Supervised Learning, pt 2.2

INST414 - Data Science Techniques

This Module's Learning Objectives

Part 2

- Define regularization and its use of improving generalizability
- Use linear regression models to predict scores for data elements
- Explain how linear regression can be adapted for classification
- Extract the class-label probabilities for model outputs

This Module's Learning Objectives

Part 2

- Define regularization and its use of improving generalizability
- Use linear regression models to predict scores for data elements
- Explain how linear regression can be adapted for classification
- Extract the class-label probabilities for model outputs

Regularization: Methods for penalizing model complexity

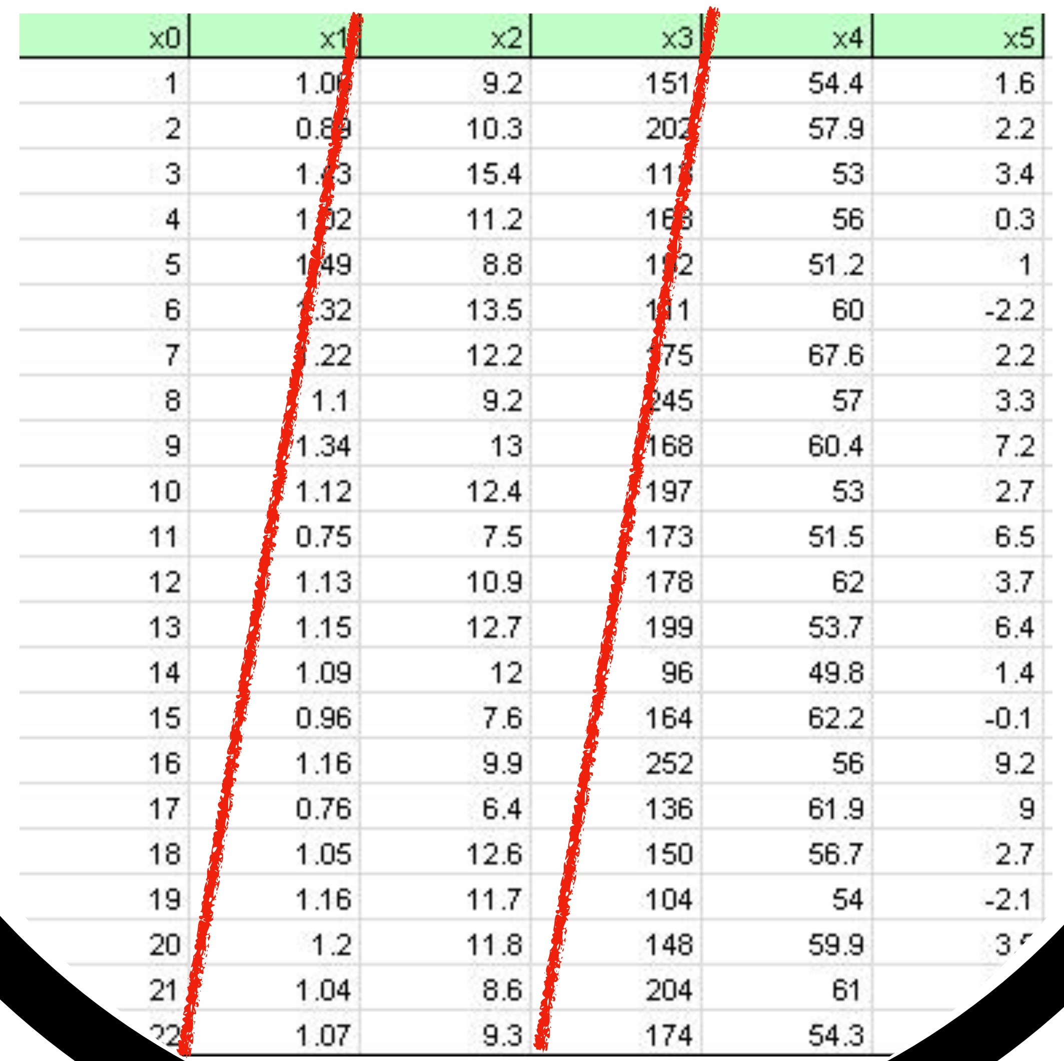
Reducing complexity reduces overfitting

Increases model interpretability

(Often) Decreases computational burden

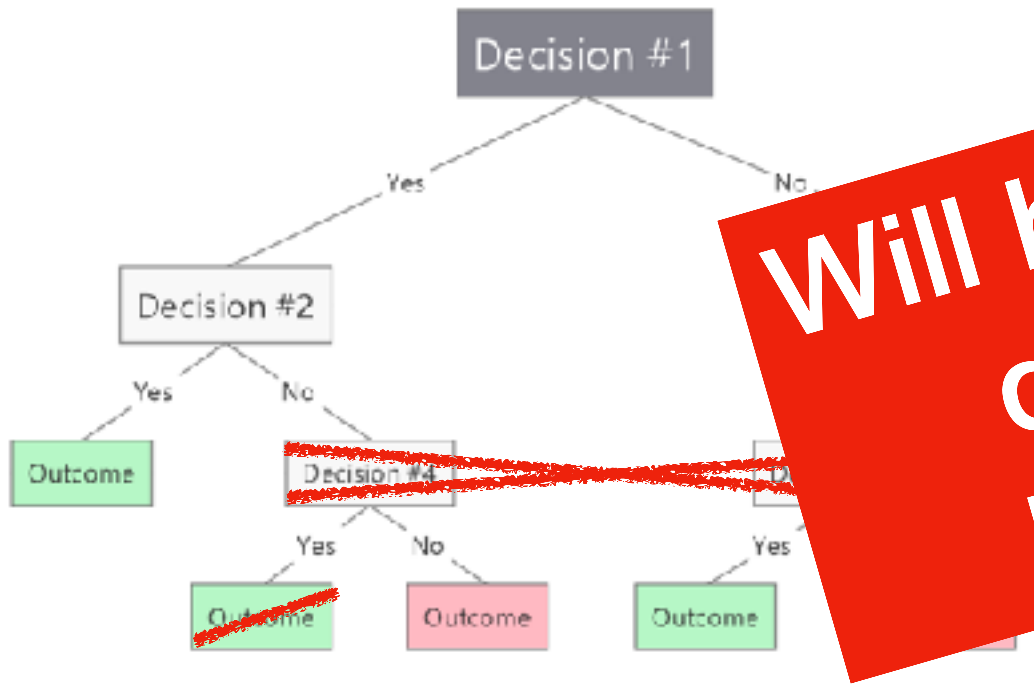
Regularization: Methods for penalizing model complexity

How might you penalize model complexity?



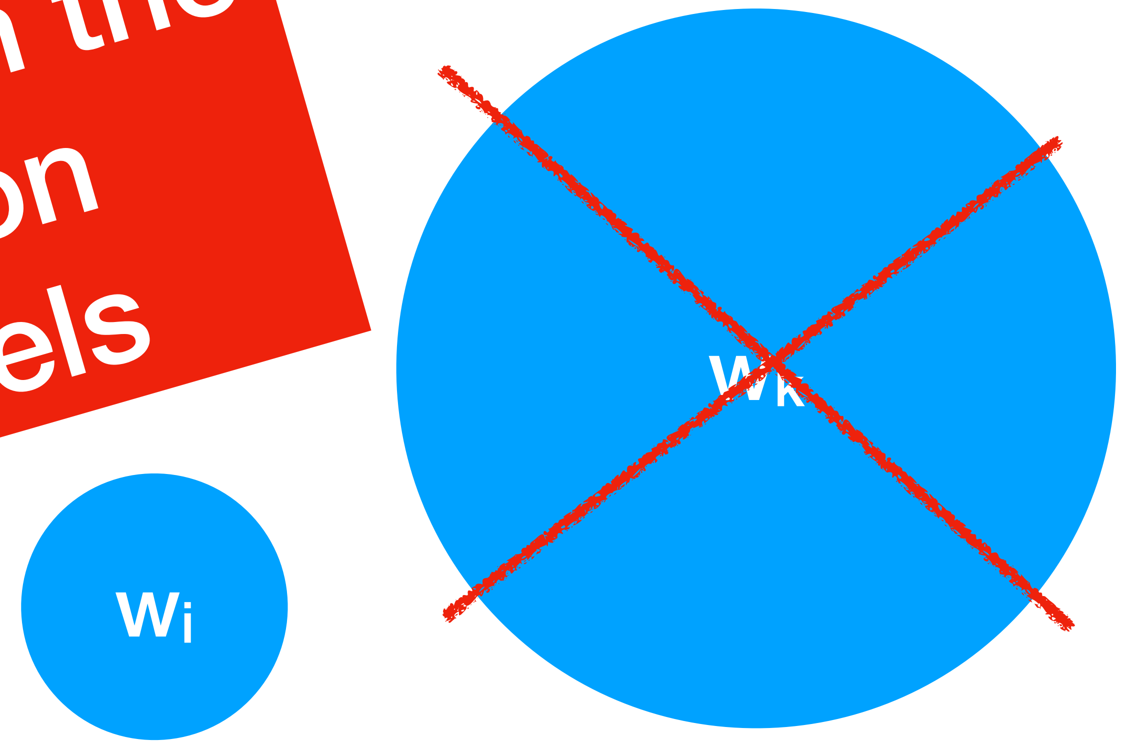
x0	x1	x2	x3	x4	x5
