Classification via Probabilities and the Logistic Function

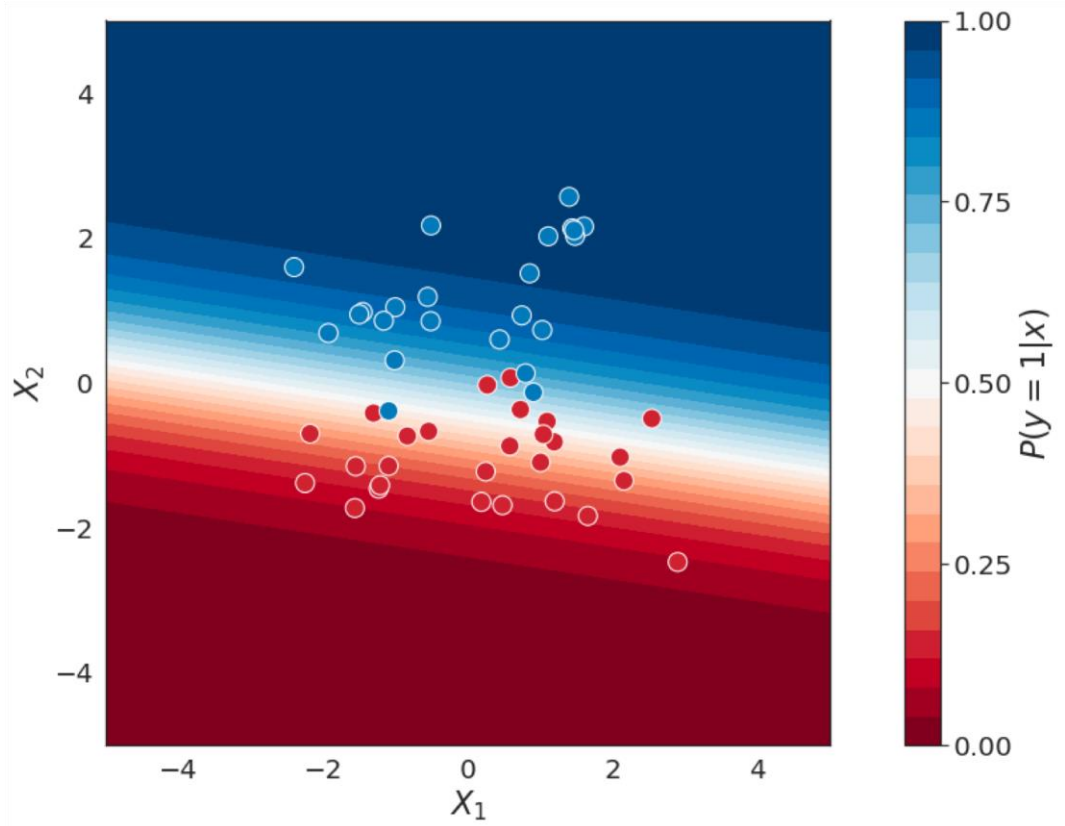
Classification Based on Probability

Class prediction

- Effective and useful
- Doesn't give prediction confidence
- Borderline cases are problematic

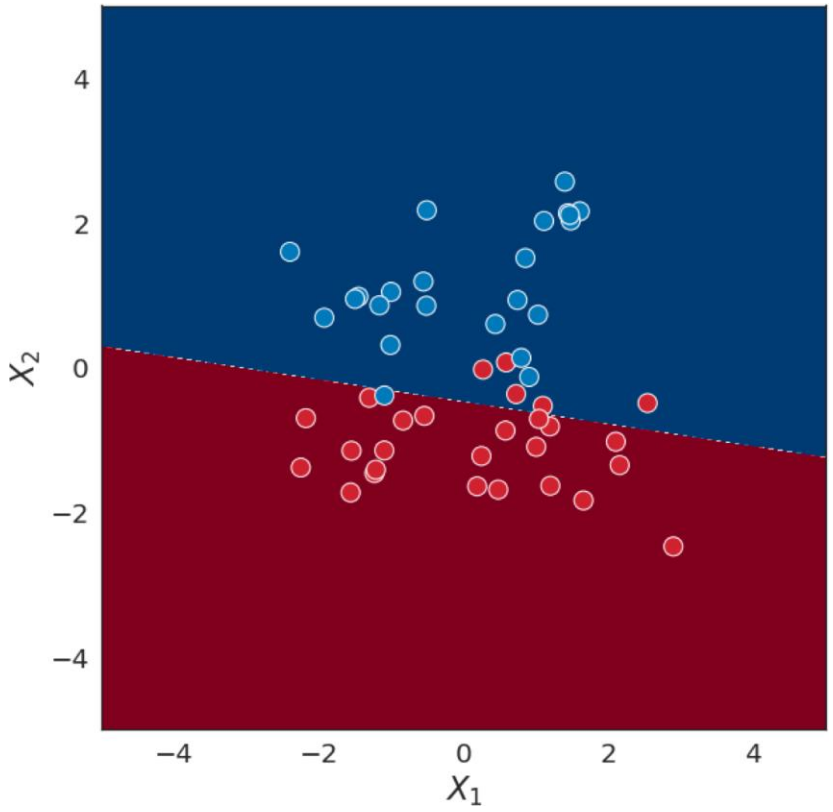
Key Idea: Instead of predicting only the class label, output the probability of the instance being within that class

