CSC311: Midterm Review

Mustafa Ammous

October 19, 2020

UNIVERSITY OF TORONTO

Based on slides from **Julyan Keller-Baruch,** Anastasia Razdaibiedina, Sargur Srihari, James Lucas and others

Midterm Review

- **For midterm review, we will go over:**
  - **ML Concepts.**
  - **Exercises**

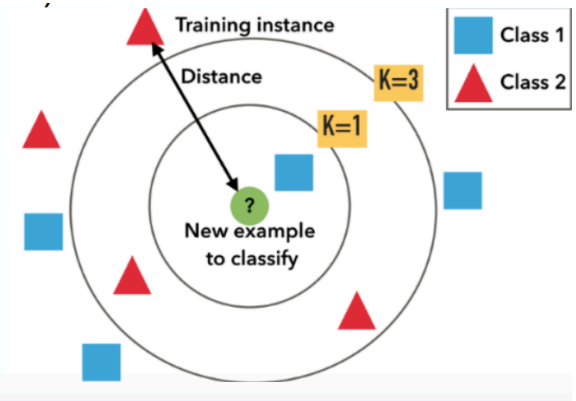
ML Concepts

- **What is supervised learning?**

  Answer: ML setting when our training set consists of inputs and their corresponding labels.

- **What does kNN do?**

  Answer: k Nearest Neighbours is an algorithm that predicts value of a new example based on its k nearest labeled neighbours.



KNN Exercise

- **Question:** A common preprocessing step of many learning algorithms is to normalize each feature to be zero mean and unit variance. Give the formula for the normalized feature $\tilde{x}_j$ as a function of the original feature x_j and the mean μ_j and the standard deviation σ_j of that feature.

Answer:

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j}$$

KNN Exercise

- **Question:** when applying K-nearest-neighbors, it is common to normalize each input dimension to unit variance.

  a) Why might it be advantageous to do this?

- Answer: If one feature has higher variance than the others, it will be treated as more important. This is problematic if the features have arbitrary scales, since the importance of a feature would depend on the unit used. Normalizing to unit variance fixes this problem.

KNN Exercise

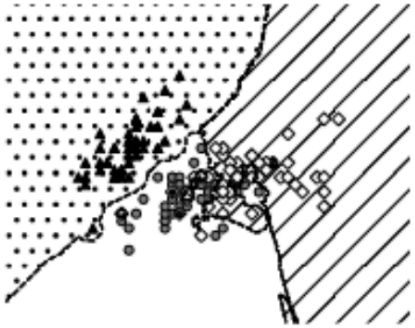
- **Question:** when applying K-nearest-neighbors, it is common to normalize each input dimension to unit variance.

  b) When might this normalization step not be a good idea? (Hint: You may want to consider the task of classifying images of handwritten digits, where the digit is centered within the image.)

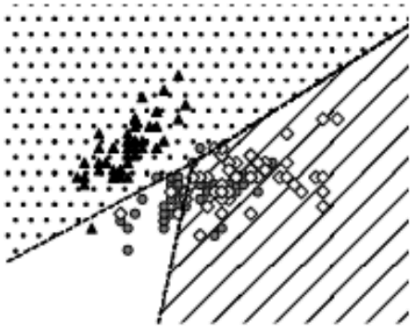
- Answer: The fact that a feature has higher variance might indicate it is actually more important. For instance, in MNIST classification, pixels near the boundary are almost always zero, and hence not very informative for the decision. Scaling them up to unit variance would just add noise to the classifications.

KNN Exercise

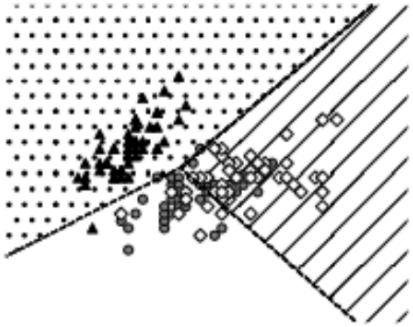
- **Question:** Which of the following decision boundaries is most likely to be generated by a k-NN?



A.



B.



D.

