Introduction to Artificial Intelligence and Machine Learning

Lecture Notes

Ninad Joshi

Abstract

Eleven short lectures covering the mathematical foundations of elementary AI and machine learning. Each lecture ends with up to three exercises.

Contents

1 Lecture 1: Artificial intelligence and rational agents **2**

2 Lecture 2: Linear classification and perceptrons **3**

3 Lecture 3: Perceptron training and convergence **4**

4 Lecture 4: Logistic regression and feature scaling **5**

5 Lecture 5: Nearest neighbors and classification metrics **6**

6 Lecture 6: Cross-validation and model selection **7**

7 Lecture 7: Decision trees **8**

8 Lecture 8: Support vector machines and kernels **9**

9 Lecture 9: Linear regression and robust fitting **10**

10 Lecture 10: Multivariate regression and LASSO **11**

11 Lecture 11: Artificial neural networks **12**

Source note **13**

1 Lecture 1: Artificial intelligence and rational agents

Artificial intelligence (AI) studies systems that perceive information and select actions to achieve objectives. A useful distinction is between fixed algorithms, adaptive systems, and agents that select actions using a model of their environment. An agent is not necessarily a machine-learning model.

Rational decision making

A goal-based agent chooses an action that achieves a stated objective; a utility-based agent compares outcomes with different values. For an action a and possible outcomes s, expected utility is

$$U(a) = \sum_s P(s \mid a)u(s, a).$$

A cost-based agent instead minimizes an expected loss $C(a)$. Decisions depend on what is counted: deaths, treatments, monetary costs, or benefits.

Reliability and rare events

An accuracy of 99.99% implies a nominal error probability of 10^{-4} per independent opportunity. Over N opportunities the expected number of errors is Np, while the probability of at least one error is $1 - (1 - p)^N$ under independence. Aggregate accuracy alone does not distinguish false negatives from false positives or specify the risk of an error.

Problem sheet

1. Classify a supermarket barcode scanner, a voice-controlled telephone menu, spelling correction, and adaptive internet routing as AI or non-AI. Justify your criteria.

2. A pedestrian detector recognizes pedestrians at crossings with 99.99% accuracy. Estimate errors over one million detection opportunities and discuss what this estimate does not establish.

3. Three medical tests differ in missed cases and unnecessary treatments. Explain how a goal-based agent minimizing deaths could choose differently from a cost-based agent.

2 Lecture 2: Linear classification and perceptrons

For a feature vector $\mathbf{x} \in \mathbb{R}^d$, a linear classifier predicts using $z = \mathbf{w}^T\mathbf{x} + b$. Its decision boundary is the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$. A threshold classifier outputs 1 when $z \geq 0$ and 0 otherwise. Multiplication of $\mathbf{w}, b$ by a positive constant preserves classification; a negative constant reverses the labels away from the boundary.

Linear separability and feature maps

Data are linearly separable if one hyperplane assigns every training point its correct class. A nonlinear feature map $\boldsymbol{\phi}(\mathbf{x})$ may turn a nonlinear boundary in the original coordinates into a linear one in feature space. For example, (x, x^2) permits a straight line to separate some classes that are inseparable using x alone.

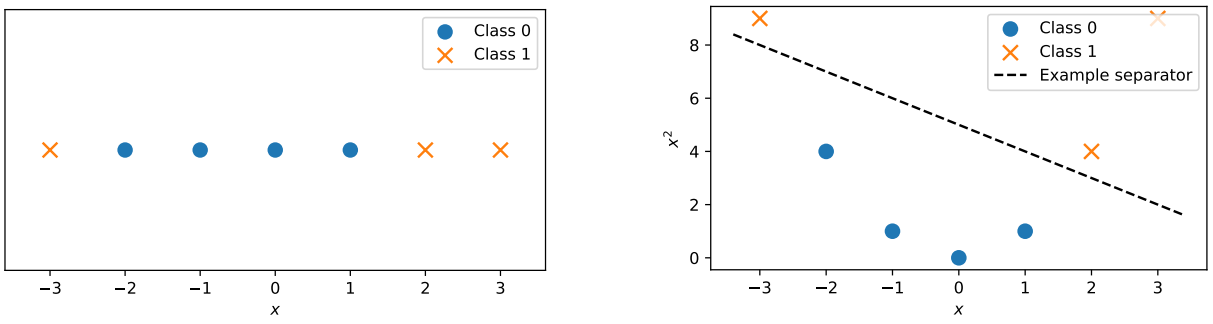


Figure 1: One-dimensional data and a two-dimensional feature transformation.

