Hard vs. Soft Decision Classifiers

- Binary classification problem:
  - Given features x, estimate class label 0 or 1
  - Ex: cat vs. dog

Cat $y = 0$ | Dog $y = 1$

- Hard decision classifier:
  - Output a class label: $\hat{y} = 0$ or 1
  - Ex: $\hat{y} = 1 \Rightarrow$ *Image is a dog!*
- Soft decision classifier:
  - Output a conditional probability $P(y = 1|x)$
  - $P(y = 1|x)$ is between 0 and 1
  - Ex: $P(y = 1|x) = 0.9 \Rightarrow$ *Given this image, there is a 90% chance it is a dog*

Why Use Soft Decision Classifiers?

- In most problems, classifiers make errors
- Example: Is the digit a 5 or 6?
  - Hard decision makes decision with certainty
  - But the decision can be wrong
- Soft decision classifiers recognize this uncertainty
- Easier to train soft decision classifier
  - Allows for error in training data
  - See next section
- Provides a confidence measure

Example from MNIST dataset

True digit = 5

Hard decision

Digit = 6!

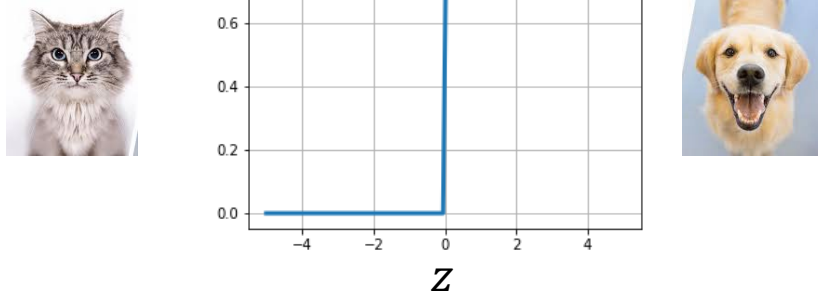
Soft decision

Digit = 5 with probability 30%

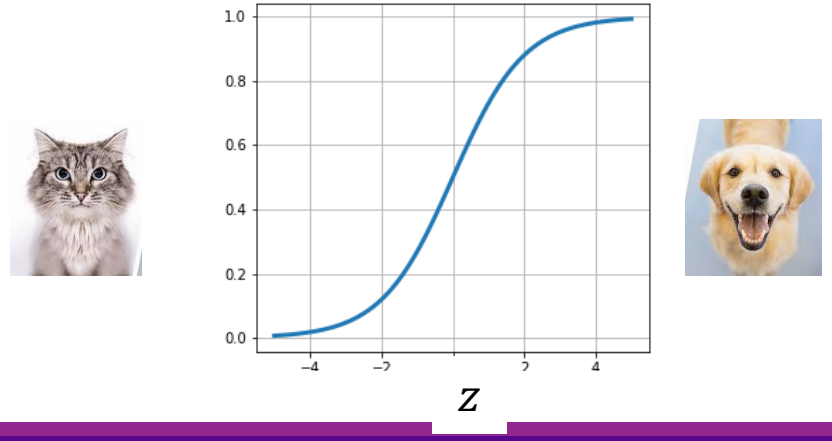
Digit = 6 with probability 70%

Logistic Model for Binary Classification

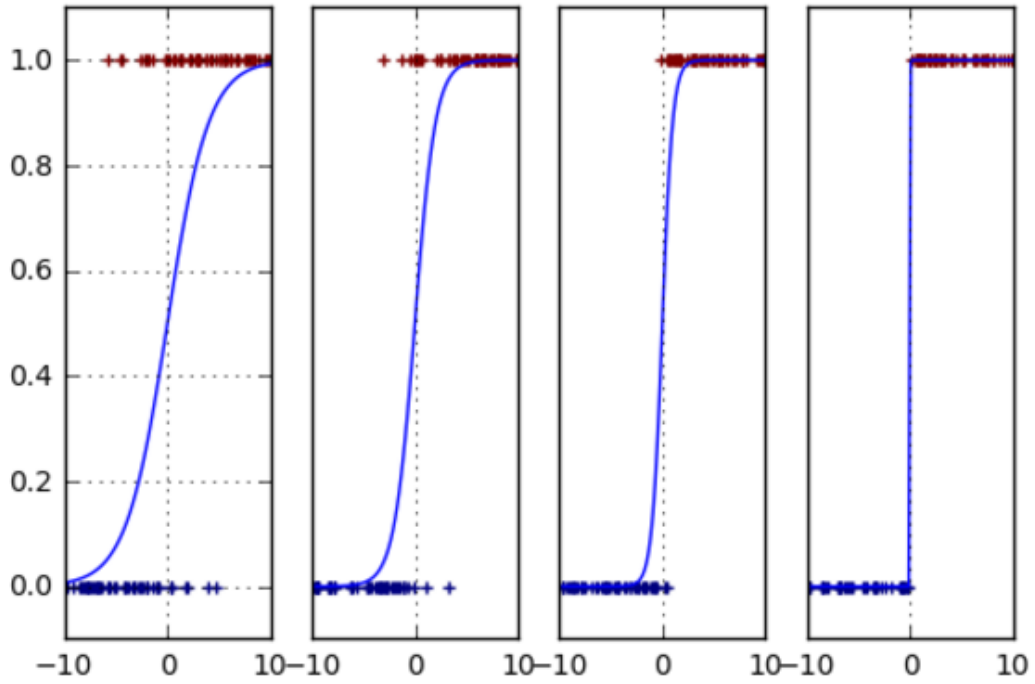
- Binary classification problem: $y = 0, 1$
