

National Technical University of Athens

Machine Learning

Regression

Linear and Logistic Models

Prediction, likelihood and model fitting

Chrysoula Zerva



Learning objectives

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

- Specify a linear regression model and use feature maps to capture curved relationships.
- Fit least-squares parameters and explain the connection to Gaussian likelihood.
- Use logistic regression to estimate binary probabilities and explain Bernoulli cross-entropy.
- Evaluate fitted models and use validation data and regularization to control overfitting.
- Use gradient descent when closed-form fitting is unavailable or impractical.

The lecture at a glance

- ① **Regression setup:** targets, features and data splits.
- ② **Linear regression:** least squares, likelihood and polynomial features.
- ③ **Logistic regression:** binary probabilities, likelihood and decisions.
- ④ **Model fitting and evaluation:** numerical optimization, regularization and validation.

The target and probability model determine the prediction function and loss.

Supervised learning

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \quad \mathbf{x}_i \in \mathbb{R}^d$$

Regression

A continuous target $y \in \mathbb{R}$.

- Reading time from article features.
- Energy demand from sensor readings.

Binary classification

A discrete label $y \in \{0, 1\}$.

- Spam detection from message features.
- Equipment fault detection from readings.

Linear regression predicts a continuous value. Logistic regression models a binary outcome probability.

Algebra recap: vectors, dot products and norms

We write vectors as columns. The transpose $\mathbf{v}^\top$ is a row.

$$\mathbf{u}^\top \mathbf{v} = \sum_{j=1}^d u_j v_j \in \mathbb{R}, \quad \|\mathbf{v}\|_2^2 = \mathbf{v}^\top \mathbf{v} = \sum_{j=1}^d v_j^2.$$

Weighted sum

$$\begin{bmatrix} 2 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = 2.$$

This is how a linear model combines features.

Squared length

$$\left\| \begin{bmatrix} 3 \\ 4 \end{bmatrix} \right\|_2^2 = 25.$$

This notation lets us express a sum of squared residuals compactly.

A dot product combines two vectors into one scalar.

Features and parameters

$$\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_d \end{bmatrix}^\top, \quad z = \mathbf{w}^\top \mathbf{x} + b$$

- **Features** describe an example: length, temperature, counts, indicators.
- **Weights** $\mathbf{w} \in \mathbb{R}^d$ determine how each feature affects the score.
- The **bias** $b \in \mathbb{R}$ sets the score when all features are zero.

$$\underbrace{z}_{\text{score}} = \underbrace{b}_{\text{bias}} + \underbrace{\sum_{j=1}^d w_j x_j}_{\text{weighted features}}$$

Discuss: Which quantities are supplied by the data, and which do we learn?

Training, validation and testing

Subset	Purpose
Training	Fit model parameters.
Validation	Choose model complexity, regularization, learning rate and stopping time.
Test	Evaluate the final model on unseen data.

Data leakage: fit preprocessing and model parameters using training data only. Validation guides model selection.

Apply the training-set scaling to validation and test data.

Splits should reflect deployment: keep related observations together, and respect time order for forecasting.

Keep the test set separate from model selection.

Linear regression

A model for continuous targets

Fitting a line

$$\hat{y} = wx + b$$

- w : change in prediction per unit of x .
- b : prediction at $x = 0$.
- The training set determines the fitted line.

Least-squares fit to illustrative data.



Residuals and squared loss

Relative to the displayed candidate line, the residual of example i is

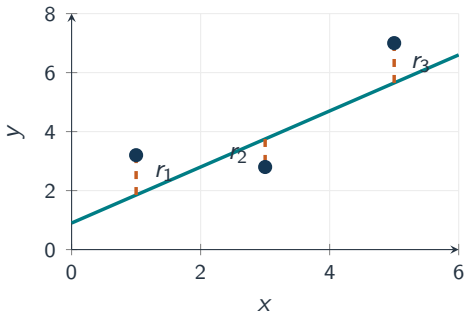
$$r_i = y_i - \hat{y}_i.$$

A common loss is

$$\ell_i = \frac{1}{2}(\hat{y}_i - y_i)^2.$$

Squaring prevents positive and negative residuals from cancelling.

Large errors receive disproportionately large penalties.



The least-squares objective

For N examples, minimize the average squared loss:

$$\mathcal{L}(w, b) = \frac{1}{2N} \sum_{i=1}^N (wx_i + b - y_i)^2.$$

- $1/N$ makes the objective an average per example.
- $1/2$ simplifies the derivative.
- Multiplying the objective by a positive constant preserves its minimizers, but scales its gradient.

Discuss: Would a sum of unsquared residuals be a useful objective?

Multiple features and the bias trick

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b.$$

Prepend a constant feature to absorb the bias:

$$\mathbf{a} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \in \mathbb{R}^m, \quad \boldsymbol{\theta} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} \in \mathbb{R}^m, \quad \hat{y} = \mathbf{a}^\top \boldsymbol{\theta}.$$

Notation for the rest of the lecture

$\boldsymbol{\theta}$ contains all trainable parameters. Each $\mathbf{a}_i$ starts with 1.

For raw features, $m = d + 1$. For a feature map $\mathbf{a}(\mathbf{x})$, m counts all mapped features, including the bias.

Algebra recap: matrix multiplication and transpose

For $A \in \mathbb{R}^{r \times k}$ and $\mathbf{v} \in \mathbb{R}^k$, $A\mathbf{v} \in \mathbb{R}^r$. Each output entry is a dot product of a row and a vector:

$$(A\mathbf{v})_i = \sum_{j=1}^k A_{ij}v_j, \quad \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} b \\ w \end{bmatrix} = \begin{bmatrix} b \\ b + 2w \end{bmatrix}.$$

Transpose rules

$$(A^\top)_{ji} = A_{ij}, \quad (AB)^\top = B^\top A^\top.$$

If A has shape $N \times m$, then $A^\top A$ has shape $m \times m$.

