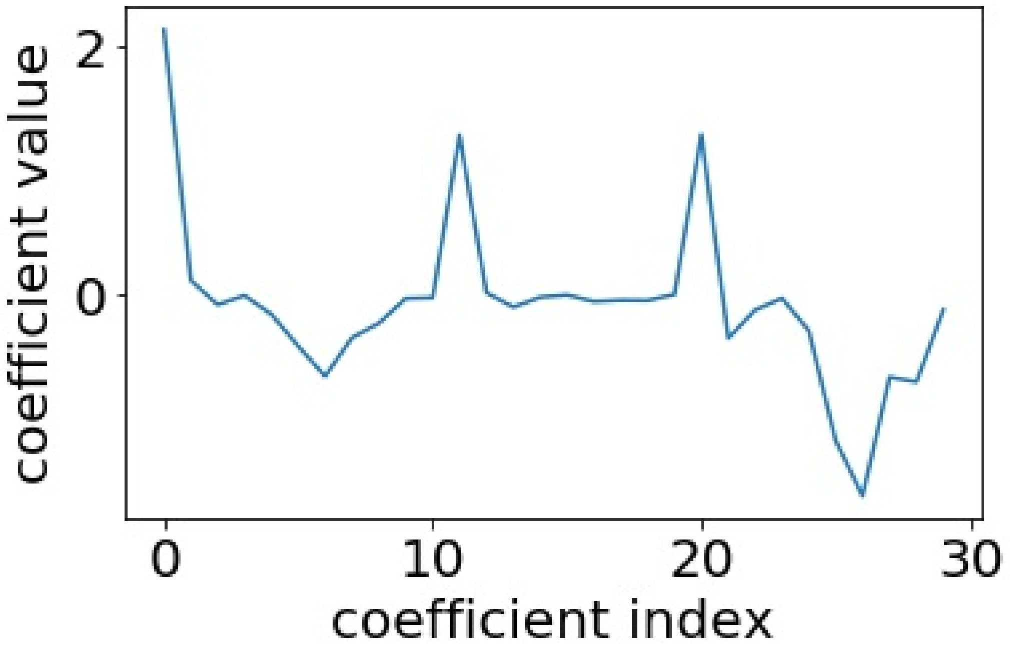
LINEAR CLASSIFIERS IN PYTHON

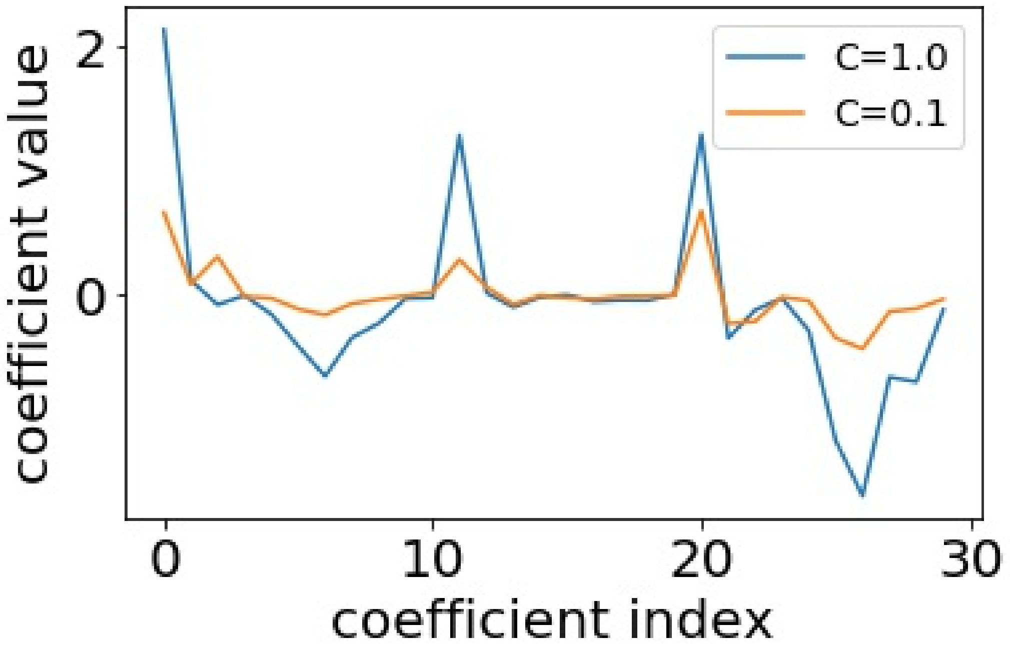
Logistic regression and regularization

Michael (Mike) Gelbart
Instructor
The University of British Columbia

Regularized logistic regression



Regularized logistic regression



How does regularization affect training accuracy?

```
In [1]: lr_weak_reg = LogisticRegression(C=100)

In [2]: lr_strong_reg = LogisticRegression(C=0.01)
```

```
In [3]: lr_weak_reg.fit(X_train, y_train)

In [4]: lr_strong_reg.fit(X_train, y_train)
```

```
In [3]: lr_weak_reg.score(X_train, y_train)
Out[3]: 1.0

In [4]: lr_strong_reg.score(X_train, y_train)
Out[4]: 0.92
```

- $\text{regularized loss} = \text{original loss} + \text{large coefficient penalty}$
- more regularization: lower training accuracy

How does regularization affect test accuracy?

```
In [5]: lr_weak_reg.score(X_test, y_test)
Out[5]: 0.86

In [6]: lr_strong_reg.score(X_test, y_test)
Out[6]: 0.88
```

- $\text{regularized loss} = \text{original loss} + \text{large coefficient penalty}$
- more regularization: lower training accuracy
- less regularization: (almost always) higher test accuracy