- Hard decision linear classifier
  - Predict a class label $\hat{y} = 0$ or 1
  - $z = w_0 + \sum_j w_j x_j$
  - $\hat{y} = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0 \end{cases}$



- Logistic soft decision classifier
  - Predict a probability $P(y = 1|x)$
  - $z = w_0 + \sum_j w_j x_j$
  - $P(y = 1|x) = \frac{1}{1+e^{-z}}$



Logistic Model as a "Soft" Classifier



- Plot of

$$P(y = 1|x) = \frac{1}{1 + e^{-z}}, \qquad z = w_1 x$$

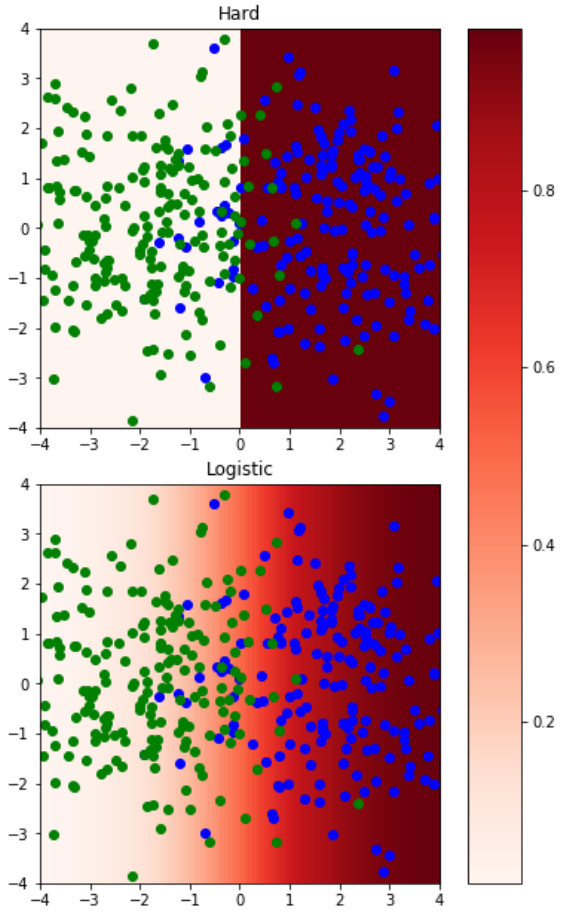
  - Markers are random samples
- Higher w_1: prob transition becomes sharper
  - Fewer samples occur across boundary
- As $w_1 \to \infty$ logistic becomes "hard" rule

$$P(y = 1|x) \approx \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases}$$

Hard vs. Soft Classification in 2D

- Hard decision:
  - Divides space into two half-space
  - One for each class label

- Soft decision
  - Gradual transition for probability 0 to 1



Multi-Class Logistic Regression

- Logistic regression easily extended to multiple classes
- Suppose $y \in 1, \ldots, K$
  - K possible classes (e.g. digits, letters, spoken words, …)
- Two parameters: $\boldsymbol{W} \in R^{K \times d}, \boldsymbol{w}_0 \in R^K$ Slope matrix and bias
- Step 1: Create K linear functions.