1	1.06	9.2	151	54.4	1.6
2	0.89	10.3	202	57.9	2.2
3	[illegible]	15.4	[illegible]	53	3.4
4	[illegible]	11.2	[illegible]	56	0.3
5	[illegible]	8.8	[illegible]	51.2	1
6	[illegible]	13.5	[illegible]	60	-2.2
7	[illegible]	12.2	[illegible]	67.6	2.2
8	1.1	9.2	245	57	3.3
9	1.34	13	168	60.4	7.2
10	1.12	12.4	197	53	2.7
11	0.75	7.5	173	51.5	6.5
12	1.13	10.9	178	62	3.7
13	1.15	12.7	199	53.7	6.4
14	1.09	12	96	49.8	1.4
15	0.96	7.6	164	62.2	-0.1
16	1.16	9.9	252	56	9.2
17	0.76	6.4	136	61.9	9
18	1.05	12.6	150	56.7	2.7
19	1.16	11.7	104	54	-2.1
20	1.2	11.8	148	59.9	[illegible]
21	1.04	8.6	204	61	
22	1.07	9.3	174	54.3	

You use fewer dimensions/features



Model enforces sparsity (mostly 0s)

Will be clearer in the discussion on Linear Models



Model forces feature weights to be small

This Module's Learning Objectives

Part 2

- Define regularization and its use of improving generalizability
- Use linear regression models to predict scores for data elements
- Explain how linear regression can be adapted for classification
- Extract the class-label probabilities for model outputs

