

INSA ROUEN NORMANDIE

Machine Learning

Linear Models

ITI 4 · Benoit Gaüzère

Learning objectives

By the end of this lecture, you should be able to:

- formulate a linear regression problem from data
- derive the normal equations for ordinary least squares
- explain why Ridge stabilizes coefficient estimates
- explain why Lasso can produce sparse coefficients
- derive logistic loss from a Bernoulli model
- distinguish probabilities, scores and class decisions.

Part 1: Linear regression and regularization

- Ordinary least squares
- Ridge regression
- Lasso regression
- Elastic Net

Part 1

Linear regression, Ridge and Lasso

A scientific regression problem

Suppose that we describe each sample by measured features and want to predict a continuous quantity.

$$\text{measurements } \mathbf{x}_i \longrightarrow \text{quantity } y_i \in \mathbb{R}$$

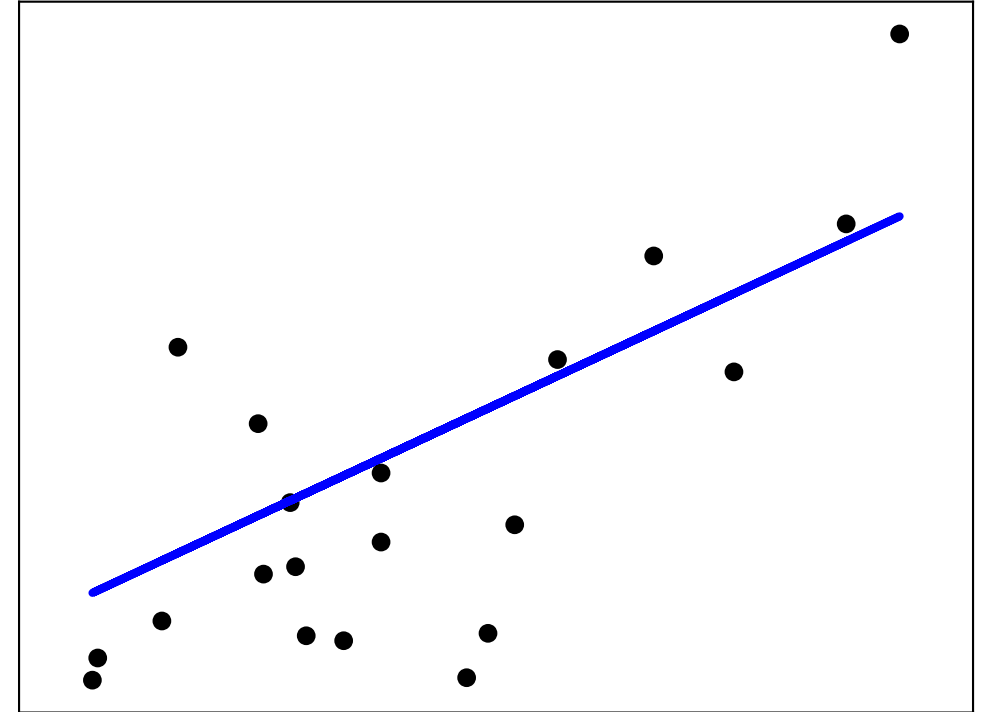
Example: predict the boiling point of a molecule from chemical characteristics.

A model with one input

For one measurement x_i , an affine model predicts

$$\hat{y}_i = wx_i + b.$$

- w controls the slope
- b controls the intercept
- w and b are parameters estimated from the observations.



Predictions and residuals

For observation i :

$$\hat{y}_i = wx_i + b, \quad r_i = y_i - \hat{y}_i.$$

The residual is the vertical difference between the observed and predicted responses.

$$r_i > 0 \Rightarrow \hat{y}_i < y_i, \quad r_i < 0 \Rightarrow \hat{y}_i > y_i.$$

Squared-error criterion

Ordinary least squares minimizes

$$J(w, b) = \frac{1}{2n} \sum_{i=1}^n (y_i - wx_i - b)^2.$$

- squaring prevents positive and negative residuals from cancelling
- large residuals are more penalized
- averaging makes datasets of different sizes comparable
- the factor $1/2$ simplifies derivatives.

The estimator

The fitted parameters minimize the criterion:

$$(\hat{w}, \hat{b}) = \arg \min_{w, b} J(w, b).$$

`min` returns the smallest value of J .

`argmin` returns the parameters that produce this value.

Several measurements per sample

With p features:

$$\mathbf{x}_i = [x_{i1} \quad \cdots \quad x_{ip}]^\top,$$
$$\hat{y}_i = b + w_1 x_{i1} + \cdots + w_p x_{ip} = \mathbf{w}^\top \mathbf{x}_i + b.$$

The model remains linear in its fitted coefficients.

Design matrix

Stack the samples by rows:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} \in \mathbb{R}^{n \times p}.$$

Then

$$\hat{\mathbf{y}} = X\mathbf{w} + b\mathbf{1}_n.$$

For the next derivations, assume that X and $\mathbf{y}$ are centered. Then $b = 0$.

Otherwise, add a column of ones to X and include b in the parameter vector.

Least squares in **matrix** form

For centered data:

$$\mathbf{r}(\mathbf{w}) = \mathbf{y} - X\mathbf{w}.$$

The scalar sum of squared residuals becomes

$$J(\mathbf{w}) = \frac{1}{2n} \|\mathbf{y} - X\mathbf{w}\|_2^2.$$

Gradient of the criterion

Using

$$\|\mathbf{y} - X\mathbf{w}\|_2^2 = (\mathbf{y} - X\mathbf{w})^\top (\mathbf{y} - X\mathbf{w}),$$

we obtain

$$\nabla J(\mathbf{w}) = \frac{1}{n} X^\top (X\mathbf{w} - \mathbf{y}).$$

Useful reference: Petersen & Pedersen, The Matrix Cookbook, Sec. 2.4.

Normal equations

At a minimum of this convex quadratic function:

$$\nabla J(\hat{\mathbf{w}}) = 0.$$

Therefore

$$X^\top X \hat{\mathbf{w}} = X^\top \mathbf{y}.$$

These are the **normal equations**.

They form a linear system $A\hat{\mathbf{w}} = \mathbf{c}$, with

$$A = X^\top X \in \mathbb{R}^{p \times p}, \quad \hat{\mathbf{w}} \in \mathbb{R}^p, \quad \mathbf{c} = X^\top \mathbf{y} \in \mathbb{R}^p.$$

Closed-form expression

If $X^\top X$ is invertible:

$$\hat{\mathbf{w}}_{\text{OLS}} = (X^\top X)^{-1} X^\top \mathbf{y}.$$

This expression characterizes the estimator.

In numerical code, use a solver rather than computing the inverse explicitly.

```
w = np.linalg.solve(X.T@X, X.T@y)
```

Geometric interpretation

The fitted vector belongs to the space spanned by the columns of X :

$$\hat{\mathbf{y}} = X\hat{\mathbf{w}} \in \text{Col}(X).$$

The normal equations imply

$$X^{\top}(\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{0}.$$

The residual vector $\mathbf{y} - \hat{\mathbf{y}}$ is orthogonal to every column of X . Thus $\hat{\mathbf{y}}$ is the orthogonal projection of $\mathbf{y}$ onto $\text{Col}(X)$.

Residual analysis

Residuals can reveal structure that the model failed to represent.