  - i.e., learn $p(y \mid \boldsymbol{x})$

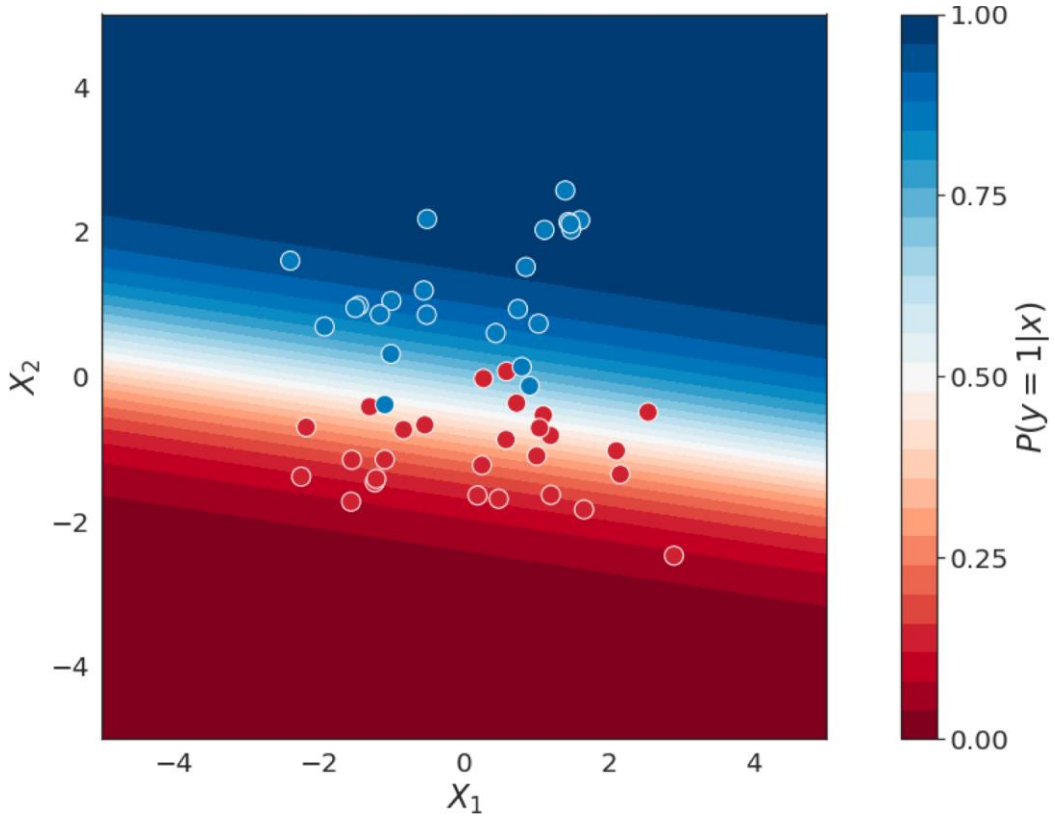


Types of Discriminative Methods

Class prediction



Probability prediction



Logistic Regression

- Takes a probabilistic approach to learning discriminative functions
  - $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $p(y = 1|\boldsymbol{x}; \boldsymbol{\theta})$
  - Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$

Can't just use linear regression with a threshold

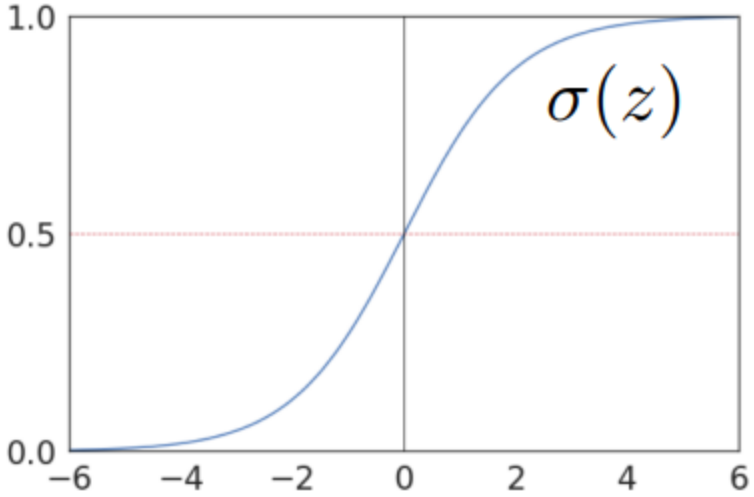
- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

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

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}}}$$

Logistic / Sigmoid Function



Interpretation of Hypothesis Output

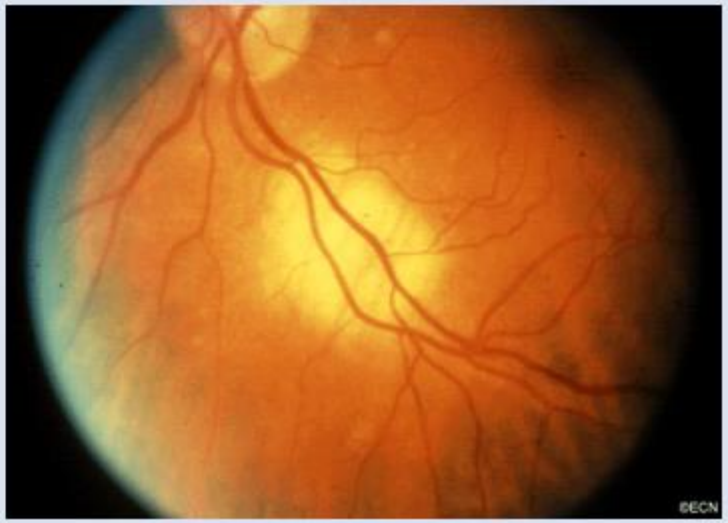
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$ = estimated $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Example: Ocular tumor diagnosis from size

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0.85$$

→ Tumor has a 70% chance of being malignant



Note that: $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) + p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1$

Therefore, $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1 - p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Another Interpretation

Equivalently, logistic regression assumes that

$$\log \frac{p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})}{p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta})} = \theta_0 + \theta_1 x_1 + \ldots + \theta_d x_d$$

odds of y = 1

Side Note: the odds in favor of an event is the quantity $p \,/\, (1 - p)$, where p is the probability of the event

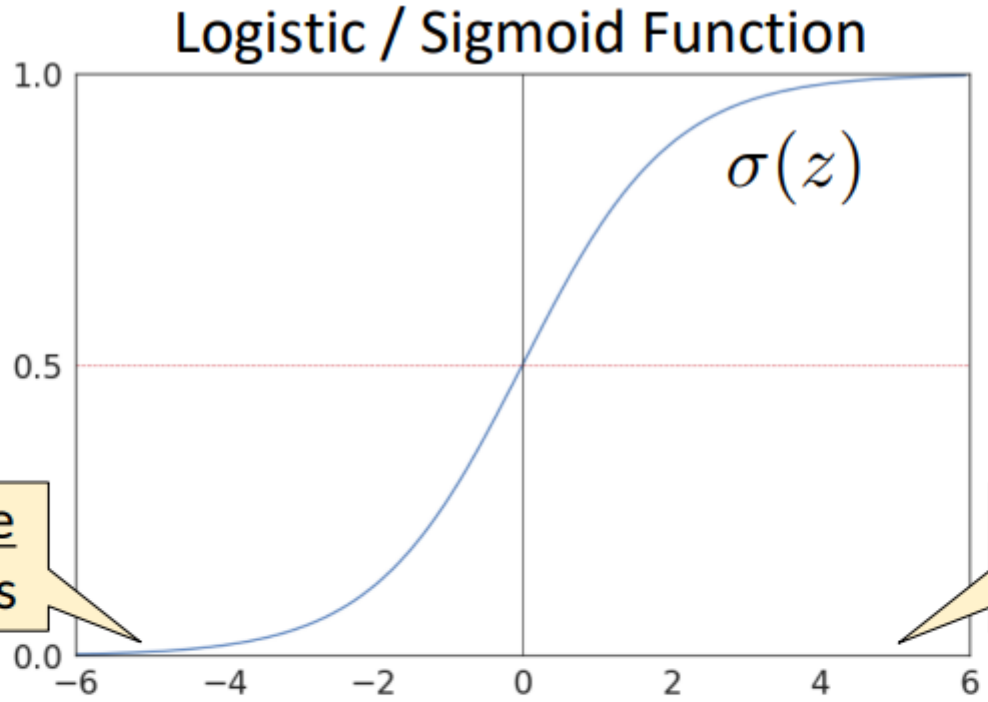
E.g., If I toss a fair dice, what are the odds that I will have a 6?

