ML4 B&I: Introduction to Machine Learning

Lecture 4- Linear Models for Classification

Murat A. Erdogdu

Vector Institute, Fall 2022

Outline

Introducing Binary Linear Classification

- Is this a spam email or not?
- **classification:** predict a discrete-valued target given a D-dimensional input $\mathbf{x} \in \mathbb{R}^D$

Introducing Binary Linear Classification

- Is this a spam email or not?
- **classification:** predict a discrete-valued target given a D-dimensional input $\mathbf{x} \in \mathbb{R}^D$
- **linear:** prediction y is a linear function of $\mathbf{x}$, followed by a threshold r:

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

$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

Introducing Binary Linear Classification

- Is this a spam email or not?
- **classification:** predict a discrete-valued target given a D-dimensional input $\mathbf{x} \in \mathbb{R}^D$
- **linear:** prediction y is a linear function of $\mathbf{x}$, followed by a threshold r:

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

$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

- **binary:** predict a binary target $t \in \{0, 1\}$
  - $t = 1$ is class 1
  - $t = 0$ is class 2.
  - $t \in \{0, 1\}$ or $t \in \{-1, +1\}$ is for notational convenience.

Simplified Model

- Trainable parameters: $\mathbf{w}, b, r$.
- Eliminating the threshold r (i.e. $r = 0$):

$$\mathbf{w}^\top \mathbf{x} + b \geq r \quad \Longleftrightarrow \quad \mathbf{w}^\top \mathbf{x} + \underbrace{b - r}_{\triangleq b'} \geq 0.$$

Simplified Model

- Trainable parameters: $\mathbf{w}, b, r$.
- Eliminating the threshold r (i.e. $r = 0$):

$$\mathbf{w}^\top \mathbf{x} + b \geq r \quad \Longleftrightarrow \quad \mathbf{w}^\top \mathbf{x} + \underbrace{b - r}_{\triangleq b'} \geq 0.$$

- Further simplifying the notation:

  Add a dummy feature $x_0 = 1$. The weight $w_0 = b$ is equivalent to a bias (same as linear regression)
- Simplified model

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

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Modeling Simple Logical Functions

- Examples: NOT, AND.
- Next lecture: XOR

Modeling NOT (Negation)

NOT

x_0	x_1	t
1	0	1
1	1	0

$$z = w_0 x_0 + w_1 x_1$$

$$y = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases}$$

- Derive two sets of values for w_0, w_1 to classify NOT.
- Which conditions on w_0, w_1 guarantee perfect classification?

Modeling NOT - Solutions

NOT

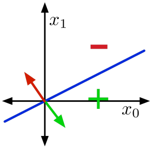
x_0	x_1	t
1	0	1
1	1	0

$$z = w_0 x_0 + w_1 x_1$$

$$\Rightarrow w_0 \geq 0$$

$$\Rightarrow w_0 + w_1 < 0 \text{ or } w_1 < -w_0$$

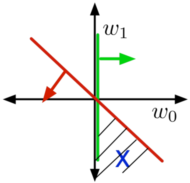
Visualizing NOT in Data Space



x_0	x_1	t
1	0	1
1	1	0

- Each training example is a point in data space.
- Data is linearly separable if a linear decision rule can perfectly separate the training examples.

Visualizing NOT in Weight Space



$w_0 \geq 0$

$w_0 + w_1 < 0$

- Each point is a set of values for the weights $\mathbf{w}$.
- Each training example $\mathbf{x}$ specifies a half-space that $\mathbf{w}$ must lie in to guarantee correct classification.
- The feasible region satisfies all the constraints.
  The problem is feasible if the feasible region is nonempty.

Modeling AND

AND

x_0	x_1	x_2	t
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$$z = w_0 x_0 + w_1 x_1 + w_2 x_2$$

$$y = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases}$$

Modeling AND - Solutions

AND

x_0	x_1	x_2	t
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

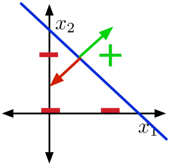
$$z = w_0x_0 + w_1x_1 + w_2x_2$$
$$\Rightarrow w_0 < 0$$
$$\Rightarrow w_0 + w_2 < 0$$
$$\Rightarrow w_0 + w_1 < 0$$
$$\Rightarrow w_0 + w_1 + w_2 \geq 0$$

Example solution: $w_0 = -1.5,\ w_1 = 1,\ w_2 = 1$

Visualizing AND in Data and Weight Spaces

Let's look at a 2-D slice of the 3-D data and weight spaces for AND.