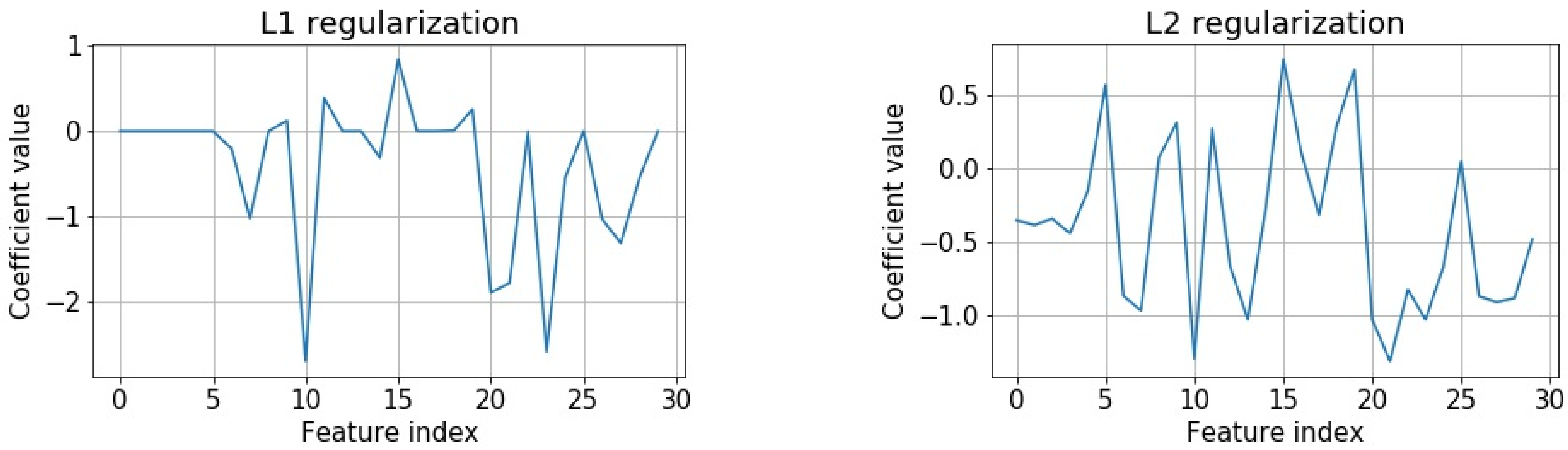
L1 vs. L2 regularization

- Lasso = linear regression with L1 regularization
- Ridge = linear regression with L2 regularization
- For other models like logistic regression we just say L1, L2, etc.

```
In [1]: lr_L1 = LogisticRegression(penalty='l1')

In [2]: lr_L2 = LogisticRegression() # penalty='l2' by default

In [3]: lr_L1.fit(X_train, y_train)

In [4]: lr_L2.fit(X_train, y_train)
```

```
In [5]: plt.plot(lr_L1.coef_.flatten())

