**CHEME 5820: Machine Learning for Engineers** **Spring 2025**

# Lecture 3a: Linear Regression, Perceptron and Binary Classification

*Lecturer: Jeffrey Varner*

**Disclaimer**: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

## 1 Introduction

In this lecture, we will introduce the perceptron algorithm for binary classification problems. The perceptron is a simple linear classifier that can be used to separate two classes of data points. The perceptron algorithm is a simple iterative algorithm that incrementally updates the weights of the linear classifier to minimize the classification error. We will first introduce the perceptron algorithm and then discuss its convergence properties, i.e., when the algorithm converges to a solution and when it does not. The key concepts covered in this lecture include:

- **Linear regression models**: A class of models used in machine learning for regression tasks, i.e., predicting a continuous output variable from one or more continuous or discrete input features. Linear regression models assume that the output variable is a linear combination of the input features.
- **Binary classification**: The problem of classifying data points into one of two classes. Binary classification is a type of supervised machine learning task that involves categorizing data points into one of two distinct classes based on their features. These features can be continuous or discrete, and the classes can be represented as binary labels, e.g., $\{-1, 1\}$ or $\{0, 1\}$.
- **Perceptron**: The perceptron algorithm for binary classification problems is a linear classifier that separates two classes of data points. The perceptron algorithm is an iterative algorithm that incrementally updates the weights of the linear classifier to minimize the classification error. The perceptron algorithm is guaranteed to converge to a solution with no mistakes in a finite number of iterations if the data set is linearly separable. However, if the data set is not linearly separable, the perceptron algorithm will not converge to a perfect solution, but rather to a solution with some classification errors.

## 2 Linear Regression Models

Linear regression models are a class of models used in machine learning for regression tasks, i.e., predicting a continuous output variable from one or more continous or discrete input features. Linear regression models assume that the output variable is a linear combination of the input features, i.e., the output variable is a linear function of the input features. Linear in this context is a misnomer in the sense that the features are not necessarily linear, but the model is linear in the parameters and the features. Suppose we have a data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$ with $m$ examples, where each example where $\mathbf{x}_i \in \mathbb{R}^n$ is a feature vector and $y_i \in \mathbb{R}$ is the output variable. The linear regression model predicts the output variable $y_i$ for feature vector $\mathbf{x}_i$ using the linear function:

$$y_i = \mathbf{x}_i^T \cdot \beta + \epsilon_i$$

where we have augmented the feature vector $\mathbf{x}_i$ with a bias term, i.e., $\mathbf{x}_i^T = \left(x_1^{(i)}, \dots, x_n^{(i)}, 1\right)$, $\beta = (w_1, \dots, w_n, b)$ is a column vector of (unknown) parameters $w_j \in \mathbb{R}$ corresponding to the importance (weight) of feature $j$ and a bias parameter $b \in \mathbb{R}$ and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is a noise term, typically assumed to be a Normal distribution with mean zero and variance $\sigma^2$. Depending upon the shape of the data and various other problem constraints, there are analytical solutions to the linear regression problem, e.g., the normal equations, or iterative solutions, e.g., gradient descent can be used to estimate the parameters $\beta$.

### 2.1 Overdetemined Linear Regression models

If the number of examples $m$ is greater than the number of features $n$, the linear regression model is said to be overdetermined. In other words, there are more examples than features. Regularized linear regression models incorporate penalty terms to constrain the size of the coefficient estimates, thereby reducing overfitting and enhancing the model's generalizability to new data. Consider an overdetermined data matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$, i.e., the case where $m > n$ (more examples than unknown parameters). A regularized least squares estimate of the unknown parameters $\beta$ for an overdetermined system will minimize a loss (objective) function of the form:

$$\hat{\beta}_{\lambda} = \arg\min_{\beta} || \ \mathbf{y} - \mathbf{X}\cdot\beta \ ||_2^2 + \lambda \cdot || \ \beta \ ||_2^2$$

where $||\star||_2^2$ is the square of the $l2$-vector norm, $\lambda \geq 0$ denotes a regularization parameter, and $\hat{\beta}_{\lambda}$ denotes the estimated parameter vector. The parameters $\hat{\beta}_{\lambda}$ that minimize the $||\star||_2^2$ loss plus penalty for overdetermined data matrix $\mathbf{X}$ are given by:

$$\hat{\beta}_{\lambda} = \left(\mathbf{X}^T\mathbf{X} + \lambda\cdot\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} - \left(\mathbf{X}^T\mathbf{X} + \lambda\cdot\mathbf{I}\right)^{-1}\mathbf{X}^T\epsilon$$

The matrix $\mathbf{X}^T\mathbf{X}+\lambda\cdot\mathbf{I}$ is the `regularized normal matrix`, while $\mathbf{X}^T\mathbf{y}$ is the `moment vector`. The inverse $\left(\mathbf{X}^T\mathbf{X} + \lambda\cdot\mathbf{I}\right)^{-1}$ must exist to obtain the estimated parameter vector $\hat{\beta}_{\lambda}$.