Data Space

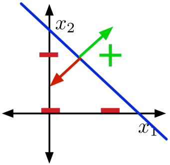


- Fix $x_0 = 1$
- Example Solution:
  $w_0 = -1.5,\ w_1 = 1,\ w_2 = 1$
- Decision Boundary:
  $w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$
  $\implies -1.5 + x_1 + x_2 = 0$

Visualizing AND in Data and Weight Spaces

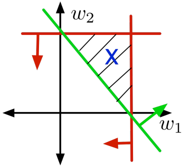
Let's look at a 2-D slice of the 3-D data and weight spaces for AND.

Data Space



- Fix $x_0 = 1$
- Example Solution: $w_0 = -1.5$, $w_1 = 1$, $w_2 = 1$
- Decision Boundary: $w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$ $\implies -1.5 + x_1 + x_2 = 0$

Weight Space



- Fix $w_0 = -1.5$
- The constraints:
  $w_0 < 0$
  $w_0 + w_2 < 0$
  $w_0 + w_1 < 0$
  $w_0 + w_1 + w_2 \geq 0$

Learning the Weights for Linearly Separable Data

Binary Linear Classification

$$z = \mathbf{w}^\top \mathbf{x}$$
$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

If data is linearly separable, we can learn the weights

- using linear programming, or
- using the perceptron algorithm (primarily of historical interest).

Unfortunately, in real life, data is almost never linearly separable.

Data Isn't Linearly Separable

What if the data-set isn't linearly separable?

- We follow the standard ML pipeline:
- Define a loss function.
- Find weights that minimize the average loss over the training examples.

First Try: 0-1 Loss

Binary linear classification with 0-1 loss:

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

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

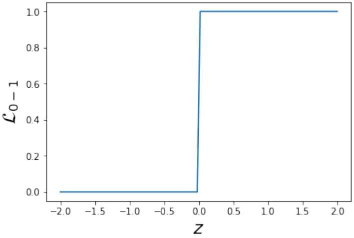
$$\mathcal{L}_{0-1}(y,t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases} = \mathbb{I}[y \neq t]$$

The cost $\mathcal{J}$ is the misclassification rate over data $\{\mathbf{x}^{(i)}, t^{(i)}\}_{i=1}^N$:

$$\mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[y^{(i)} \neq t^{(i)}]$$

Problems with 0-1 loss

- We need to minimize the cost with gradient descent.
- But, the gradient is zero almost everywhere!
  Changing the weights has no effect on the loss.
- Also, 0-1 loss is discontinuous at $z = 0$,
  where the gradient is undefined.



Second Try: Squared Loss for Linear Regression

- Choose an easier to optimize loss function.
- How about the squared loss for linear regression?

$$z = \mathbf{w}^\top \mathbf{x}$$
$$\mathcal{L}_{\mathrm{SE}}(z, t) = \frac{1}{2}(z - t)^2$$

Second Try: Squared Loss for Linear Regression

- Choose an easier to optimize loss function.
- How about the squared loss for linear regression?

$$z = \mathbf{w}^\top \mathbf{x}$$
$$\mathcal{L}_{\mathrm{SE}}(z, t) = \frac{1}{2}(z - t)^2$$

- Treat the binary targets as continuous values.

Second Try: Squared Loss for Linear Regression

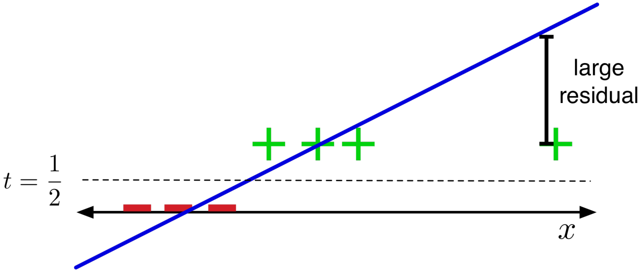
- Choose an easier to optimize loss function.
- How about the squared loss for linear regression?

$$z = \mathbf{w}^\top \mathbf{x}$$
$$\mathcal{L}_{\text{SE}}(z, t) = \frac{1}{2}(z - t)^2$$

- Treat the binary targets as continuous values.
- Make final predictions y by thresholding z at $\frac{1}{2}$.