The inner dimensions must match. The outer dimensions give the output shape.

The design matrix

Stack examples into rows:

$$A = \begin{bmatrix} \mathbf{a}_1^\top \\ \vdots \\ \mathbf{a}_N^\top \end{bmatrix} \in \mathbb{R}^{N \times m}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N.$$

Then

$$\underbrace{\hat{\mathbf{y}}}_{N \times 1} = \underbrace{A}_{N \times m} \underbrace{\boldsymbol{\theta}}_{m \times 1}, \quad \boxed{\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2N} \|A\boldsymbol{\theta} - \mathbf{y}\|_2^2.}$$

Each row is an example. Each column is a feature.

Deriving the normal equations

For symmetric M , $\nabla_{\theta}(\theta^{\top} M \theta) = 2M\theta$. Also $\nabla_{\theta}(\mathbf{c}^{\top} \theta) = \mathbf{c}$.

$$\mathcal{L}(\theta) = \frac{1}{2N} \left(\theta^{\top} A^{\top} A \theta - 2\theta^{\top} A^{\top} \mathbf{y} + \mathbf{y}^{\top} \mathbf{y} \right),$$

Deriving the normal equations

For symmetric M , $\nabla_{\theta}(\theta^{\top} M \theta) = 2M\theta$. Also $\nabla_{\theta}(\mathbf{c}^{\top} \theta) = \mathbf{c}$.

$$\mathcal{L}(\theta) = \frac{1}{2N} \left(\theta^{\top} A^{\top} A \theta - 2\theta^{\top} A^{\top} \mathbf{y} + \mathbf{y}^{\top} \mathbf{y} \right),$$

$$\nabla_{\theta} \mathcal{L} = \frac{1}{N} A^{\top} (A\theta - \mathbf{y}).$$

Deriving the normal equations

For symmetric M , $\nabla_{\theta}(\theta^{\top} M \theta) = 2M\theta$. Also $\nabla_{\theta}(\mathbf{c}^{\top} \theta) = \mathbf{c}$.

$$\mathcal{L}(\theta) = \frac{1}{2N} \left(\theta^{\top} A^{\top} A \theta - 2\theta^{\top} A^{\top} \mathbf{y} + \mathbf{y}^{\top} \mathbf{y} \right),$$

$$\nabla_{\theta} \mathcal{L} = \frac{1}{N} A^{\top} (A\theta - \mathbf{y}).$$

$$\nabla_{\theta} \mathcal{L} = 0 \quad \Longrightarrow \quad \boxed{A^{\top} A \theta = A^{\top} \mathbf{y}.}$$

These are the **normal equations**. The objective is a convex quadratic, so every solution minimizes it.

Closed-form solution and its conditions

When a square matrix M is invertible, its inverse satisfies $M^{-1}M = MM^{-1} = I$, where I is the identity matrix.

If A has full column rank, $A^{\top}A$ is invertible:

$$\theta^* = (A^{\top}A)^{-1}A^{\top}\mathbf{y}.$$

- Full column rank means linearly independent columns. It requires $N \geq m$, but this alone is insufficient.
- If A is rank deficient, the unregularized minimizer is not unique.
- The pseudoinverse gives the minimum-norm solution $\theta^* = A^+\mathbf{y}$.

In code, solve least squares using QR or SVD rather than forming an explicit inverse.

A small least-squares example

x_i	y_i
0	1
1	2
2	2

$$\hat{y} = b + wx$$

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, \quad A^T A = \begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix}, \quad A^T \mathbf{y} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}.$$

$$\begin{cases} 3b + 3w = 5 \\ 3b + 5w = 6 \end{cases} \implies w = \frac{1}{2}, \quad b = \frac{7}{6}.$$

Predictions: $[7/6, 5/3, 13/6]^T$. Residuals: $[-1/6, 1/3, -1/6]^T$ sum to zero.

Discuss: Why do the residuals sum to zero when the model includes a bias?

Probability recap: continuous variables and Gaussian noise

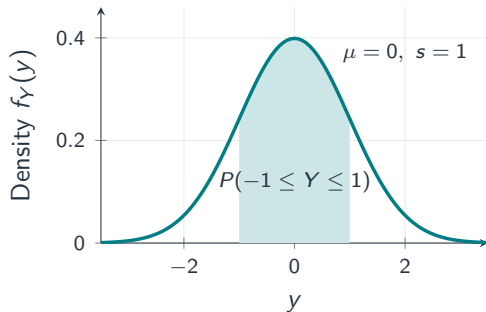
We describe a continuous target using a **density** $f_Y(y)$.

$$P(a \leq Y \leq b) = \int_a^b f_Y(y) dy.$$

The total area is 1. For a continuous target, $P(Y = y) = 0$ even where the density is positive.

For $Y \sim \mathcal{N}(\mu, s^2)$, $\mathbb{E}[Y] = \mu$ and $\text{Var}(Y) = \mathbb{E}[(Y - \mu)^2] = s^2$. The standard deviation is $s > 0$.

$$f_Y(y) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{(y - \mu)^2}{2s^2}\right).$$



Probability recap: independence, likelihood and logarithms

For observations independent conditional on their inputs and parameters,

$$\Lambda(\boldsymbol{\theta}) = \prod_{i=1}^N p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta}).$$

Here p is a probability mass for discrete targets or a density for continuous targets.

- The **likelihood** treats the observed data as fixed and varies the parameters.
- The logarithm turns a product of positive terms into a sum.

$$\log \Lambda(\boldsymbol{\theta}) = \sum_i \log p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta}).$$

Because log is increasing, maximizing likelihood and maximizing log-likelihood select the same parameters.

Minimizing negative log-likelihood gives a loss that adds over examples.