- So, logistic regression assumes that the log odds is a linear function of $\boldsymbol{x}$

Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

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

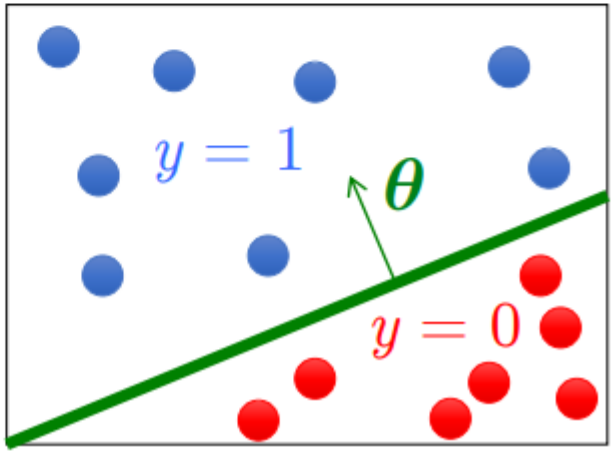


$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large negative values for negative instances

$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large positive values for positive instances

Predicting a class: Assume a threshold and...

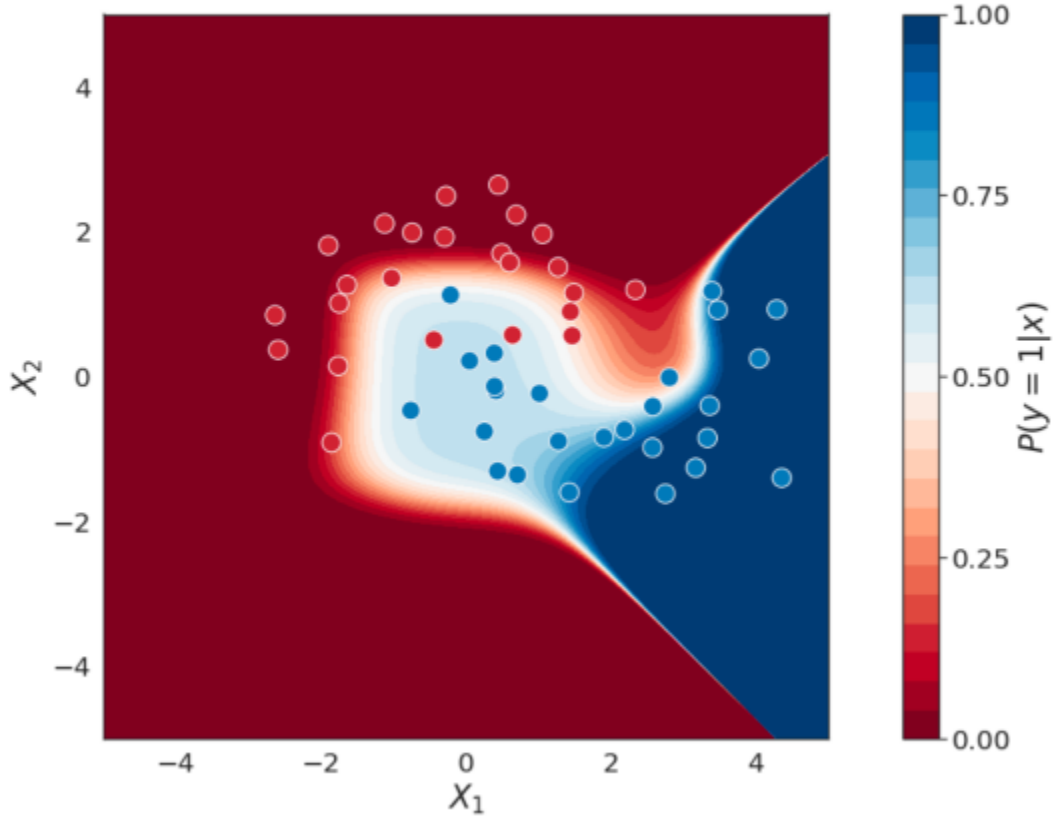
- Predict $y = 1$ if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
- Predict $y = 0$ if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Non-Linear Decision Boundary

Can apply basis expansion to features, same as with linear regression

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ \vdots \end{bmatrix}$$



Logistic Regression

- Given $\left\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\right\}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \{0, 1\}$

- Model: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma\left(\boldsymbol{\theta}^\intercal \boldsymbol{x}\right)$; $\sigma(z) = \dfrac{1}{1 + e^{-z}}$

$$= \frac{1}{1 + e^{-\boldsymbol{\theta}^\intercal \boldsymbol{x}}}$$

where

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

Maximum Likelihood Estimation for Logistic Regression

Optimizing Logistic Regression

- Can't just use squared loss as in linear regression:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2$$

  - Using the logistic regression model $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \dfrac{1}{1+e^{-\boldsymbol{\theta}^\top \boldsymbol{x}}}$ in the above results in a non-convex optimization

- We need another way; one that takes probability into account...

Deriving the Cost Function via Maximum Likelihood Estimation

- Likelihood of data is given by: $l(\boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$

Since each probability is in [0,1], this is a very small number

Potential for numeric underflow

- We are looking for the $\boldsymbol{\theta}$ that maximizes the likelihood

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$$

Deriving the Cost Function via Maximum Likelihood Estimation

- Need to solve $\boldsymbol{\theta}_{\mathrm{MLE}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$

- We can take the log without changing the solution:

$$\boldsymbol{\theta}_{\mathrm{MLE}} = \arg\max_{\boldsymbol{\theta}} \log \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$$

This is called the log likelihood

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$$

Sum avoids underflow

Deriving the Cost Function via Maximum Likelihood Estimation

- Need to solve $\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$

- We can take the log without changing the solution:

