CS 1810 Spring 2026 Section 3
Classification

1 Classification

In the previous sections, we predicted continuous targets using KNN, kernelized regression and OLS. We now turn to classification, where the target y is discrete, such as identifying bird species from audio recordings or classifying galaxies by morphology.

1.1 Linear Classification

Given an input vector $\boldsymbol{x}$, linear classification assigns it to one of K discrete classes C_k. The input space is divided into decision regions separated by linear decision boundaries.

In binary linear classification, two classes (labeled -1 and 1) are separated by a single hyperplane. The discriminant function directly assigns each vector to a class:

$$\hat{y} = \text{sign}(h(\boldsymbol{x}; \boldsymbol{w}, w_0)) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + w_0)$$

where $\text{sign}(z) = 1$ if $z \geq 0$ and $\text{sign}(z) = -1$ if $z < 0$. The weight vector $\boldsymbol{w}$ is normal to the decision surface and determines its orientation.

1.2 Perceptron

The perceptron is a discriminative algorithm for binary classification that finds a linear decision boundary, if one exists.

To define the loss, we use the rectified linear function $\text{ReLU}(z) = \max\{0, z\}$. With $h(\mathbf{x}_i; \mathbf{w}, w_0) = \mathbf{w}^\top \mathbf{x}_i + w_0$ and label y_i, the perceptron loss is:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{N} \text{ReLU}(-h(\mathbf{x}_i; \mathbf{w}, w_0) y_i) \\ &= -\sum_{i:\, y_i \neq \hat{y}_i} (\mathbf{w}^\top \mathbf{x}_i + w_0) y_i\end{aligned}$$

For any misclassified point, $-h(\mathbf{x}_i; \mathbf{w}, w_0) y_i$ is positive, so minimizing this loss drives the decision boundary toward correct classification.

We minimize the loss using stochastic gradient descent. Correctly classified points have zero gradient; for a misclassified point $(\mathbf{x}_i, y_i)$, the updates are:

$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i \\ w_0 &\leftarrow w_0 + \eta y_i\end{aligned}$$

where η is the learning rate.

If the training data are linearly separable, the perceptron is guaranteed to converge in finitely many updates.

1.2.1 Concept Question

Why do we choose the ReLU function instead of the $0/1$ function when formulating the loss function?

1.3 Exercise: Boolean Functions and the Perceptron

A perceptron classifies inputs $\mathbf{x} = (x_1, x_2)$ with $x_1, x_2 \in \{0, 1\}$ via the rule $\hat{y} = \text{sign}(\mathbf{w}^\top \mathbf{x} + w_0)$.

1. Find weights w_1, w_2 and bias w_0 such that the perceptron computes the AND function (output $+1$ if and only if $x_1 = 1$ and $x_2 = 1$).
2. Find weights such that the perceptron computes the OR function (output $+1$ if and only if at least one of x_1, x_2 equals 1).
3. Prove that no choice of weights can compute the XOR function (output $+1$ if and only if exactly one of x_1, x_2 equals 1).
4. What does this tell us about the perceptron algorithm on XOR-labeled data?

2 Probabilistic Classification

There are two distinct probabilistic approaches to modeling: discriminative and generative.

2.1 Discriminative Models

A discriminative model learns $p(y|\mathbf{x})$ without modeling how $\mathbf{x}$ is generated.

2.1.1 Logistic Regression

Logistic regression is a discriminative model for binary classification. The two classes are labeled 0 and 1. Given weights $\mathbf{w}$ and intercept w_0, we model the probability of label y given features $\mathbf{x}$:

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + w_0)$$
$$p(y = 0|\mathbf{x}) = 1 - \sigma(\mathbf{w}^\top \mathbf{x} + w_0)$$

σ denotes the sigmoid function:

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

The sigmoid takes any value $z \in \mathbb{R}$ and returns an output in $(0, 1)$.

To find the best weights $\mathbf{w}$, we maximize the likelihood of our data. Unlike OLS, there is no closed-form solution. We minimize the negative log-likelihood (NLL): negating turns the maximization into a minimization, and the logarithm converts the product of likelihoods into a sum without changing the optimum.