Problems with Squared Loss

- If $t = 1$, a greater loss for $z = 10$ than $z = 0$.
- Making a correct prediction with high confidence should be good, but incurs a large loss.

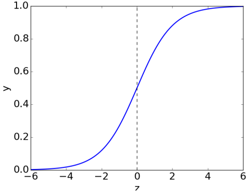


Third Try: Logistic Activation Function

For binary targets, no reason to predict values outside $[0, 1]$.
Let's squash predictions y into $[0, 1]$.

The logistic function is a sigmoid (S-shaped) function.

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

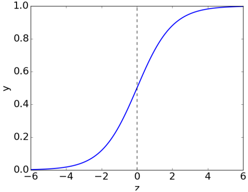


Third Try: Logistic Activation Function

For binary targets, no reason to predict values outside $[0, 1]$.
Let's squash predictions y into $[0, 1]$.

The logistic function is a sigmoid (S-shaped) function.

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



This results in a linear model with a logistic non-linearity:

$$z = \mathbf{w}^\top \mathbf{x}$$
$$y = \sigma(z)$$
$$\mathcal{L}_{\text{SE}}(y, t) = \frac{1}{2}(y - t)^2.$$

σ is called an activation function.

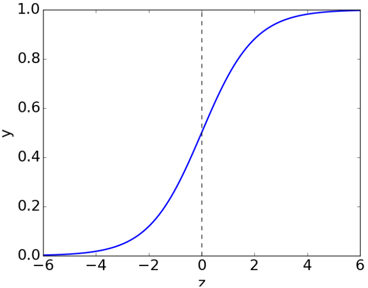
Problems with Logistic Activation Function

Suppose that $t = 1$ and z is very negative ($z \ll 0$).

Then, the prediction $y \approx 0$ is really wrong.

However, the weights appears to be at a critical point:

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_j} \approx 0 \frac{\partial z}{\partial w_j} = 0$$



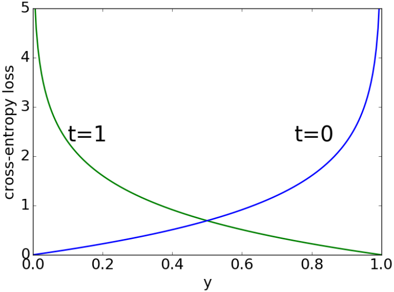
Final Try: Cross-Entropy Loss

- Interpret $y \in [0, 1]$ as the estimated probability that $t = 1$.
- Heavily penalize the extreme mis-classification cases when $t = 0, y = 1$ or $t = 1, y = 0$.

Final Try: Cross-Entropy Loss

- Interpret $y \in [0, 1]$ as the estimated probability that $t = 1$.
- Heavily penalize the extreme mis-classification cases when $t = 0, y = 1$ or $t = 1, y = 0$.
- Cross-entropy loss (a.k.a. log loss) captures this intuition:

$$\mathcal{L}_{\mathrm{CE}}(y, t) = \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1 - y) & \text{if } t = 0 \end{cases}$$
$$= -t \log y - (1 - t) \log(1 - y)$$



Logistic Regression

$$z = \mathbf{w}^\top \mathbf{x}$$
$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$
$$\mathcal{L}_{\text{CE}} = -t \log y - (1-t) \log(1-y)$$

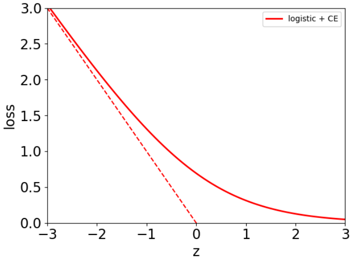


Figure: Cross-Entropy Loss w.r.t z, assuming $t = 1$

Numerical Instabilities

- Implementing logistic regression naively can cause numerical instabilities.
- Suppose that $t = 1$ and $z \ll 0$.
- If y is small enough, it may be numerically zero.
  This can cause very subtle and hard-to-find bugs.