Least squares as maximum likelihood

Assume independent Gaussian noise with fixed variance $s^2 > 0$:

$$y_i = \mathbf{a}_i^\top \boldsymbol{\theta} + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, s^2).$$

Thus $p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta}) = \mathcal{N}(y_i; \mathbf{a}_i^\top \boldsymbol{\theta}, s^2)$.

$$-\log p(\mathbf{y} \mid A; \boldsymbol{\theta}) = \underbrace{\frac{N}{2} \log(2\pi s^2)}_{\text{constant in } \boldsymbol{\theta}} + \frac{1}{2s^2} \sum_i (y_i - \mathbf{a}_i^\top \boldsymbol{\theta})^2.$$

Under this model, $\mathbb{E}[Y \mid \mathbf{X} = \mathbf{x}] = \mathbf{a}(\mathbf{x})^\top \boldsymbol{\theta}$ and $\text{Var}(Y \mid \mathbf{X} = \mathbf{x}) = s^2$.

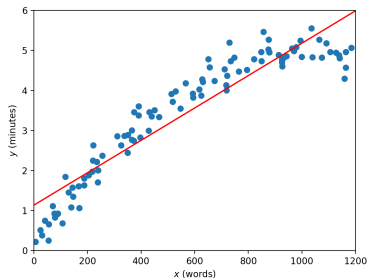
More generally, with a finite second moment, the conditional mean minimizes expected squared prediction error. Gaussian noise is unnecessary for this property.

Maximum likelihood gives least squares under this noise model.

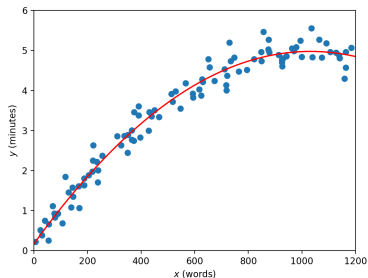
Polynomial regression is linear in its parameters

For degree 2, use $\mathbf{a}(x) = [1, x, x^2]^\top$ ($m = 3$) and $\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2$.

Degree 1: straight line



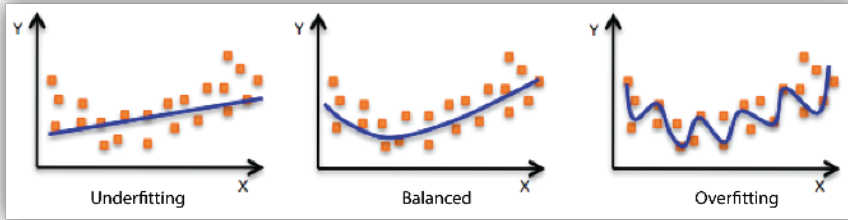
Degree 2: curved prediction



Example axes: x is article length (words), and y is reading time (min).

Any fixed features, such as x^2 , $\sin x$ or interactions, retain linearity in θ .

Model capacity and generalization



- A simple feature map can underfit.
- A flexible feature map can fit noise in a small training set.
- Validation performance helps select model capacity.

Lower training loss alone does not establish better predictions on new data.

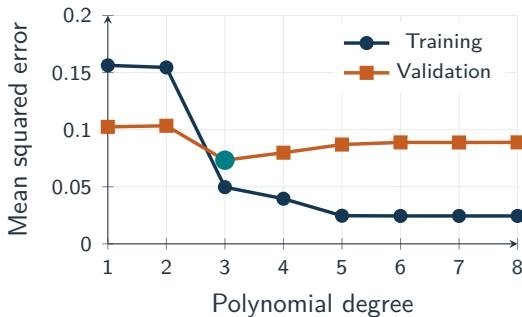
Selecting model capacity with validation data

Fit each polynomial degree on the same training set.

- Evaluate each fit on held-out validation data.
- Select the degree with the lowest validation error.
- Use the test set once the choice is fixed.

In this example, validation selects degree 3.

Synthetic noisy data. Error curves depend on the data and split.



Checkpoint: linear regression

- 1 Is $\hat{y} = \theta_0 + \theta_1 \sin(x) + \theta_2 x^2$ a linear regression model?
- 2 When does $(A^\top A)^{-1}$ exist?
- 3 Does zero training error imply zero test error?

Checkpoint: linear regression

- ❶ Is $\hat{y} = \theta_0 + \theta_1 \sin(x) + \theta_2 x^2$ a linear regression model?
- ❷ When does $(A^\top A)^{-1}$ exist?
- ❸ Does zero training error imply zero test error?

Answers

1. Yes, for fixed features.
2. When A has full column rank.
3. No. Test error depends on generalization.

Logistic regression

A regression model for binary outcomes

Why a linear score needs a probability link

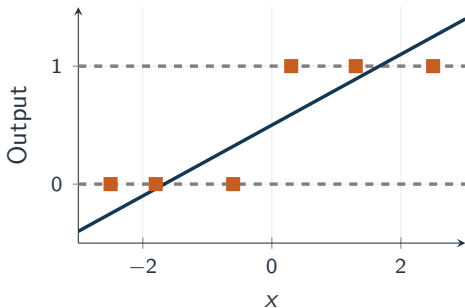
For $y \in \{0, 1\}$, a score

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

can take any real value.

- Values outside $[0, 1]$ cannot be probabilities.
- Thresholding a score gives a label, but does not define a probability model.

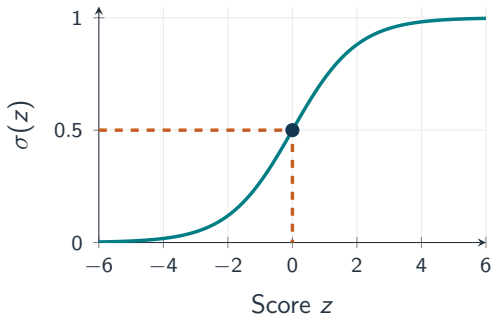
Logistic regression maps the score to a model probability.



