Machine Learning (CE 40717)

Fall 2025

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

November 1, 2025

Binary Classification Problem

- Consider a **binary classification** task:
  - Email classification: Spam / Not Spam
  - Online transactions: Fraudulent / Genuine
  - Tumor diagnosis: Malignant / Benign

Define the target variable formally:

$$y \in \{0, 1\}, \quad \begin{cases} 0 & \text{Negative class (e.g., benign tumor)} \\ 1 & \text{Positive class (e.g., malignant tumor)} \end{cases}$$

Linear Regression for Classification

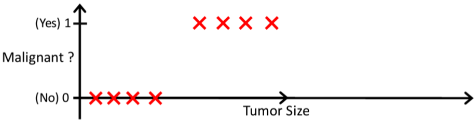
- A natural approach is to use linear regression:

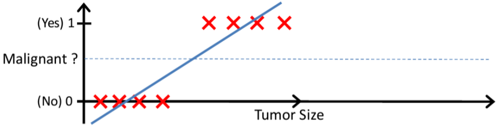
$$h_\theta(x) = \theta_0 + \theta_1 x$$

and define a threshold at 0.5 for prediction:

$$\hat{y} = \begin{cases} 1, & h_\theta(\mathbf{x}) \geq 0.5 \\ 0, & h_\theta(\mathbf{x}) < 0.5 \end{cases}$$

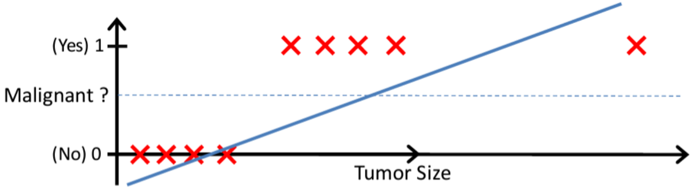
- Here, $\hat{y}$ is the **predicted class label** (i.e., the model's guess for y). It converts the continuous output of $h_\theta(\mathbf{x})$ into a discrete class (0 or 1).





Limitations of Linear Regression for Classification

- The model's output, $h_\theta(\mathbf{x})$, is unbounded and can produce predictions greater than 1 or less than 0.
- Linear regression does not provide probabilistic outputs.
- The decision boundary may be highly sensitive to outliers.



Requirement: $0 \leq h_\theta(\mathbf{x}) \leq 1$

Limitations of Perceptron with Step Activation for Classification

- As we saw earlier, perceptron Uses a step activation

$$f(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

- The step function has **slope = 0** almost everywhere (non-differentiable at 0).
- No gradient means it cannot use gradient-based optimization.
- Produces hard class labels (0/1), not probabilities, thus we cannot measure model confidence — probabilistic outputs are preferred.

Properties of a Good Classifier

Therefore, we prefer a model that:

- Outputs **probabilities** for each class, so it has **bounded outputs** in $[0, 1]$, making it less sensitive to outliers.
- **Shows model confidence** through its probabilistic output.
- Has a **differentiable, convex objective with non-zero derivative**, allowing efficient optimization with gradient-based methods.

Introduction

- Suppose we have a binary classification task (so $K = 2$).
- By observing age, gender, height, weight and BMI we try to distinguish if a person is overweight or not overweight.

Age	Gender	Height (cm)	Weight (kg)	BMI	Overweight
25	Male	175	80	25.3	0
30	Female	160	60	22.5	0
		...			
35	Male	180	90	27.3	1

- We denote the features of a sample with vector x and the label with y.
- In logistic regression we try to find an $\sigma(\mathbf{w}^\top \mathbf{x})$ which predicts **posterior** probabilities $\mathbb{P}(y = 1|\mathbf{x})$.

Introduction (cont.)

- $\sigma(\mathbf{w}^\top x)$: probability that $y = 1$ given x (parameterized by $\mathbf{w}$)

$$\mathbb{P}(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^\top x)$$
$$\mathbb{P}(y = 0|\mathbf{x}, \mathbf{w}) = 1 - \sigma(\mathbf{w}^\top x)$$

- We need to look for a function which gives us an output in the range $[0, 1]$. (like a probability).
- Let's denote this function with $\sigma(.)$ and call it the **activation function**.

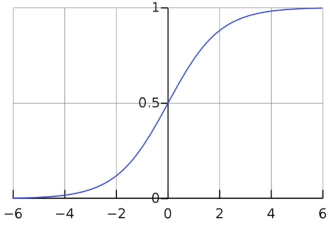
Introduction (cont.)