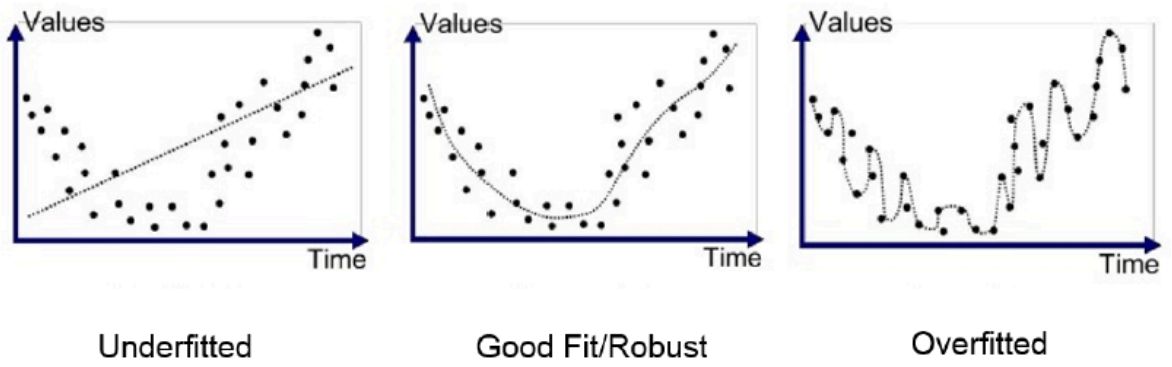
ML Concepts

- **Difference between regression / classification?**

  Answer: in ***classification*** we are predicting a discrete target (like cat or dog class), while in ***regression*** we are predicting a continuous-valued target (like temperature).

- **What is overfitting and underfitting?**

  Answer: When the model gets good performance on a particular dataset by "memorizing" it, but fails to generalize to new data.



ML Concepts

- **Write a model for binary linear classification …**

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

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

- **What are the two ways of finding good values for model's parameters (w, b)?**

  A. direct solution

  B. iterative solution - gradient descent

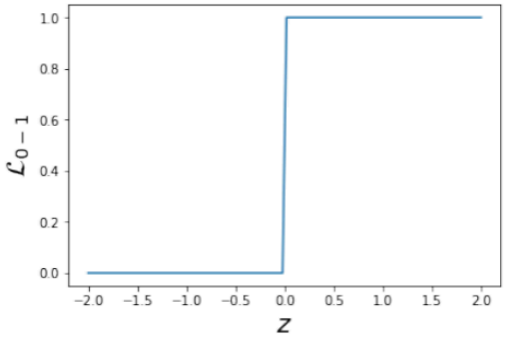
- **What is loss function?**

Answer: it's a function that evaluates how well specific algorithm models the given data; loss function takes predicted values and target values as inputs.

- **Write 0-1 loss function. Why is it bad?**

Answer: 0-1 loss is bad because it's not informative - its derivative is 0 everywhere it's defined.

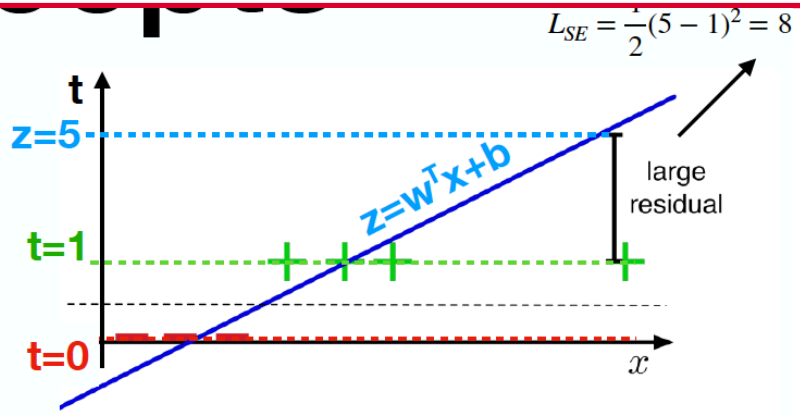
$$\mathcal{L}_{0-1}(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases}$$



ML Concepts

- **What are the problems with squared error loss function in classification?**

Answer: squared error loss gives a big penalty for correct predictions that are made with high confidence.



A solution?

Predict values only in [0,1] interval. For that we use **sigmoid function** to squash y into [0,1]:

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

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

$$y = \sigma(z)$$

$$\mathcal{L}_{\mathrm{SE}}(y,t) = \frac{1}{2}(y-t)^2.$$

(t=1)