The sigmoid function

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

- $0 < \sigma(z) < 1$ for finite z .
 - $\sigma(0) = 1/2$.
 - $\sigma(-z) = 1 - \sigma(z)$.
- $$\sigma'(z) = \sigma(z)(1 - \sigma(z)).$$



Probability recap: conditioning and binary outcomes

A **conditional probability** describes an outcome given information:

$$p(\mathbf{x}) = P(Y = 1 \mid \mathbf{X} = \mathbf{x}), \quad P(Y = 0 \mid \mathbf{X} = \mathbf{x}) = 1 - p(\mathbf{x}).$$

- Conditioning on $\mathbf{x}$ means that we use the features of this example.
- Given $\mathbf{X} = \mathbf{x}$, $Y \in \{0, 1\}$ follows a **Bernoulli distribution** with parameter $p(\mathbf{x})$.
- The conditional mean is $\mathbb{E}[Y \mid \mathbf{X} = \mathbf{x}] = 0(1 - p) + 1p = p(\mathbf{x})$.

The conditional variance is $p(\mathbf{x})(1 - p(\mathbf{x}))$. For $p = 0.8$, the mean is 0.8 and the variance is 0.16.

Logistic regression learns how the Bernoulli parameter depends on the input.

Binary logistic regression: prediction and decision

Model probability

$$z_i = \mathbf{a}_i^\top \boldsymbol{\theta}, \quad p_i = \sigma(z_i).$$

The model estimates $P(Y_i = 1 \mid \mathbf{X}_i = \mathbf{x}_i)$.

Decision at $\tau \in (0, 1)$

$$\hat{y}_i = \begin{cases} 1 & p_i \geq \tau, \\ 0 & p_i < \tau. \end{cases}$$

For $\tau = 0.5$, predict 1 when $z_i \geq 0$.

One-feature example

$$w = 0.8, \quad b = -2, \quad x = 3.$$

$$z = 0.8(3) - 2 = 0.4, \quad p \approx 0.599.$$

Threshold	Predicted label
0.5	1
0.7	0

The threshold changes the label while the estimated probability stays the same.

The decision boundary and the threshold

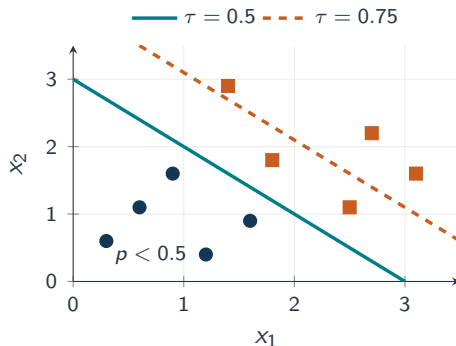
For $\mathbf{w} \neq 0$, the boundary at threshold τ is

$$\mathbf{w}^\top \mathbf{x} + b = \log \frac{\tau}{1 - \tau}.$$

Here $z = x_1 + x_2 - 3$:

- $\tau = 0.5$: $x_1 + x_2 = 3$.
- $\tau = 0.75$: $x_1 + x_2 \approx 4.10$.

Increasing τ shrinks the region predicted as class 1.



Circles and squares show observed classes 0 and 1. Choose the threshold on validation data to reflect error costs.

The boundary is a hyperplane in feature space. Nonlinear features can give a curved boundary in the original inputs.

Log-odds interpretation

The **odds** compare the probabilities of the two classes:

$$\frac{p}{1-p} = e^z, \quad \boxed{\log \frac{p}{1-p} = \mathbf{w}^\top \mathbf{x} + b.}$$

Holding the other features fixed, increasing x_j by one unit:

- changes the log-odds by w_j ,
- multiplies the odds by e^{w_j} .

For example, $w_j = \log 2$ doubles the odds for a one-unit increase in x_j . This describes an association, not a causal effect.

Weight signs determine how the probability changes, holding other features fixed.

The Bernoulli likelihood

A single expression covers both possible labels:

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

- For $y_i = 1$, this reduces to p_i .
- For $y_i = 0$, this reduces to $1 - p_i$.

This is a **discriminative** model: it specifies $P(Y \mid \mathbf{X})$, without modeling $P(\mathbf{X})$. With conditionally independent outcomes,

$$P(\mathbf{y} \mid A; \boldsymbol{\theta}) = \prod_{i=1}^N p_i^{y_i} (1 - p_i)^{1-y_i}.$$

Maximum likelihood fits parameters to the observed labels.

Negative log-likelihood and cross-entropy

Take the negative logarithm of the likelihood and average:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^N [y_i \log p_i + (1 - y_i) \log(1 - p_i)].$$

This is **binary cross-entropy** (BCE) between target probabilities $(1 - y_i, y_i)$ and model probabilities $(1 - p_i, p_i)$.

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

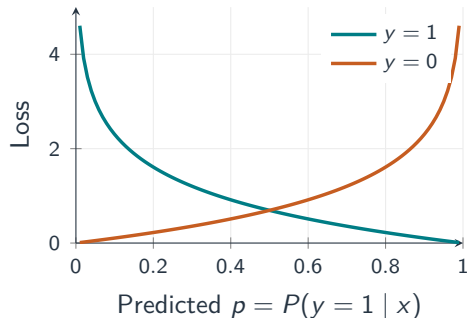
- Correct, confident predictions have small loss.
- Wrong, confident predictions receive a large penalty.

How cross-entropy penalizes mistakes

True y	p	BCE
1	0.9	0.105
1	0.5	0.693
1	0.1	2.303
0	0.9	2.303

Here and throughout, $\log$ is the natural logarithm.

Discuss: Two predictions give the same label. Can their losses differ?



Cross-entropy directly in terms of the score

With $p = \sigma(z)$,

$$\ell(y, z) = \log(1 + e^z) - yz.$$

- The raw score z is the **logit**. The term $\log(1 + e^z)$ is **softplus**.
- This form makes the gradient especially simple.
- Directly computing e^z can overflow for large positive z .