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$$

This is called the log likelihood

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$$

Sum avoids underflow

Deriving the Cost Function via Maximum Likelihood Estimation

- Need to solve $\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$$

$$= \arg\min_{\boldsymbol{\theta}} -\sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$$

$$= \arg\min_{\boldsymbol{\theta}} -\sum_{i=1}^{n} \Big[y_i \log p(y_i = 1 \mid \boldsymbol{x}_i; \boldsymbol{\theta}) + (1 - y_i) \log \left(1 - p(y_i = 1 \mid \boldsymbol{x}_i; \boldsymbol{\theta})\right) \Big]$$

Taking the negative turns a maximization problem into a minimization problem

Exploits that y_i is either 0 or 1, so only one term is non-zero

Intuition Behind the Objective

Logreg objective : $\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1-y_i)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)\right]$

- Cost of a single instance:

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

- Can re-write objective function as $\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\right)$

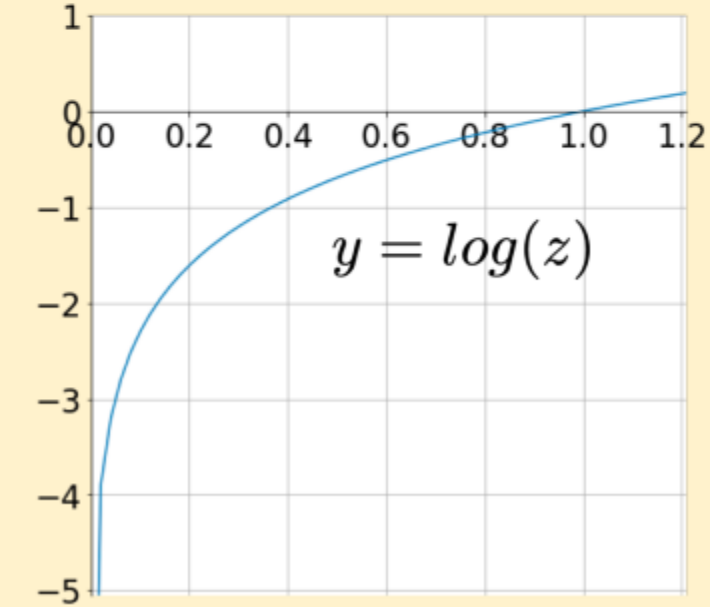
Compare to linear regression: $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)^2$

Intuition Behind the Objective

Logreg objective : $\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\right)$

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

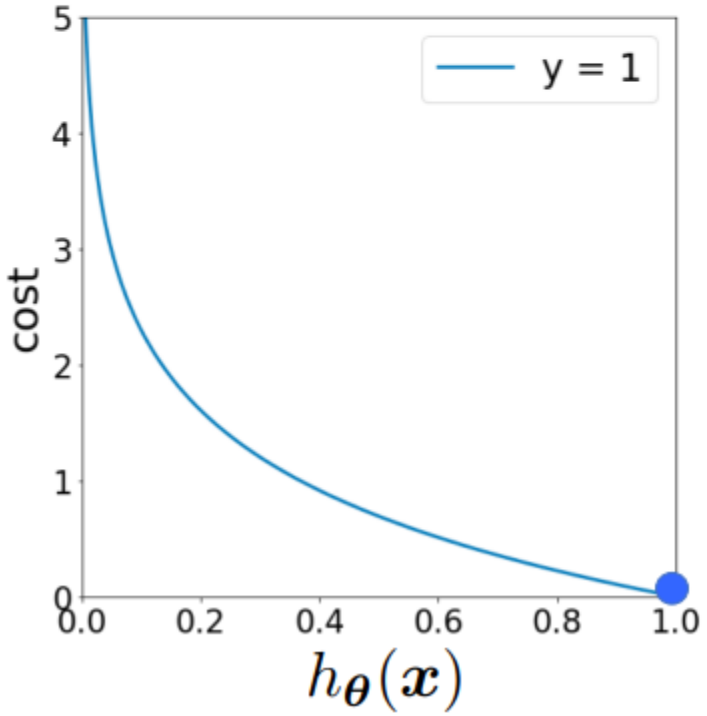
Recall the plot of $\log()$



Intuition Behind the Objective

Logreg objective : $\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\right)$

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$



For positive instances ($y = 1$)

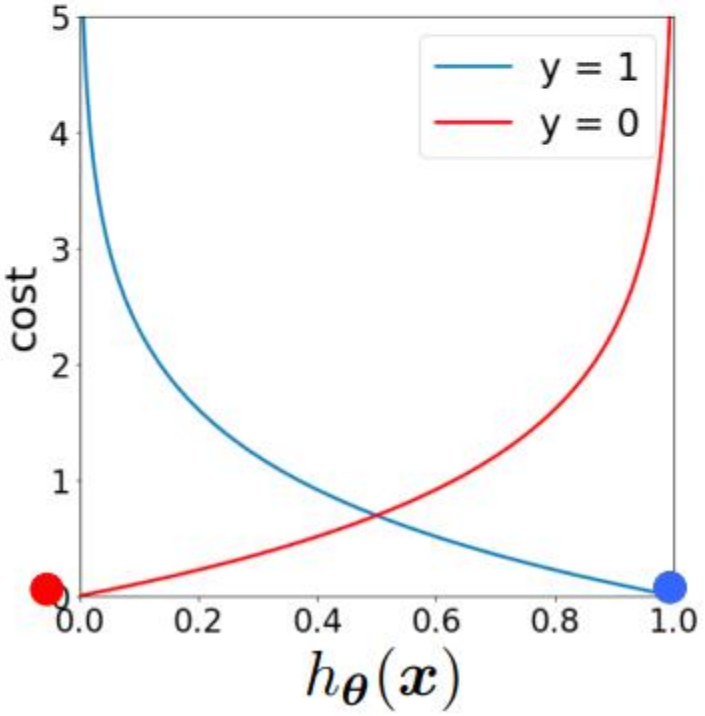
- Cost = 0 if predicted positive ($h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 1$)
- As $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \to 0, \text{cost} \to \infty$

Larger mistakes get larger penalties

Intuition Behind the Objective

Logreg objective : $\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\right)$

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$



For positive instances ($y = 1$)

- Cost = 0 if predicted positive ($h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 1$)
- As $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \to 0, \text{cost} \to \infty$

For negative instances ($y = 0$)

- Cost = 0 if predicted negative ($h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0$)
- As $(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \to 0, \text{cost} \to \infty$

Regularization and Gradient Descent for Logistic Regression

Copyright © 2020 Eric Eaton

Regularized Logistic Regression

Logreg objective : $\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1-y_i)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)\right]$

Use the natural logarithm ($\ln = \log_e$) to cancel with the $\exp()$ in $h_{\boldsymbol{\theta}}(\boldsymbol{x})$

