Day 4: Linear Classifiers

Summer STEM: Machine Learning

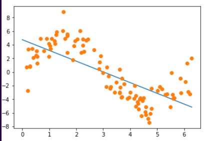
Department of Electrical and Computer Engineering
NYU Tandon School of Engineering
Brooklyn, New York

June 24, 2020

Outline

Polynomial Fitting

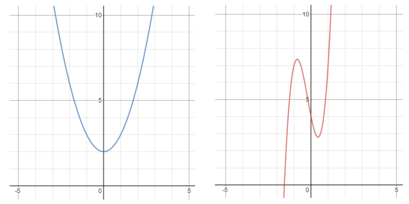
- We have been using straight lines to fit our data. But it doesn't work well every time
- Some data have more complex relation that cannot be fitted well using a straight line



- Can we use some other model to fit this data?

Polynomial Fitting

- Can we use a polynomial to fit our data?
- Polynomial: A sum of different powers of a variable
  - Examples: $y = x^2 + 2$, $y = 5x^3 - 3x^2 + 4$



NYU TANDON SCHOOL OF ENGINEERING

Polynomial Fitting

- Polynomials of x: $\hat{y} = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_M x^M$
- M is called the order of the polynomial.
- The process of fitting a polynomial is similar to linearly fitting multivariate data.

Polynomial fitting

- Rewrite in matrix-vector form
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \approx \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2 & x_2^2 & \cdots & x_2^M \\ \vdots & & \ddots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^M \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}$$
- This can still be written as
$$Y \approx X\mathbf{w}$$
- Loss $J(\mathbf{w}) = \frac{1}{N} \|Y - X\mathbf{w}\|^2$
- The i-th row of the design matrix X is simply a transformed feature $\phi(x_i) = (1, x_i, x_i^2, \cdots, x_i^M)$

Polynomial Fitting

- Original design matrix: $\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$
- Design matrix after feature transformation:
$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2 & x_2^2 & \cdots & x_2^M \\ \vdots & & \ddots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^M \end{bmatrix}$$
- For the polynomial fitting, we just added columns of features that are powers of the original feature

NYU TANDON SCHOOL OF ENGINEERING

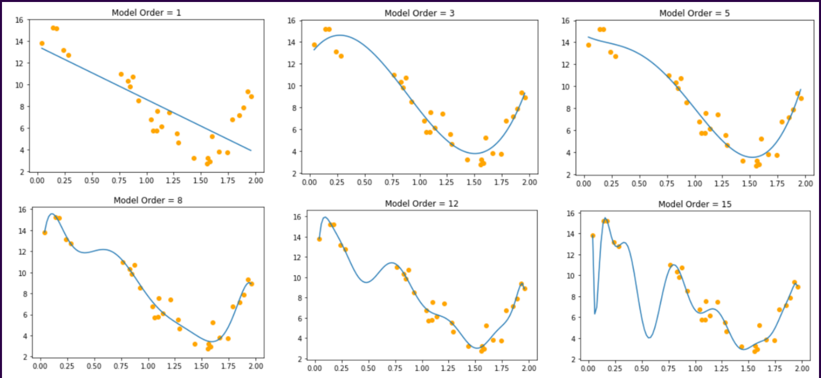
Linear Regression

- Model $\hat{y} = \mathbf{w}^T \phi(\mathbf{x})$
- Loss $J(\mathbf{w}) = \dfrac{1}{N} \|Y - X\mathbf{w}\|^2$
- Find $\mathbf{w}$ that minimizes $L(\mathbf{w})$

Overfitting

- We learned how to fit our data using polynomials of different order
- With a higher model order, we can fit the data with increasing accuracy
- As you increase the model order, at certain point it is possible find a model that fits your data perfectly (ie. zero error)
- What could be the problem?

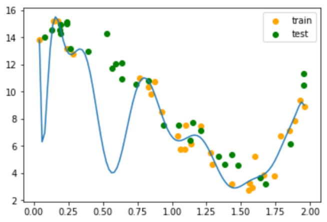
Overfitting



- Which of these model do you think is the best? Why?