An equivalent stable form is

$$\ell(y, z) = \max(z, 0) - yz + \log(1 + e^{-|z|}).$$

Use a library loss that accepts logits, such as BCE with logits.

Checkpoint: logistic regression

- ① A score is $z = 0$. What probability does the model output?
- ② Is the probability p linear in x ?
- ③ Does a probability of 0.9 guarantee a correct prediction?
- ④ What happens to the boundary when the threshold changes?

Checkpoint: logistic regression

- ① A score is $z = 0$. What probability does the model output?
- ② Is the probability p linear in $\mathbf{x}$?
- ③ Does a probability of 0.9 guarantee a correct prediction?
- ④ What happens to the boundary when the threshold changes?

Answers

1. $p = 0.5$.
2. No, the sigmoid is nonlinear.
3. No, this is a model estimate.
4. The boundary becomes $\mathbf{w}^\top \mathbf{x} + b = \log \frac{\tau}{1-\tau}$.

Fitting regression models

Numerical fitting, regularization and evaluation

Closed-form and numerical model fitting

Fitting means minimizing the chosen regression objective:

$$\theta^* \in \operatorname{argmin}_{\theta} \mathcal{L}(\theta) \quad \text{when a finite minimizer exists.}$$

Linear regression

Least squares gives the normal equations.
Compute a solution using QR or SVD, or
use iterative updates for large datasets.

Logistic regression

Cross-entropy usually requires iterative
fitting. Regularization can ensure a finite
solution when unregularized fitting has none.

Gradient descent is a fitting method for these models.

Gradient descent for model fitting

The gradient $\nabla \mathcal{L}$ collects the partial derivatives of the fitting objective. At a nonzero gradient, a sufficiently small step along its negative decreases the loss.

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla \mathcal{L}(\theta^{(t)}), \quad \eta_t > 0.$$

- ① Compute the current model predictions.
- ② Calculate the gradient of squared loss or cross-entropy.
- ③ Update all parameters together and recompute predictions.
- ④ Stop using a convergence criterion or a validation-based rule.

The prediction model stays the same. Iterative fitting changes its parameters.

Fitting linear and logistic regression

	Linear regression	Logistic regression
Score	$z_i = \mathbf{a}_i^\top \boldsymbol{\theta}$	$z_i = \mathbf{a}_i^\top \boldsymbol{\theta}$
Prediction	$\hat{y}_i = z_i$	$p_i = \sigma(z_i)$
Loss	$\frac{1}{2}(z_i - y_i)^2$	$\log(1 + e^{z_i}) - y_i z_i$
Gradient	$(\hat{y}_i - y_i)\mathbf{a}_i$	$(p_i - y_i)\mathbf{a}_i$

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \frac{1}{N} \sum_i \nabla \ell_i.$$

Each gradient uses prediction minus target. For linear regression, this is $-r_i$.

Regularization of regression coefficients

For either model, add a penalty to the average data loss:

$$\mathcal{L}_\lambda(\mathbf{w}, b) = \frac{1}{N} \sum_i \ell_i(\mathbf{w}, b) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2, \quad \lambda \geq 0.$$

Linear regression: ridge

The penalty stabilizes fits with correlated or redundant features and can reduce overfitting.

Leave the bias unpenalized. Scale features on training data and tune λ on validation data.

For ℓ_1 , add $\lambda \|\mathbf{w}\|_1 = \lambda \sum_j |w_j|$. Some coefficients can become exactly zero (lasso for linear regression).

ℓ_2 shrinks coefficients. ℓ_1 can also select features.

Logistic regression

With both classes present and $\lambda > 0$, the penalty gives a finite solution even for separable data.

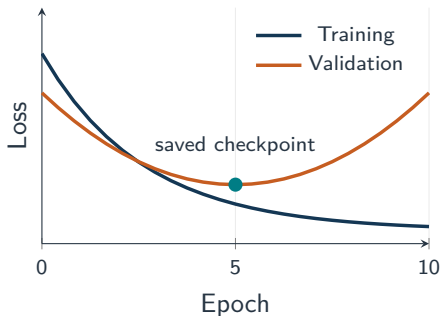
Evaluating a fitted regression model

Optimization

Track the training objective, including any penalty. Rising full-batch loss suggests checking the learning rate and feature scaling.

Generalization

Track validation data loss, excluding the weight penalty. For iterative fitting, save its best checkpoint and stop after a chosen patience.



Schematic data-loss curves. On an evaluation set of size n , $\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2}$ has the target's units. Classification accuracy depends on the threshold and class balance.

Linear and logistic regression at a glance

	Linear regression	Logistic regression
Target	Continuous value	Binary outcome
Output	$\hat{y} = \mathbf{a}^\top \boldsymbol{\theta}$	$p = \sigma(\mathbf{a}^\top \boldsymbol{\theta})$
Likelihood used here	Gaussian with fixed variance	Bernoulli
Data loss	Average squared loss	Average binary cross-entropy
Fitting	Least-squares solver or iterative updates	Usually iterative updates
Evaluation	RMSE on unseen data	Cross-entropy and classification metrics

Both models estimate conditional means. Least squares itself does not require Gaussian noise.

Checkpoint: regression models

- ➊ Why is polynomial regression still linear in its parameters?
- ➋ When is the least-squares parameter vector unique?
- ➌ Which probability models lead to squared loss and binary cross-entropy?
- ➍ How can training loss fall while validation loss rises?
- ➎ Why does logistic regression usually need iterative fitting?

Checkpoint: regression models

- ➊ Why is polynomial regression still linear in its parameters?
- ➋ When is the least-squares parameter vector unique?
- ➌ Which probability models lead to squared loss and binary cross-entropy?
- ➍ How can training loss fall while validation loss rises?
- ➎ Why does logistic regression usually need iterative fitting?

Key points

Fixed features, full column rank, Gaussian versus Bernoulli likelihoods, overfitting, and the absence of a general closed-form logistic solution.