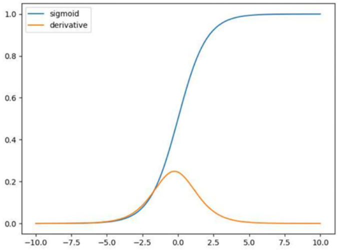
- Sigmoid (logistic) function:

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

- Smooth, bounded in $(0,1)$, and **differentiable**.

$$\sigma'(z) = \frac{e^{-z}}{(1+e^{-z})^2} = \underbrace{\frac{1}{1+e^{-z}}}_{\sigma(z)} \underbrace{\frac{e^{-z}}{1+e^{-z}}}_{1-\sigma(z)}$$
$$= \sigma(z)\big(1-\sigma(z)\big)$$





Introduction (cont.)

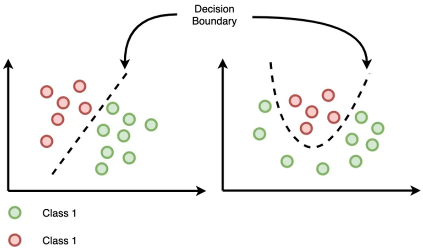
- The sigmoid function takes a number as input but we have:

$$\mathbf{x} = [x_0 = 1, x_1, \ldots, x_d]$$
$$\mathbf{w} = [w_0, w_1, \ldots, w_d]$$

- So we can use the **dot product** of $\mathbf{x}$ and $\mathbf{w}$.
- We have $0 \leq \sigma(\mathbf{w}^\top \mathbf{x}) \leq 1$. which is the estimated probability of $y = 1$ on input x.
- An Example : A basketball game (Win, Lose)
  - $\sigma(\mathbf{w}^\top \mathbf{x}) = 0.7$
  - In other terms 70 percent chance of winning the game.

Decision surface

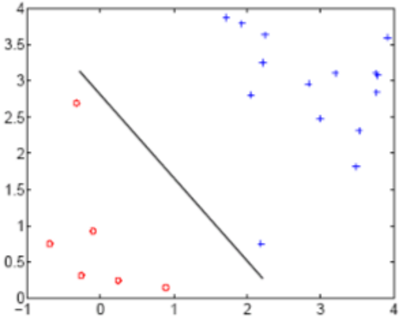
- Decision surface or decision boundary is the region of a problem space in which the output label of a classifier is ambiguous. (could be linear or non-linear)
- In binary classification it is where the probability of a sample belonging to each $y = 0$ and $y = 1$ is equal.



- Decision boundary hyperplane always has **one less dimension** than the feature space.

Decision surface (cont.)

- An example of linear decision boundaries:



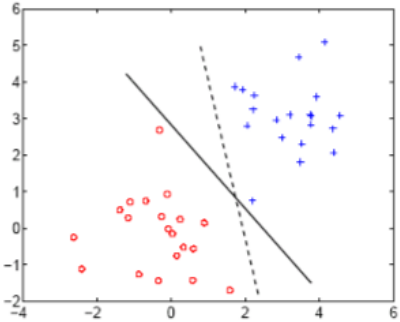


Figure adapted from Eric Xing, Machine Learning, CMU

Decision surface (cont.)

- Back to our logistic regression problem.
- Decision surface $\sigma(\mathbf{w}^\top \mathbf{x})$ = **constant**.

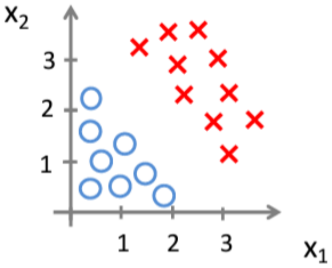
$$\sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x})}} = 0.5$$

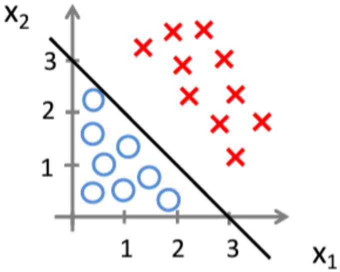
- Decision surfaces are **linear functions** of x
  - if $\sigma(\mathbf{w}^\top \mathbf{x}) \geq 0.5$ then $\hat{y} = 1$, else $\hat{y} = 0$
  - Equivalently, since $\sigma(z) \geq 0.5$ only when $z \geq 0$, this means:
    - if $\mathbf{w}^\top \mathbf{x} \geq 0$ then decide $\hat{y} = 1$, else $\hat{y} = 0$