After fitting, we predict the class for a new point $\mathbf{x}^*$ by computing

$$p(y^* = 1|\mathbf{x}^*; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}^* + w_0)$$
$$p(y^* = 0|\mathbf{x}^*; \mathbf{w}) = 1 - p(y^* = 1|\mathbf{x}^*; \mathbf{w})$$

and assigning whichever class has the higher probability. Although the sigmoid function is nonlinear, the decision boundary of logistic regression is linear: it is the set of points where $\mathbf{w}^\top \mathbf{x} + w_0 = 0$, a hyperplane.

We can extend logistic regression to K classes using the softmax function, which generalizes the sigmoid:

$$\text{softmax}_k(\mathbf{z}) = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}$$

Notation. We write $\hat{y}_i = p(y_i = 1|\mathbf{x}_i) = \sigma(\mathbf{w}^\top \mathbf{x}_i + w_0)$ for the predicted probability. We use $\mathcal{L}$ for the loss function, L for the likelihood and ℓ for the log-likelihood.

2.1.2 Exercise: Deriving the Negative Log-Likelihood for Logistic Regression

In this exercise, we derive the loss function for logistic regression from first principles using maximum likelihood estimation (MLE).

Given a training dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with $y_i \in \{0, 1\}$, and the logistic regression model $p(y = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x})$ (absorbing the intercept into $\mathbf{w}$), let $\hat{y}_i = \sigma(\mathbf{w}^\top \mathbf{x}_i)$.

1. **Single data point likelihood.** Using the "power trick," show that the likelihood for a single data point $(\mathbf{x}_i, y_i)$ can be written compactly as:

$$p(y_i \mid \mathbf{x}_i; \mathbf{w}) = \hat{y}_i^{y_i}(1 - \hat{y}_i)^{1-y_i}$$

   Verify that this expression is correct for both $y_i = 1$ and $y_i = 0$.

2. **Joint likelihood.** Assuming the data points are independent and identically distributed (i.i.d.), write down the likelihood of the entire dataset $L(\mathbf{w})$.

3. **Log-likelihood.** Take the logarithm of the joint likelihood to obtain the log-likelihood $\ell(\mathbf{w})$. Why is it helpful to work with the log-likelihood instead of the likelihood directly?

4. **Negative log-likelihood (NLL).** Write down the NLL loss $\mathcal{L}_{\text{NLL}}(\mathbf{w}) = -\ell(\mathbf{w})$ and expand it in terms of $\hat{y}_i = \sigma(\mathbf{w}^\top \mathbf{x}_i)$.

5. **Sigmoid derivative.** Show that $\sigma'(z) = \sigma(z)(1 - \sigma(z))$.

6. **Gradient of the NLL.** Using this property and the chain rule, compute the gradient $\nabla_{\mathbf{w}} \mathcal{L}_{\text{NLL}}(\mathbf{w})$ and show that it simplifies to:

$$\nabla_{\mathbf{w}} \mathcal{L}_{\text{NLL}}(\mathbf{w}) = \sum_{i=1}^{N} (\hat{y}_i - y_i)\mathbf{x}_i$$

2.2 Generative Models

A discriminative model directly specifies the posterior $p(y|\mathbf{x})$. A generative model instead specifies the joint distribution $p(\mathbf{x}, y)$, then uses Bayes' Rule to recover the posterior and pick the most likely label y:

$$p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y)p(y)}{p(\boldsymbol{x})} \propto p(\boldsymbol{x}|y)p(y)$$

$p(y)$ is the class prior, the probability of each class before observing features. In binary classification, this is a Bernoulli distribution. $p(\mathbf{x}|y)$ is the class-conditional distribution: given a class y, it tells us how likely we are to see features $\mathbf{x}$. We classify by picking the class k that maximizes $p(y = k|\mathbf{x})$.