Optional: the gradient gives local sensitivity

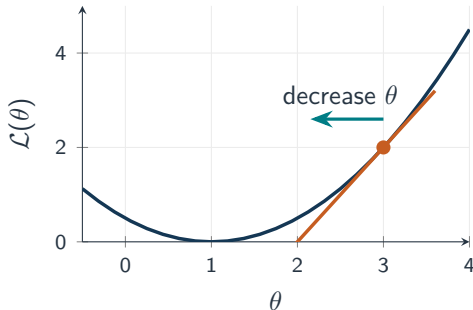
For one parameter, $\mathcal{L}'(\theta)$ is the local slope.

$$\mathcal{L}(\theta + \Delta) \approx \mathcal{L}(\theta) + \mathcal{L}'(\theta)\Delta.$$

For several parameters,

$$\nabla \mathcal{L} = \begin{bmatrix} \partial \mathcal{L} / \partial \theta_0 \\ \vdots \\ \partial \mathcal{L} / \partial \theta_{m-1} \end{bmatrix}.$$

A sufficiently small step along $-\nabla \mathcal{L}$ decreases the loss when the gradient is nonzero.



Optional: a numerical gradient descent example

Let $\mathcal{L}(\theta) = \frac{1}{2}(\theta - 3)^2$, so $\mathcal{L}'(\theta) = \theta - 3$. With $\theta^{(0)} = 0$ and $\eta = 0.5$:

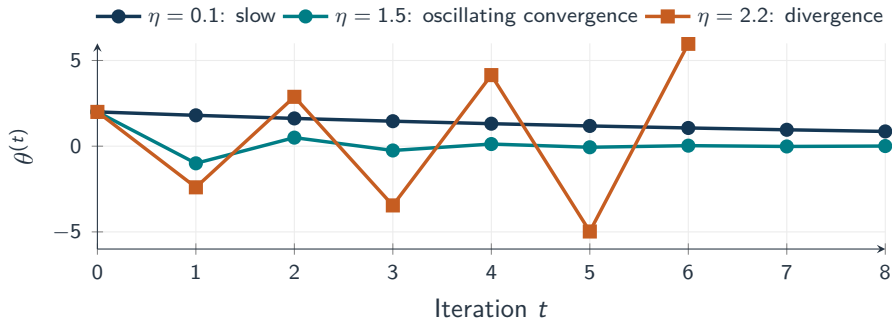
$$\theta^{(t+1)} = \theta^{(t)} - 0.5(\theta^{(t)} - 3).$$

t	$\theta^{(t)}$	$\mathcal{L}'(\theta^{(t)})$	$\mathcal{L}(\theta^{(t)})$
0	0	-3	4.5
1	1.5	-1.5	1.125
2	2.25	-0.75	0.28125
3	2.625	-0.375	0.0703125

Discuss: What is the next parameter value? What happens from $\theta^{(0)} = 0$ with $\eta = 2$?

Optional: the learning rate controls the dynamics

For $\mathcal{L}(\theta) = \frac{1}{2}\theta^2$, the update is $\theta^{(t+1)} = (1 - \eta)\theta^{(t)}$.



From the shown initial value, convergence to 0 requires $0 < \eta < 2$.

Optional: convexity and the learning rate

- For fixed features, squared loss and logistic cross-entropy are convex in θ .
- Any local minimum is global. Uniqueness requires additional conditions.
- An excessive learning rate can make gradient descent diverge even for a convex objective.

For these smooth convex losses, if a finite minimizer exists, a sufficiently small constant step gives convergence in objective value.

For twice continuously differentiable f on $\mathbb{R}^m$, $\nabla^2 f \succeq 0$ everywhere is equivalent to convexity. Positive definiteness everywhere is sufficient for strict convexity, but is not necessary.

Convexity describes the objective. The learning rate controls the algorithm.

Optional: separable data and regularization

A direction θ strictly separates the data when

$$(2y_i - 1)\mathbf{a}_i^\top \theta > 0 \quad \text{for every } i.$$

Scaling that direction gives $\ell_i(\alpha\theta) \rightarrow 0$ as $\alpha \rightarrow \infty$. The loss approaches 0 but never reaches it at finite parameters.

$$\mathcal{L}_\lambda(\mathbf{w}, b) = \mathcal{L}(\mathbf{w}, b) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2, \quad \lambda > 0.$$

- The penalty discourages large weights. Select λ on validation data.
- With both classes present and an unpenalized bias, this objective has a finite minimizer.

The weight gradient gains $\lambda\mathbf{w}$. The bias gradient stays unchanged.

Optional: batch, stochastic and mini-batch gradients

For $\mathcal{L} = \frac{1}{N} \sum_i \ell_i$, choose how many examples to use per update:

Batch: all N examples

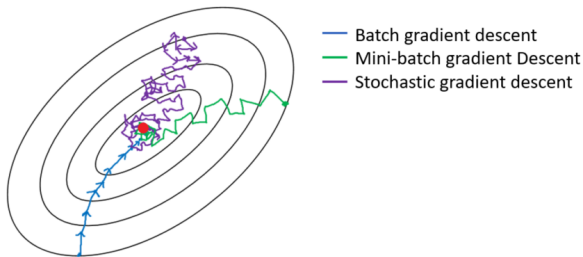
$$\frac{1}{N} \sum_{i=1}^N \nabla \ell_i$$

SGD: one random example

$$\nabla \ell_i$$

Mini-batch: a random batch B_t

$$\frac{1}{|B_t|} \sum_{i \in B_t} \nabla \ell_i$$

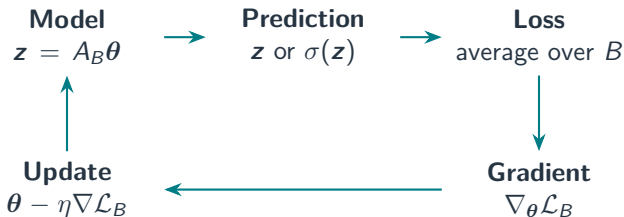


Illustrative paths on loss contours; starting points differ.