Residual pattern	Possible interpretation
no visible structure	linear mean may be adequate
curved structure	missing nonlinear relation
increasing spread	non-constant noise variance
isolated large residual	atypical observation or missing variable

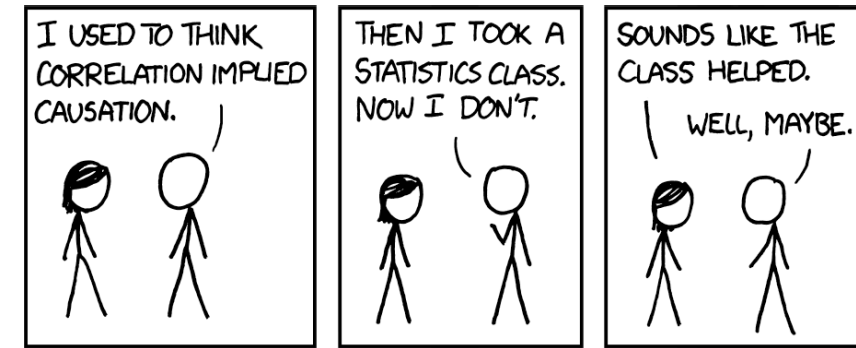
What OLS does and does not guarantee

OLS minimizes the **training** squared error among linear predictors.

It does not guarantee:

- stable coefficients
- good predictions on new samples
- a causal interpretation of the coefficients (correlation is not causality)

Regularization adds control over the magnitude of the coefficients.



Ridge regression

Ridge adds a penalty on coefficient magnitude:

$$J_{\text{Ridge}}(\mathbf{w}) = \frac{1}{2n} \|\mathbf{y} - X\mathbf{w}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2.$$

The intercept is not penalized. We continue to assume centered data.

λ is a **hyperparameter**.

Ridge solution

The first-order condition is

$$\frac{1}{n} X^\top (X \hat{\mathbf{w}}_\lambda - \mathbf{y}) + \lambda \hat{\mathbf{w}}_\lambda = 0.$$

Hence

$$\hat{\mathbf{w}}_\lambda = (X^\top X + n\lambda I)^{-1} X^\top \mathbf{y}.$$

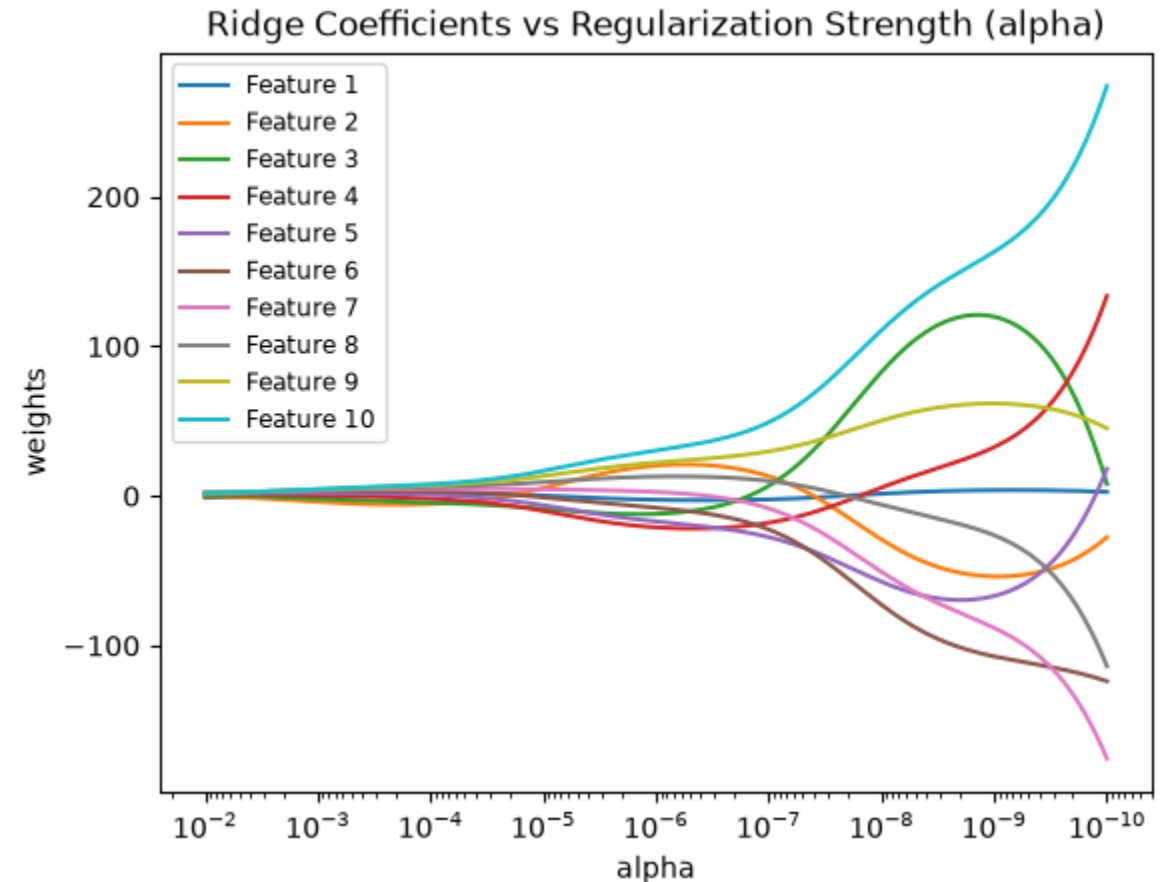
For $\lambda > 0$, the solution is unique.

Ridge regularization path

Each curve represents one component of $\hat{\mathbf{w}}_\lambda$.

- left: strong regularization, coefficients are close to zero
- right: $\lambda \rightarrow 0$, Ridge approaches OLS and unstable directions reappear