$$z \ll 0 \Rightarrow y = \sigma(z) \approx 0$$
$$\mathcal{L}_{\text{CE}} = -t \log y - (1 - t) \log(1 - y) \approx -1 \log 0$$

Numerically Stable Version

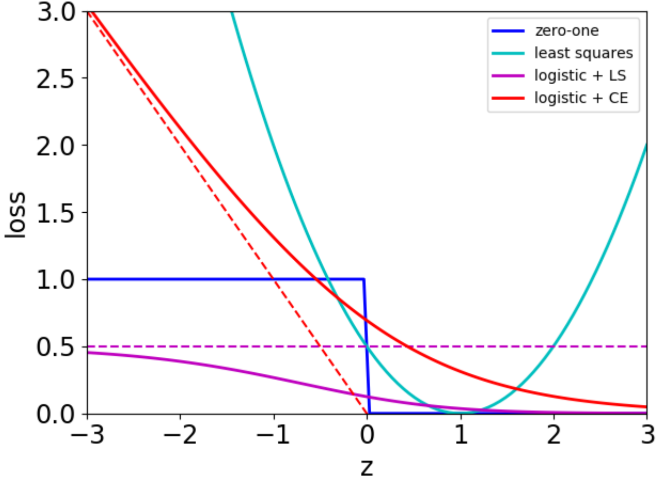
- Instead, we combine the logistic activation function and the cross-entropy loss into a single logistic-cross-entropy function.

$$\mathcal{L}_{\mathrm{LCE}}(z,t) = \mathcal{L}_{\mathrm{CE}}(\sigma(z),t) = t\log(1+e^{-z}) + (1-t)\log(1+e^{z})$$

- Numerically stable computation:

```
E = t * np.logaddexp(0, -z) + (1-t) * np.logaddexp(0, z)
```

Comparing Loss Functions for t = 1



Gradient Descent for Logistic Regression

- How do we minimize the cost $\mathcal{J}$ for logistic regression? Unfortunately, no direct solution.

- Use gradient descent
  - initialize the weights to something reasonable and repeatedly adjust them in the direction of steepest descent.
  - A standard initialization is $\mathbf{w} = 0$.

Gradient of Logistic Loss

Back to logistic regression:

$$\begin{aligned}\mathcal{L}_{\mathrm{CE}}(y,t) =& -t\log(y) - (1-t)\log(1-y)\\ y =& 1/(1+e^{-z}) \quad \text{and} \quad z = \mathbf{w}^\top \mathbf{x}\end{aligned}$$

Therefore

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial w_j} = \frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w_j} =& \left(-\frac{t}{y} + \frac{1-t}{1-y}\right) \cdot y(1-y) \cdot x_j\\ =& (y-t)x_j\end{aligned}$$

Gradient of Logistic Loss

Back to logistic regression:

$$\mathcal{L}_{\mathrm{CE}}(y,t) = -\,t\log(y) - (1-t)\log(1-y)$$
$$y = 1/(1+e^{-z}) \quad \text{and} \quad z = \mathbf{w}^\top \mathbf{x}$$

Therefore

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial w_j} = \frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w_j} = \left(-\frac{t}{y} + \frac{1-t}{1-y}\right) \cdot y(1-y) \cdot x_j$$
$$= (y-t)x_j$$

Gradient descent update for logistic regression:

$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{J}}{\partial w_j}$$
$$= w_j - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, x_j^{(i)}$$

Comparing Gradient Descent Updates

- Linear regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, \mathbf{x}^{(i)}$$

- Logistic regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, \mathbf{x}^{(i)}$$

They are both examples of generalized linear models.

Main Takeaways on Logistic Regression 1/2

What is the main motivation for using logistic regression?

Main Takeaways on Logistic Regression 1/2

What is the main motivation for using logistic regression?

- When data isn't linearly separable, cannot classify data perfectly.
- Use a loss function and minimize average loss.

Why did we try 0-1 loss first? What's the problem with it?

Main Takeaways on Logistic Regression 1/2

What is the main motivation for using logistic regression?

- When data isn't linearly separable, cannot classify data perfectly.
- Use a loss function and minimize average loss.

Why did we try 0-1 loss first? What's the problem with it?

