit function is a better fit for binary results

This function is a better fit for binary results:

$$f(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

Large t -> 1
Small t -> 0



This function results in (0,1)
for all real numbers (like a probability)

And if t has this form:

$t = w_0 + x_1 w_1$ (a line) → $f(x_i, w0, w1) = \frac{1}{1 + e^{-(w0 + x1w1)}}$ (s shaped curve)

If w1 is small → slow rise
If w1 is large → fast rise

- Think of f(x,w) as probability y = 1 given any x
- Prob (y=1) = ½ when $e^{-(w0 + xi,1\ w1)} = 1$, ie $w_0 + x_1 w_1 = 0$
- Choose w0, w1,..wd to get best fit
- Still need a loss function and then solve for best W

Note: Loss Function Used in Logistic Regression

$$\text{Cross Entropy} = -t\ln(f(\vec{x})) - (1-t)\ln(1-f(\vec{x}))$$

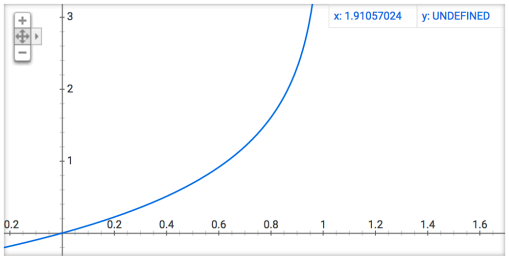
Actual Output: y (output)

Estimated y (output)

y is -1 or +1,
t = (1+y)/2, so
t ranges from 0 to 1

if t = 0: loss = -ln(1-f(x))
if f(x) is 1 -> loss = e,
if f(x) is 0 -> loss = 0

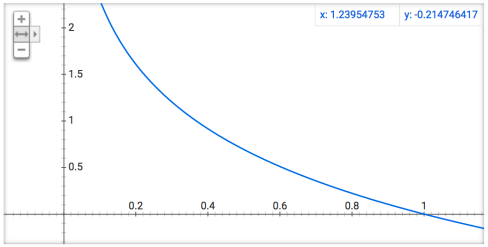
-ln(1-f(x)) when t = 0



Larger loss

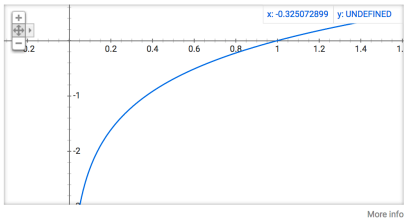
If t = 1: loss = - ln(f(x))
if f(x) is near 0 ->
loss = large
if f(x) is near 1 ->
loss = approx. = 0

-ln(f(x)) when t = 1

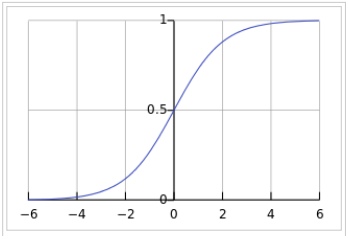


Larger loss

Graph for ln(x)



$$f(x_i, w0, w1) = \frac{1}{1 + e^{-(w0 + x1w1)}}$$



Other Famous Loss Functions

- Logistic Loss $= \frac{1}{\ln 2} \ln(1 + e^{-yf(\vec{x})})$

y f(x) = 1 for match: proportional to exp(1+e^{-1})
y f(x) = -1 for no match, proportional to exp (1+e^{1})

- Cross Entropy $= -t \ln(f(\vec{x})) - (1 - t) \ln(1 - f(\vec{x}))$

t = (1+y)/2, so t ranges from 0 to 1

if t = 0: loss = -ln(1-f(x)) = e, when f(x) is 1

If t = 1: loss = - ln(f(x) = large when score f(x) is near 0
approx. = 0, when f(x) = 1

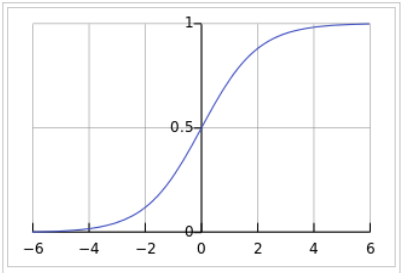
Note: Used for Logistic Regression

The logit function can be extended for multiple features (dimensions)

Logit function:

$$f(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

Large t -> 1, Small t -> 0



And if t has this form:

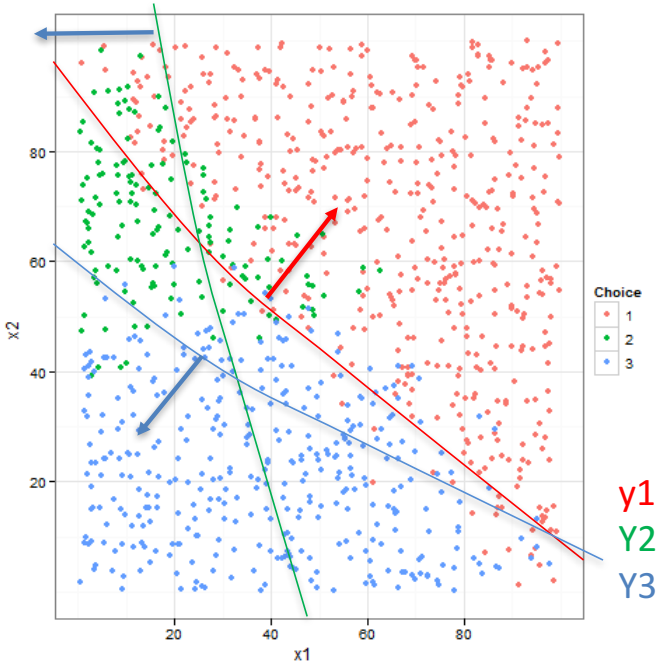
$t = w_0 + x_1 w_1$

or

$t = Wx_i$ (in matrix form) $= w_1 x_{i,1} + w_2 x_{i,2} \ldots$

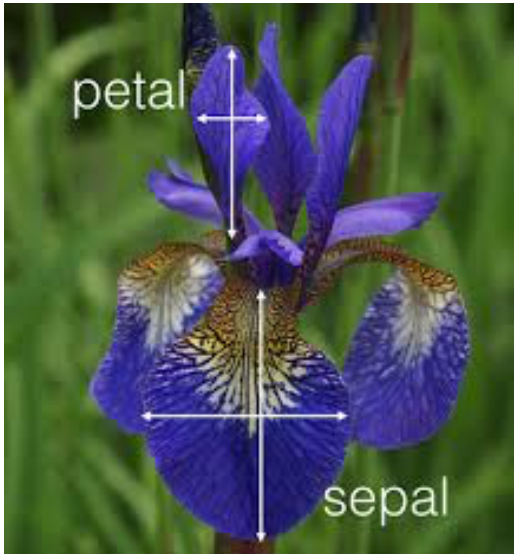
$$f(x_i, W) = \frac{1}{1 + e^{-(w0 + xi,1\ w1 + xi,2\ w2\ \ldots)}}$$

- Easily extends to multiple features (x1, x2, ..)
- And multiple parameter weights



y1 boundary
Y2 boundary
Y3 boundary

Example Code Sample with Logistic Regression Classifier



Input data

X: Attribute Information:
- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm

Y:
0 = 'setosa',
1 = 'versicolor',
2 = 'virginica'

```
print type (X)
print X[0:5]

<type 'numpy.ndarray'>
[[ 5.1  3.5  1.4  0.2]
 [ 4.9  3.   1.4  0.2]
 [ 4.7  3.2  1.3  0.2]
 [ 4.6  3.1  1.5  0.2]
 [ 5.   3.6  1.4  0.2]]


print Y[0:5]
[0 0 0 0 0]
```

Example Code Sample with Logistic Regression Classifier

```
import numpy as np
from sklearn import linear_model, datasets

X = iris.data[:, 1:3]  # only the first two features.
Y = iris.target

# https://en.wikipedia.org/wiki/Logistic_regression
logreg = linear_model.LogisticRegression(C=1e5)

# we create an instance of Neighbours Classifier and
fit the data.
logreg.fit(X, Y)

# predict a category for every row in X
Z = logreg.predict(X)
```

1 → 2 → 3 → 4 →

Class sklearn.linear_model.LogisticRegression

(penalty='l2',
dual=False,
tol=0.0001,
C=1.0,
fit_intercept=True,
intercept_scaling=1,
class_weight=None,
random_state=None,
solver='liblinear', max_iter=100,
multi_class='ovr', verbose=0,
warm_start=False,
n_jobs=1)

http://scikit-learn.org/stable/modules/generatedsklearn.linear_model.LogisticRegression.html

* Z[2] will be the predicted number for row X[2]