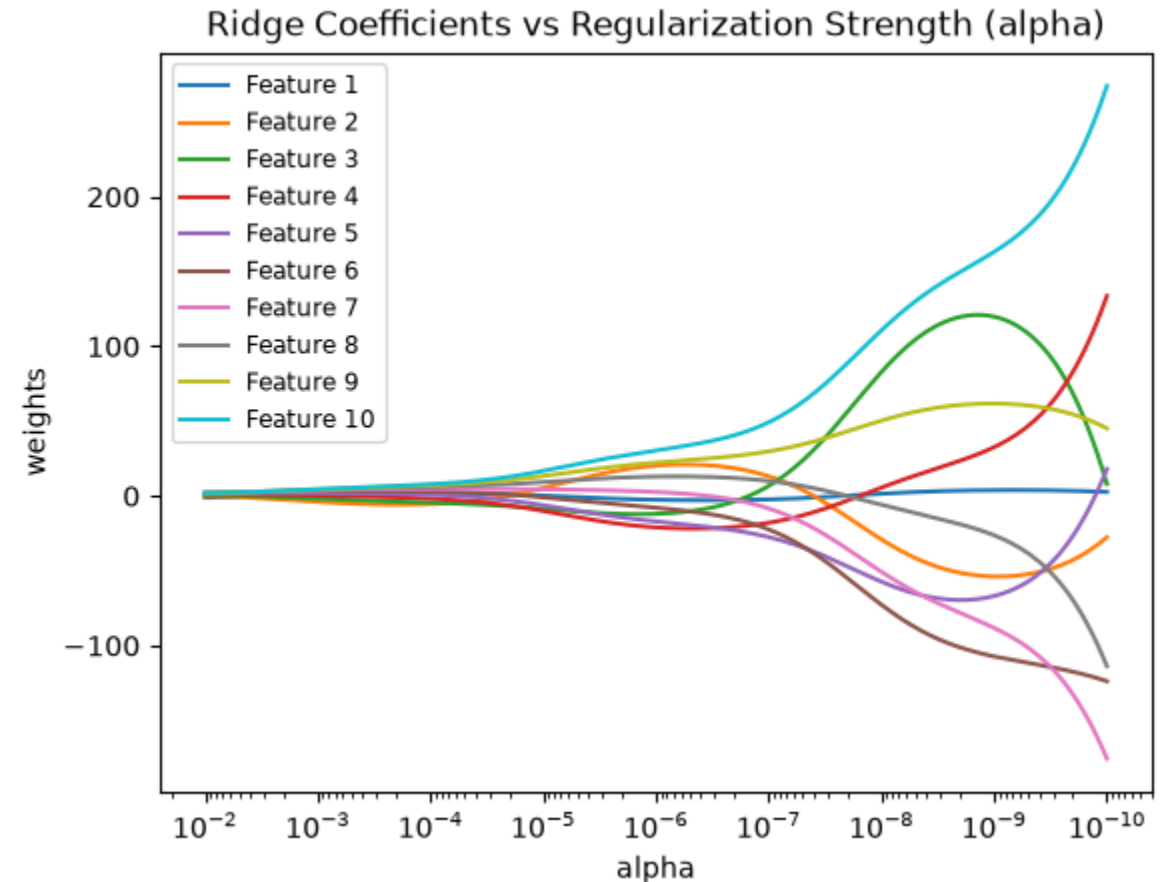
Source and reproducible example: [scikit-learn](#).
Course script: `generate_ridge_path.py`.



Effect of the Ridge parameter

Regularization	Expected behavior
$\lambda = 0$	ordinary least squares
small λ	weak coefficient shrinkage
large λ	strong coefficient shrinkage
$\lambda \rightarrow \infty$	penalized coefficients approach zero

Ridge usually keeps every coefficient nonzero.



From coefficient stability to prediction stability

Consider two fits obtained from slightly different training samples:

$$\delta \hat{\mathbf{w}} = \hat{\mathbf{w}}^{(1)} - \hat{\mathbf{w}}^{(2)}.$$

For a new input $\mathbf{x}$, the change in prediction $\delta \hat{y}(\mathbf{x}) = \mathbf{x}^\top \hat{\mathbf{w}}^{(1)} - \mathbf{x}^\top \hat{\mathbf{w}}^{(2)}$ is bounded by

$$\|\delta \hat{y}(\mathbf{x})\|_2 = \|\mathbf{x}^\top \delta \hat{\mathbf{w}}\|_2 \leq \|\mathbf{x}\|_2 \|\delta \hat{\mathbf{w}}\|_2.$$

By reducing variations in the coefficients, Ridge often reduces variations in predictions.

Bias and variance: repeated training samples

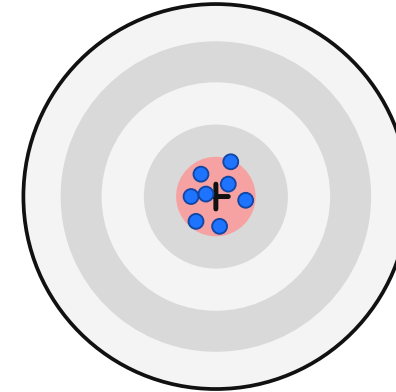
Let $\hat{f}_{\mathcal{D}}(\mathbf{x})$ be the model fitted on a random training set $\mathcal{D}$.

$$\text{Bias}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(\mathbf{x})] - f^*(\mathbf{x}),$$

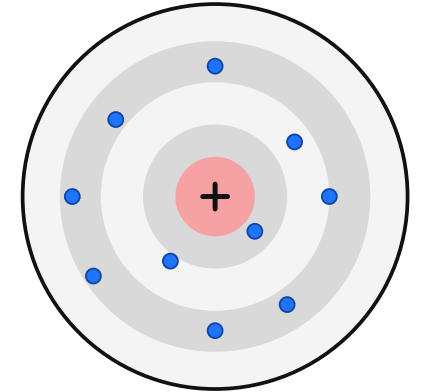
$$\text{Var}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left[(\hat{f}_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(\mathbf{x})])^2\right].$$

- **bias**: systematic displacement from the target function
- **variance**: sensitivity to the particular training sample

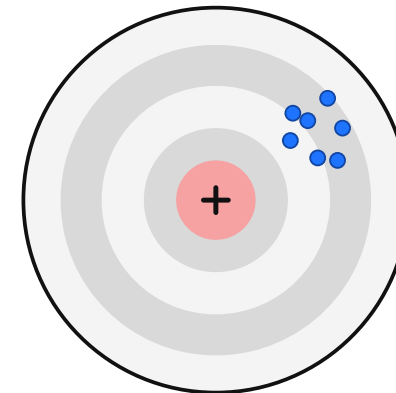
Low bias / low variance



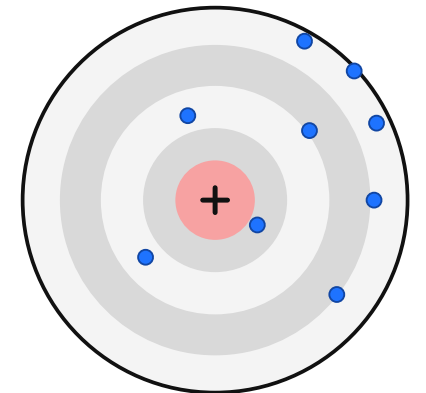
Low bias / high variance



High bias / low variance



High bias / high variance



Ridge: a bias–variance trade-off

If $Y = f^*(\mathbf{x}) + \varepsilon$, with noise variance σ^2 , then

$$\mathbb{E}_{\mathcal{D}, Y} \left[(Y - \hat{f}_{\mathcal{D}}(\mathbf{x}))^2 \right] = \underbrace{\sigma^2}_{\text{noise}} + \underbrace{\text{Bias}(\mathbf{x})^2}_{\text{squared bias}} + \underbrace{\text{Var}(\mathbf{x})}_{\text{variance}}.$$

λ	Regularization bias	Variance	Typical behavior
0	none	potentially high	OLS may be unstable
moderate	some	often lower	prediction may improve
very large	high	low	underfitting

The best predictive model need **not** have the smallest training error.

Hygiene mentale youtube video

Lasso regression

Lasso uses an L_1 penalty on $\mathbf{w}$:

$$J_{\text{Lasso}}(\mathbf{w}) = \frac{1}{2n} \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1,$$

where

$$\|\mathbf{w}\|_1 = \sum_{j=1}^p |w_j|.$$

How is Lasso fitted?

There is no closed-form matrix expression in the general case.

`sklearn.linear_model.Lasso` uses **coordinate descent**:

1. update one coefficient while keeping the others fixed
2. cycle through the coefficients
3. stop when the optimization tolerance is reached.

The optimization details are outside the scope of this lecture.

scikit-learn documentation: Lasso

Two penalties

Ridge	Lasso
$\frac{\lambda}{2} \sum_j w_j^2$	$\lambda \sum_j w_j $
smooth at zero	corner at zero
shrinks coefficients	shrinks and may set coefficients to zero

The shape of the function changes the structure of the solution.