Perceptron learning

For target $y \in \{0, 1\}$ and prediction $\hat{y}$, an online update is

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(y - \hat{y})\mathbf{x}, \qquad b \leftarrow b + \eta(y - \hat{y}).$$

Here $\eta > 0$ is the learning rate. On linearly separable finite data, standard perceptron learning converges under suitable assumptions. A single threshold unit cannot represent XOR in the original two-dimensional Boolean inputs.

Problem sheet

1. For $x = -3, -2, -1, 0, 1, 2, 3$ and labels $1, 0, 0, 0, 0, 1, 1$, examine separability in x, x^2, and (x, x^2).

2. For $\mathbf{w} = (1, 1)$ and $b = 2$, sketch the decision boundary and the region classified as 1. What happens under positive and negative scaling?

3. Construct the disjunctive normal form of a Boolean truth table and explain how a two-layer network of threshold units can implement it.

3 Lecture 3: Perceptron training and convergence

Training proceeds through repeated passes (*epochs*) over labeled examples. In online learning, weights change after each example. In batch learning, updates are accumulated across the dataset and applied together; these procedures need not converge at the same speed.

Learning rate and order

The learning rate sets update size. For a basic perceptron started from zero, changing a positive learning rate may mainly rescale the resulting weights rather than change the separating hyperplane, although stopping rules and initialization can affect observed epoch counts. Shuffling changes the presentation order, so empirical comparisons should use multiple runs.

Digit recognition

An image can be flattened into a vector of pixel intensities. A binary perceptron can distinguish two digit classes when the selected samples are linearly separable. Pairwise digit classification is different from recognizing one digit against all remaining digits: the latter may be inseparable. Finite iteration limits cannot prove mathematical nonseparability.

Problem sheet

1. Compare online and batch perceptron updates for two simple labeled observations. State precisely when weights are changed.

2. Explain why shuffled and unshuffled training may require different numbers of epochs. Propose a fair comparison protocol.

3. Discuss why successful pairwise digit separation does not guarantee successful one-versus-rest separation.

4 Lecture 4: Logistic regression and feature scaling

A perceptron returns a hard decision; logistic regression models a conditional probability. With $z = \mathbf{w}^T\mathbf{x} + b$,

$$p(y = 1 \mid \mathbf{x}) = \sigma(z) = \frac{1}{1 + e^{-z}}.$$

For observations $(\mathbf{x}_i, y_i)$, binary cross-entropy is

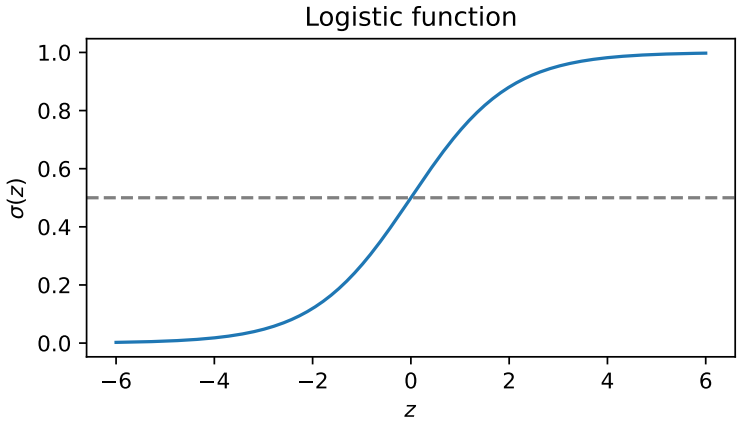


Figure 2: The logistic sigmoid.

$$\mathcal{L} = -\sum_i \left[y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \right].$$

Minimizing this loss is equivalent to maximizing the Bernoulli log-likelihood. Its gradient is $\nabla_{\mathbf{w}}\mathcal{L} = \sum_i (p_i - y_i)\mathbf{x}_i$.

Scaling

Min–max scaling maps a feature to $[0, 1]$ or $[-1, 1]$. Standardization uses $x' = (x - \mu)/s$, where μ and s are the training-set mean and standard deviation. Scaling can substantially change numerical optimization speed; it must be fitted only on training data when evaluating a model.