Small batches give cheaper updates; larger batches usually reduce gradient noise. A noisy update can increase the full training loss.

Mini-batch training is the usual compromise.

Optional: the learning loop and neural networks



Here $\mathcal{L}_B = |B|^{-1} \sum_{i \in B} \ell_i$, and σ acts elementwise.

In a neural network, earlier layers learn the features. Backpropagation computes gradients through those layers using the chain rule.

With learned features, the same training loop usually optimizes a nonconvex objective.

Optional: gradient of linear regression

For one example,

$$\ell_i = \frac{1}{2}(\mathbf{a}_i^\top \boldsymbol{\theta} - y_i)^2.$$

Apply the chain rule:

$$\nabla_{\boldsymbol{\theta}} \ell_i = \underbrace{(\hat{y}_i - y_i)}_{\text{prediction error}} \underbrace{\mathbf{a}_i}_{\text{features}}.$$

Average over the training set:

$$\nabla \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_i (\hat{y}_i - y_i) \mathbf{a}_i = \frac{1}{N} \mathbf{A}^\top (\mathbf{A}\boldsymbol{\theta} - \mathbf{y}).$$

Each example contributes its error multiplied by its features.

Optional: one update for linear regression

Use the earlier dataset $(x_i, y_i) \in \{(0, 1), (1, 2), (2, 2)\}$. At $b = w = 0$, all predictions equal zero:

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{3}(-1 - 2 - 2) = -\frac{5}{3}, \quad \frac{\partial \mathcal{L}}{\partial w} = \frac{1}{3}(0 - 2 - 4) = -2.$$

With $\eta = 0.1$,

$$b^{(1)} = \frac{1}{6}, \quad w^{(1)} = 0.2.$$
$$\mathcal{L}^{(0)} = 1.5, \quad \mathcal{L}^{(1)} \approx 0.9028.$$

A single update reduces the loss. Further updates approach the least-squares solution for a suitable learning rate.

Optional: gradient of logistic regression

For one example, $z = \mathbf{a}^\top \boldsymbol{\theta}$ and $p = \sigma(z)$:

$$\ell(y, z) = \log(1 + e^z) - yz.$$

Optional: gradient of logistic regression

For one example, $z = \mathbf{a}^\top \boldsymbol{\theta}$ and $p = \sigma(z)$:

$$\ell(y, z) = \log(1 + e^z) - yz.$$

$$\frac{\partial \ell}{\partial z} = \frac{e^z}{1 + e^z} - y = p - y,$$

Optional: gradient of logistic regression

For one example, $z = \mathbf{a}^\top \boldsymbol{\theta}$ and $p = \sigma(z)$:

$$\ell(y, z) = \log(1 + e^z) - yz.$$

$$\frac{\partial \ell}{\partial z} = \frac{e^z}{1 + e^z} - y = p - y,$$

$$\nabla_{\boldsymbol{\theta}} \ell = \frac{\partial \ell}{\partial z} \nabla_{\boldsymbol{\theta}} z = \boxed{(p - y) \mathbf{a}}.$$

Again: prediction error multiplied by features. The prediction is now a probability.

Optional: one update for logistic regression

One example: $x = 2$, $y = 1$. Initially $b = w = 0$.

$$\mathbf{a} = [1, 2]^\top, \quad z = 0, \quad p = 0.5.$$

$$\nabla \ell = (p - y)\mathbf{a} = -0.5[1, 2]^\top = [-0.5, -1]^\top.$$

With $\eta = 0.1$,

$$b^{(1)} = 0.05, \quad w^{(1)} = 0.1, \quad z^{(1)} = 0.25.$$

$$p^{(1)} = \sigma(0.25) \approx 0.5622, \quad \ell^{(1)} \approx 0.5759 < 0.6931.$$

Discuss: For the same input with $y = 0$, which direction would the update take?

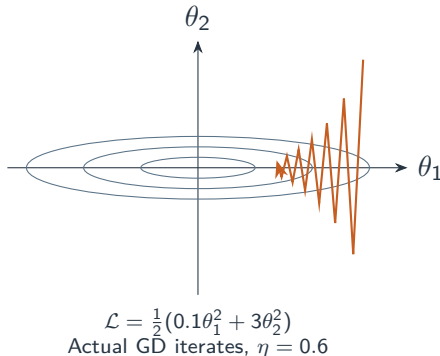
Optional: feature scaling and gradient descent

Standardize a nonconstant feature:

$$x'_j = \frac{x_j - \mu_j}{s_j}, \quad s_j > 0.$$

Fit μ_j, s_j on training data. Keep the bias feature equal to 1.

- Better scaling can improve conditioning and convergence.
- Correlated features can still make optimization slow.
- A weight penalty depends on the feature scales.



Optional: why a stochastic gradient can work

If i is sampled uniformly from $\{1, \dots, N\}$,

$$\mathbb{E}_i[\nabla \ell_i(\boldsymbol{\theta})] = \frac{1}{N} \sum_i \nabla \ell_i(\boldsymbol{\theta}) = \nabla \mathcal{L}(\boldsymbol{\theta}).$$

Thus the sample gradient is an **unbiased estimate** of the full gradient.

- A single update may increase the full training loss.
- Gradient noise does not generally disappear at the optimum.
- Under standard assumptions, decreasing step sizes can support convergence. A constant step size can leave fluctuations near the optimum.

This identity holds for fixed $\boldsymbol{\theta}$, or for a fresh independent draw at each update. Random reshuffling generally loses conditional unbiasedness within an epoch.