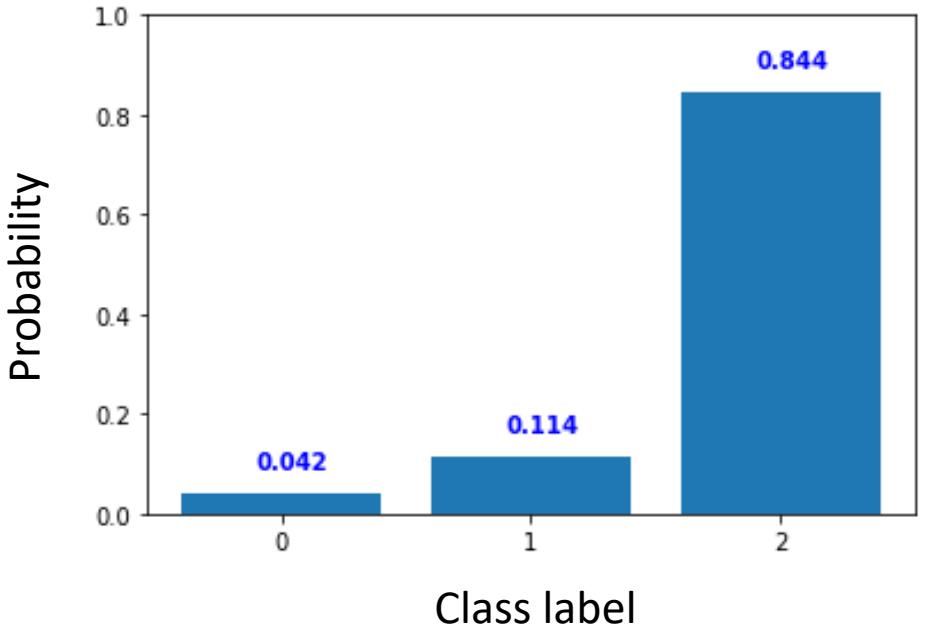
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}_0$$

  - Called scores or logits
- Step 2: Predict probabilities via softmax function

$$P(y = k|\boldsymbol{x}) = \frac{e^{z_k}}{\sum_{\ell=1}^{K} e^{z_\ell}}$$

Softmax Example

- Suppose:
  - Three class $y \in \{0,1,2\}$
  - For some x, the logits are $z = [-1, 0, 2]$
- Then $\exp(z) = [0.36, 1, 7.36]$
- Softmax probabilities:
  - $P(y = 0|x) = \frac{0.36}{0.36+1+7.36} \approx 0.042$
  - $P(y = 1|x) = \frac{1}{0.36+1+7.36} \approx 0.114$
  - $P(y = 2|x) = \frac{7.36}{0.36+1+7.36} \approx 0.844$



Softmax Properties

- Consider soft-max function:

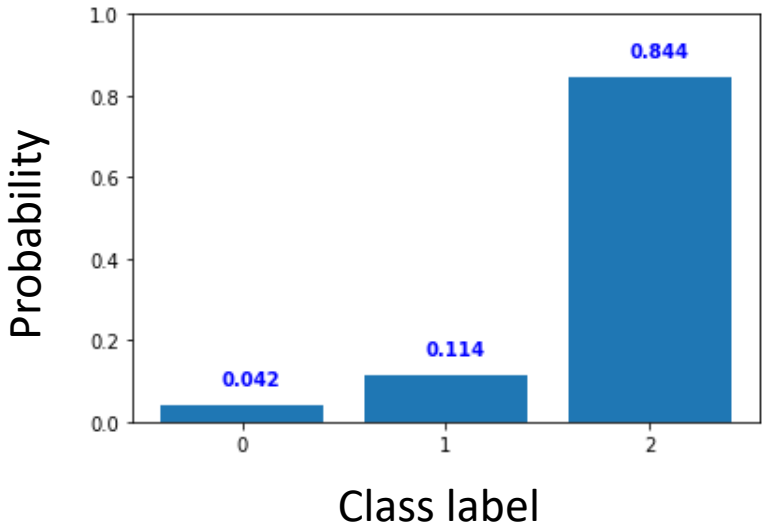
$$g_k(\boldsymbol{z}) = \frac{e^{z_k}}{\sum_{\ell=1}^{K} e^{z_\ell}}$$