Methods for LogisticRegression

Methods

`decision_function`(X)	Predict confidence scores for samples.
`densify`()	Convert coefficient matrix to dense array format.
`fit`(X, y[, sample_weight])	Fit the model according to the given training data.
`fit_transform`(X[, y])	Fit to data, then transform it.
`get_params`([deep])	Get parameters for this estimator.
`predict`(X)	Predict class labels for samples in X.
`predict_log_proba`(X)	Log of probability estimates.
`predict_proba`(X)	Probability estimates.
`score`(X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
`set_params`(**params)	Set the parameters of this estimator.
`sparsify`()	Convert coefficient matrix to sparse format.
`transform`(*args, **kwargs)	DEPRECATED: Support to use estimators as feature selectors will be removed in version 0.19.

fit(*X, y, sample_weight=None*) [source]

Fit the model according to the given training data.

Parameters: **X** : {array-like, sparse matrix}, shape (n_samples, n_features)

Training vector, where n_samples is the number of samples and n_features is the number of features.

y : array-like, shape (n_samples,)

Target vector relative to X.

sample_weight : array-like, shape (n_samples,) optional

Array of weights that are assigned to individual samples. If not provided, then each sample is given unit weight.

New in version 0.17: sample_weight support to LogisticRegression.

Returns: **self** : object

Returns self.

predict(*X*) [source]

Predict class labels for samples in X.

Parameters: **X** : {array-like, sparse matrix}, shape = [n_samples, n_features]

Samples.

Returns: **C** : array, shape = [n_samples]

Predicted class label per sample.

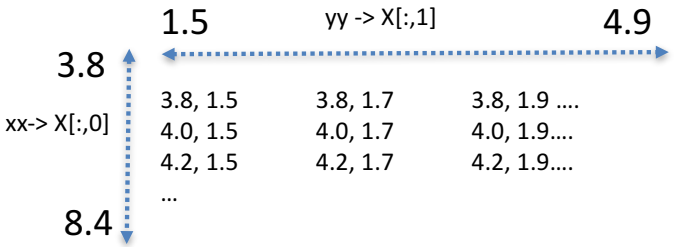
Fit and predict from ScikitLearn

Code Samples with SciKit Learn

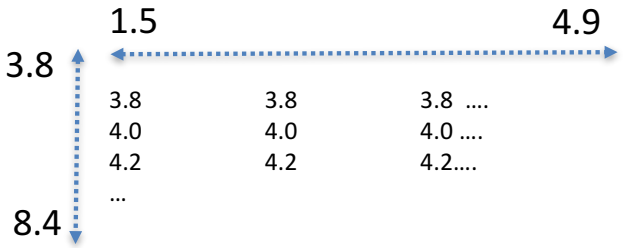
```
# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])
# numpy.ravel: Return a contiguous flattened array.
```

xx shape is (171, 231)
yy shape is (171, 231)

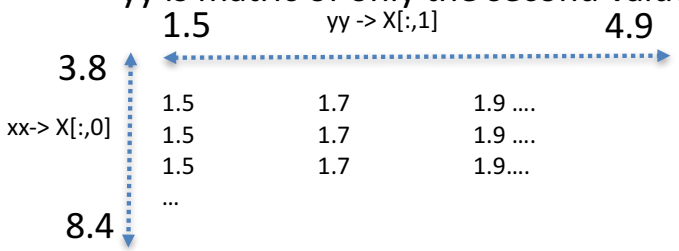
np.c returns shape (39501, 2)
[[3.8 1.5]
[3.82 1.5]
[3.84 1.5] …]
Z shape is (39501,)



xx is a matrix of all the first values



yy is matric of only the second values



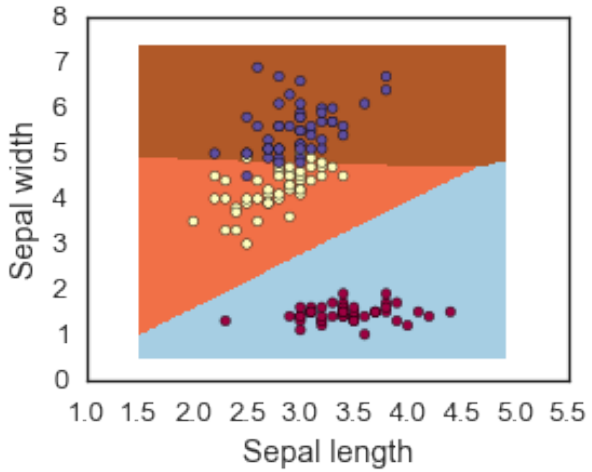
Plotting the Results

```
# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

# Plot also the training points
plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k',
cmap=get_cmap("Spectral"))
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

#plt.xlim(xx.min(), xx.max())
#plt.ylim(yy.min(), yy.max())
#plt.xticks(())
#plt.yticks(())

plt.show()
```