Overfitting

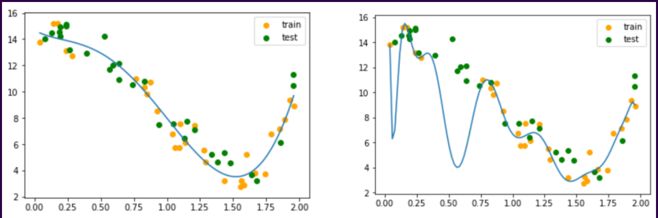
- The problem is that we are only fitting our model using data that is given
- Data usually contains noise
- When a model becomes too complex, it will start to fit the noise in the data
- What happens if we apply our model to predict some data that the model has never seen before? It will not work well.
- This is called over-fitting



NYU TANDON SCHOOL OF ENGINEERING

Overfitting

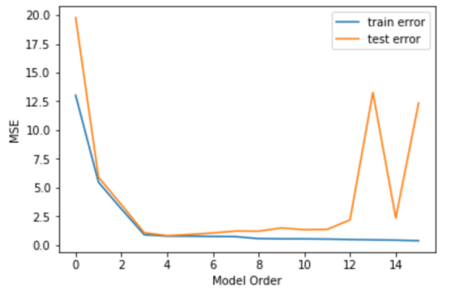
- Split the data set into a train set and a test set
- Train set will be used to train the model
- The test set will not be seen by the model during the training process
- Use test set to evaluate the model when a model is trained



- With the training and test sets shown, which one do you think is the better model now?

Train and Test Error

- Plot of train error and test error for different model order
- Initially both train and test error go down as model order increase
- But at a certain point, test error start to increase because of overfitting



Outline

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization**: methods to prevent overfitting

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization**: methods to prevent overfitting
  - We just covered regularization by model order selection

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization**: methods to prevent overfitting
  - We just covered regularization by model order selection
- Is there another way? Talk among your classmates.

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization**: methods to prevent overfitting
  - We just covered regularization by model order selection
- Is there another way? Talk among your classmates.
  - Solution: We can change our cost function.

Weight Based Regularization

- Looking back at the polynomial overfitting
- Notice that weight-size increases with overfitting

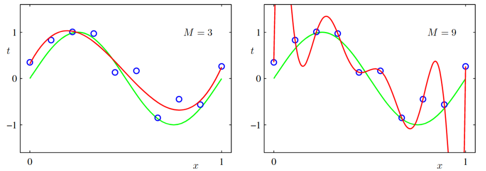


Table 1.1 Table of the coefficients $\mathbf{w}^{\star}$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^{\star}$	0.19	0.82	0.31	0.35
$w_1^{\star}$		-1.27	7.99	232.37
$w_2^{\star}$			-25.43	-5321.83
$w_3^{\star}$			17.37	48568.31
$w_4^{\star}$				-231639.30
$w_5^{\star}$				640042.26
$w_6^{\star}$				-1061800.52
$w_7^{\star}$				1042400.18
$w_8^{\star}$				-557682.99
$w_9^{\star}$				125201.43

New Cost Function

$$J(\mathbf{w}) = \frac{1}{N} \|Y - X\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2$$

- Penalize complexity by simultaneously minimizing weight values.
- We call λ a **hyper-parameter**
  - λ determines relative importance

Table 1.2 Table of the coefficients $\mathbf{w}^\star$ for $M = 9$ polynomials with various values for the regularization parameter λ. Note that $\ln \lambda = -\infty$ corresponds to a model with no regularization, i.e., to the graph at the bottom right in Figure 1.4. We see that, as the value of λ increases, the typical magnitude of the coefficients gets smaller.