  - K inputs $\boldsymbol{z} = (z_1, \ldots, z_K)$
  - K outputs $g(\boldsymbol{z}) = (g(\boldsymbol{z})_1, \ldots, g(\boldsymbol{z})_K)$

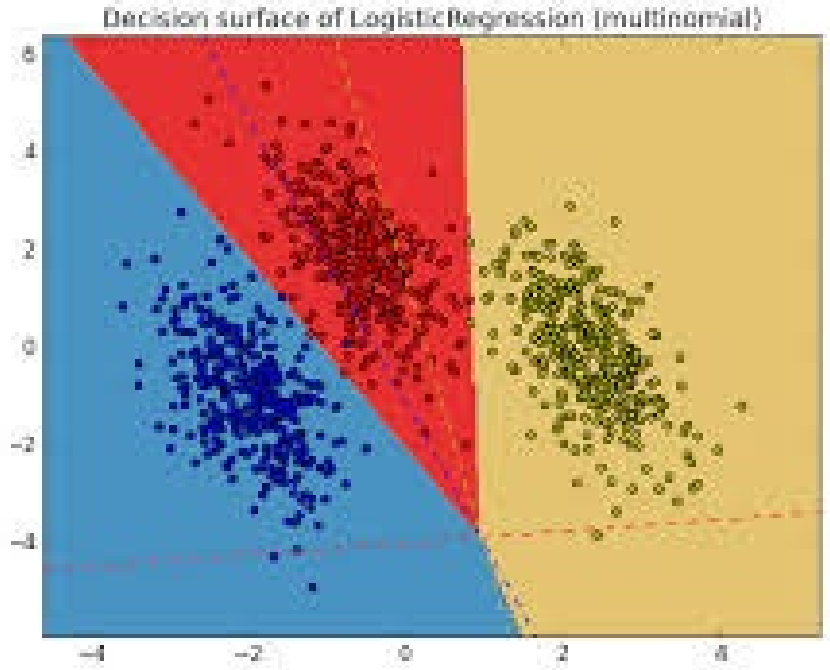
- $g(\boldsymbol{z})$ is like a PMF on the labels $[0,1, \ldots, K-1]$
  - $g_k(\boldsymbol{z}) \in [0,1]$ for each component k
  - $\sum_{k=1}^{K} g_k(\boldsymbol{z}) = 1$

- Softmax property: When $z_k \gg z_\ell$ for all $\ell \neq k$:
  - $g_k(\boldsymbol{z}) \approx 1$
  - $g_\ell(\boldsymbol{z}) \approx 0$ for all $\ell \neq k$

- Assigns highest probability to class k when z_k is largest



Multi-Class Logistic Regression Decision Regions



- Each decision region defined by set of hyperplanes
- Intersection of linear constraints
- Sometimes called a polytope

Transform Linear Models

- As in regression, logistic models can be applied to transform features
- Step 1: Map $\boldsymbol{x}$ to some transform features, $\phi(\boldsymbol{x}) = \left[\phi_1(\boldsymbol{x}), \ldots, \phi_p(\boldsymbol{x})\right]^T$ ← Additional transform step
- Step 2: Linear weights: $z_k = \sum_{j=1}^{p} W_{kj}\phi_j(\boldsymbol{x})$
- Step 3: Soft-max $P(y = k|\boldsymbol{z}) = g_k(\boldsymbol{z})$, $g_k(\boldsymbol{z}) = \frac{e^{z_k}}{\sum_\ell e^{z_\ell}}$

- Example transforms:
  - Standard regression $\phi(\boldsymbol{x}) = \left[1, x_1, \ldots, x_j\right]^T$ (j original features, j+1 transformed features)
  - Polynomial regression: $\phi(\boldsymbol{x}) = [1, x, \ldots, x^d]^T$ (1 original feature, $d+1$ transformed features)

Using Transformed Features

- Enables richer class boundaries
- Example: Fig B is not linearly separable
- But, consider nonlinear features
  - $\phi(\boldsymbol{x}) = [1, x_1, x_2, x_1^2, x_2^2]^T$
- Then can discriminate classes with linear function
  - $w = [-r^2, 0, 0, 1, 1]$
  - $z = w^T \phi(x) = x_1^2 + x_2^2 - r^2$
  - Blue when $z \leq 0$ and Green when $z > 0$