$\hat{y}$ is the predicted label

Decision boundary example

$$\sigma(\mathbf{w}^\top \mathbf{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$$



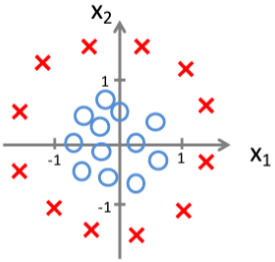


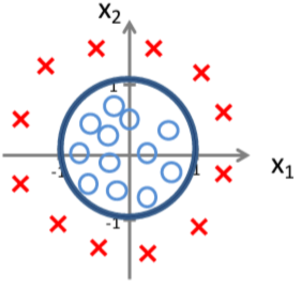
Predict $y = 1$ if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundary example

$$\sigma(\mathbf{w}^\top \mathbf{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$$

We can learn more complex decision boundaries when having higher order terms





Predict $y = 1$ if $-1 + x_1^2 + x_2^2 \geq 0$

Maximum Likelihood Estimation (MLE)

- For n independent samples, the likelihood is:

$$\mathscr{L}(\mathbf{w}) = \prod_{i=1}^{n} \mathbb{P}(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w})$$

- For binary classification ($y \in \{0, 1\}$):

$$\mathbb{P}(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})^{y^{(i)}} (1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}))^{1-y^{(i)}}$$

- This compact form works because $y^{(i)} \in \{0, 1\}$:
  - If $y^{(i)} = 1$, the expression becomes $P(y = 1|\mathbf{x}^{(i)}, \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})$.
  - If $y^{(i)} = 0$, the expression becomes $P(y = 0|\mathbf{x}^{(i)}, \mathbf{w}) = 1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})$.
- Log-likelihood:

$$\log \mathscr{L}(\mathbf{w}) = \sum_{i=1}^{n} \left[y^{(i)} \log \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})) \right]$$

From Likelihood to Cost Function

- Maximizing the likelihood ⇔ minimizing the negative log-likelihood (NLL):

$$J_{\text{MLE}}(\mathbf{w}) = -\log \mathscr{L}(\mathbf{w})$$

- Can be written as an integral over the data distribution:

$$J_{\text{MLE}}(\mathbf{w}) = -\int \mathbb{P}(\mathbf{x}, y) \log \mathbb{P}(y|\mathbf{x}, \mathbf{w})\, d\mathbf{x}\, dy$$

- Empirical estimate (training data):

$$J_{\text{MLE}}(\mathbf{w}) = -\frac{1}{n} \sum_{i=1}^{n} \log \mathbb{P}(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w})$$

- This is exactly the **cross-entropy loss** used in classification.

Maximum A Posteriori Estimation (MAP)

- MAP incorporates prior knowledge about parameters:

$$\mathbf{w}_{\text{MAP}} = \arg\max_{\mathbf{w}} \mathbb{P}(\mathbf{w}|D)$$

- Using Bayes' rule:

$$\mathbb{P}(\mathbf{w}|D) \propto P(D|\mathbf{w})P(\mathbf{w})$$

- Equivalently:

$$\mathbf{w}_{\text{MAP}} = \arg\max_{\mathbf{w}} \left[\log \mathbb{P}(D|\mathbf{w}) + \log \mathbb{P}(\mathbf{w})\right]$$

- Cost function:

$$J_{\text{MAP}}(\mathbf{w}) = -\sum_{i=1}^{n} \log \mathbb{P}(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) - \log \mathbb{P}(\mathbf{w})$$

MAP with Gaussian Prior (L2 Regularization)

- Gaussian prior: $\mathbf{w} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$

$$\mathbb{P}(\mathbf{w}) \propto \exp\left(-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}\right)$$

- MAP cost:

$$J_{\text{MAP}}(\mathbf{w}) = J_{\text{MLE}}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

- L2 penalty encourages **weight shrinkage** (smooth regularization, no sparsity).

MAP with Laplace Prior (L1 Regularization)

- Laplace prior: $\mathbf{w} \sim \text{Laplace}(0, b)$

$$\mathbb{P}(\mathbf{w}) \propto \exp\left(-\frac{\|\mathbf{w}\|_1}{b}\right)$$

- MAP cost:

$$J_{\text{MAP}}(\mathbf{w}) = J_{\text{MLE}}(\mathbf{w}) + \lambda \|\mathbf{w}\|_1$$

- L1 penalty encourages **sparsity** (feature selection).