Penalized and constrained formulations

Penalized problem:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) + \lambda \Omega(\mathbf{w}).$$

Corresponding constrained problem:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \quad \text{subject to} \quad \Omega(\mathbf{w}) \leq t.$$

Any penalized solution $\hat{\mathbf{w}}_\lambda$ also solves the constrained problem with $t = \Omega(\hat{\mathbf{w}}_\lambda)$.

For convex problems, constrained solutions can generally be associated with a penalty parameter λ . Larger λ typically corresponds to a smaller radius t .

Shapes of the L_2 and L_1 balls

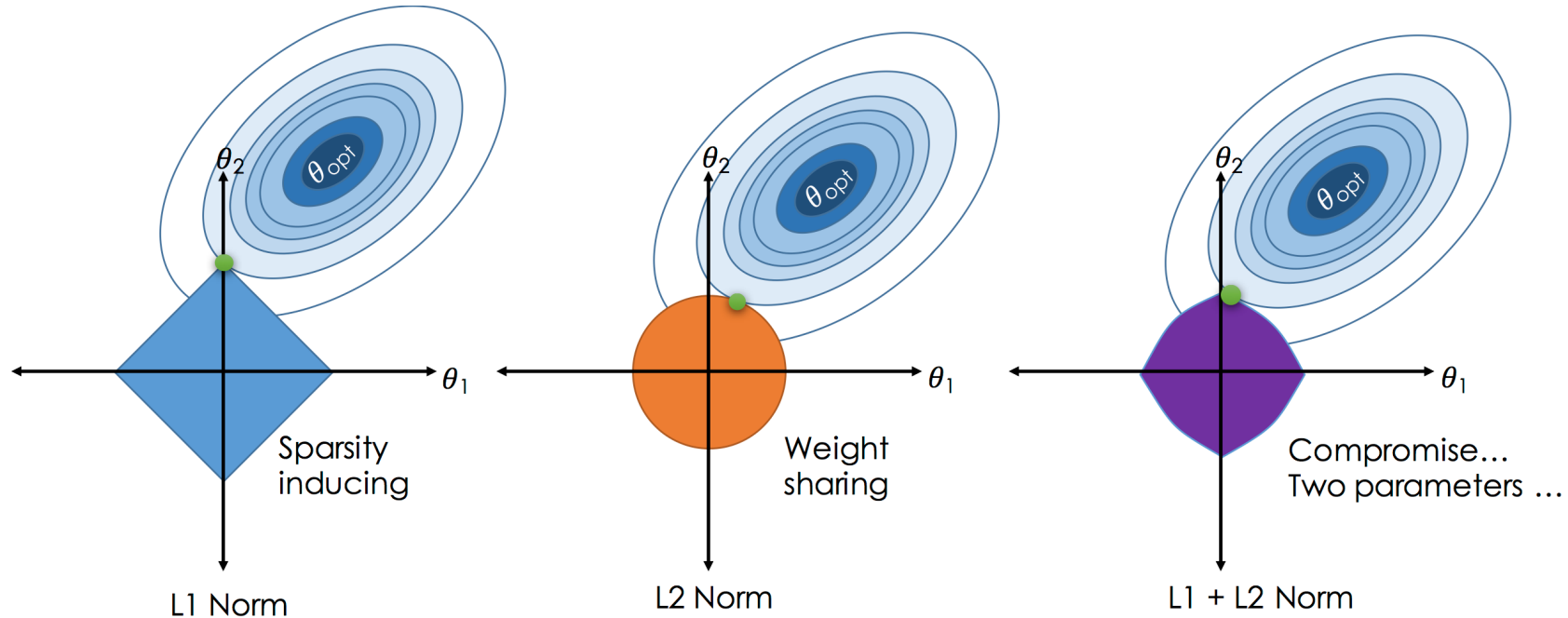
With two coefficients $\mathbf{w} = (w_1, w_2)$:

$\mathcal{B}_2(t) = \{\mathbf{w} : w_1^2 + w_2^2 \leq t^2\}$ is a disk,

$\mathcal{B}_1(t) = \{\mathbf{w} : |w_1| + |w_2| \leq t\}$ is a diamond.

- the L_2 boundary is smooth
- the L_1 boundary has corners on the coordinate axes.

Why Lasso produces zeros



The solution is the first contact between a loss ellipse and the constraint. An L_1 corner lies on an axis, so one coefficient is exactly zero.

Source figure: Li, C., Liu, J. & Qu, Z. ICCPNet: inter-layer coupling channel pruning network based on elastic scaling scoring for complex traffic scene. Appl Intell 56, 175 (2026). <https://doi.org/10.1007/s10489-026-07191-4>.

Sparsity

For a sufficiently large λ , Lasso sets some coefficients exactly to zero:

$$\hat{w}_j = 0.$$

The remaining nonzero coefficients define the active set:

$$\mathcal{A} = \{j : \hat{w}_j \neq 0\}.$$

This gives a sparse predictive model.

Lasso with correlated predictors

If several features carry similar information, Lasso may keep one and discard the others.

A small change in the sample may change which feature is selected.

Therefore:

sparse coefficients $\nRightarrow$ unique scientific mechanism.

Elastic Net

Elastic Net combines both penalties:

$$J_{\text{EN}}(\mathbf{w}) = \frac{1}{2n} \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda \left[\rho \|\mathbf{w}\|_1 + \frac{1 - \rho}{2} \|\mathbf{w}\|_2^2 \right],$$

with $0 \leq \rho \leq 1$.

- $\rho = 1$: Lasso
- $\rho = 0$: Ridge, up to the regularization convention
- intermediate values combine sparsity and stability

Elastic Net is useful when predictors are numerous and correlated.

Check-up 1: Ridge or Lasso?

Choose the most natural starting point:

1. Many correlated measurements should contribute jointly.
2. A sparse set of predictors is required.
3. Stable prediction matters more than variable selection.

Check-up 1: answer

1. **Ridge** tends to distribute weight across correlated predictors.
2. **Lasso** can produce an exactly sparse coefficient vector.
3. **Ridge** is often more stable when predictors are strongly correlated.

The final choice must still be validated on the target prediction problem.

Selecting the regularization strength

The parameter λ is a hyperparameter. It must be selected from validation data, typically by cross-validation on the development set.

For every candidate value:

1. fit the scaler and model on each training fold
2. evaluate the pipeline on the corresponding validation fold
3. aggregate the validation scores
4. select λ without using the test set.

Regression models in scikit-learn

```
from sklearn.linear_model import Ridge
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

model = Pipeline([
    ("scale", StandardScaler()),
    ("regressor", Ridge(alpha=1.0)),
])
```

`alpha` controls the Ridge regularization strength in scikit-learn. Its numerical scale depends on the loss convention used by the implementation.