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

NYU TANDON SCHOOL OF ENGINEERING

Tuning Hyper-parameters

- Motivation: never determine a hyper-parameter based on training data
- **Hyper-Parameter**: a parameter of the algorithm that is not a model-parameter solved for in optimization.
  - Ex: λ weight regularization value vs. model weights ($\mathbf{w}$)
- Solution: split dataset into three
  - **Training set**: to compute the model-parameters ($\mathbf{w}$)
  - **Validation set**: to tune hyper-parameters (λ)
  - **Test set**: to compute the performance of the algorithm (MSE)

Outline

1. Leftovers from Day 3
2. Regularization
3. Non-linear Optimization
4. Logistic Regression
5. Lab: Diagnosing Breast Cancer
6. Multiclass Classificaiton
7. Lab: Iris Dataset

Motivation

- Cannot rely on closed form solutions
  - Computation efficiency: operations like inverting a matrix is not efficient
  - For more complex problems such as neural networks, a closed-form solution is not always available
- Need an optimization technique to find an optimal solution
  - Machine learning practitioners use **gradient**-based methods

Gradient Descent Algorithm

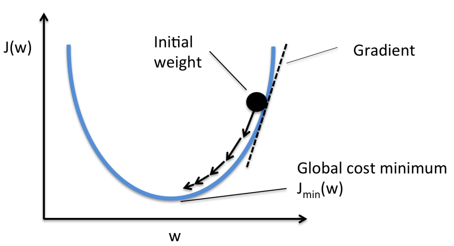
- Update Rule

  $Repeat\{$

  $$\mathbf{w}_{new} = \mathbf{w} - \alpha \nabla_{\mathbf{w}} J$$

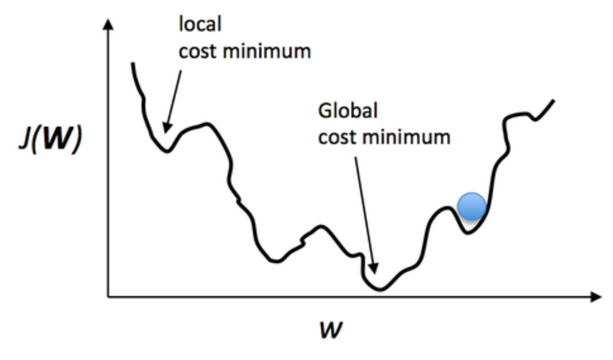
  $\}$

  α is the learning rate

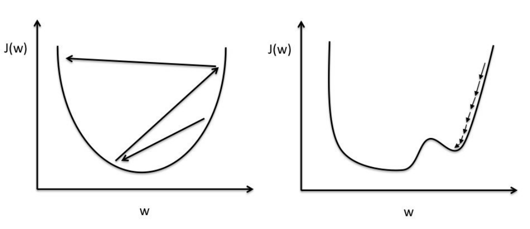


General Loss Function Contours

- Most loss function contours are not perfectly parabolic
- Our goal is to find a solution that is very close to global minimum by the right choice of hyper-parameters

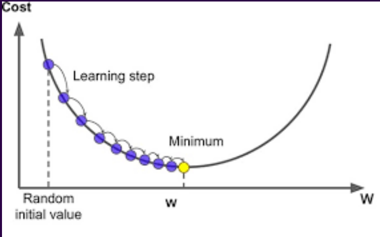


Understanding Learning Rate



Large learning rate: Overshooting.

Small learning rate: Many iterations until convergence and trapping in local minima.



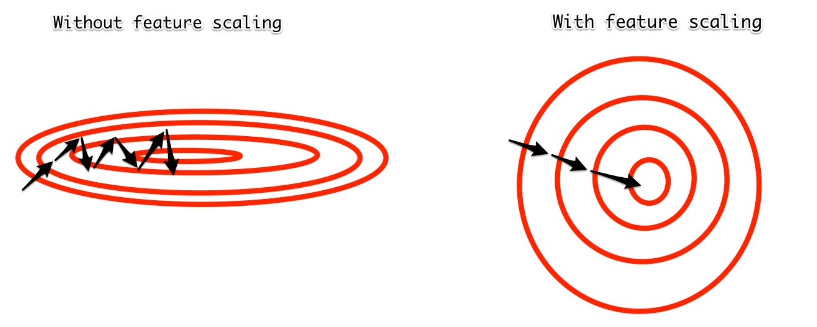
Correct learning rate

Some Animations

- Demonstrate gradient descent animation

Importance of Feature Normalization (Optional)