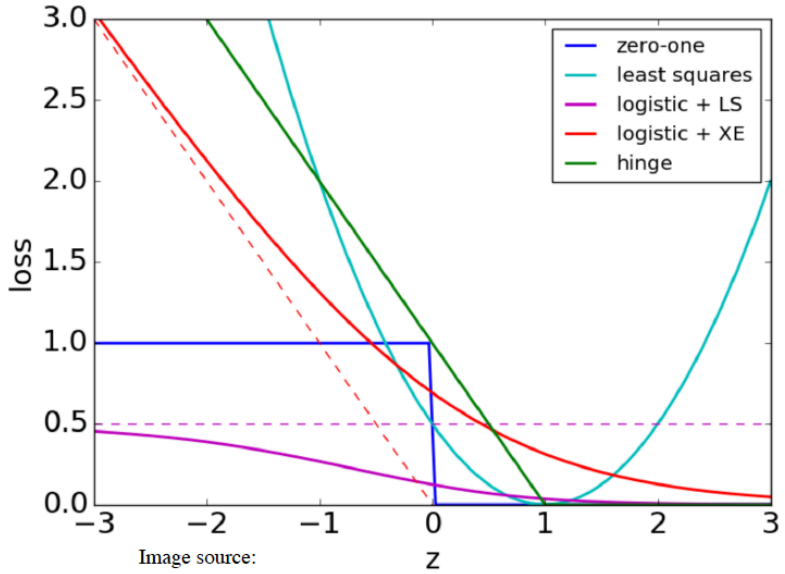


Image source:
http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec4.pdf

ML Concepts

Another solution:

Cross entropy loss.

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

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

$$y = \sigma(z)$$

$$L_{CE} = -t\log(y) - (1-t)\log(1-y)$$

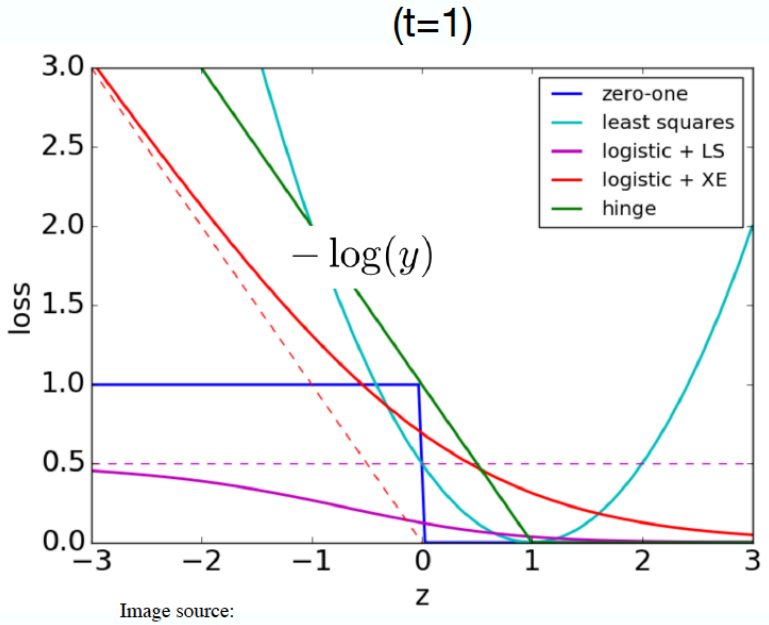


Image source:
http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec4.pdf

ML Concepts

- **What is the difference between parameters / hyper parameters of the model?**

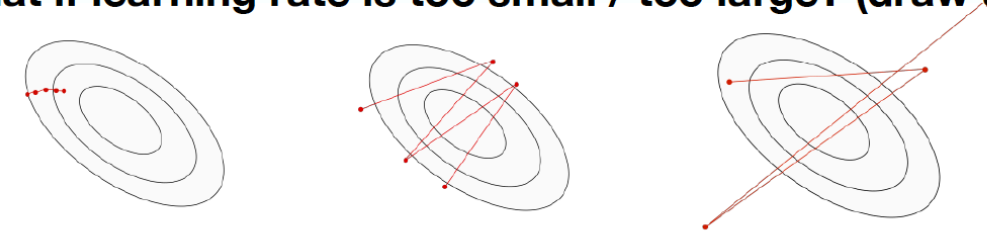
Answer: ***parameters*** are learned through training (by iteratively performing gradient descent updates) - weights and biases, ***hyperparameters*** are "manually" adjusted and set before training - number of hidden layers of a neural network, k for kNN, learning rate etc.

- **What is learning rate?**

Answer: learning rate is a hyper parameter that controls how much the weights are updated at each iteration.

$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{J}}{\partial w_j}$$

- **What if learning rate is too small / too large? (draw a picture)**



α too small: slow progress

α too large: oscillations

α much too large: instability

ML Concepts

- **What is regularization? Why do we need it?**

Answer: regularization is a technique of adding an extra term to the loss function. It reduces overfitting by keeping the weights of the model smaller.