- Natural choice for classification.
- Gradient zero almost everywhere. Has a discontinuity.

Why did we try squared loss next? What's the problem with it?

Main Takeaways on Logistic Regression 1/2

What is the main motivation for using logistic regression?

- When data isn't linearly separable, cannot classify data perfectly.
- Use a loss function and minimize average loss.

Why did we try 0-1 loss first? What's the problem with it?

- Natural choice for classification.
- Gradient zero almost everywhere. Has a discontinuity.

Why did we try squared loss next? What's the problem with it?

- Easier to optimize.
- Large penalty for a correct prediction with high confidence.

Main Takeaways on Logistic Regression 2/2

Why did we try logistic activation function next? What's the problem with it?

Main Takeaways on Logistic Regression 2/2

Why did we try logistic activation function next? What's the problem with it?

- Prediction $\in [0, 1]$.
- An extreme mis-classification case appears optimal.

Why did we try cross-entropy loss next?

Main Takeaways on Logistic Regression 2/2

Why did we try logistic activation function next? What's the problem with it?

- Prediction $\in [0, 1]$.
- An extreme mis-classification case appears optimal.

Why did we try cross-entropy loss next?

- Heavily penalizes extreme mis-classification.

How do we apply gradient descent to logistic regression?

Main Takeaways on Logistic Regression 2/2

Why did we try logistic activation function next? What's the problem with it?

- Prediction $\in [0, 1]$.
- An extreme mis-classification case appears optimal.

Why did we try cross-entropy loss next?

- Heavily penalizes extreme mis-classification.

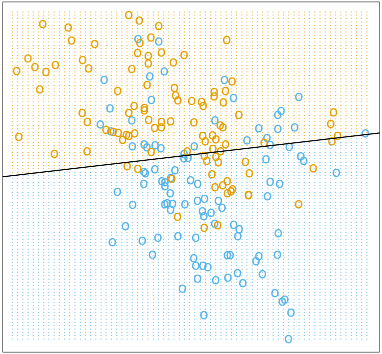
How do we apply gradient descent to logistic regression?

- Derive the update rule.

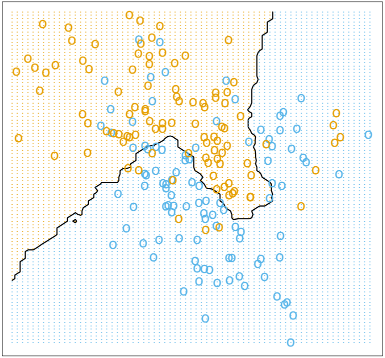
Linear Classifiers vs. KNN

Linear classifiers and KNN have very different decision boundaries:

Linear Classifier



K Nearest Neighbours



Parametric v.s. Non-Parametric Algorithms

- A parametric algorithm:
  the hypothesis space $\mathcal{H}$ is defined using a finite set of parameters.
  - Examples: linear regression, logistic regression.
  - Other examples: neural networks, Gaussian mixture models.
  - Work better in high-dimensions.
- A non-parametric algorithm:
  the hypothesis space $\mathcal{H}$ is defined in terms of the data.
  - Examples: k-nearest neighbors, decision trees.
  - Other examples: Gaussian processes, kernel density estimation
  - Suffers from curse of dimensionality

Main Takeaways on Basic Concepts

Compare and contrast KNN and Linear Classifiers.

Main Takeaways on Basic Concepts

Compare and contrast KNN and Linear Classifiers.

- kNN is a non-parametric model, logistic regression is parametric.
- kNN has nonlinear decision boundary, that of logistic regression is linear.
- We expect logistic regression to work better in high-dimensions.
- It is harder to train logistic regression.

Main Takeaways on Basic Concepts

Compare and contrast KNN and Linear Classifiers.

- kNN is a non-parametric model, logistic regression is parametric.
- kNN has nonlinear decision boundary, that of logistic regression is linear.
- We expect logistic regression to work better in high-dimensions.
- It is harder to train logistic regression.

Define parametric and non-parametric algorithms. Give examples.

Main Takeaways on Basic Concepts

Compare and contrast KNN and Linear Classifiers.