- Helps improve the performance of gradient based optimization



Outline

Classification Vs. Regression

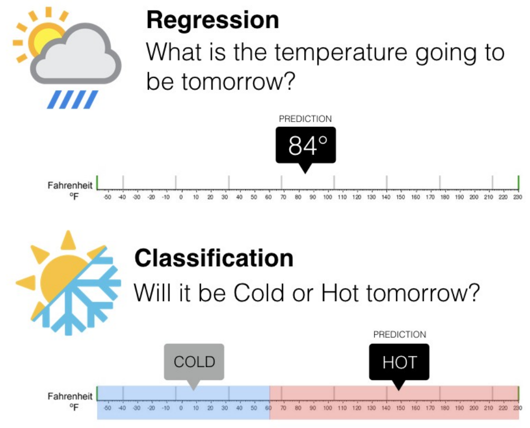
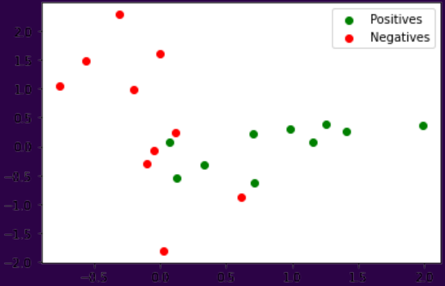


Figure: https://www.pinterest.com/pin/672232681855858622/?lp=true

Classification

Given the dataset (x_i, y_i) for $i = 1, 2, \ldots, N$, find a function $f(x)$ (model) so that it can predict the label $\hat{y}$ for some input x, even if it is not in the dataset, i.e. $\hat{y} = f(x)$.

- Positive : $y = 1$
- Negative : $y = 0$



Classification via regression

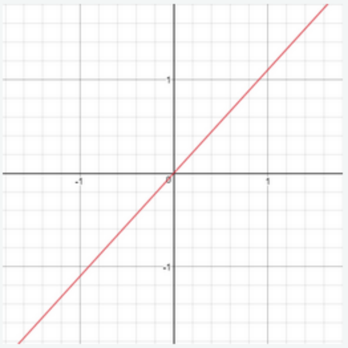
- Proposal: train a model to fit the data with linear regression (potentially with polynomial features)!

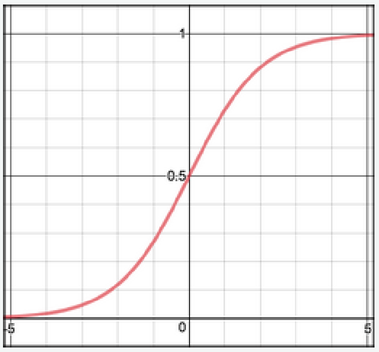
Classification via regression

- Proposal: train a model to fit the data with linear regression (potentially with polynomial features)!
- What could be the problem?

Sigmoid Function

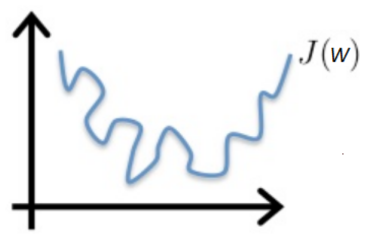
- Recall from linear regression $z = w_0 + w_1 x$
- By applying the sigmoid function to z, we enforce $0 \leq \hat{y} \leq 1$
    - $\hat{y} = sigmoid(z) = \frac{1}{1+e^{-z}}$

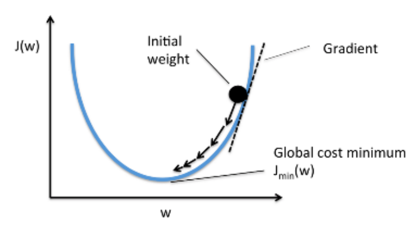




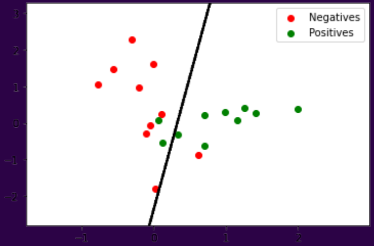
Classification Loss Function

- Cannot use the same cost function that we used for linear regression
  - MSE of a logistic function has many local minima
- Use $\frac{1}{N}\sum_{i=1}^{N}\left[-ylog(\hat{y})-(1-y)log(1-\hat{y})\right]$
  - This loss function is called binary cross entropy loss
  - This loss function has only one minimum