[scikit-learn: linear models](#)

OLS, Ridge and Lasso

Property	OLS	Ridge	Lasso
minimizes training squared error	yes	no	no
shrinks coefficients	no	yes	yes
produces exact zeros	no	usually no	yes
regularization parameter	none	λ	λ
standardization before fitting	optional	recommended	recommended

Part 2

Logistic regression

Binary classification

We observe labels

$$y_i \in \{0, 1\}.$$

Logistic regression estimates the conditional probability

$$p(\mathbf{x}) = P(Y = 1 \mid \mathbf{x}),$$

then a decision rule converts this probability into a class.

$$p(\mathbf{x}) \xrightarrow{\text{threshold } \tau} \hat{y}_\tau(\mathbf{x}).$$

A linear score

Logistic regression starts with

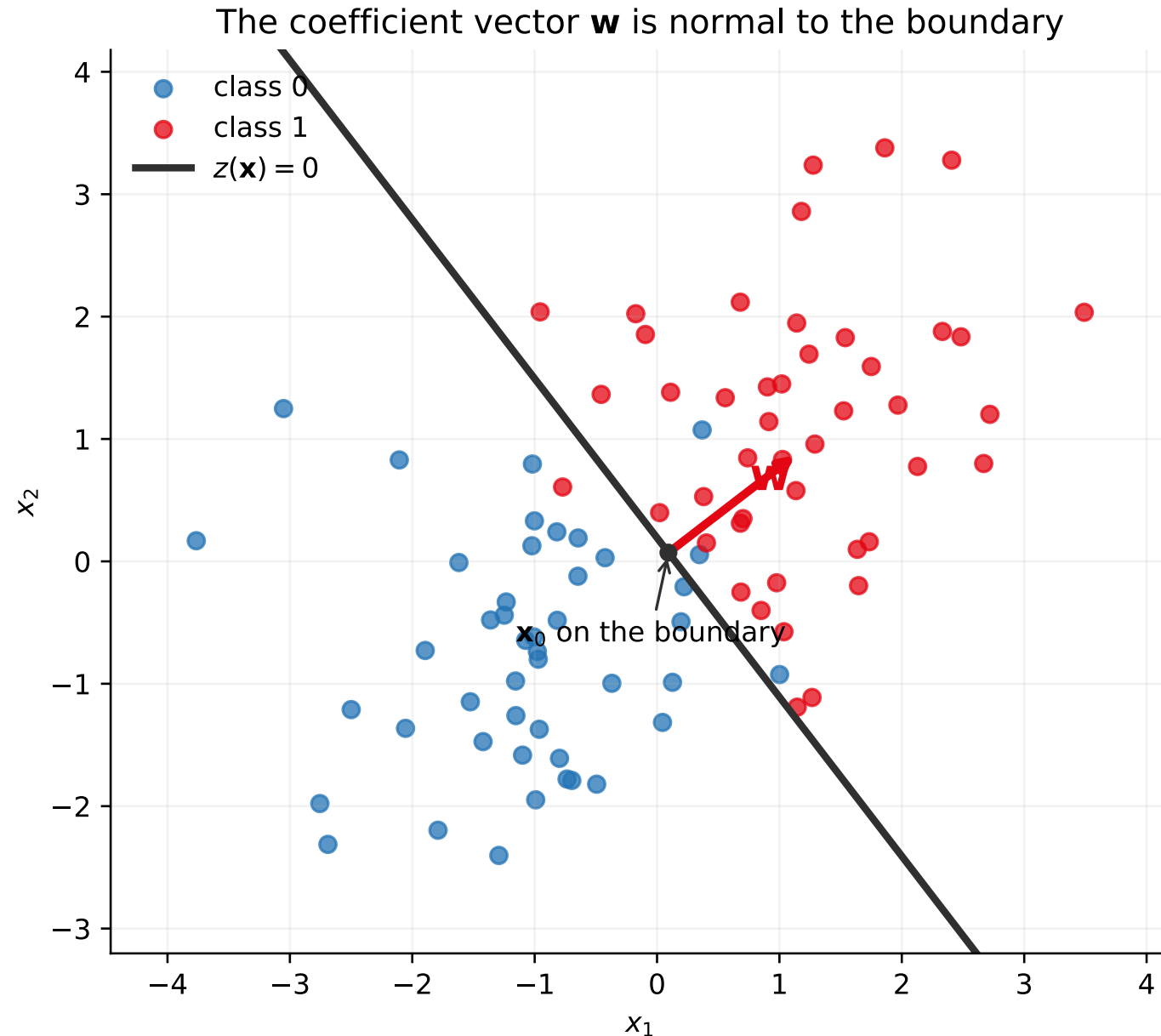
$$z(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b.$$

In two dimensions, the boundary is

$$w_1x_1 + w_2x_2 + b = 0 \iff x_2 = -\frac{w_1x_1 + b}{w_2}.$$

```
w = model.coef_[0]
b = model.intercept_[0]
x2 = -(w[0] * x1 + b) / w[1]
```

$\mathbf{w}$ is normal to the boundary and points toward increasing scores.



Score and geometric distance

Let $\mathbf{x}_0$ be the orthogonal projection of $\mathbf{x}$ onto the hyperplane. Since $\mathbf{w}$ is normal to it,

$$d(\mathbf{x}) = \frac{\mathbf{w}^\top (\mathbf{x} - \mathbf{x}_0)}{\|\mathbf{w}\|_2}.$$

Because $\mathbf{w}^\top \mathbf{x}_0 + b = 0$,

$$d(\mathbf{x}) = \frac{\mathbf{w}^\top \mathbf{x} + b}{\|\mathbf{w}\|_2} = \frac{z(\mathbf{x})}{\|\mathbf{w}\|_2}.$$

The sign indicates the side of the hyperplane. The absolute value $|d(\mathbf{x})|$ is the Euclidean distance.

Odds and log-odds

For $p = P(Y = 1 \mid \mathbf{x})$, the **odds** of class 1 are

$$\text{odds}(p) = \frac{p}{1 - p}.$$

Taking the natural logarithm gives the **logit**:

$$\text{logit}(p) = \log \frac{p}{1 - p}.$$

p	odds	logit
0.2	1/4	$-\log 4$
0.5	1	0
0.8	4	$\log 4$

The logistic modeling assumption

The central assumption is a **linear relation between the features and the log-odds**:

$$\text{logit}(p(\mathbf{x})) = \log \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} = z(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

The probability itself is not linear in $\mathbf{x}$.

From logit to sigmoid

Invert the logit equation:

$$\begin{aligned}\log \frac{p}{1-p} = z &\iff \frac{p}{1-p} = e^z \\ \iff p = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}} = \sigma(z).\end{aligned}$$