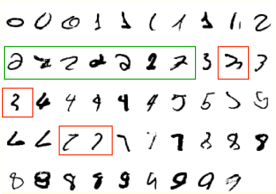
- kNN is a non-parametric model, logistic regression is parametric.
- kNN has nonlinear decision boundary, that of logistic regression is linear.
- We expect logistic regression to work better in high-dimensions.
- It is harder to train logistic regression.

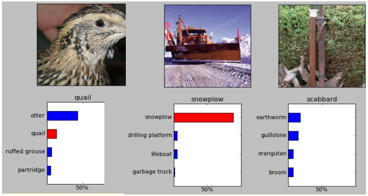
Define parametric and non-parametric algorithms. Give examples.

- A parametric algorithm has parameters (weights).
  Examples: linear regression, logistic regression.
- A non-parametric algorithm has no parameter is defined in terms of the data, and certain set of rules.
  Examples: k-nearest-neighbours, decision trees.

Multi-class Classification

Task is to predict a discrete(> 2)-valued target.





Targets in Multi-class Classification

- Targets form a discrete set $\{1, \ldots, K\}$.
- Represent targets as one-hot vectors or one-of-K encoding:

$$\mathbf{t} = \underbrace{(0, \ldots, 0, 1, 0, \ldots, 0)}_{\text{entry } k \text{ is } 1} \in \mathbb{R}^K$$

Linear Function of Inputs

Vectorized form:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b} \text{ or}$$
$$\mathbf{z} = \mathbf{W}\mathbf{x} \text{ with dummy } x_0 = 1$$

Non-vectorized form:

$$z_k = \sum_{j=1}^{D} w_{kj} x_j + b_k \quad \text{for} \quad k = 1, 2, ..., K$$

- $\mathbf{W}$: K x D matrix.
- $\mathbf{x}$: D x 1 vector.
- $\mathbf{b}$: K x 1 vector.
- $\mathbf{z}$: K x 1 vector.

Generating a Prediction

Interpret z_k as how much the model prefers the k-th prediction.

$$y_i = \begin{cases} 1, & \text{if } i = \arg\max_k z_k \\ 0, & \text{otherwise} \end{cases}$$

How does the $K = 2$ case relate to the binary linear classifiers?

Softmax Regression

- Soften the predictions for optimization.
- A natural activation function is the softmax function, a generalization of the logistic function:

$$y_k = \mathrm{softmax}(z_1, \dots, z_K)_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$$

Softmax Regression

- Soften the predictions for optimization.
- A natural activation function is the softmax function, a generalization of the logistic function:

$$y_k = \mathrm{softmax}(z_1, \ldots, z_K)_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$$

- Inputs z_k are called the logits.
- Interpret outputs as probabilities.
- If z_k is much larger than the others, then $\mathrm{softmax}(\mathbf{z})_k \approx 1$ and it behaves like argmax.

What does the $K = 2$ case look like?

Cross-Entropy as Loss Function

Use cross-entropy as the loss function.

$$\mathcal{L}_{\mathrm{CE}}(\mathbf{y}, \mathbf{t}) = -\sum_{k=1}^{K} t_k \log y_k = -\mathbf{t}^\top (\log \mathbf{y}),$$

where the log is applied element-wise.

Gradient Descent Updates for Softmax Regression

Softmax Regression:

$$
\begin{aligned}
\mathbf{z} &= \mathbf{W}\mathbf{x} \\
\mathbf{y} &= \mathrm{softmax}(\mathbf{z}) \\
\mathcal{L}_{\mathrm{CE}} &= -\mathbf{t}^{\top}(\log \mathbf{y})
\end{aligned}
$$

Gradient Descent Updates:

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial \mathbf{w}_k} &= \frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} = (y_k - t_k) \cdot \mathbf{x} \\
\mathbf{w}_k &\leftarrow \mathbf{w}_k - \alpha \frac{1}{N} \sum_{i=1}^{N} (y_k^{(i)} - t_k^{(i)}) \mathbf{x}^{(i)}
\end{aligned}
$$

Conclusions

- Introduced logistic regression, a linear classification algorithm.
- Exemplified some recurring themes
  - Can define a surrogate loss function if the one we care about is intractable.
  - Think about whether a loss function penalizes certain mistakes too much or too little.
  - Can be useful to view the classfier's output as probabilities.
- Easily generalizes to multiclass classification.