- We can regularize logistic regression exactly as before:

$$\mathcal{L}_{\text{regularized}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \lambda \sum_{j=1}^{d} \theta_j^2$$

$$= \mathcal{L}(\boldsymbol{\theta}) + \lambda \left\|\boldsymbol{\theta}_{[1:d]}\right\|_2^2$$

Gradient Descent for Logistic Regression

$$\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\Big[y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1-y_i)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)\Big] + \lambda \left\|\boldsymbol{\theta}_{[1:d]}\right\|_2^2$$

Want $\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{reg}}(\boldsymbol{\theta})$

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{L}_{\text{reg}}(\boldsymbol{\theta})$$

simultaneous update for j = 0 ... d

Gradient Descent for Logistic Regression

$$\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1-y_i)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)\right] + \lambda \left\|\boldsymbol{\theta}_{[1:d]}\right\|_2^2$$

Want $\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{reg}}(\boldsymbol{\theta})$

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right) x_{ij} + \lambda\theta_j\right]$$

simultaneous update for $j = 0 \ldots d$

Gradient Descent for Logistic Regression

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right)$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right) - y_i\right) x_{ij} + \lambda \theta_j\right]$$

simultaneous update for $j = 0 \ldots d$

This looks IDENTICAL to linear regression!!!

- Ignoring the $1/n$ constant
- However, the form of the model is very different: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \dfrac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}}}$

L1 Regularization and Sparsity

Copyright © 2020 Eric Eaton

Datasets with Many Features

1,000+

SEQN	RIDAGEYR	BMXWAIST	BMXHT	LBXTC	BMXLEG	BMXWT	BMXBMI	RIDRETH1	BPQ020	ALQ120Q	DMDEDUC2	RIAGENDR		DIABETIC
73557	69.0	100.0	171.3	167.0	39.2	78.3	26.7	Non-Hispanic Black	yes	1.0	high school graduate / GED	male	•••	yes
73558	54.0	107.6	176.8	170.0	40.0	89.5	28.6	Non-Hispanic White	yes	7.0	high school graduate / GED	male	•••	yes
73559	72.0	109.2	175.3	126.0	40.0	88.9	28.9	Non-Hispanic White	yes	0.0	some college or AA degree	male	•••	yes
73562	56.0	123.1	158.7	226.0	34.2	105.0	41.7	Mexican American	yes	5.0	some college or AA degree	male	•••	no
73564	61.0	110.8	161.8	168.0	37.1	93.4	35.7	Non-Hispanic White	yes	2.0	college graduate or above	female	•••	no
73566	56.0	85.5	152.8	278.0	32.4	61.8	26.5	Non-Hispanic White	no	1.0	high school graduate / GED	female	•••	no
73567	65.0	93.7	172.4	173.0	40.0	65.3	22.0	Non-Hispanic White	no	4.0	9th-11th grade	male	•••	no
73568	26.0	73.7	152.5	168.0	34.4	47.1	20.3	Non-Hispanic White	no	2.0	college graduate or above	female	•••	no
73571	76.0	122.1	172.5	167.0	35.5	102.4	34.4	Non-Hispanic White	yes	2.0	college graduate or above	male	•••	yes

- L_2 regularization will assign a weight to every feature
  - Many features with little weight → complex model
- The need for feature selection
  - Often, it is only a few features that we really want
  - Irrelevant features should be assigned a coefficient of 0

This is called sparsity

Copyright © 2020 Eric Eaton

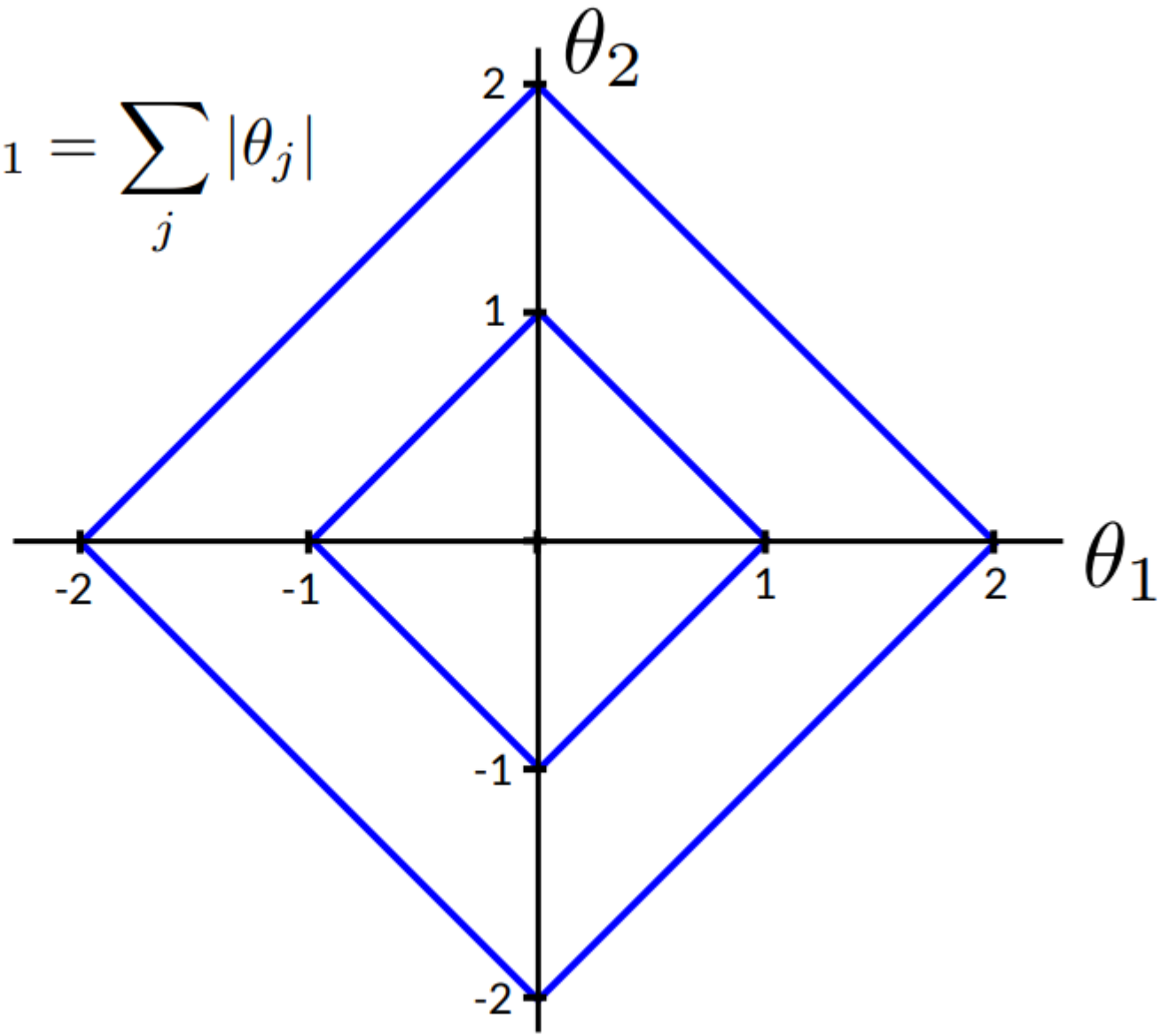
L_1 Regularization

- What we really want is the L_0 norm (count of non-zero coefficients)
  - But, it is discontinuous, non-convex, and generally difficult to compute
- Instead, approximate using the L_1 norm: $\|\boldsymbol{v}\|_1 = \sum_j |v_j|$
- Can incorporate into the logistic regression objective:

$$\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \Big[y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1 - y_i) \log \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right) \Big] + \lambda \left\| \boldsymbol{\theta}_{[1:d]} \right\|_1$$