In [6]: plt.plot(lr_L2.coef_.flatten())
```

L2 vs. L1 regularization



LINEAR CLASSIFIERS IN PYTHON

Let's practice!

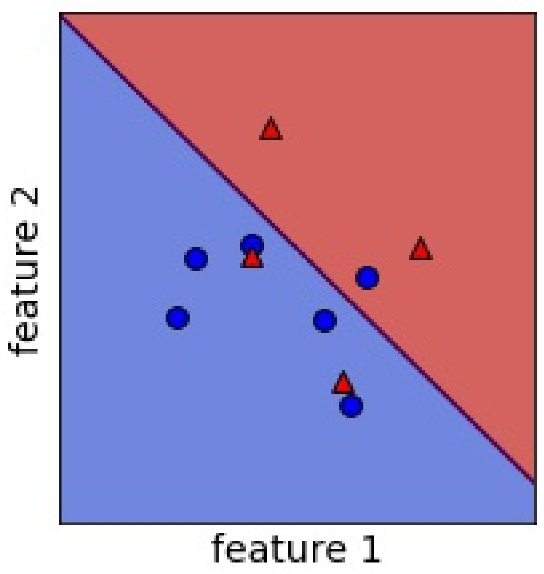
LINEAR CLASSIFIERS IN PYTHON

Logistic regression and probabilities

Michael (Mike) Gelbart
Instructor
The University of British Columbia

Logistic regression probabilities

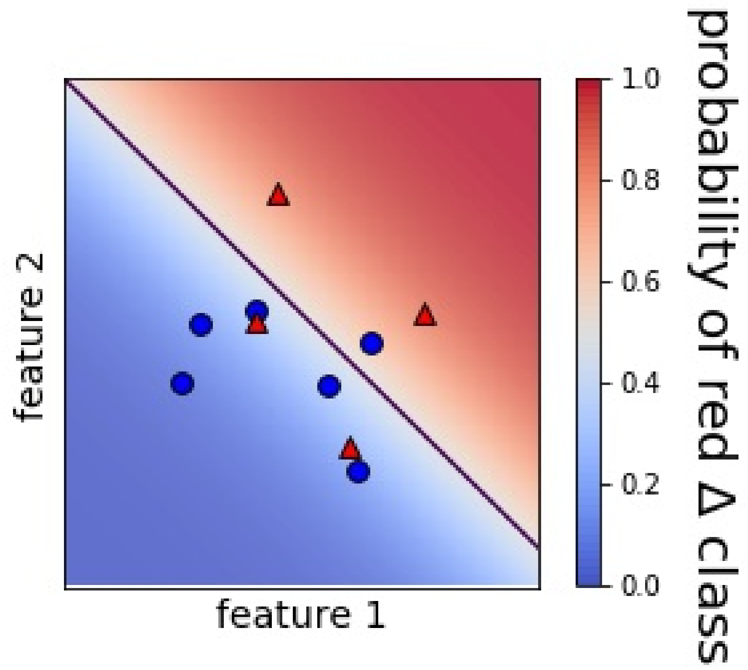
Without regularization ($C = 10^8$):



- model coefficients: [[1.55 1.57]]
- model intercept: [-0.64]