2.2.1 Exercise: Shapes of Decision Boundaries I

Consider a generative model with $K > 2$ classes and output labels encoded as one-hot vectors $\boldsymbol{y}$ of length K. The class prior is $p(\boldsymbol{y} = C_k) = \pi_k$ for $k \in \{1, \ldots, K\}$, and the class-conditional distributions are Gaussian:

$$p(\boldsymbol{x} \,|\, \boldsymbol{y} = C_k) = \mathcal{N}(\boldsymbol{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad k \in \{1, \ldots, K\}$$

We classify a new example $\boldsymbol{x}$ by the highest posterior probability $p(\boldsymbol{y} = C_k \,|\, \boldsymbol{x})$. Show that this is equivalent to finding the class k that maximizes:

$$f_k(\boldsymbol{x}) = \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k).$$

Derive f_k from the posterior $p(\boldsymbol{y} = C_k \,|\, \boldsymbol{x})$. What can we claim about the shape of the decision boundary?

2.2.2 Exercise: Shapes of Decision Boundaries II

Now suppose every class has the same covariance matrix, so $\mathbf{\Sigma}_\ell = \mathbf{\Sigma}_{\ell'}$ for all classes C_ℓ and $C_{\ell'}$. Simplify this formula further. What can we claim about the shape of the decision boundaries now?

3 Confusion Matrices, TPR, FPR and AUC

Simple accuracy is misleading when classes are imbalanced: a 99% accuracy rate means little if 99% of examples belong to one class. Accuracy also treats all errors alike, when in practice some are far more costly than others.

3.1 Confusion Matrix

A confusion matrix summarizes classification performance using four counts.

	Predicted Positive	Predicted Negative
Actually Positive	True Positive (TP)	False Negative (FN)
Actually Negative	False Positive (FP)	True Negative (TN)

3.2 TPR and FPR

The true positive rate (TPR) is the fraction of actual positive instances that the model correctly identifies. TPR is also known as sensitivity or recall.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The false positive rate (FPR) is the fraction of actual negative instances that the model incorrectly classifies as positive:

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

The true negative rate (TNR) is the fraction of actual negative instances that the model correctly identifies:

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} = 1 - \text{FPR}$$

The false negative rate (FNR) is the fraction of actual positive instances that the model incorrectly classifies as negative:

$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} = 1 - \text{TPR}$$

3.3 ROC Curve and AUC

The receiver operating characteristic (ROC) curve plots TPR against FPR at different classification thresholds. Lowering the threshold classifies more examples as positive, which increases both the true positive rate and the false positive rate.

At threshold 0, every example is classified positive ($\text{TPR} = \text{FPR} = 1$); at threshold 1, none are ($\text{TPR} = \text{FPR} = 0$). A perfect classifier reaches the top-left corner of the ROC curve, catching every positive ($\text{TPR} = 1$) with no false alarms ($\text{FPR} = 0$).

The area under the ROC curve (AUC) quantifies the tradeoff between TPR and FPR across all thresholds; it ranges from 0 to 1.

Formally, AUC is the integral of the ROC curve over FPR $\in [0, 1]$. A random classifier's ROC curve follows the diagonal from $(0, 0)$ to $(1, 1)$, giving AUC $= 0.5$; a perfect classifier has AUC $= 1$. AUC is useful for comparing classifiers against each other, not only against a random baseline.

3.4 Exercise: Computing ROC Curves and AUC

A bank uses a classifier to flag potentially fraudulent credit card transactions. The classifier outputs a score between 0 and 1, where higher scores indicate higher confidence that the transaction is fraudulent. A transaction is flagged as fraud ("positive") if its score is $\geq t$ for some threshold t. The results for four transactions are:

Transaction	Score	Ground Truth
A	0.9	Fraud
B	0.6	Legitimate
C	0.4	Fraud
D	0.1	Legitimate

1. At threshold $t = 0.5$, write down the confusion matrix and compute the TPR and FPR.
2. Compute the TPR and FPR at each distinct threshold value. Plot the ROC curve.
3. Compute the AUC.
4. The fraud prevention team wants to catch every fraudulent transaction (maximize TPR). The customer experience team wants to ensure no legitimate transaction is ever blocked (minimize FPR). Using the ROC curve, discuss what threshold each team would prefer.