Regularization

Why: To avoid over-fitting

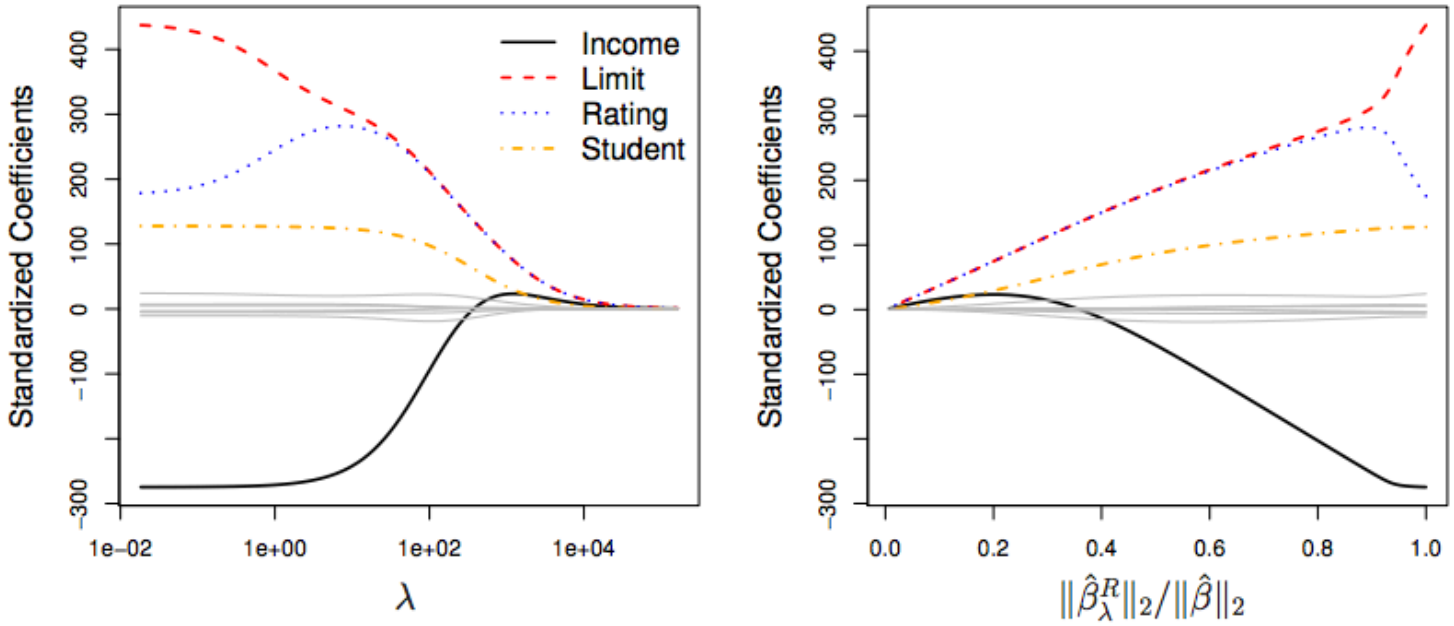
How: You penalize your loss function by adding a multiple of an L1 (LASSO) or an L2 (Ridge) norm of your weights vector w

Your new loss function = $L(X,Y) + \lambda N(w)$

Tuning the regularization term λ: Cross-validation:

- divide your training data,
- train your model for a fixed value of λ and test it on the remaining subsets
- repeat this procedure while varying λ.
  Then you select the best λ that minimizes your loss function.

Shrinkage Methods II: An example



References

- The material presented in this lecture references lecture material draws on the materials the following courses:
- UC Berkeley – CS 294-129 (Designing, Visualizing, and Understanding Deep Neural Networks): https://bcourses.berkeley.edu/courses/1453965/pages/cs294-129-designing-visualizing-and-understanding-deep-neural-networks
- Stanford – CS231n (Convolutional Neural Networks for Visual Recognition): http://cs231n.stanford.edu/