### 2.2 Underdetermined Linear Regression models

Assume the data matrix $\mathbf{X}$ is `underdetermined`, i.e., $m < n$ (more columns than rows), and the error vector $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\cdot\mathbf{I})$. Then, an ordinary least squares estimate of the unknown parameters is the *smallest* parameter vector $\beta$ that satisfies the original equations:

$$\begin{aligned} \text{minimize} \quad & ||\,\beta\,|| \\ \text{subject to} \quad & \mathbf{X}\cdot\beta = \mathbf{y} \end{aligned}$$

The least-norm problem has an analytical estimate for the unknown parameter vector $\hat{\beta}$ given by:

$$\hat{\beta} = \mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}\cdot\mathbf{y} - \mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}\cdot\epsilon$$

where inverse $\left(\mathbf{X}\mathbf{X}^T\right)^{-1}$ must exist to obtain the estimated model parameter vectors $\hat{\beta}$.

## 3 The Perceptron and Binary Classification

The Perceptron (1) is a simple yet powerful algorithm used in machine learning for binary classification tasks. The Perceptron (Rosenblatt, 1957) takes the (scalar) output of a linear regression model $y_i \in \mathbb{R}$ and transforms it, using a transform function $\sigma(\star) = \text{sign}(\star)$, into a discrete value representing a category, e.g., $\sigma : \mathbb{R} \rightarrow \{-1, 1\}$ in the binary classification case. Suppose there exists a data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$ with $m$ *labeled* examples, where each example $1, 2, \dots, m$ has been labeled by an expert, i.e., a human to be in a category $\hat{y}_i \in \{-1, 1\}$, given the feature vector $\mathbf{x}_i \in \mathbb{R}^n$. The Perceptron *incrementally* learns a linear decision boundary between two classes of possible objects (binary classification) in $\mathcal{D}$ by repeatedly processing the data. During each pass, a regression parameter vector $\beta$ is updated until it makes no more than a specified number of mistakes.

The Perceptron computes the label $\hat{y}_i$ for feature vector $\mathbf{x}_i$ using the $\sigma(\star) = \text{sign}(\star)$ function:

$$\hat{y}_i = \text{sign}\left(\mathbf{x}_i^T \cdot \beta\right)$$

where $\beta = (w_1, \dots, w_n, b)$ is a column vector of (unknown) weight parameters $w_j \in \mathbb{R}$ corresponding to the importance of feature $j$ and a bias parameter $b \in \mathbb{R}$, the features $\mathbf{x}_i^T = \left(x_1^{(i)}, \dots, x_n^{(i)}, 1\right)$ is the $n+1$-dimensional feature (row) vector (features augmented with bias term), and $\text{sign}(z)$ is the `sign` function:

$$\text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

If data set $\mathcal{D}$ is linearly separable, the Perceptron will find a separating hyperplane in a finite number of passes through $\mathcal{D}$. However, if the data set $\mathcal{D}$ is not linearly separable, the Perceptron will not converge. Pusedo code for the perceptron algorithm is shown in Algorithm 1.

**Algorithm 1** The Perceptron Algorithm

```
Input: $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, tolerance $\epsilon \geq 0$, maximum iterations maxiter
Features: $\mathbf{x}_i = (x_{i1}, \dots, x_{in}, 1)$ are augmented with a bias term, labels $y_i \in \{-1, 1\}$.
Output: Classifier parameters $\beta = (w_1, \dots, w_n, b)$
$\beta \leftarrow$ rand                                ▷ Initialize parameter vector $\beta$ to a random vector
$i \leftarrow 0$                                       ▷ Initialize the loop counter to zero
while true do                                          ▷ Repeat until stopping criterion is met
    error $\leftarrow 0$                               ▷ Initialize the error count to zero for this pass through $\mathcal{D}$
    for $(\mathbf{x}, y) \in \mathcal{D}$ do           ▷ Iterate over each pair $(\mathbf{x}, y)$ in data set $\mathcal{D}$
        if $y \cdot \left(\mathbf{x}^T \cdot \beta\right) \leq 0$ then   ▷ Ooops! The data pair $(\mathbf{x}, y)$ is misclassified
            $\beta \leftarrow \beta + y \cdot \mathbf{x}$              ▷ Update the weight vector $\beta$
            error $\leftarrow$ error $+ 1$                             ▷ Increment the error count
        end if
    end for
    if error $\leq \epsilon$ or $i \geq$ maxiter then  ▷ Stopping criterion: tolerance or max iterations?
        break                                          ▷ Exit the training loop
    end if
    $i \leftarrow i + 1$                               ▷ Increment the loop counter and repeat
end while
```

## 4 Summary

In this lecture, we introduced the perceptron algorithm for binary classification problems. The perceptron is a simple linear classifier that can be used to separate two classes of data points. The perceptron algorithm is a simple iterative algorithm that incrementally updates the weights of the linear classifier to minimize the classification error. Thus, it is one of the first examples of on online learning algorithm, i.e., an algorithm that learns from data in an incremental fashion. The perceptron algorithm is guaranteed to converge to a solution if the data set is linearly separable. However, if the data set is not linearly separable, the perceptron algorithm will not converge to a perfect solution. If we are willing to accept some classification errors, we can use the perceptron algorithm to find a separating hyperplane in a finite number of passes through the data set, even if the data set is not linearly separable.

## References

1. Rosenblatt F. Perceptron Simulation Experiments. Proceedings of the IRE. 1960;48(3):301–309. doi:10.1109/JRPROC.1960.287598.