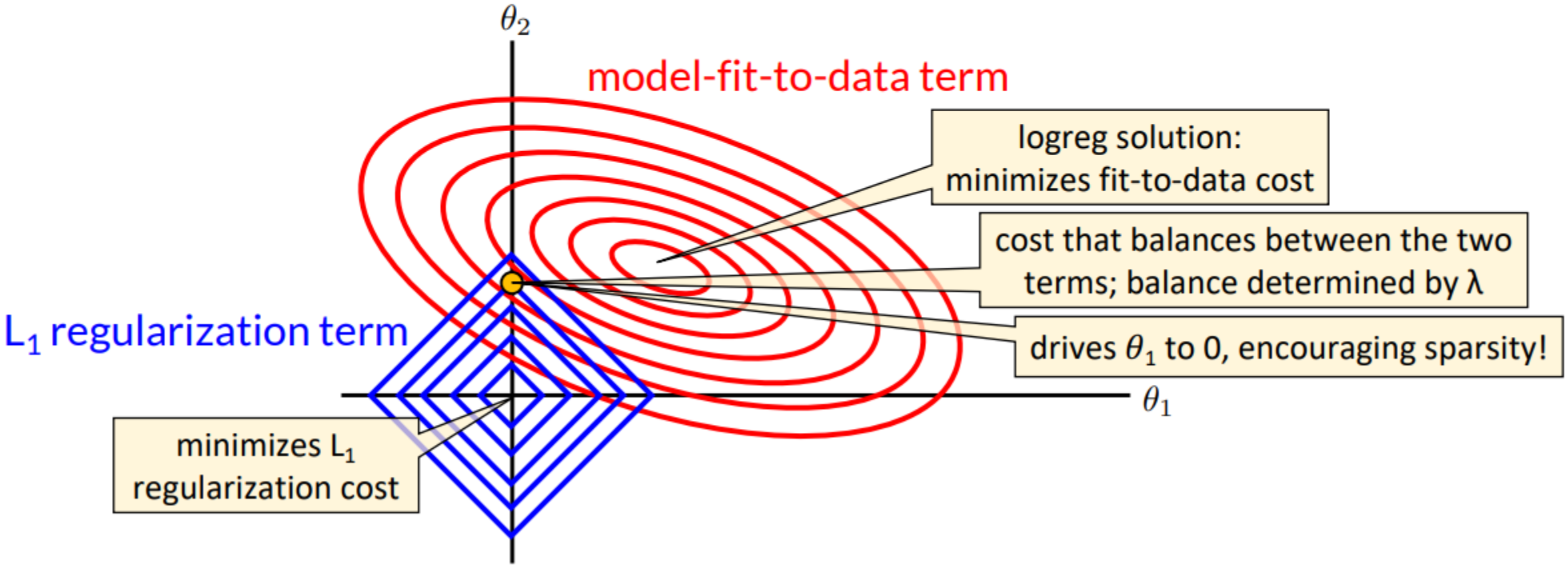
Understanding L_1 Regularization

L_1 norm: $\|\boldsymbol{\theta}\|_1 = \sum_j |\theta_j|$



Understanding L_1 Regularization

$$\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\Big[y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1-y_i)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)\Big] + \lambda \sum_{j=1}^{d} |\theta_j|$$



Stochastic Gradient Descent

Consider Learning with Numerous Data

- Logistic regression objective:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \Big[\underbrace{y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1 - y_i) \log \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)}_{\text{cost}_{\boldsymbol{\theta}}(\mathbf{x}_i, y_i)} \Big]$$

- Fit via gradient descent:

$$\theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i\right) x_{ij}$$

- What is the computational complexity in terms of n?

Gradient Descent

Logreg objective: $\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1-y_i)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)\right]$

Batch Gradient Descent

Initialize $\boldsymbol{\theta}$

Repeat {

$$\theta_j \leftarrow \theta_j - \alpha \underbrace{\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i\right) x_{ij}}_{\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})} \quad \forall j$$