Problem sheet

1. Derive the gradient of binary cross-entropy with respect to a logistic-regression weight.

2. For a feature with minimum 2, maximum 10, mean 6, and standard deviation 2, transform $x = 8$ by three scaling methods.

3. Explain why fixing a weight can restrict the set of models and why scaling affects optimization.

5 Lecture 5: Nearest neighbors and classification metrics

The nearest-neighbor classifier assigns a query the label of its closest training observation. With k neighbors, classification typically uses a majority vote. The Minkowski distance is

$$d_p(\mathbf{x}, \mathbf{z}) = \left(\sum_j |x_j - z_j|^p \right)^{1/p}.$$

For $p = 2$ this is Euclidean distance. A Voronoi diagram partitions space into regions nearest to each reference point.

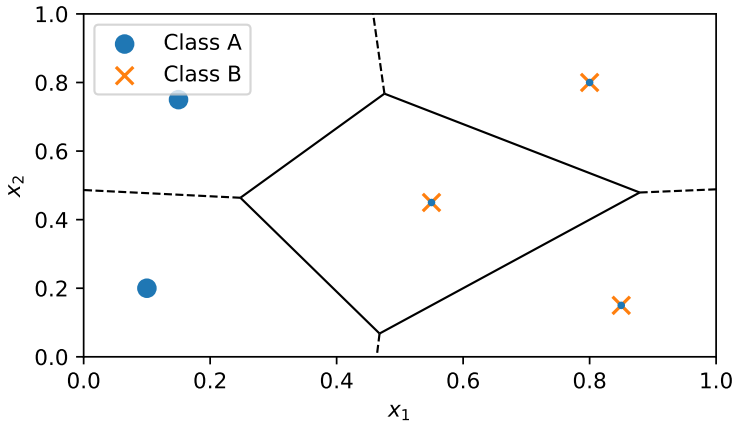


Figure 3: Voronoi regions for example reference points.

Confusion matrix

With true positives TP, false positives FP, true negatives TN, and false negatives FN:

$$\text{Recall} = \frac{TP}{TP + FN}, \quad \text{Specificity} = \frac{TN}{TN + FP},$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad F_1 = \frac{2PR}{P + R}.$$

Sensitivity is another name for recall. The false-positive rate is $1 - \text{Specificity}$. A ROC curve requires varying a decision threshold over a score; one fixed binary decision gives only one operating point.

Problem sheet

1. Draw a Voronoi partition for five labeled points and identify the 1-nearest-neighbor class boundary.

2. For TP=33, FP=0, FN=28, TN=61117, calculate recall, specificity, precision and F_1.

3. Explain the difference between precision and sensitivity in a rare-disease screening problem.

6 Lecture 6: Cross-validation and model selection

A hyperparameter is selected before fitting model coefficients; examples include k in kNN and the distance exponent p. In K-fold cross-validation, the training data are partitioned into K folds. Each fold is held out once while the model is fitted on the others. The mean validation score estimates performance for that training procedure.

$$\overline{S} = \frac{1}{K} \sum_{j=1}^{K} S_j.$$

Leave-one-out cross-validation uses one held-out observation per fold. Repeated random splits can reduce dependence on a particular partition, at additional computational cost.

Avoiding leakage

Choose hyperparameters and fit preprocessing inside the training folds. Reserve an untouched test set for final assessment. When tuning multiple interacting hyperparameters, joint optimization is usually preferable to a one-at-a-time search. A small score tolerance can justify selecting a simpler model among nearly equivalent candidates.

Problem sheet

1. Compare five-fold and leave-one-out cross-validation for a dataset with 100 observations.

2. Explain why feature standardization must be fitted separately within each training fold.

3. Design a joint cross-validation search for k, Minkowski exponent p, and distance-weight exponent α in weighted kNN.

7 Lecture 7: Decision trees

A decision tree recursively partitions the feature space using questions about features. Internal nodes test a condition; leaf nodes assign a class or prediction. Trees can capture nonlinear boundaries without explicit feature transformations.

Impurity measures

For class proportions p_c at a node, entropy and Gini impurity are

$$H = -\sum_c p_c \log_2 p_c, \qquad G = 1 - \sum_c p_c^2.$$

A candidate split is judged by the decrease in impurity, weighting each child by its fraction of observations. Greedy splitting need not produce the globally smallest tree.

Overfitting control

A deep tree can memorize training examples. Restricting maximum depth, minimum samples per leaf, or maximum leaf nodes limits complexity. These hyperparameters should be evaluated together by cross-validation.