Optional: a mini-batch training algorithm

```
initialize theta
for each epoch:
    shuffle training examples
    for each batch B:
        z = A_B @ theta
        prediction = z                # linear regression
        # prediction = sigmoid(z)    # logistic regression
        g = A_B.T @ (prediction - y_B) / len(B)
        theta = theta - eta * g
    monitor training and validation loss
```

With ℓ_2 regularization: add $\lambda \mathbf{w}$ to the weight-gradient entries before updating.

An epoch is one pass over the training set. An update uses one batch.

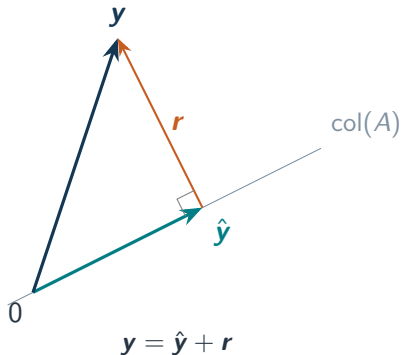
Optional: least squares as a projection

At a least-squares solution,

$$A^T(\mathbf{y} - A\boldsymbol{\theta}^*) = 0.$$

The residual has zero dot product with every column of A .

- $\hat{\mathbf{y}}$ is the projection onto $\text{col}(A)$.
- Fitted values remain unique when parameters are not.
- With a bias, $\sum_i (y_i - \hat{y}_i) = 0$.



Optional: the logistic Hessian

$$\nabla \mathcal{L} = \frac{1}{N} A^\top (\mathbf{p} - \mathbf{y}), \quad \nabla^2 \mathcal{L} = \frac{1}{N} A^\top D A,$$

Here the Hessian collects second derivatives, and $D = \text{diag}(p_i(1 - p_i))$ is diagonal. For any vector $\mathbf{v}$,

$$\mathbf{v}^\top \nabla^2 \mathcal{L} \mathbf{v} = \frac{1}{N} \sum_i p_i(1 - p_i) (\mathbf{a}_i^\top \mathbf{v})^2 \geq 0.$$

- This quadratic-form inequality defines positive semidefiniteness and proves convexity.
- Strictly convex when A has full column rank, at finite parameters.
- Strict convexity does not imply that a finite minimizer exists.

Optional: ridge regression

Let $R = \text{diag}(0, 1, \dots, 1)$ to leave the bias unpenalized.

$$\mathcal{L}_\lambda(\boldsymbol{\theta}) = \frac{1}{2N} \|A\boldsymbol{\theta} - \mathbf{y}\|^2 + \frac{\lambda}{2} \boldsymbol{\theta}^\top R \boldsymbol{\theta}.$$

Then

$$\nabla \mathcal{L}_\lambda = \frac{1}{N} A^\top (A\boldsymbol{\theta} - \mathbf{y}) + \lambda R \boldsymbol{\theta},$$

$$\boxed{(A^\top A + N\lambda R)\boldsymbol{\theta}^* = A^\top \mathbf{y}.}$$

For this bias-augmented design, $N \geq 1$ and $\lambda > 0$ make the matrix positive definite.

The factor N follows from using an average data loss.

Optional: multiclass logistic regression

For K mutually exclusive classes, define one score per class:

$$\mathbf{z} = W\mathbf{a}, \quad W \in \mathbb{R}^{K \times m}.$$

The **softmax** maps scores to probabilities:

$$p_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}, \quad \sum_k p_k = 1.$$

With one-hot target $\mathbf{e}_y$ (a 1 in position y , zeros elsewhere),

$$\ell = -\log p_y, \quad \boxed{\nabla_W \ell = (\mathbf{p} - \mathbf{e}_y)\mathbf{a}^\top.}$$

Use shifted scores for softmax and log-sum-exp for the loss to avoid overflow and $\log(0)$.

Optional: two-class softmax and identifiability

With two scores z_0, z_1 ,

$$p_1 = \frac{e^{z_1}}{e^{z_0} + e^{z_1}} = \sigma(z_1 - z_0).$$

Only the score difference matters.

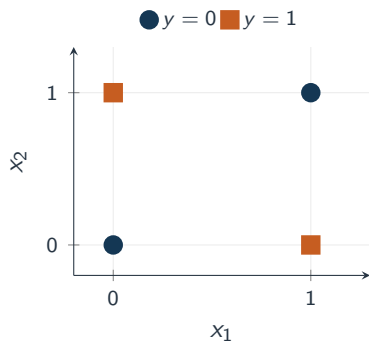
More generally, adding the same score offset to every class leaves softmax unchanged:

$$\text{softmax}(\mathbf{z} + c\mathbf{1}) = \text{softmax}(\mathbf{z}).$$

- The unconstrained softmax parameterization is not identifiable.
- Fixing a reference class removes this common-offset redundancy. Rank deficiencies can still make parameters non-unique.

Multiclass cross-entropy is convex in fixed linear scores, but not automatically strictly convex in all parameters.

Optional: XOR and transformed features



No straight line separates the original inputs.
Choose an interaction feature ($m = 4$):

$$\mathbf{a}(\mathbf{x}) = [1, x_1, x_2, x_1x_2]^\top.$$

A separating score is

$$z = -0.5 + x_1 + x_2 - 2x_1x_2.$$

It gives $z = -0.5$ for class 0 and $z = 0.5$ for class 1.

The classifier is linear in the transformed features. Its boundary in the original inputs can be nonlinear.

Optional: short exercises

- ➊ Derive $\sigma'(z) = \sigma(z)(1 - \sigma(z))$.
- ➋ Perform two full-batch updates for the three-point regression dataset using $\eta = 0.1$.
- ➌ For logistic regression, derive the bias gradient and compare it with the weight gradient.
- ➍ Explain why increasing polynomial degree can reduce training loss without improving validation error.
- ➎ Show that ℓ_2 regularization adds $\lambda \mathbf{w}$ to the weight gradient.

Further reading

Linear and logistic regression

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
Chapters 3 and 4.

Gradient descent and convex optimization

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. Chapter 9.

Connections to neural networks

Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016.
Chapters 5 and 8.