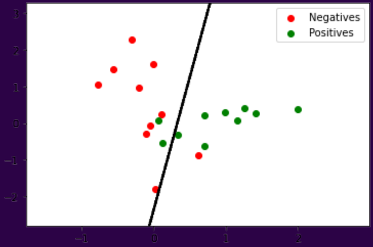
Decision Boundary



- Evaluation metric :

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}}$$

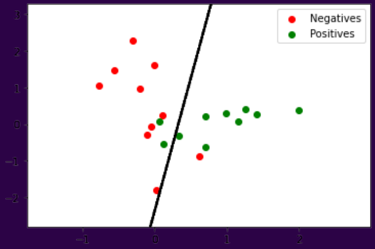
Decision Boundary



- Evaluation metric :

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}}$$

- What is the accuracy in this example ?



- Evaluation metric :

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}} = \frac{17}{20} = 0.85 = 85\%$$

Classifier

- How to deal with uncertainty ?

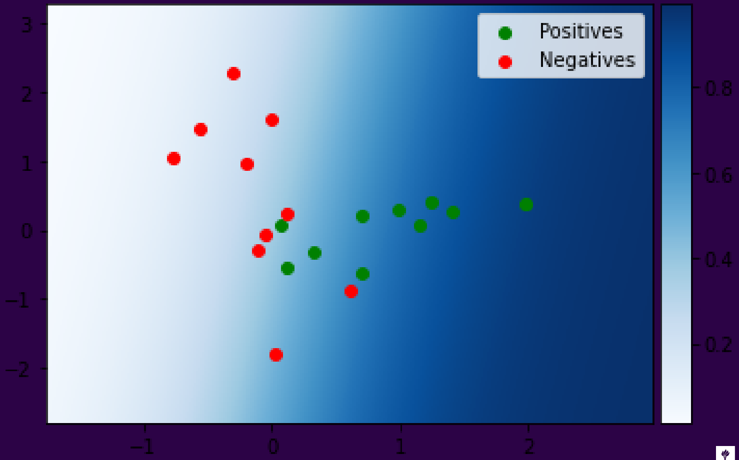
Classifier

- How to deal with uncertainty ?
  - $\hat{y} = f(x)$ should be between 0 and 1.

Classifier

- How to deal with uncertainty ?
    - $\hat{y} = f(x)$ should be between 0 and 1.
- If $\hat{y}$ is close to 0, the data is probably negative
- If $\hat{y}$ is close to 1, the data is probably positive
- If $\hat{y}$ is around 0.5, we are not sure.

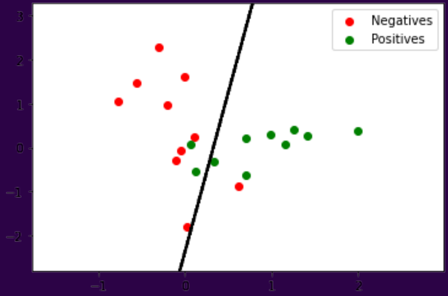
Classifier



Types of Errors in Classification

- Correct predictions:
  - True Positive (TP) : Predict $\hat{y} = 1$ when $y = 1$
  - True Negative (TN) : Predict $\hat{y} = 0$ when $y = 0$
- Two types of errors:
  - False Positive/ False Alarm (FP): $\hat{y} = 1$ when $y = 0$
  - False Negative/ Missed Detection (FN): $\hat{y} = 0$ when $y = 1$

Example



- How many True Positive (TP) are there ?
- How many True Negative (TN) are there ?
- How many False Positive (FP) are there ?
- How many False Negative (FN) are there ?