Recall from last time…

x_i => ith row in $\boldsymbol{X}$

x_{ij} => jth feature of the ith row in $\boldsymbol{X}$

$\boldsymbol{X}$ is an n x d matrix, where d is the number of features

$$\underbrace{\text{mass} \quad \text{width} \quad \text{height} \quad \text{color\_score}}_{X} \quad \underbrace{\text{fruit\_label}}_{y}$$

mass	width	height	color_score	fruit_label
192	8.4	7.3	0.55	1
180	8.0	6.8	0.59	1
176	7.4	7.2	0.60	1
86	6.2	4.7	0.80	2
84	6.0	4.6	0.79	2
80	5.8	4.3	0.77	2
80	5.9	4.3	0.81	2
76	5.8	4.0	0.81	2
178	7.1	7.8	0.92	1
172	7.4	7.0	0.89	1
166	6.9	7.3	0.93	1
172	7.1	7.6	0.92	1
154	7.0	7.1	0.88	1
164	7.3	7.7	0.70	1
152	7.6	7.3	0.69	1
156	7.7	7.1	0.69	1
156	7.6	7.5	0.67	1
168	7.5	7.6	0.73	1
162	7.5	7.1	0.83	1
162	7.4	7.2	0.85	1

Linear Regression in simplest form

Linear regression is the process of finding these weights w_j such that…

Function of weights w times a row in X plus some constant value

$$= wx_i + b | x_i \in X, y_i \in y$$

We minimize this error

Actual Value: y_i , Predicted Value: $\hat{y}_i$

$$min(y_i - \hat{y}_i)$$

Linear Regression: Assumes relationship between X and y is <u>linear</u>

May (likely) not be true

Linear Regression is “easy” to interpret

Feature weights w_j tell you how much that feature contributes to the outcome

$$y_i = \sum_{j \in d} w_j x_{i,j} + b$$

`sklearn.linear_model`.LinearRegression

class sklearn.linear_model.LinearRegression(*, *fit_intercept=True*, *normalize='deprecated'*, *copy_X=True*, *n_jobs=None*, *positive=False*) [source]

Ordinary least squares Linear Regression.

LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Parameters:

fit_intercept : *bool, default=True*
Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

normalize : *bool, default=False*
This parameter is ignored when `fit_intercept` is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use `StandardScaler` before calling `fit` on an estimator with `normalize=False`.

Deprecated since version 1.0: `normalize` was deprecated in version 1.0 and will be removed in 1.2.

copy_X : *bool, default=True*
If True, X will be copied; else, it may be overwritten.

n_jobs : *int, default=None*
The number of jobs to use for the computation. This will only provide speedup in case of sufficiently large problems, that is if firstly `n_targets > 1` and secondly `X` is sparse or if `positive` is set to `True`. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

positive : *bool, default=False*
When set to `True`, forces the coefficients to be positive. This option is only supported for dense arrays.

New in version 0.24.

Attributes:

coef_ : *array of shape (n_features,) or (n_targets, n_features)*
Estimated coefficients for the linear regression problem. If multiple targets are passed during the fit (y 2D), this is a 2D array of shape (n_targets, n_features), while if only one target is passed, this is a 1D array of length n_features.

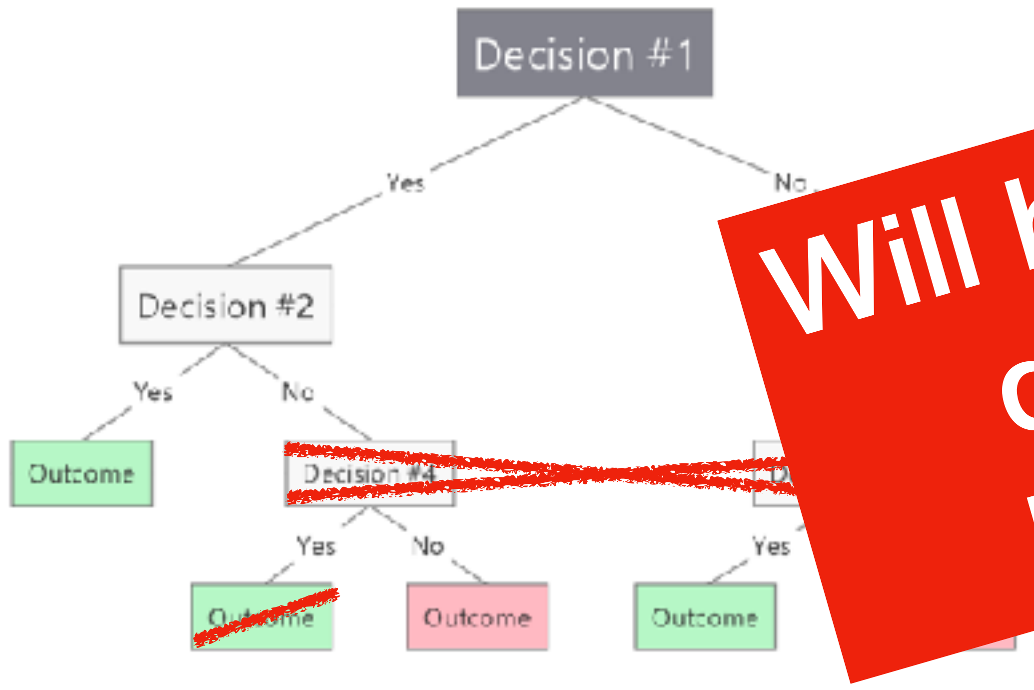
Feature weights w_j tell you how much that feature contributes to the outcome

Regularization: Methods for penalizing model complexity

How might you penalize model complexity?