Therefore

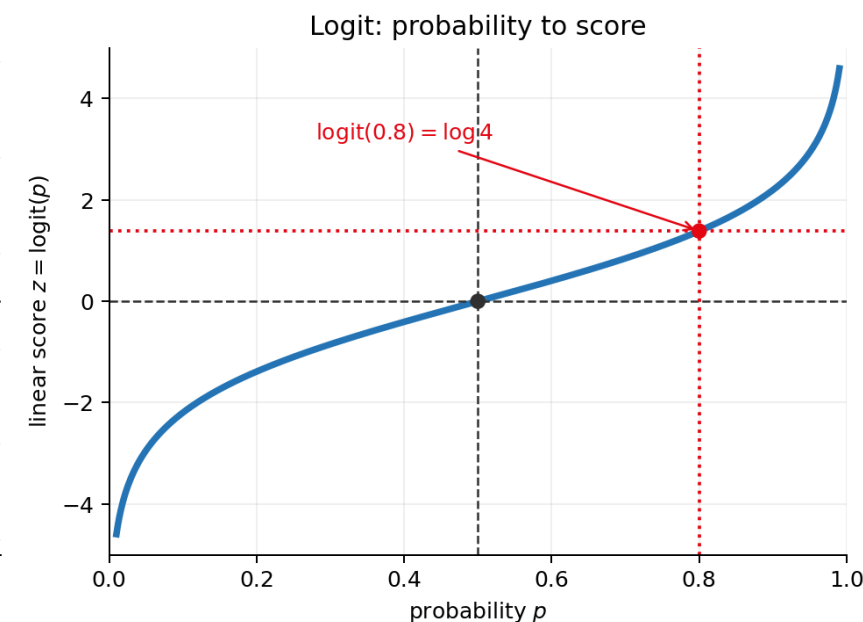
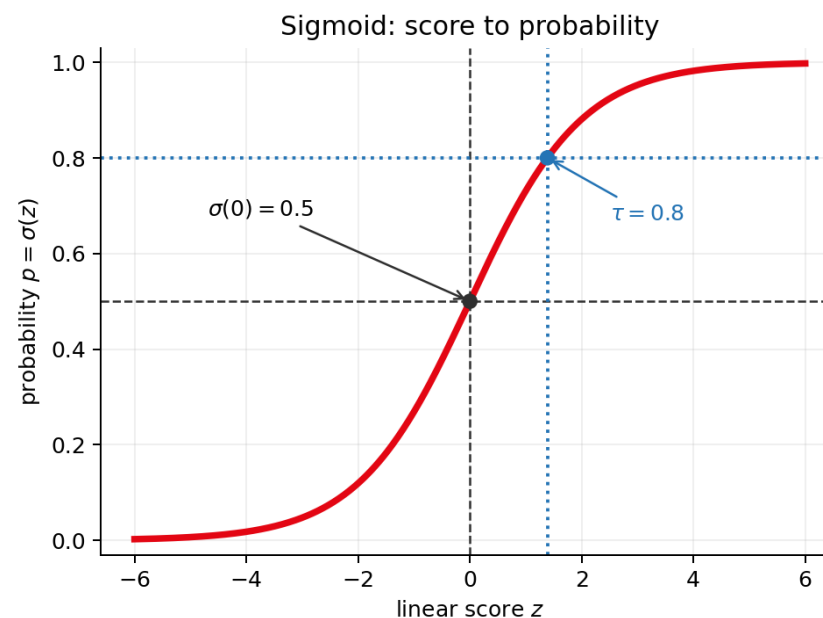
$$P(Y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b).$$

Logit and sigmoid are inverse functions

```
def sigmoid(z):  
    return 1 / (1 + np.exp(-z))
```

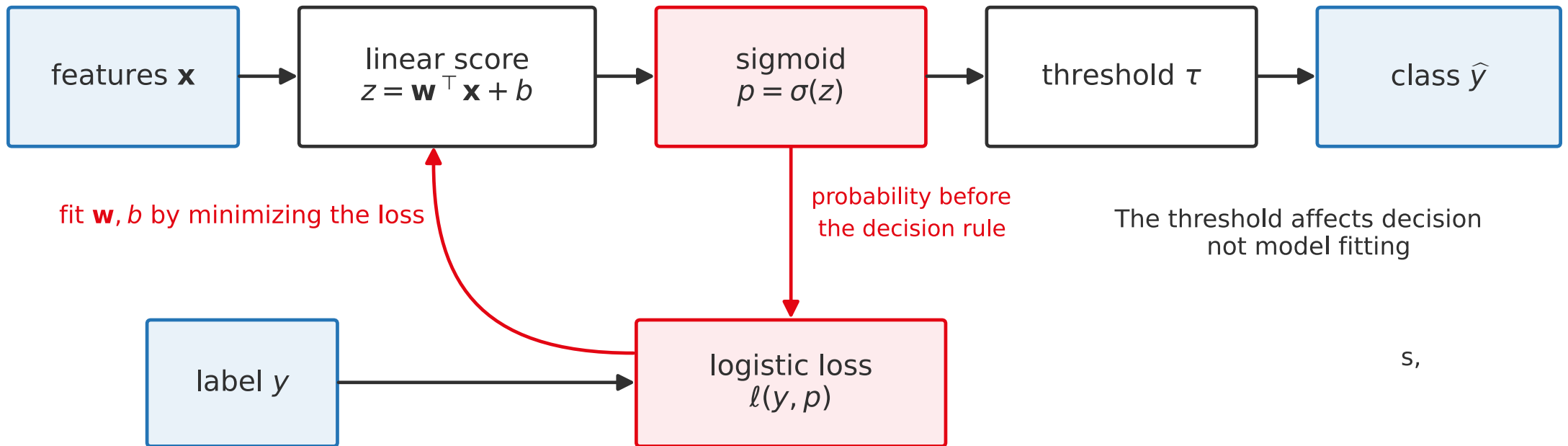
$$\sigma(0) = 0.5, \quad \sigma'(z) > 0.$$

The transformation preserves the ranking induced by z .



Logistic regression architecture

Prediction



The threshold affects decision
not model fitting

S,

Adapted from Raschka et al., *Machine Learning with PyTorch and Scikit-Learn*, Figure 3.3.

Check-up 2: score and probability

Without a calculator:

1. Associate $p(\mathbf{x})$ and $z(\mathbf{x})$:

$p(\mathbf{x}) : 0.5, < 0.5, > 0.5$

$z(\mathbf{x}) : -2, 3, 0$

2. What is the score associated with $p(\mathbf{x}) = 0.8$?

Which observation lies on the default decision boundary?

Check-up 2: answer

Score	Probability
-2	below 0.5
0	exactly 0.5
3	above 0.5

For $p = 0.8$:

$$z = \text{logit}(0.8) = \log 4 \approx 1.386.$$

The observation with $z = 0$ lies on the default decision boundary.

Bernoulli model

To learn $\mathbf{w}$ and b , we need a criterion that rewards probabilities consistent with the observed labels.

Start with a probabilistic model for $Y_i \mid \mathbf{x}_i$. Let

$$p_i = P(Y_i = 1 \mid \mathbf{x}_i).$$

For $y_i \in \{0, 1\}$:

$$P(Y_i = y_i \mid \mathbf{x}_i) = p_i^{y_i} (1 - p_i)^{1-y_i}.$$

This single expression covers both possible labels.

Likelihood of the observations

Assuming conditionally independent observations:

$$\mathcal{L}(\mathbf{w}, b) = \prod_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i}.$$

The parameters should assign high probability to the observed labels.

Maximum likelihood chooses $(\mathbf{w}, b)$ that maximize $\mathcal{L}$.

Logistic loss

Maximizing the likelihood is equivalent to minimizing the negative log-likelihood:

$$J(\mathbf{w}, b) = -\frac{1}{n} \sum_{i=1}^n [y_i \log p_i + (1 - y_i) \log(1 - p_i)].$$

This is the **binary cross-entropy** or logistic loss.