Performance metrics for a classifier

- Accuracy of a classifier:
  - (TP + TF)/(TP+FP+TN+FN) (percentage of correct classification)
- Why accuracy alone is not a good measure for assessing the model

Performance metrics for a classifier

- Accuracy of a classifier:
  - (TP + TF)/(TP+FP+TN+FN) (percentage of correct classification)
- Why accuracy alone is not a good measure for assessing the model
  - There might be an overwhelming proportion of one class over another (unbalanced classes)
  - Example: A rare disease occurs 1 in ten thousand people
  - A test that classifies everyone as free of the disease can achieve 99.999% accuracy when tested with people drawn randomly from the entire population

Other metrics

Some other metrics

- Sensitivity/Recall/TPR = TP/(TP+FN) (How many positives are detected among all positive?)
- Precision = TP/(TP+FP) (How many detected positives are actually positive?)
- Specificity/TNR = TN/(TN+FP) (How many negatives are detected among all negatives?)

Exercise: think of tasks for which sensitivity, precision, or specificity is a better metric.

Outline

Lab: Diagnosing Breast Cancer

- We're going to use the breast cancer dataset to predict whether the patients' scans show a malignant tumour or a benign tumour.
- Let's try to find the best linear classifier using logistic regression.

Outline

Multiclass Classificaiton

- Previous model: $f(\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$
- Representing Multiple Classes:
  - One-hot / 1-of-K vectors, ex : 4 Class
  - Class 1 : $\mathbf{y} = [1, 0, 0, 0]$
  - Class 2 : $\mathbf{y} = [0, 1, 0, 0]$
  - Class 3 : $\mathbf{y} = [0, 0, 1, 0]$
  - Class 4 : $\mathbf{y} = [0, 0, 0, 1]$

Multiclass Classificaiton

- Previous model: $f(\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$
- Representing Multiple Classes:
  - One-hot / 1-of-K vectors, ex : 4 Class
  - Class 1 : $\mathbf{y} = [1, 0, 0, 0]$
  - Class 2 : $\mathbf{y} = [0, 1, 0, 0]$
  - Class 3 : $\mathbf{y} = [0, 0, 1, 0]$
  - Class 4 : $\mathbf{y} = [0, 0, 0, 1]$
- Multiple outputs: $f(\mathbf{x}) = \text{softmax}(W^T \phi(\mathbf{x}))$
- Shape of $W^T \phi(\mathbf{x})$: $(K, 1) = (K, D) \times (D, 1)$
- $\text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$

Multiclass Classificaiton

- Multiple outputs: $f(\mathbf{x}) = \text{softmax}(\mathbf{z})$ with $\mathbf{z} = W^T\phi(\mathbf{x})$
- $\text{softmax}(\mathbf{z})_k = \frac{e^{\mathbf{z}_k}}{\sum_j e^{\mathbf{z}_j}}$
- Softmax example: If $\mathbf{z} = \begin{bmatrix} -1 \\ 2 \\ 1 \\ -4 \end{bmatrix}$ then,

$$\text{softmax}(z) = \begin{bmatrix} \frac{e^{-1}}{e^{-1}+e^{2}+e^{1}+e^{-4}} \\ \frac{e^{2}}{e^{-1}+e^{2}+e^{1}+e^{-4}} \\ \frac{e^{1}}{e^{-1}+e^{2}+e^{1}+e^{-4}} \\ \frac{e^{-4}}{e^{-1}+e^{2}+e^{1}+e^{-4}} \end{bmatrix} \approx \begin{bmatrix} 0.035 \\ 0.704 \\ 0.259 \\ 0.002 \end{bmatrix}$$

Cross-entropy

- Multiple outputs: $\hat{\mathbf{y}}_{\mathbf{i}} = \text{softmax}(W^T \phi(\mathbf{x}_i))$
- Cross-Entropy: $J(W) = -\sum_{i=1}^{N} \sum_{k=1}^{K} \mathbf{y}_{ik} log(\hat{\mathbf{y}}_{ik})$
- Example : $\text{K} = 4$

$$\text{If, } \mathbf{y}_i = [0, 0, 1, 0] \text{ then, } \sum_{k=1}^{K} \mathbf{y}_{ik} log(\hat{\mathbf{y}}_{ik}) = log(\hat{\mathbf{y}}_{i3})$$

Outline