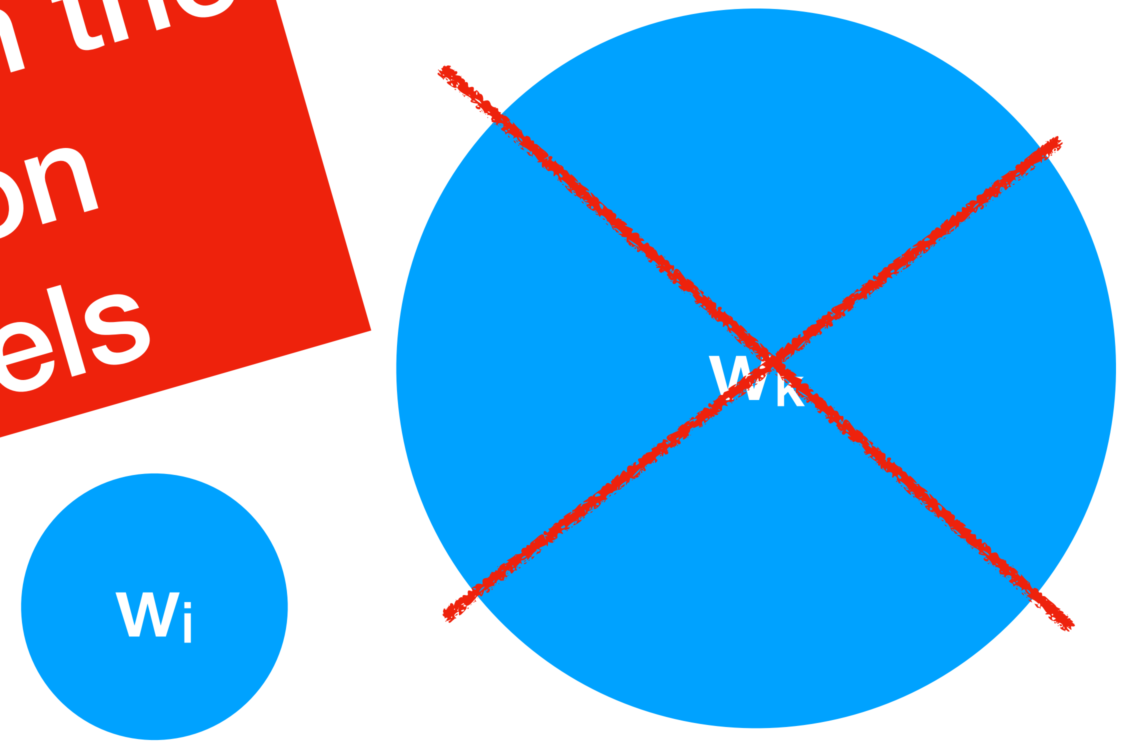
x0	x1	x2	x3	x4	x5
1	[illegible]	9.2	151	54.4	1.6
2	[illegible]	10.3	[illegible]	57.9	2.2
3	[illegible]	15.4	[illegible]	53	3.4
4	[illegible]	11.2	[illegible]	56	0.3
5	[illegible]	8.8	[illegible]	51.2	1
6	[illegible]	13.5	[illegible]	60	-2.2
7	[illegible]	12.2	[illegible]	67.6	2.2
8	1.1	9.2	[illegible]	57	3.3
9	1.34	13	168	60.4	7.2
10	1.12	12.4	197	53	2.7
11	0.75	7.5	173	51.5	6.5
12	1.13	10.9	178	62	3.7
13	1.15	12.7	199	53.7	6.4
14	1.09	12	96	49.8	1.4
15	0.96	7.6	164	62.2	-0.1
16	1.16	9.9	252	56	9.2
17	0.76	6.4	136	61.9	9
18	1.05	12.6	150	56.7	2.7
19	1.16	11.7	104	54	-2.1
20	1.2	11.8	148	59.9	[illegible]
21	1.04	8.6	204	61	
22	1.07	9.3	174	54.3	

You use fewer dimensions/features



Will be clearer in the discussion on Linear Models

Model enforces sparsity (mostly 0s)



Model forces feature weights to be small

This Module's Learning Objectives

Part 2

- Define regularization and its use of improving generalizability
- Use linear regression models to predict scores for data elements
- Explain how linear regression can be adapted for classification
- Extract the class-label probabilities for model outputs