Ridge vs. Lasso (Recap)

Ridge Regression (L2)	**Lasso Regression (L1)**
Constraint: $\|\mathbf{w}\|_2^2 \le t$ (L2 ball).	Constraint: $\|\mathbf{w}\|_1 \le t$ (L1 ball).
Contours of $J(\mathbf{w})$ typically touch the constraint boundary at smooth points.	Contours often hit the sharp **corners** of the diamond.
Produces small but nonzero coefficients (no exact sparsity).	Produces **sparse** solutions (some coefficients exactly zero).
Differentiable everywhere; the regularizer $\|\mathbf{w}\|_2^2$ is smooth.	Not differentiable at $w_i = 0$; $\|\mathbf{w}\|_1$ has sharp corners.
Strongly convex $\Rightarrow$ guarantees a **unique optimum**.	Not strongly convex $\Rightarrow$ may yield **multiple optima**.

Gradient descent

- Remember from previous slides:

$$J(\mathbf{w}) = \sum_{i=1}^{n} -y^{(i)} \log(\sigma(\mathbf{w}^\top \mathbf{x}^{(i)})) - (1 - y^{(i)}) \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}))$$

- Update rule for **gradient descent**:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla_{\mathbf{w}} J(\mathbf{w}^t)$$

- With $J(\mathbf{w})$ definition for logistic regression we get:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{i=1}^{n} (\sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) - y^{(i)}) \mathbf{x}^{(i)}$$

Gradient descent (cont.)

Let's find the derivative from the previous slide step by step:

- For one sample, we have:

$$J_i = -y^{(i)} \log \sigma(z^{(i)}) - (1 - y^{(i)}) \log(1 - \sigma(z^{(i)})), \quad z^{(i)} = \mathbf{w}^\top \mathbf{x}^{(i)}$$

- Differentiate w.r.t. $z^{(i)}$:

$$\frac{\partial J_i}{\partial z^{(i)}} = -\frac{y^{(i)}}{\sigma(z^{(i)})}\sigma'(z^{(i)}) + \frac{1 - y^{(i)}}{1 - \sigma(z^{(i)})}\sigma'(z^{(i)}) = \sigma(z^{(i)}) - y^{(i)}$$

- By chain rule we get:

$$\frac{\partial J_i}{\partial \mathbf{w}} = \frac{\partial J_i}{\partial z^{(i)}} \cdot \frac{\partial z^{(i)}}{\partial \mathbf{w}} = (\sigma(z^{(i)}) - y^{(i)})\,\mathbf{x}^{(i)}$$

- Summing over all samples gives the result:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{i=1}^{n} (\sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) - y^{(i)})\,\mathbf{x}^{(i)}$$

Gradient descent (cont.)

- Compare the gradient of logistic regression with the gradient of SSE in linear regression :

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{i=1}^{n} (\sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) - y^{(i)}) \mathbf{x}^{(i)}$$

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{i=1}^{n} (\mathbf{w}^\top \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

Loss function

- Loss function is a single overall measure of loss incurred for taking our decisions (over entire dataset).
- This is the **cross-entropy** (or log) loss for a single sample:

$$J(y, \sigma(\mathbf{w}^\top \mathbf{x})) = -y \log(\sigma(\mathbf{w}^\top \mathbf{x})) - (1-y)\log(1-\sigma(\mathbf{w}^\top \mathbf{x}))$$

- Since in binary classification $y \in \{0, 1\}$, the loss simplifies:

$$J(y, \sigma(\mathbf{w}^\top \mathbf{x})) = \begin{cases} -\log(\sigma(\mathbf{w}^\top \mathbf{x})) & \textbf{if } y = 1 \\ -\log(1-\sigma(\mathbf{w}^\top \mathbf{x})) & \textbf{if } y = 0 \end{cases}$$

Convexity of Cross-Entropy Loss

This is the logistic regression loss over the dataset:

$$J(\mathbf{w}) = \sum_{i=1}^{n} \Big[-y^{(i)} \log \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) - (1 - y^{(i)}) \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})) \Big],$$

Define $\hat{y}^{(i)} = \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})$. Then the gradient is:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})\, \mathbf{x}^{(i)}.$$