Logistic regression probabilities

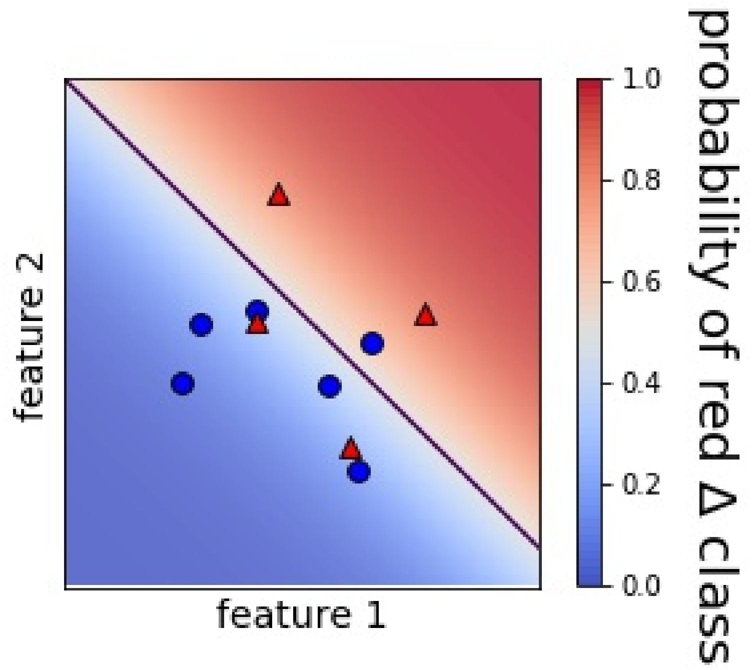
Without regularization ($C = 10^8$):



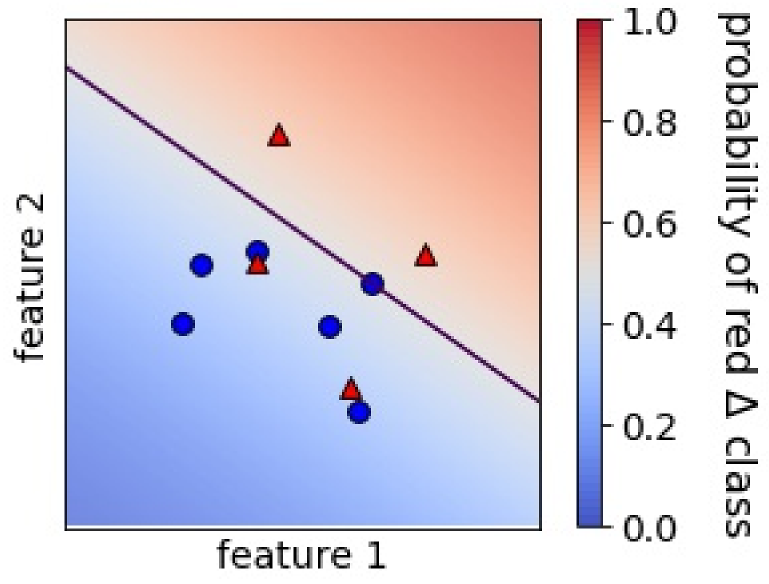
- model coefficients: [[1.55 1.57]]
- model intercept: [-0.64]

Logistic regression probabilities

Without regularization ($C = 10^8$):



With regularization ($C = 1$):