Problem sheet

1. For a node with six positive and four negative examples, calculate entropy and Gini impurity.

2. A split creates children with class counts (4 positive, 1 negative) and (2 positive, 3 negative). Calculate the weighted Gini decrease.

3. Explain the different effects of maximum depth, minimum leaf size and maximum leaf count.

8 Lecture 8: Support vector machines and kernels

A support vector machine (SVM) seeks a separating hyperplane with a large geometric margin. For labels $y_i \in \{-1, +1\}$, the hard-margin constraints are $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$ and the objective is to minimize $\|\mathbf{w}\|^2/2$. Soft-margin SVMs allow violations using slack variables ξ_i and a penalty C:

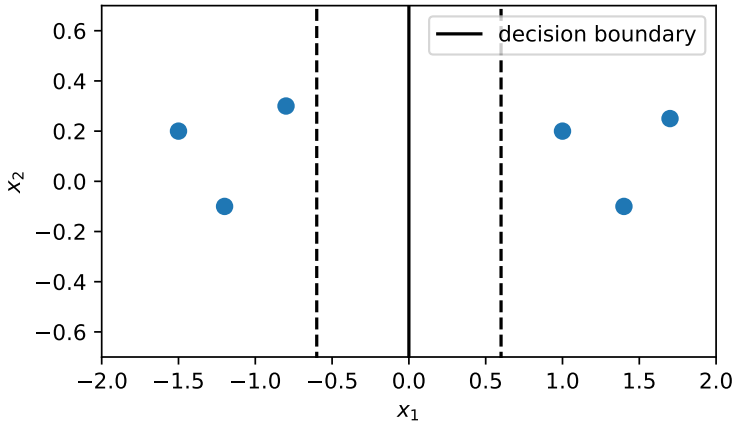


Figure 4: Illustrative decision boundary and margins.

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i.$$

A feature transformation can make a curved original-space boundary linear in feature space. The kernel trick computes inner products in a feature space without explicitly constructing every coordinate. A common kernel is $K(\mathbf{x}, \mathbf{z}) = \exp(-\gamma\|\mathbf{x} - \mathbf{z}\|^2)$. Both C and γ require model selection.

Problem sheet

1. Expand the shifted-circle equation $(x_1 - a)^2 + (x_2 - b)^2 = r^2$ as a hyperplane in suitable polynomial features.

2. Find a feature transformation that makes XOR with labels $-1, +1$ linearly separable.

3. Explain qualitatively how C, γ and feature scaling affect an RBF-kernel SVM.

9 Lecture 9: Linear regression and robust fitting

Regression predicts a continuous quantity rather than a class. A linear model is $\hat{y} = b + \mathbf{w}^T\mathbf{x}$. Ordinary least squares minimizes

$$L_2 = \sum_i (y_i - \hat{y}_i)^2.$$

A large residual is strongly penalized, making least squares sensitive to outliers. Absolute error uses $L_1 = \sum_i |y_i - \hat{y}_i|$ and is more robust to isolated extreme residuals. Huber loss is quadratic

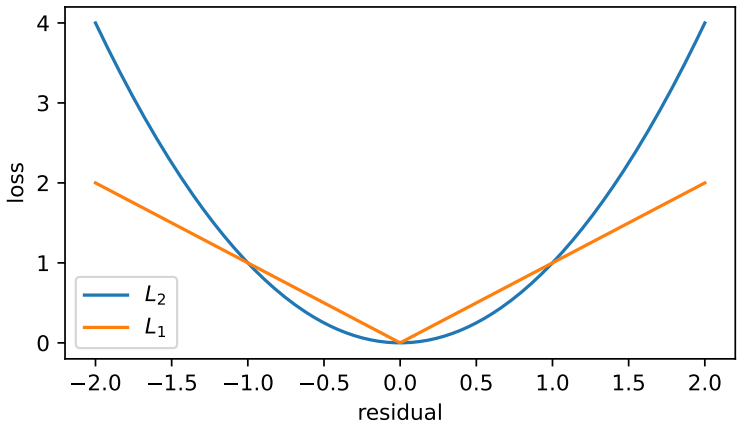


Figure 5: Absolute and squared loss for a single residual.

near zero and linear for large residuals:

$$\ell_\delta(r) = \begin{cases} \frac{1}{2}r^2, & |r| \le \delta, \\ \delta(|r| - \delta/2), & |r| > \delta. \end{cases}$$

RANSAC repeatedly fits a model on small subsets and identifies observations consistent with it. Its success depends on a suitable residual threshold and sufficient inliers.