“Transform” Linear Regression for Classification

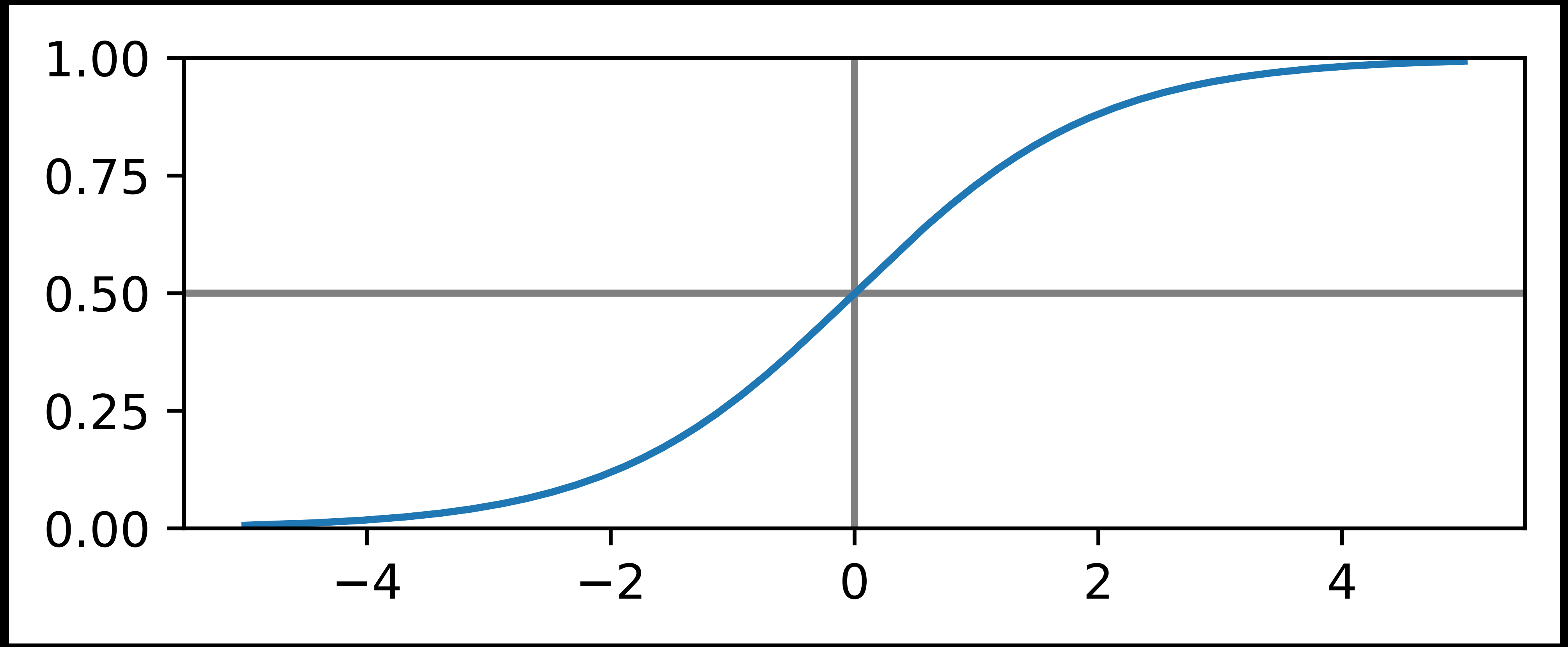
- Regression: $X \rightarrow y, y \in (-\infty, +\infty)$
- Binary classification: $X \rightarrow y, y \in \{0,1\}$
- Can we construct a mapping of $(-\infty, +\infty) \rightarrow \{0,1\}$?

"Transform" Linear Regression for Classification

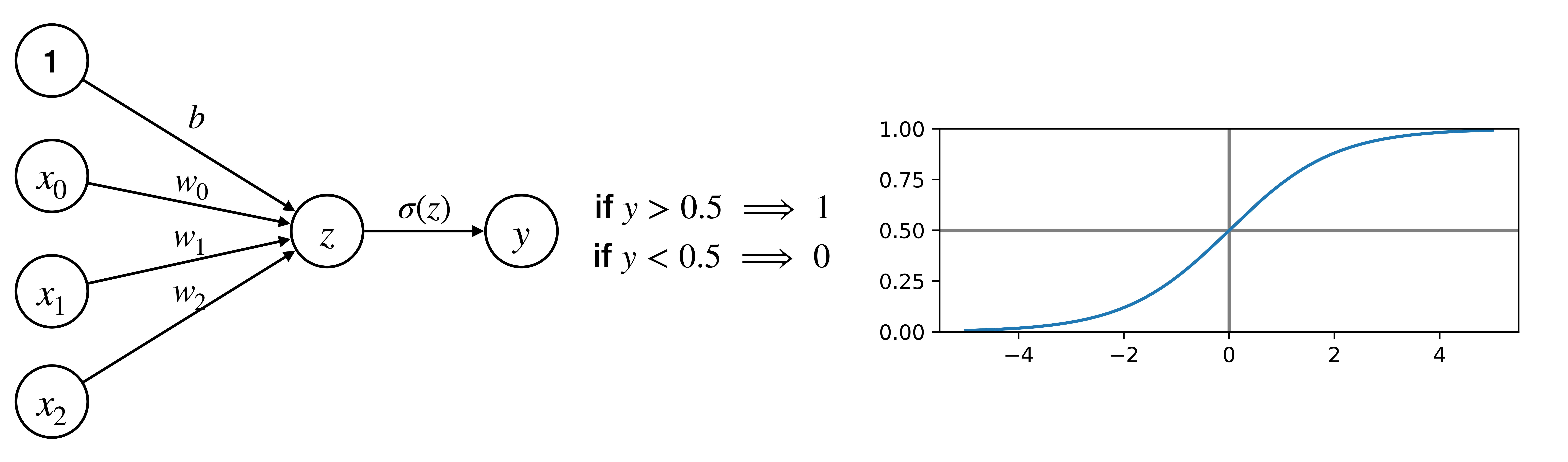
- Can we construct a mapping of $(-\infty, +\infty) \rightarrow \{0,1\}$?