The Hessian (second derivative matrix) is:

$$\nabla^2_{\mathbf{w}} J(\mathbf{w}) = \sum_{i=1}^{n} \hat{y}^{(i)} (1 - \hat{y}^{(i)})\, \mathbf{x}^{(i)} (\mathbf{x}^{(i)})^\top = \mathbf{X}^\top \mathbf{D} \mathbf{X},$$

where $\mathbf{D} = \operatorname{diag}(\hat{y}^{(i)}(1 - \hat{y}^{(i)}))$ and each $\hat{y}^{(i)}(1 - \hat{y}^{(i)}) \geq 0$. For any vector $\mathbf{v}$,

$$\mathbf{v}^\top \nabla^2_{\mathbf{w}} J(\mathbf{w})\, \mathbf{v} = \sum_{i=1}^{n} \hat{y}^{(i)} (1 - \hat{y}^{(i)})\, (\mathbf{v}^\top \mathbf{x}^{(i)})^2 \geq 0.$$

Hence, the Hessian is **positive semidefinite (PSD)**, and $J(\mathbf{w})$ is **convex**.

Loss function (cont.)

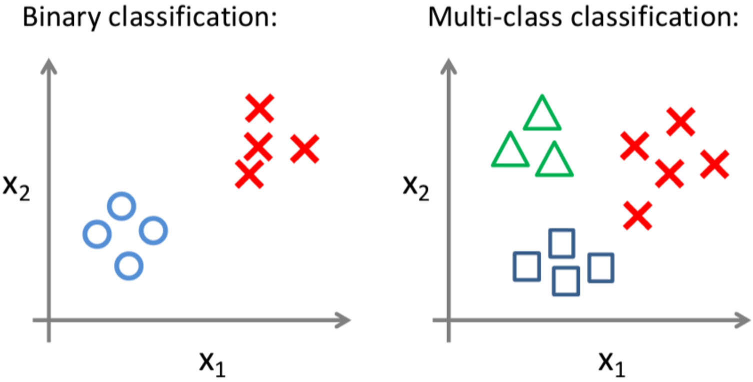
- Note that this is different from the **zero-one loss**, which simply counts misclassifications:

$$J_{0-1}(y, \hat{y}) = \begin{cases} 1 & \textbf{if } y \neq \hat{y} \\ 0 & \textbf{if } y = \hat{y} \end{cases}$$

  ($\hat{y}$ is the predicted label and y is the true label)
- We use cross-entropy (logistic loss) instead of zero-one loss because the zero-one loss function is **non-differentiable** and **non-convex**.
- The cross-entropy loss is a smooth, convex, and differentiable substitute, which allows us to use optimization methods like gradient descent.

Multi-class logistic regression

- Now consider a problem where we have K classes and every sample only belongs to one class (for simplicity).



Multi-class logistic regression (cont.)

- For each class k, $\sigma_k(\mathbf{x};\mathbf{W})$ predicts the probability of $y = k$.
  - i.e., $\mathbb{P}(y = k|\mathbf{x},\mathbf{W})$
- For each data point $\mathbf{x}$, $\sum_{k=1}^{K} \mathbb{P}(y = k|\mathbf{x},\mathbf{W})$ must be 1
  - $\mathbf{W}$ denotes a matrix of $\mathbf{w}_i$'s in which each $\mathbf{w}_i$ is a weight vector dedicated for class label i.
- On a new input x, to make a prediction, we pick the class that maximizes $\sigma_k(\mathbf{x};\mathbf{W})$:

$$\alpha(\mathbf{x}) = \underset{k=1,\dots,K}{\arg\max}\ \sigma_k(\mathbf{x};\mathbf{W})$$

$$\textbf{if } \sigma_k(\mathbf{x};\mathbf{W}) > \sigma_j(\mathbf{x};\mathbf{W})\ \forall j \neq k \textbf{ then decide } C_k$$

Multi-class logistic regression (cont.)

- $K > 2$ and $y \in \{1, 2, \ldots, K\}$

$$\sigma_k(\mathbf{x}, \mathbf{W}) = \mathbb{P}(y = k|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_k^\top \mathbf{x}\right)}{\sum_{j=1}^{K} \exp\left(\mathbf{w}_j^\top \mathbf{x}\right)}$$

- Normalized exponential (Aka **Softmax**)
- if $\mathbf{w}_k^\top \mathbf{x} \gg \mathbf{w}_j^\top \mathbf{x}$ for all $j \neq k$ then $\mathbb{P}(C_k|\mathbf{x}) \approx 1$ and $\mathbb{P}(C_j|x) \approx 0$
- Note : remember from Bayes theorem:

$$\mathbb{P}(C_k|x) = \frac{\mathbb{P}(\mathbf{x}|C_k)\mathbb{P}(C_k)}{\sum_{j=1}^{K} \mathbb{P}(\mathbf{x}|C_j)\mathbb{P}(C_j)}$$