L1 vs L2 Regularization

L1:

$$O = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} X_{i,j} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

L2:

$$O = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} X_{i,j} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

ML Concepts (Multi-Class Classification)

- Now there are D input dimensions and K output dimensions, so we need $K \times D$ weights, which we arrange as a weight matrix $\mathbf{W}$.
- Also, we have a K-dimensional vector $\mathbf{b}$ of biases.

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

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

- Softmax regression:

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

- Gradient descent updates can be derived for each row of $\mathbf{W}$:

$$\frac{\partial \mathcal{L}_{\text{CE}}}{\partial \mathbf{w}_k} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} = (y_k - t_k) \cdot \mathbf{x}$$

$$\mathbf{w}_k \leftarrow \mathbf{w}_k - \alpha \frac{1}{N} \sum_{i=1}^{N} (y_k^{(i)} - t_k^{(i)}) \mathbf{x}^{(i)}$$

- Similar to linear/logistic reg (no coincidence) (verify the update)

ML Concepts (Multi-Class Classification)

- **What is softmax? Calculate** $softmax(\begin{bmatrix} 2 \\ 1 \\ 0.1 \end{bmatrix})$

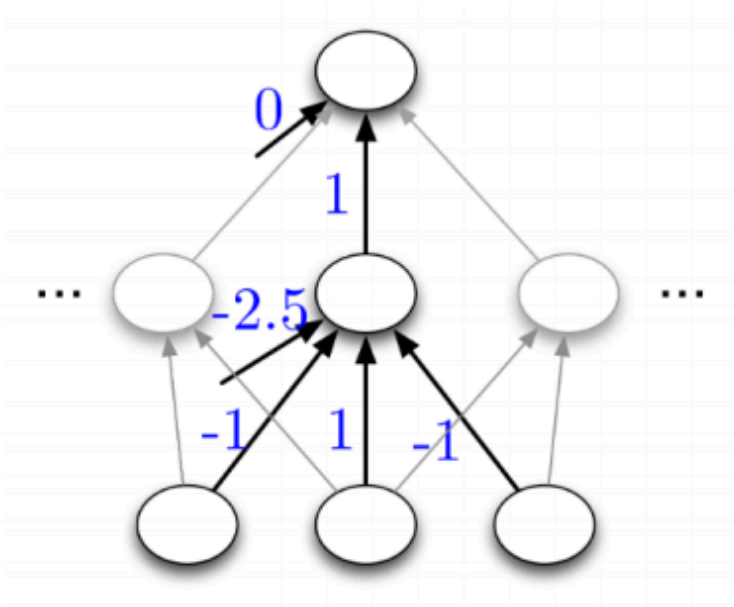
Answer: softmax is an ***activation function*** for multi-class classification that maps input ***logits*** to probabilities.

$$softmax(\begin{bmatrix} 2 \\ 1 \\ 0.1 \end{bmatrix}) = \begin{bmatrix} e^2/(e^2+e^1+e^{0.1}) \\ e^1/(e^2+e^1+e^{0.1}) \\ e^{0.1}/(e^2+e^1+e^{0.1}) \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$$

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

ML Concepts (Neural Networks)

1. Weights and neurons
2. Activation functions
3. Depth and expressive power
4. Backpropagation



Examples

Suppose we somehow know that the weights for a two-dimensional regression problem should lie on the unit circle. We can parameterize the weights in terms of the angle θ, i.e. let $(w_1, w_2) = (\cos\theta, \sin\theta)$. The model and loss function are as follows:

$$
\begin{aligned}
w_1 &= \cos\theta \\
w_2 &= \sin\theta \\
y &= w_1 x_1 + w_2 x_2 \\
\mathcal{L} &= \frac{1}{2}(y - t)^2
\end{aligned}
$$

(a) **[1pt]** *Draw the computation graph relating θ, w_1, w_2, y, and $\mathcal{L}$.*

(b) **[2pts]** *Determine the backprop update rules which let you compute the derivative $\mathrm{d}\mathcal{L}/\mathrm{d}\theta$.*

Your equations should refer to previously computed values (e.g. your formula for $\overline{z}$ should be a function of $\overline{y}$). You do not need to show your work, but it may help you get partial credit. The first two steps have been filled in for you.