Sigmoid function

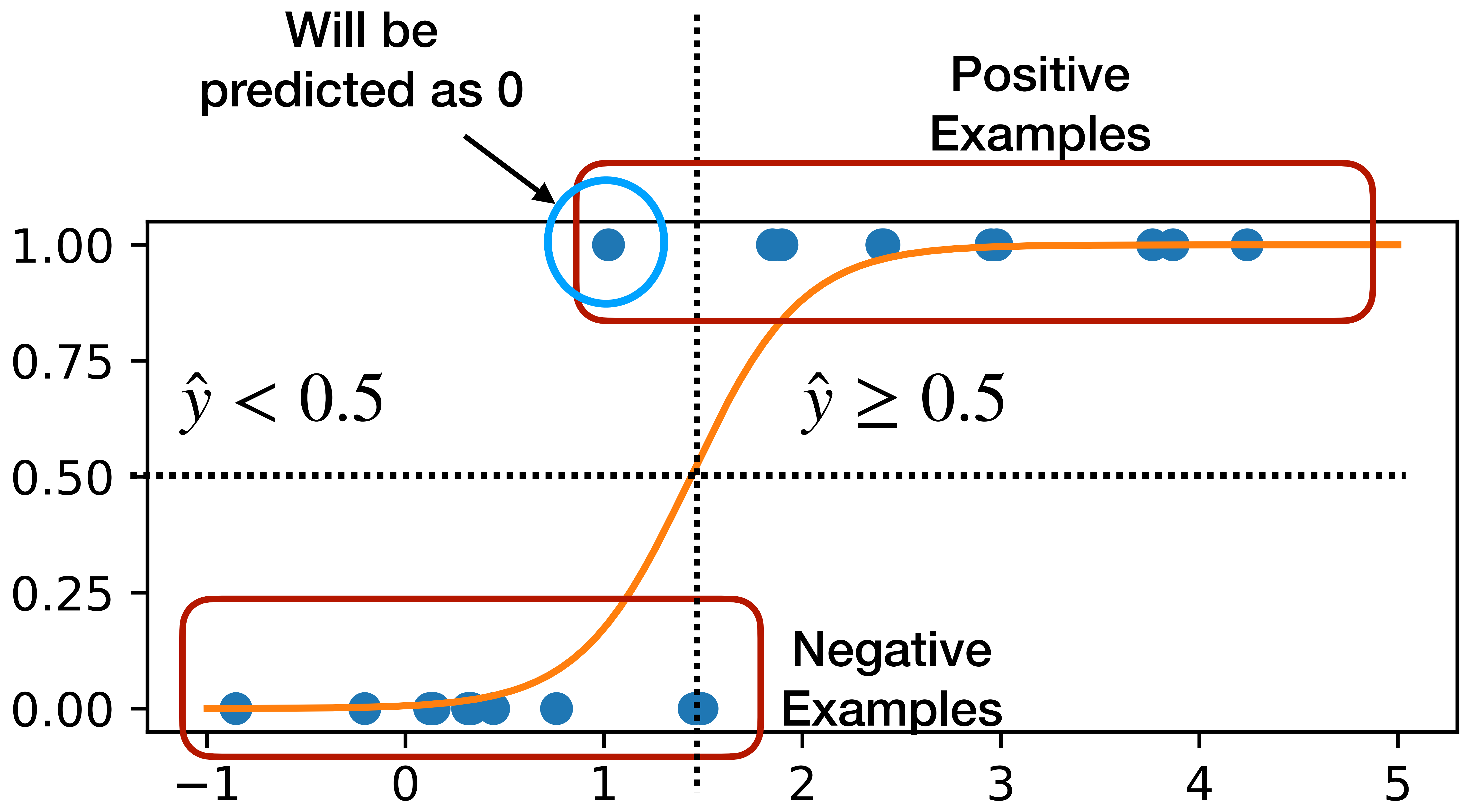
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