Problem sheet

1. For observations $2, 4, 4, 9, 11, 100$, find the constant prediction minimizing squared loss and the set minimizing absolute loss.

2. Explain why Huber loss can outperform squared loss when a dataset contains outliers.

3. Describe the basic steps of RANSAC and the role of its inlier threshold.

10 Lecture 10: Multivariate regression and LASSO

Linear regression can use many features or basis functions:

$$\hat{y}(\mathbf{x}) = b + \sum_{j=1}^{m} w_j \phi_j(\mathbf{x}).$$

The model is linear in its coefficients even when the basis functions, such as x^2 or $\sin x$, are nonlinear in the input. Additional features increase flexibility but may also increase variance and coefficient instability.

Regularization

LASSO adds an L_1 penalty to least squares:

$$\min_{\mathbf{w},b} \frac{1}{2n} \sum_i (y_i - \hat{y}_i)^2 + \lambda \sum_j |w_j|.$$

The penalty can set coefficients exactly to zero, performing feature selection. Feature scaling matters because coefficients depend on feature units. Choose λ using validation and assess the final model on held-out observations. Feature selection and refitting must not use the test set.

Problem sheet

1. Construct a linear-in-parameters model using $1, x, x^2, \sin x$ and $x \sin x$.

2. Explain why LASSO may prefer a sparse model over ordinary least squares.

3. Discuss how changing the scale of one input feature affects the LASSO penalty.

11 Lecture 11: Artificial neural networks

A feed-forward neural network combines affine transformations and nonlinear activation functions. A layer computes

$$\mathbf{a}^{(\ell)} = g\Big(W^{(\ell)}\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}\Big).$$

Common activations include sigmoid, hyperbolic tangent, and ReLU $g(z) = \max(0, z)$. The

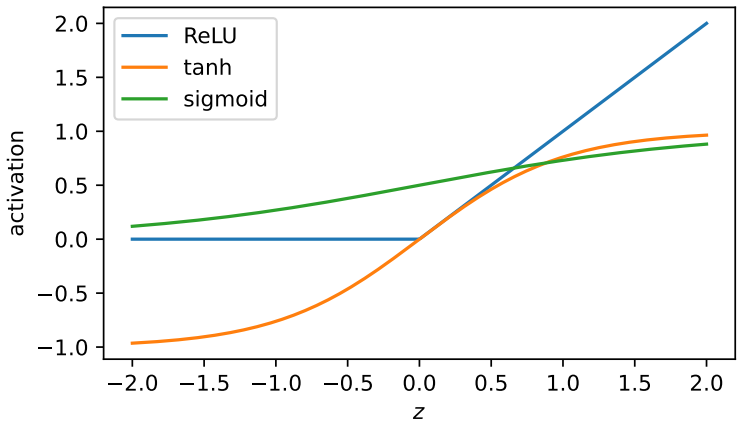


Figure 6: Common nonlinear activation functions.

derivative of the sigmoid is $\sigma'(z) = \sigma(z)(1 - \sigma(z))$. ReLU has derivative zero for $z < 0$ and one for $z > 0$; it is not differentiable at zero, where a subgradient convention is used.

Network size

For N inputs, $L \geq 1$ fully connected hidden layers of width n, and one scalar output, including biases, the parameter count is

$$n(N + 1) + (L - 1)(n^2 + n) + (n + 1).$$

Greater width and depth increase representational capacity but also training cost and overfitting risk. Nonlinear activation is essential: a composition of purely linear layers remains a linear transformation.

Problem sheet

1. Calculate the output of a two-input, two-hidden-neuron, two-output ReLU network for $\mathbf{x} = (1, 2)$ using weights of your choice; show each intermediate activation.

2. Derive the number of parameters in a network with N inputs, L hidden layers of width n, and one output.

3. Differentiate sigmoid, ReLU, softplus and tanh. Identify any relations between the derivatives.

Source note

The sequence and problem themes are adapted from the uploaded *Künstliche Intelligenz I* homework sheets 1–12 (winter semester 2023/24). Sheets 9 and 10 were distributed together. Explanatory text and illustrations are newly written; exercise wording is shortened or adapted, not reproduced verbatim. The original materials contain additional computational tasks that are intentionally excluded here.