Examples

Suppose we somehow know that the weights for a two-dimensional regression problem should lie on the unit circle. We can parameterize the weights in terms of the angle θ, i.e. let $(w_1, w_2) = (\cos\theta, \sin\theta)$. The model and loss function are as follows:

$$
\begin{aligned}
w_1 &= \cos\theta \\
w_2 &= \sin\theta \\
y &= w_1 x_1 + w_2 x_2 \\
\mathcal{L} &= \frac{1}{2}(y - t)^2
\end{aligned}
$$

(a) **[1pt]** *Draw the computation graph relating θ, w_1, w_2, y, and $\mathcal{L}$.*

(b) **[2pts]** *Determine the backprop update rules which let you compute the derivative $\mathrm{d}\mathcal{L}/\mathrm{d}\theta$.*

Your equations should refer to previously computed values (e.g. your formula for $\overline{z}$ should be a function of $\overline{y}$). You do not need to show your work, but it may help you get partial credit. The first two steps have been filled in for you.

Solution:

$$
\begin{aligned}
\overline{\mathcal{L}} &= 1 \\
\overline{y} &= \overline{\mathcal{L}} \cdot (y - t) \\
\overline{w_1} &= \overline{y}\, x_1 \\
\overline{w_2} &= \overline{y}\, x_2 \\
\overline{\theta} &= -\overline{w_1} \sin\theta + \overline{w_2} \cos\theta
\end{aligned}
$$

Examples

Find a linear classifier with weights w1, w2, w3, and b which correctly classifies all of these training examples:

x_1	x_2	x_3	t
0	0	0	1
0	1	0	0
0	1	1	1
1	1	1	0

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + b \gtrless 0$$

Answer: write a system of inequalities and find one solution (there would be many possible answers).

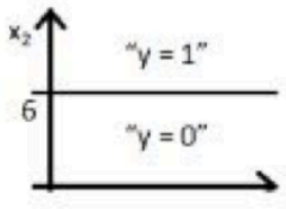
$$\begin{aligned} b &> 0 \\ w_2 + b &< 0 \\ w_2 + w_3 + b &> 0 \\ w_1 + w_2 + w_3 + b &< 0 \end{aligned} \qquad \begin{aligned} b &= 1 \\ w_1 &= -2 \\ w_2 &= -2 \\ w_3 &= 2 \end{aligned}$$

Examples

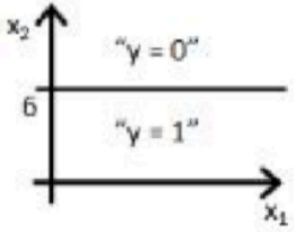
Suppose you train a logistic regression classifier and the learned hypothesis function is

$$h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2),$$

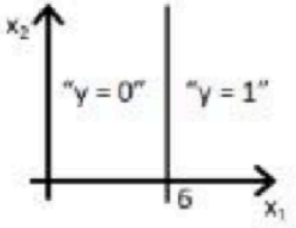
where $\theta_0 = 6, \theta_1 = 0, \theta_2 = -1$. Which of the following represents the decision boundary for $h_\theta(x)$?



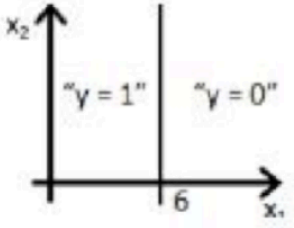
A



B



C



D

Examples

nt) Consider the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$. The derivative $f'(x)$ is

A. $f(x)\log f(x) + (1 - f(x))\log(1 - f(x))$

B. $f(x)(1 - f(x))$

C. $f(x)\log f(x)$

D. $f(x)(1 + f(x))$

Examples

True/False:

1. Gradient descent is guaranteed to find the global minimum of a loss function.

Examples

True/False:

1. Gradient descent is guaranteed to find the global minimum of a loss function.
   FAISE

2. K nearest neighbours produces a linear decision boundary.

Examples

True/False:

1. Gradient descent is guaranteed to find the global minimum of a loss function.
   FAISE

2. K nearest neighbours produces a linear decision boundary.
   FALSE

3. Both the training and testing errors are too large, if the a function underfits the data.

Examples

True/False:

1. Gradient descent is guaranteed to find the global minimum of a loss function.
   FAISE

2. K nearest neighbours produces a linear decision boundary.
   FALSE

3. Both the training and testing errors are too large, if the a function underfits the data.
   TRUE

4. Multi-class logistic regression applied to a 2-class problem is equivalent to binary logistic regression.

Examples

True/False:

1. Gradient descent is guaranteed to find the global minimum of a loss function.
   FAISE

2. K nearest neighbours produces a linear decision boundary.
   FALSE

3. Both the training and testing errors are too large, if the a function underfits the data.
   TRUE

4. Multi-class logistic regression applied to a 2-class problem is equivalent to binary logistic regression.
   TRUE

Midterm Review

good luck