“Transform” Linear Regression for Classification



Logistic Regression: 1-D Example

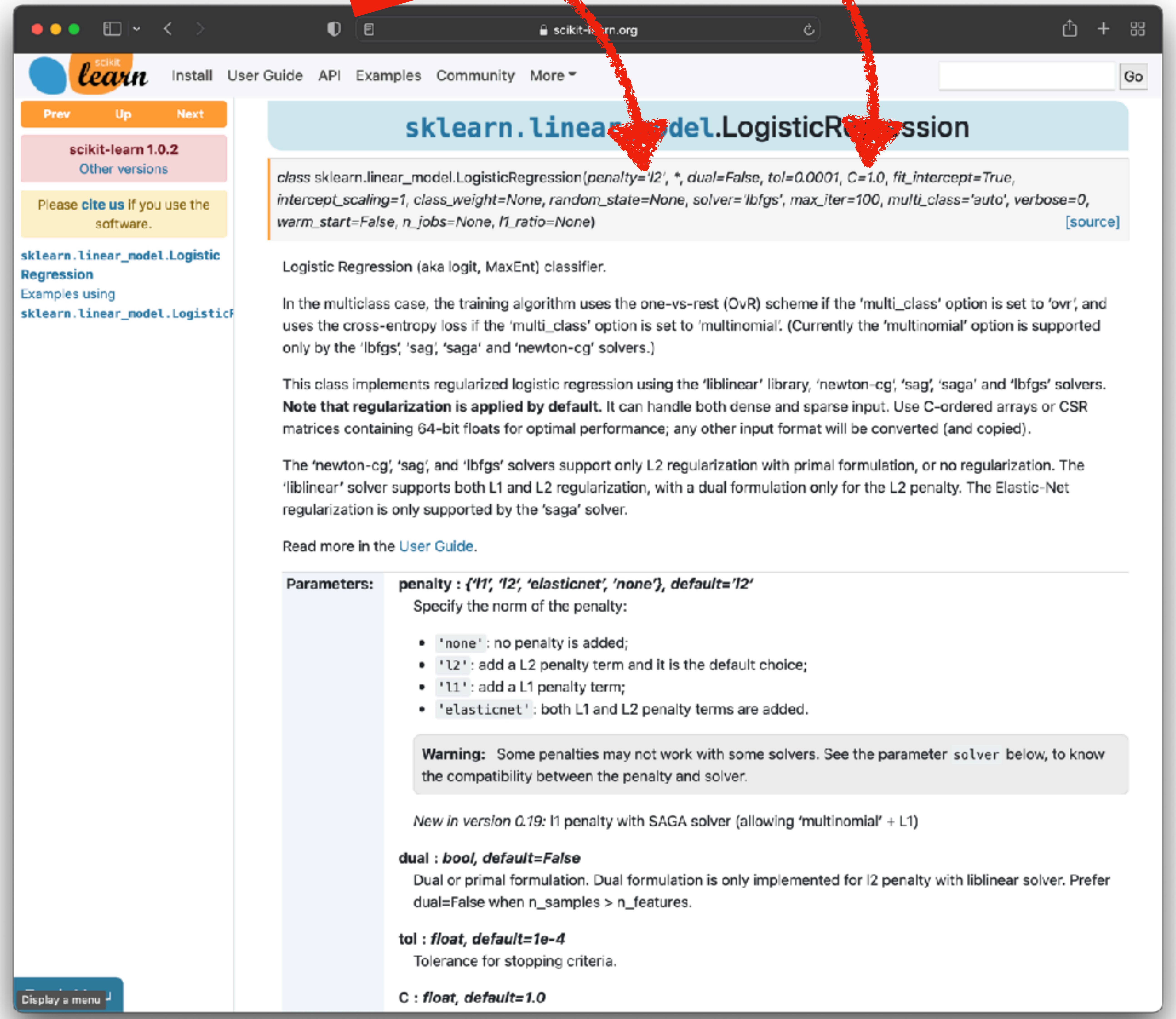


Logistic Regression: Regularization

L2 Reg/ Penalty

Lambda : c

- L2 regularization is "on" by default in scikit-learn
  - Parameter C controls the strength of the regularization.



scikit-learn.org

Install User Guide API Examples Community More

Prev Up Next

scikit-learn 1.0.2
Other versions

Please **cite us** if you use the software.

sklearn.linear_model.LogisticRegression
Examples using sklearn.linear_model.LogisticR

sklearn.linear_model.LogisticRegression

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='lbfgs'*, *max_iter=100*, *multi_class='auto'*, *verbose=0*, *warm_start=False*, *n_jobs=None*, *l1_ratio=None*) [source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default.** It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation, or no regularization. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty. The Elastic-Net regularization is only supported by the 'saga' solver.

Read more in the User Guide.

Parameters:

penalty : *{'l1', 'l2', 'elasticnet', 'none'}, default='l2'*
Specify the norm of the penalty:

- `'none'`: no penalty is added;
- `'l2'`: add a L2 penalty term and it is the default choice;
- `'l1'`: add a L1 penalty term;
- `'elasticnet'`: both L1 and L2 penalty terms are added.

Warning: Some penalties may not work with some solvers. See the parameter `solver` below, to know the compatibility between the penalty and solver.

New in version 0.19: l1 penalty with SAGA solver (allowing 'multinomial' + L1)

dual : *bool, default=False*
Dual or primal formulation. Dual formulation is only implemented for l2 penalty with liblinear solver. Prefer dual=False when n_samples > n_features.

tol : *float, default=1e-4*
Tolerance for stopping criteria.