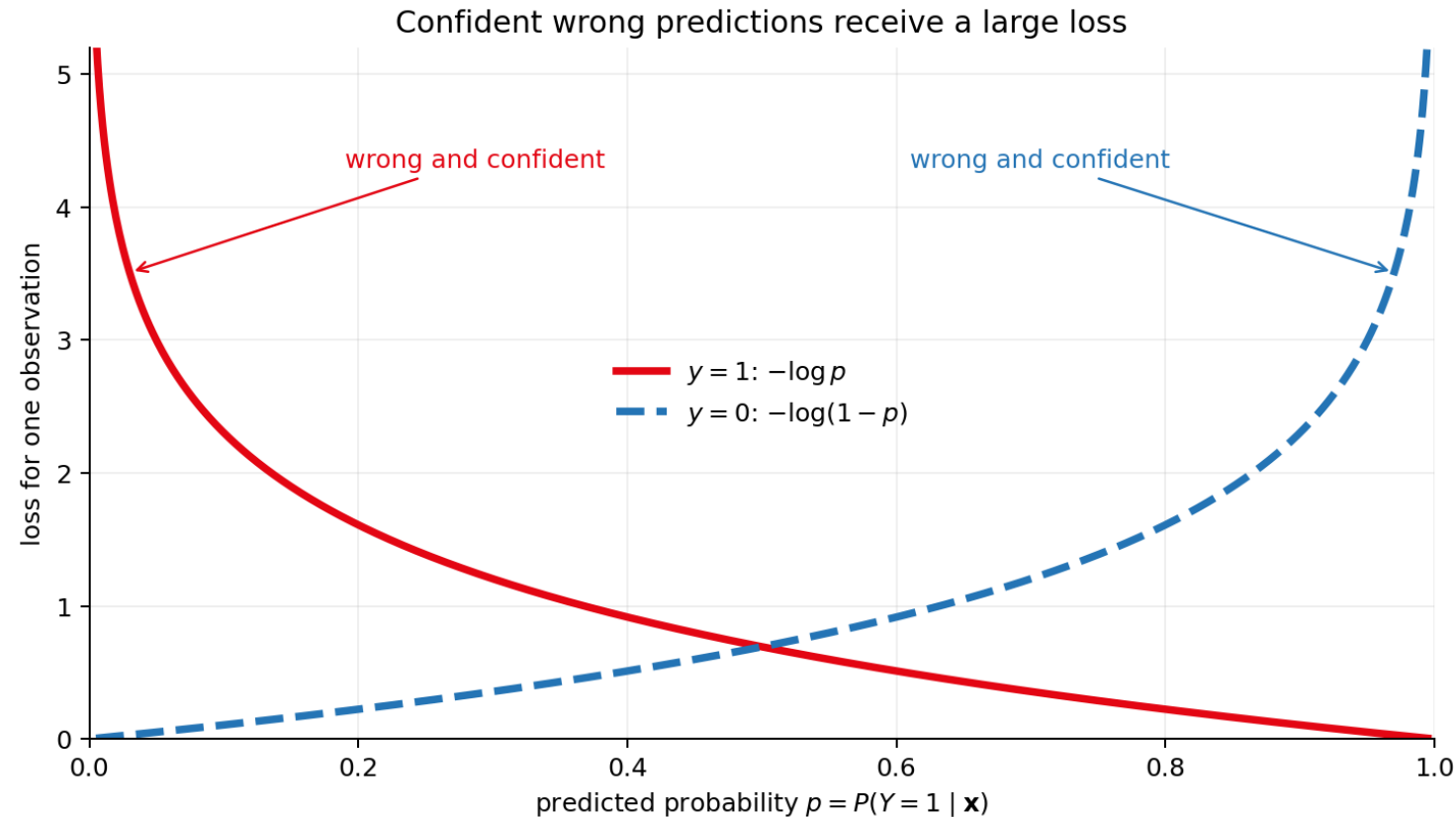
Loss for one observation

For one predicted probability p :

$$\ell(y, p) = -y \log p - (1 - y) \log(1 - p).$$

```
loss_if_one = -np.log(p)  
loss_if_zero = -np.log1p(-p)
```

A confident wrong prediction receives an increasingly large penalty.



Fitting the logistic model

The fitted parameters solve

$$(\hat{\mathbf{w}}, \hat{b}) = \arg \min_{\mathbf{w}, b} J(\mathbf{w}, b).$$

The objective is convex, but it has no general closed-form minimizer.

A numerical solver estimates the parameters.

From probability to decision

For a threshold $\tau \in (0, 1)$:

$$\hat{y}_{\tau}(\mathbf{x}) = \begin{cases} 1 & \text{if } p(\mathbf{x}) \geq \tau, \\ 0 & \text{otherwise.} \end{cases}$$

The model produces $p(\mathbf{x}) = \sigma(z(\mathbf{x}))$; the threshold produces the class.

Changing τ changes the decisions without changing the fitted model.

Threshold and decision boundary

Because the sigmoid is increasing:

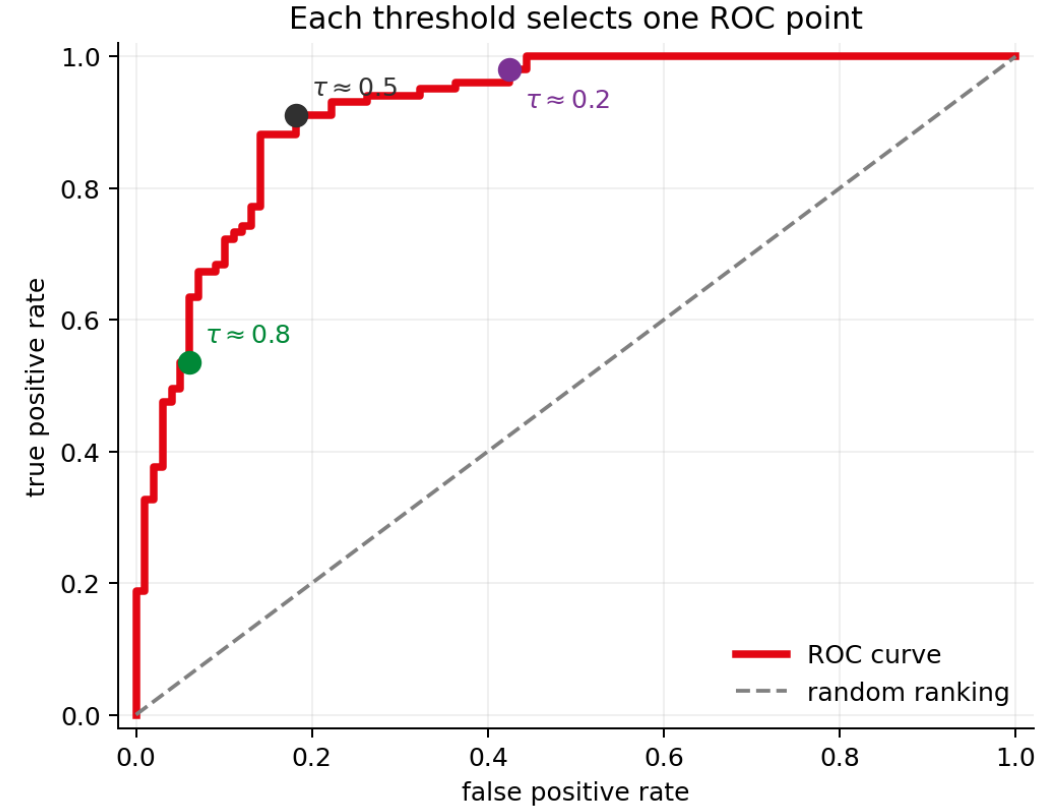
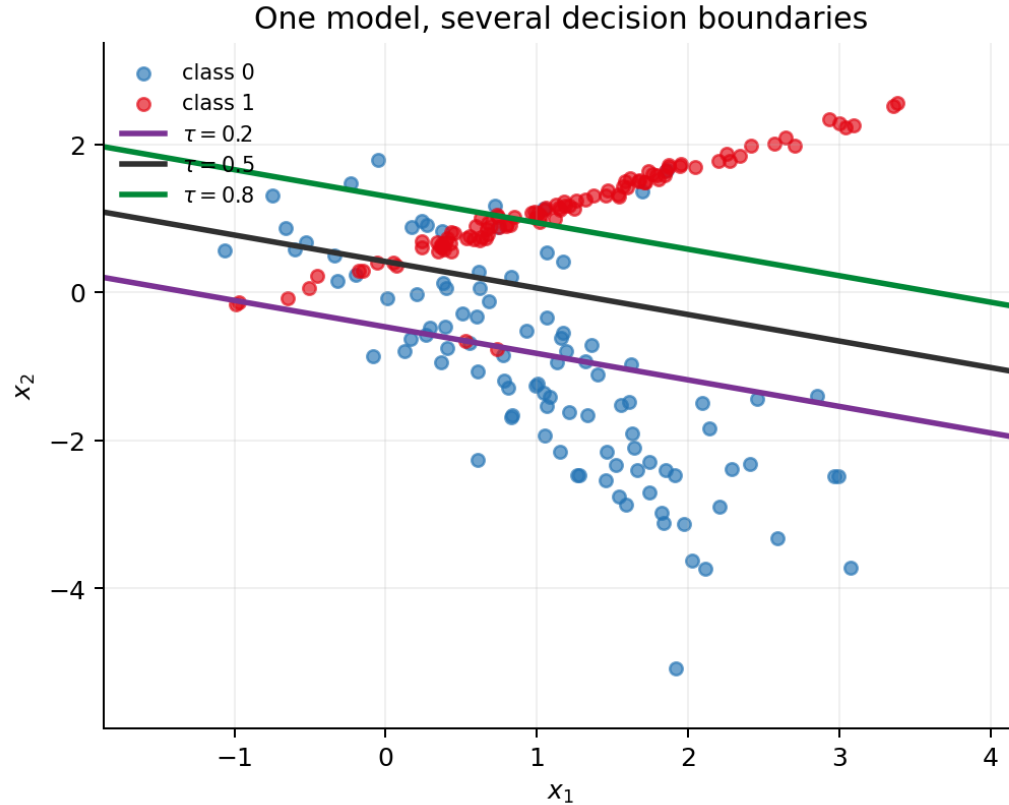
$$p(\mathbf{x}) \geq \tau \iff z(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b \geq \text{logit}(\tau) = \log \frac{\tau}{1 - \tau}.$$