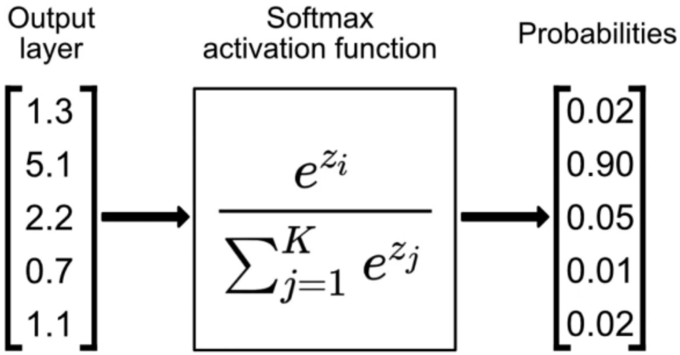
Multi-class logistic regression (cont.)

- Softmax function **smoothly** highlights the maximum probability and is differentiable.
- Compare it with max(.) function which is strict and non-differentiable
- Softmax can also handle negative values because we are using exponential function
- And it gives us probability for each class since:

$$\sum_{k=1}^{K} \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})} = 1$$

Multi-class logistic regression (cont.)

- An example of applying softmax (note that $z_i = \mathbf{w}^\top \mathbf{x}^{(i)}$):



Multi-class logistic regression (cont.)

- Again we set $J(\mathbf{W})$ as negative of log likelihood.
- We need $\mathbf{W}_{\text{MLE}} = \underset{\mathbf{W}}{\arg\min}\ J(\mathbf{W})$

$$
\begin{aligned}
J(W) &= -\log \prod_{i=1}^{n} \mathbb{P}(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}, \mathbf{W}) \\
&= -\log \prod_{i=1}^{n} \prod_{k=1}^{K} \sigma_k(\mathbf{x}^{(i)}; \mathbf{W})^{y_k^{(i)}} \\
&= -\sum_{i=1}^{n} \sum_{k=1}^{K} y_k^{(i)} \log(\sigma_k(\mathbf{x}^{(i)}; \mathbf{W}))
\end{aligned}
$$

- If **i-th** sample belongs to class k then $y_k^{(i)}$ is 1 else 0.
- Again no closed-from solution for $\mathbf{W}_{\text{MLE}}$

Multi-class logistic regression (cont.)

- From previous slides we have:

$$J(\mathbf{W}) = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_k^{(i)} \log(\sigma_k(\mathbf{x}^{(i)};\mathbf{W}))$$

- In which:

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K], \quad \mathbf{Y} = \begin{pmatrix} \mathbf{y}^{(1)} \\ \mathbf{y}^{(2)} \\ \vdots \\ \mathbf{y}^{(n)} \end{pmatrix} = \begin{pmatrix} y_1^{(1)} & \cdots & y_K^{(1)} \\ y_1^{(2)} & \cdots & y_K^{(2)} \\ \vdots & \ddots & \vdots \\ y_1^{(n)} & \cdots & y_K^{(n)} \end{pmatrix}$$

- $\mathbf{y}$ is a vector of length K (1-of-K encoding)
  - For example $\mathbf{y} = [0, 0, 1, 0]^\top$ when the target class is C_3.

Multi-class logistic regression (cont.)

- Update rule for gradient descent:

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t - \eta \nabla_{\mathbf{W}} J(\mathbf{W}^t)$$

$$\nabla_{\mathbf{w}_j} J(\mathbf{W}) = \sum_{i=1}^{n} (\sigma_j(\mathbf{x}^{(i)}; \mathbf{W}) - y_j^{(i)})\mathbf{x}^{(i)}$$

- $\mathbf{w}_j^t$ denotes the weight vector for class j (since in multi-class LR, each class has its own weight vector) in the t-th iteration

Contributions

- **These slides are authored by:**
  - Alireza Mirrokni
  - Danial Gharib
  - Amir Malek Hosseini
  - Aida Jalali

[1] M. Soleymani Baghshah, “Machine learning.” Lecture slides.

[2] A. Ng, “Ml-005, lecture 6.” Lecture slides.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Information Science and Statistics, New York, NY: Springer, 1 ed., Aug. 2006.

[4] S. Fidler, “Csc411.” Lecture slides.

[5] A. Ng and T. Ma, *CS229 Lecture Notes*. Updated June 11, 2023.