C : *float, default=1.0*

Display a menu

This Module's Learning Objectives

Part 2

Define regularization and its use of improving generalizability

Use linear regression models to predict scores for data elements

Explain how linear regression can be adapted for classification

Extract the class-label probabilities for model outputs

This Module's Learning Objectives

Part 2

Define regularization and its use of improving generalizability

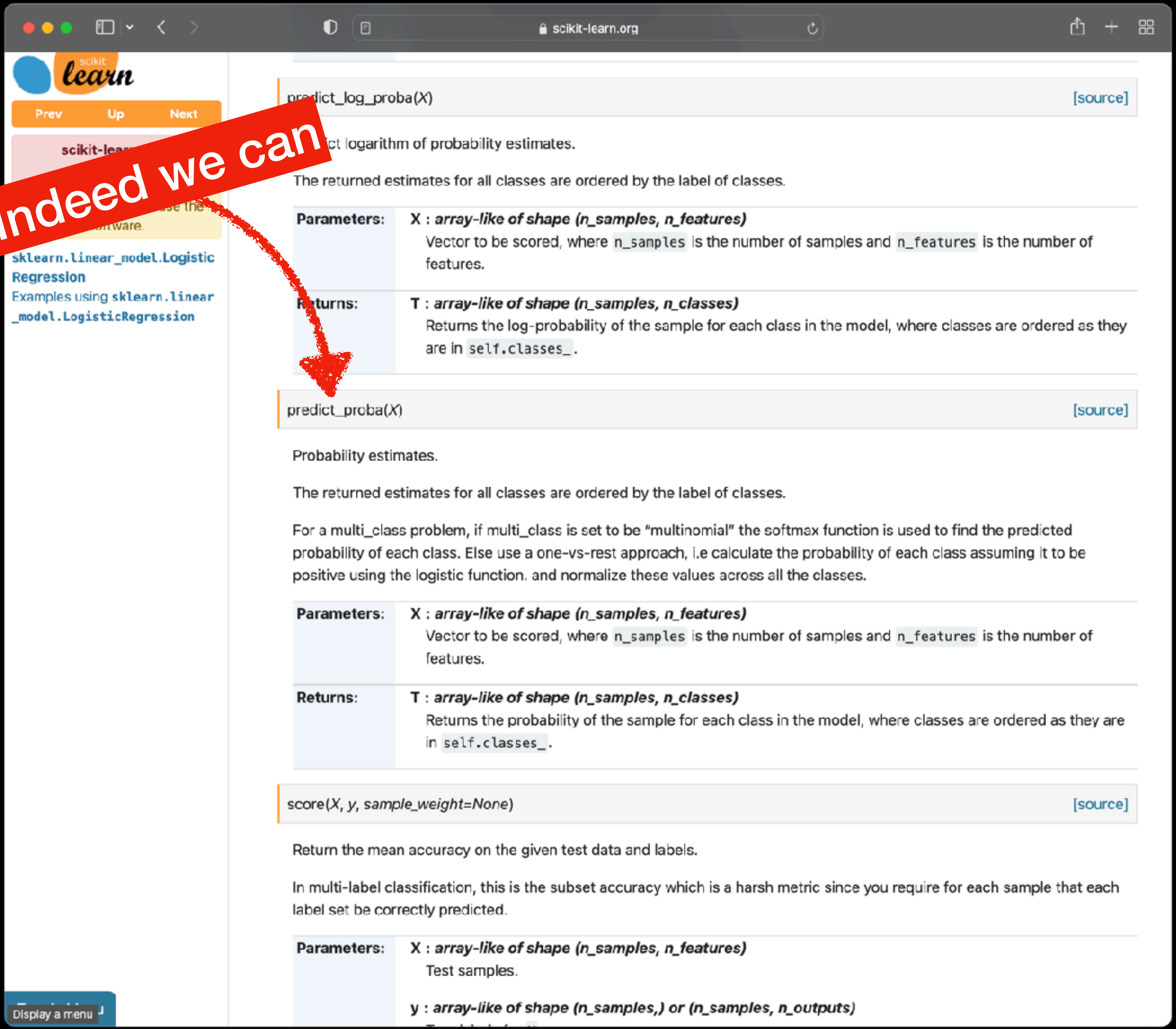
Use linear regression models to predict scores for data elements

Explain how linear regression can be adapted for classification

Extract the class-label probabilities for model outputs

If Logistic Regression classifies labels via probability thresholds…

Can we extract these probabilities directly?


scikit-learn.org

scikit learn

Prev Up Next

scikit-lear

se the
ftware.

sklearn.linear_model.Logistic Regression

Examples using **sklearn.linear_model.LogisticRegression**

Display a menu

Indeed we can

predict_log_proba(*X*) [source]

Predict logarithm of probability estimates.

The returned estimates for all classes are ordered by the label of classes.

Parameters:	**X : *array-like of shape (n_samples, n_features)*** Vector to be scored, where `n_samples` is the number of samples and `n_features` is the number of features.
Returns:	**T : *array-like of shape (n_samples, n_classes)*** Returns the log-probability of the sample for each class in the model, where classes are ordered as they are in `self.classes_`.

predict_proba(*X*) [source]

Probability estimates.

The returned estimates for all classes are ordered by the label of classes.

For a multi_class problem, if multi_class is set to be "multinomial" the softmax function is used to find the predicted probability of each class. Else use a one-vs-rest approach, i.e calculate the probability of each class assuming it to be positive using the logistic function. and normalize these values across all the classes.

Parameters:	**X : *array-like of shape (n_samples, n_features)*** Vector to be scored, where `n_samples` is the number of samples and `n_features` is the number of features.
Returns:	**T : *array-like of shape (n_samples, n_classes)*** Returns the probability of the sample for each class in the model, where classes are ordered as they are in `self.classes_`.

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

Return the mean accuracy on the given test data and labels.

In multi-label classification, this is the subset accuracy which is a harsh metric since you require for each sample that each label set be correctly predicted.

Parameters:	**X : *array-like of shape (n_samples, n_features)*** Test samples. **y : *array-like of shape (n_samples,) or (n_samples, n_outputs)***

Why would you want these probabilities?

Perhaps you only care about instances with high confidence

Instances with low confidence (prob. ~ 0.5) are interesting/hard samples

How else might we calculate these probabilities?

Train many slightly different models and assess predictions

Premise for “random forest” classification method

This Module's Learning Objectives

Part 2

Define regularization and its use of improving generalizability

Use linear regression models to predict scores for data elements

Explain how linear regression can be adapted for classification

Extract the class-label probabilities for model outputs

Questions?

Prof. Cody Buntain | @codybuntain | cbuntain@umd.edu
Director, Information Ecosystems Lab