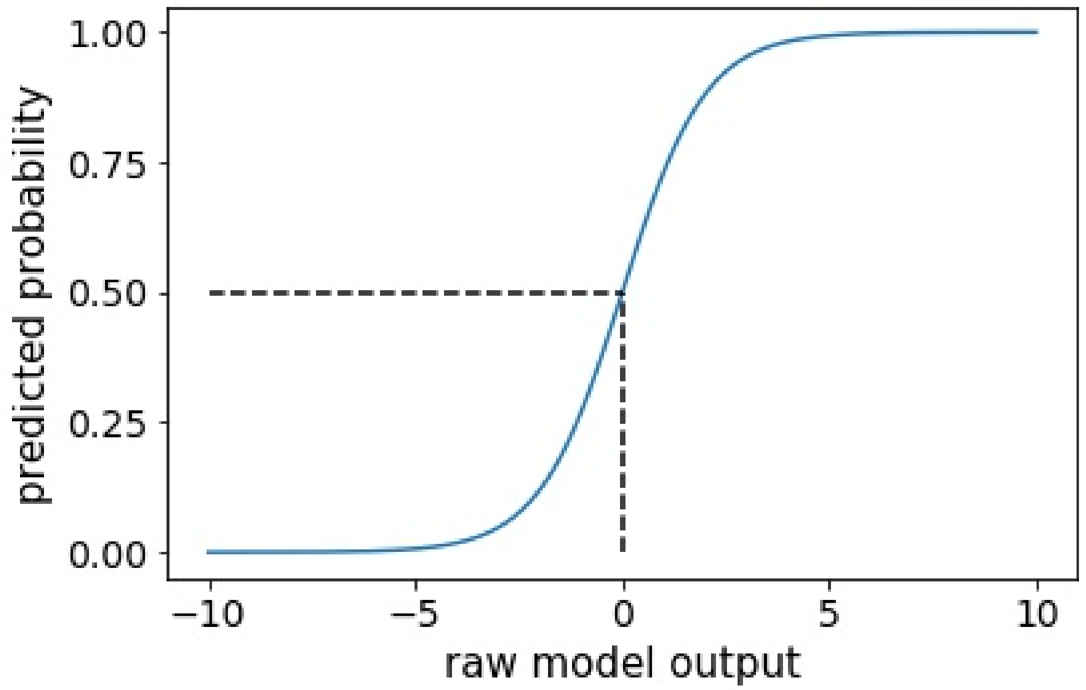


- model coefficients: [[1.55 1.57]]
- model intercept: [-0.64]

- model coefficients: [[0.45 0.64]]
- model intercept: [-0.26]

How are these probabilities computed?

- logistic regression predictions: sign of raw model output
- logistic regression probabilities: "squashed" raw model output



LINEAR CLASSIFIERS IN PYTHON

Let's practice!

LINEAR CLASSIFIERS IN PYTHON

Multi-class logistic regression

Michael (Mike) Gelbart
Instructor
The University of British Columbia

Combining binary classifiers with one-vs-rest

```
In [1]: lr0.fit(X, y==0)

In [2]: lr1.fit(X, y==1)

In [3]: lr2.fit(X, y==2)
```

```
In [4]: lr0.decision_function(X)[0] # get raw model output
Out[4]: 6.124

In [5]: lr1.decision_function(X)[0]
Out[5]: -5.429

In [6]: lr2.decision_function(X)[0]
Out[6]: -7.532
```

```
In [7]: lr.fit(X, y)
```

```
In [8]: lr.predict(X)[0]
Out[8]: 0
```

One-vs-rest vs. multinomial/softmax

One-vs-rest:

- fit a binary classifier for each class
- predict with all, take largest output
- pro: simple, modular
- con: not directly optimizing accuracy
- common for SVMs as well
- can produce probabilities

"Multinomial" or "softmax":

- fit a single classifier for all classes
- prediction directly outputs best class
- con: more complicated, new code
- pro: tackle the problem directly
- possible for SVMs, but less common

Model coefficients for multi-class

```
In [1]: lr_ovr = LogisticRegression() # one-vs-rest by default

In [2]: lr_ovr.fit(X,y)

In [3]: lr_ovr.coef_.shape
Out[3]: (3,13)

In [4]: lr_ovr.intercept_.shape
Out[4]: (3,)
```

```
In [5]: lr_mn = LogisticRegression(multi_class="multinomial",solver="lbfgs")

In [6]: lr_mn.fit(X,y)

In [7]: lr_mn.coef_.shape
Out[7]: (3,13)

In [8]: lr_mn.intercept_.shape
Out[8]: (3,)
```

LINEAR CLASSIFIERS IN PYTHON

Let's practice!