}

Stochastic Gradient Descent

Initialize $\boldsymbol{\theta}$

Randomly shuffle dataset

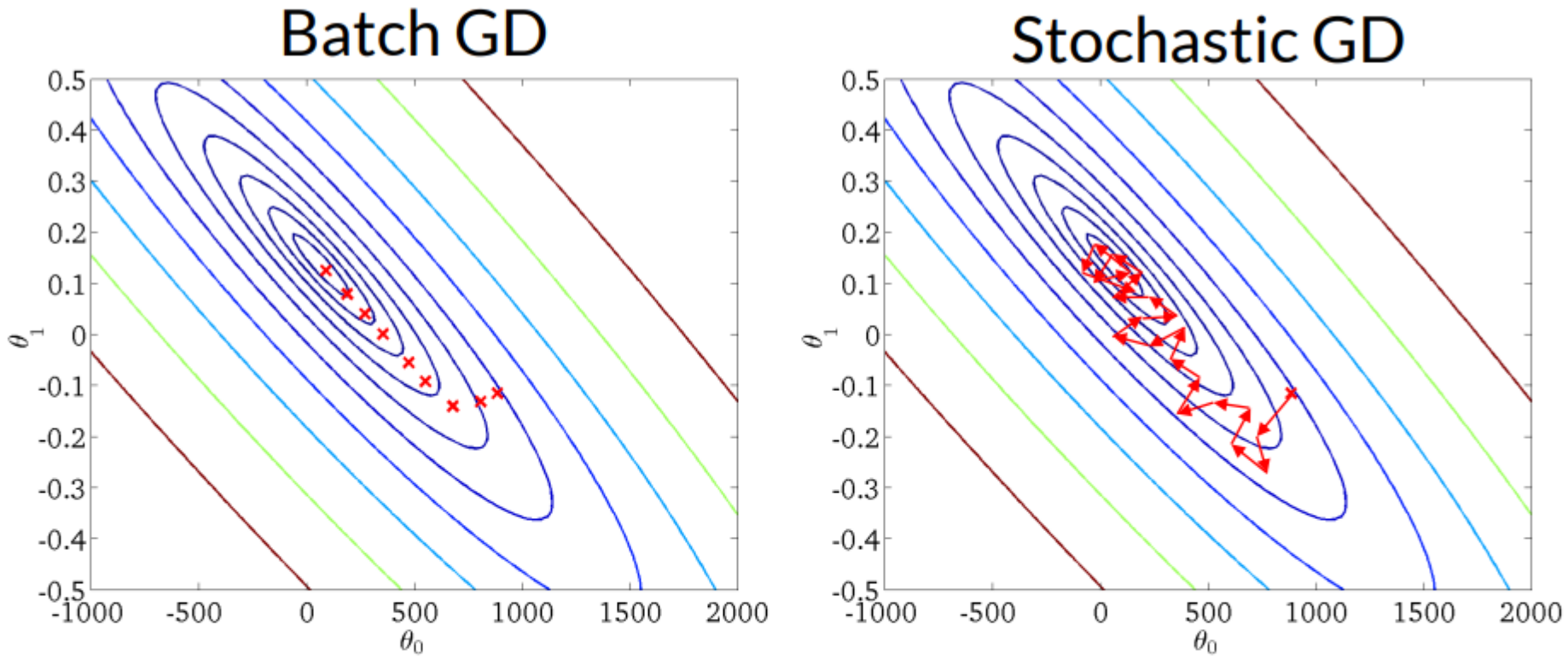
Repeat { (Typically 1 – 10x)

For $i = 1 \ldots n$, do

$$\theta_j \leftarrow \theta_j - \alpha \underbrace{\left(h_{\boldsymbol{\theta}}\left(\mathbf{x}_i\right) - y_i\right) x_{ij}}_{\frac{\partial}{\partial \theta_j} \text{cost}_{\boldsymbol{\theta}}(\boldsymbol{x}_i, y_i)} \quad \forall j$$

}

Batch vs Stochastic GD



- Learning rate α is typically held constant
- Can slowly decrease α over time to force $\boldsymbol{\theta}$ to converge:

e.g., $\alpha_t = \dfrac{\text{constant1}}{\text{iterationNumber} + \text{constant2}}$

Based on slide by Andrew Ng

Adagrad

New Stochastic Gradient Algorithms

So far, we have considered:
- a constant learning rate α
- a time-dependent learning rate α_t via a pre-set formula

- AdaGrad adjusts the learning rate based on historical information
- **Adagrad's Key idea:** "learn slowly" from frequent features but "pay attention" to rare but informative features
  - Frequently occurring features in the gradients get small learning rates
  - Infrequent features get higher learning rates
- Define a per-feature learning rate for feature j as:

$$\alpha_{t,j} = \frac{\alpha}{\sqrt{G_{t,j}}} \quad \text{where} \quad G_{t,j} = \sum_{k=1}^{t} g_{k,j}^2 \qquad \frac{\partial}{\partial \theta_j} \text{cost}_{\boldsymbol{\theta}}(\mathbf{x}_k, y_k)$$

$G_{t,j}$ is the sum of squares of gradients of feature j through time t

New Stochastic Gradient Algorithms

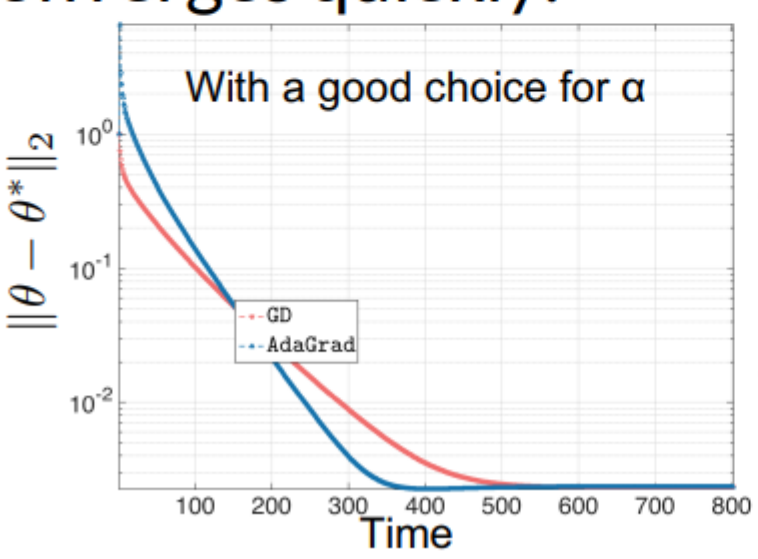
Adagrad per-feature learning rate: $\alpha_{t,j} = \frac{\alpha}{\sqrt{G_{t,j}}}$ where $G_{t,j} = \sum_{k=1}^{t} g_{k,j}^2$

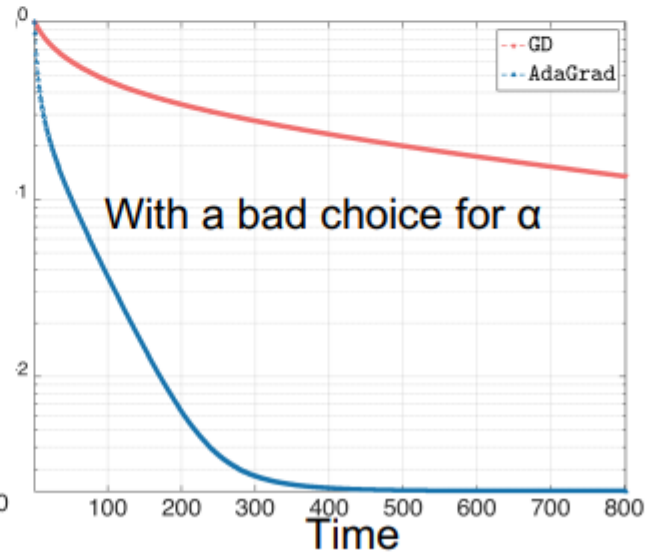
- Adagrad changes the update rule for SGD at time t from $\theta_j \leftarrow \theta_j - \alpha g_{t,j}$ to

$$\theta_j \leftarrow \theta_j - \frac{\alpha}{\sqrt{G_{t,j}} + \zeta} g_{t,j}$$

In practice, we add a small constant $\zeta > 0$ to prevent dividing by zero errors

- Adagrad converges quickly:



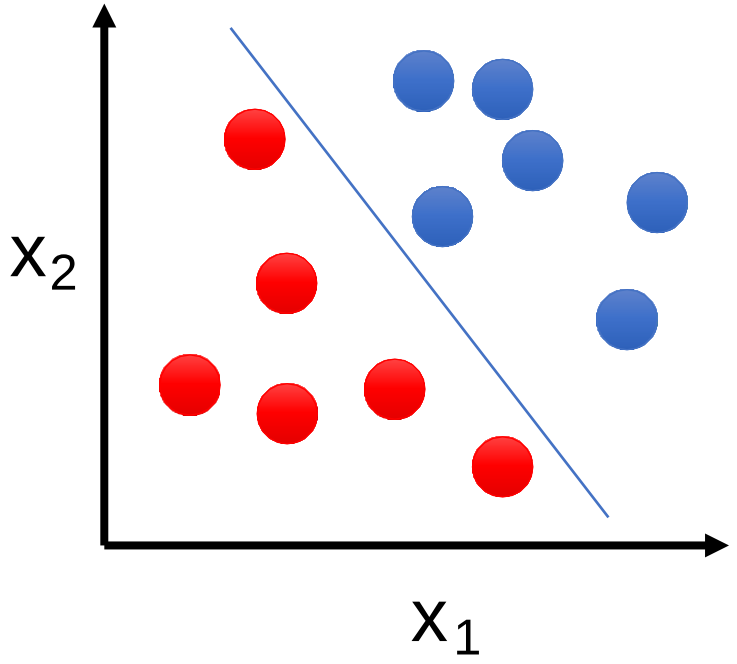


Plots from http://akyrillidis.github.io/notes/AdaGrad

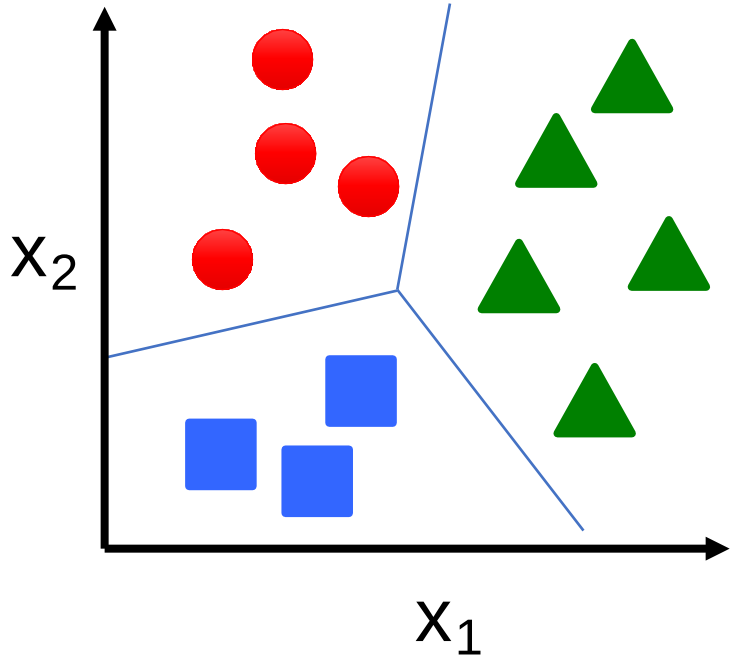
Multi-Class Classification

Multi-Class Classification

Binary classification:



Multi-class classification:



Disease diagnosis: healthy / cold / flu / pneumonia

Object classification: desk / chair / monitor / bookcase

Multi-Class Logistic Regression

- For 2 classes:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x})} = \frac{\exp(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x})}{1+\exp(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x})}$$

weight assigned to $y = 0$

weight assigned to $y = 1$

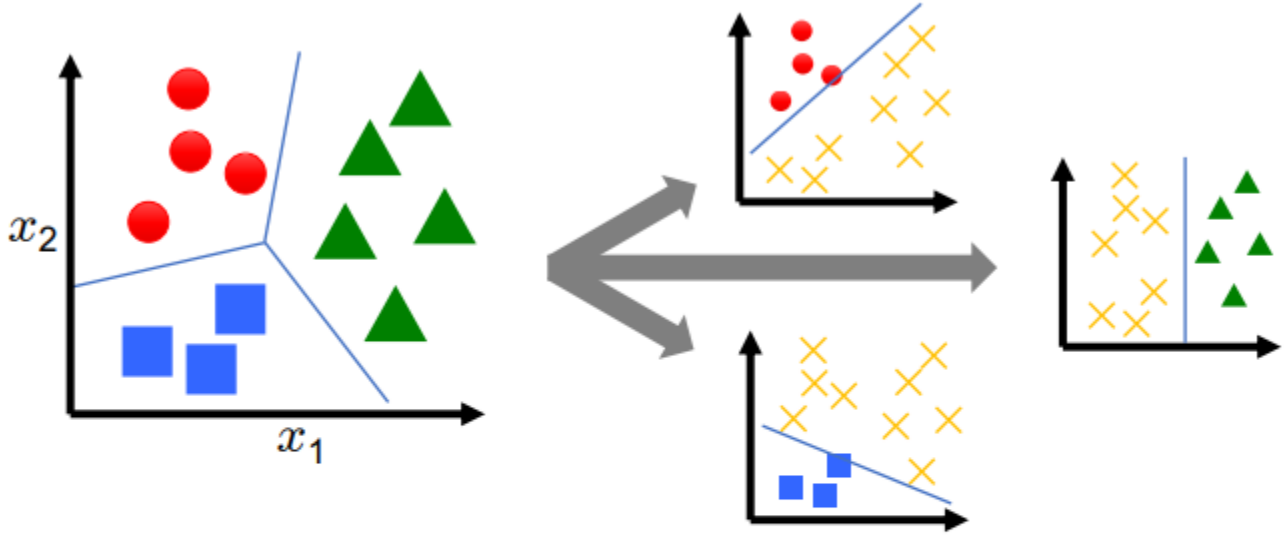
- For C classes $\{1, \ldots, C\}$:

$$p(y = c \mid \boldsymbol{x}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_C) = \frac{\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}$$

  - Called the **softmax** function

Multi-Class Logistic Regression

- Split into One vs Rest:



- Train a logistic regression classifier for each class i to predict the probability that $y = i$ with

$$h_c(\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}$$

Implementing Multi-Class Logistic Regression

- Use $h_c(\boldsymbol{x}) = \dfrac{\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}$ as the model for class c

- Gradient descent simultaneously updates all parameters for all models
  - Same derivative as before, just with the above $h_c(\boldsymbol{x})$

- Predict class label as the most probable label $\max_c h_c(\boldsymbol{x})$