The decision boundary is therefore

$$\mathbf{w}^\top \mathbf{x} + b = \text{logit}(\tau).$$

For $\tau = 0.5$, the boundary is $z(\mathbf{x}) = 0$. Changing τ moves it parallel to itself.

Thresholds and the ROC curve



The same fitted probabilities produce one (FPR, TPR) point for each threshold.

Sweeping the threshold

```
from sklearn.metrics import roc_curve  
  
fpr, tpr, thresholds = roc_curve(y_dev, p_dev)
```

- Lower τ : more positive predictions, usually higher TPR and FPR.
- Higher τ : fewer positive predictions, usually lower TPR and FPR.
- Sweeping τ traces the ROC curve.
- Select an operating threshold on development data; keep the test set untouched.

Check-up 3: changing the threshold

The threshold changes from 0.5 to 0.8.

1. Must the logistic model be refitted?
2. What score replaces the boundary $z = 0$?
3. Do positive predictions normally become more or less frequent?
4. In which direction does the corresponding ROC point usually move?

Check-up 3: answer

The fitted model remains unchanged.

The new score boundary is

$$z = \log \frac{0.8}{0.2} = \log 4 \approx 1.386.$$

Fewer samples normally reach the higher threshold, so positive predictions become less frequent.

Both TPR and FPR usually decrease: the ROC point moves down and to the left.

Regularized logistic regression

The same penalties apply to logistic loss:

$$\min_{\mathbf{w}, b} [J(\mathbf{w}, b) + \lambda \Omega(\mathbf{w})].$$

Common choices are

$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2 \quad \text{or} \quad \Omega(\mathbf{w}) = \|\mathbf{w}\|_1.$$

For scikit-learn with an L_2 penalty and uniform sample weights, an equivalent convention is

$$\min_{\mathbf{w}, b} \left[\sum_{i=1}^n \ell_i + \frac{1}{2C} \|\mathbf{w}\|_2^2 \right].$$

Thus, smaller **C** means stronger regularization.

Logistic regression in scikit-learn

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

model = Pipeline([
    ("scale", StandardScaler()),
    ("classifier", LogisticRegression(C=1.0, max_iter=1000)),
])

model.fit(X_train, y_train)
z_dev = model.decision_function(X_dev)
p_dev = model.predict_proba(X_dev)[: , 1]
```

`decision_function` returns z ; `predict_proba` returns $\sigma(z)$.

scikit-learn: `LogisticRegression`

Multiclass logistic regression

For K classes, define one score per class:

$$s_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + b_k.$$

The softmax model is

$$P(Y = k \mid \mathbf{x}) = \frac{e^{s_k(\mathbf{x})}}{\sum_{\ell=1}^K e^{s_\ell(\mathbf{x})}}.$$

The predicted class has the largest probability.

Linear and logistic regression

	Linear regression	Logistic regression
Target	$y \in \mathbb{R}$	$y \in \{0, 1\}$
Linear quantity	predicted response	log-odds
Data-fit criterion	squared loss	logistic loss
Model output	real value	probability
Regularization	Ridge or Lasso	L_2 or L_1

Main results

1. OLS minimizes the sum of squared residuals.
2. Correlated predictors can make OLS coefficients unstable.
3. Ridge stabilizes estimates by penalizing squared coefficients.
4. Lasso can produce an exactly sparse coefficient vector.
5. Logistic regression assumes $\text{logit}(p(\mathbf{x})) = \mathbf{w}^\top \mathbf{x} + b$.
6. The sigmoid transforms the linear score into a probability.
7. A threshold converts probabilities into decisions; sweeping it traces the ROC curve.

Bonus: derivative of the logistic loss

For one observation, $p_i = \sigma(z_i)$ and

$$\ell_i = -y_i \log p_i - (1 - y_i) \log(1 - p_i).$$

Using $\sigma'(z_i) = p_i(1 - p_i)$:

$$\frac{\partial \ell_i}{\partial z_i} = \frac{\partial \ell_i}{\partial p_i} \frac{\partial p_i}{\partial z_i} = \left(-\frac{y_i}{p_i} + \frac{1 - y_i}{1 - p_i} \right) p_i(1 - p_i) = p_i - y_i.$$

Bonus: gradient descent update

With $\mathbf{z} = X\mathbf{w} + b\mathbf{1}$ and $\mathbf{p} = \sigma(\mathbf{z})$:

$$\nabla_{\mathbf{w}} J = \frac{1}{n} X^\top (\mathbf{p} - \mathbf{y}), \quad \frac{\partial J}{\partial b} = \frac{1}{n} \mathbf{1}^\top (\mathbf{p} - \mathbf{y}).$$

```
z = X @ w + b
p = sigmoid(z)
grad_w = X.T @ (p - y) / len(y)
grad_b = np.mean(p - y)
w -= learning_rate * grad_w
b -= learning_rate * grad_b
```

For L_2 regularization, add $\lambda\mathbf{w}$ to $\nabla_{\mathbf{w}} J$.