Universität
Konstanz

Lecture 3 Regularization and multi-class logistic regression

Dr. Hanhe Lin
Dept. of Computer and Information Science
University of Konstanz

Logistic regression

- Hypothesis: $h(x) = \dfrac{1}{1 + e^{-(wx+b)}}$
- Parameter: w, b
- Loss function:

$$L(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left(y\log(h(x)) + (1-y)\log(1-h(x))\right)$$

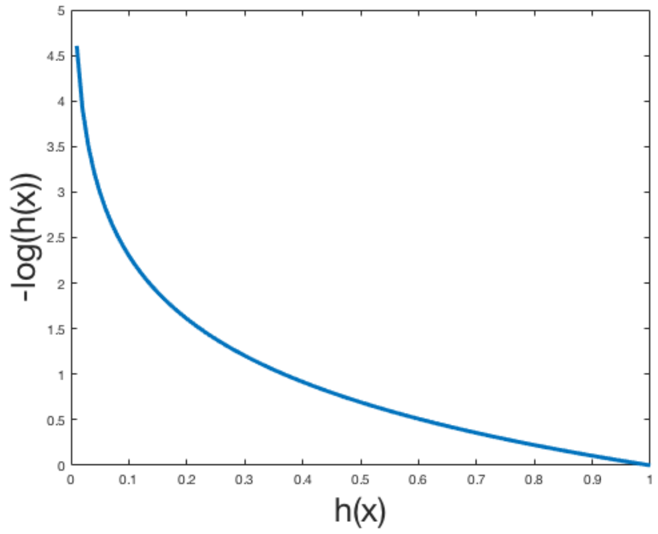
- Goal: $\min\limits_{w,b} L(w,b)$

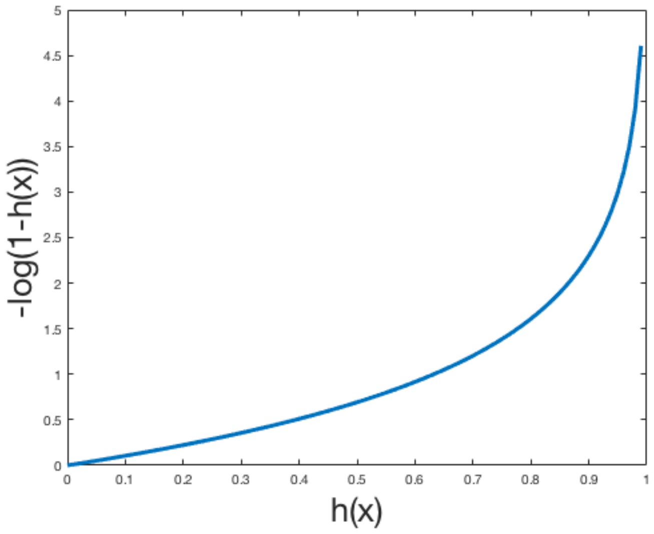
Loss function of logistic regression

- Loss function:

$$\text{Loss}(h(x), y) = \begin{cases} -\log(h(x)) \; \mathit{if} \;\; y = 1 \\ -\log(1 - h(x)) \; \mathit{if} \;\; y = 0 \end{cases}$$

- Intuition: if $h(x) = 0$, but $y = 1$, the learning algorithm will be penalized by a vary large loss





Gradient descent for logistic regression

- Given the loss function:

$$L(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left(y\log(h(x)) + (1-y)\log(1-h(x))\right)$$

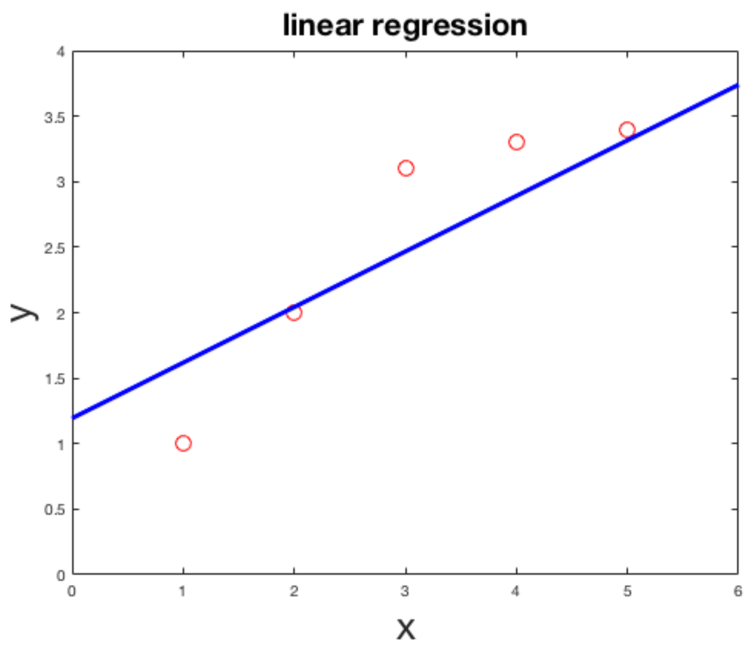
our objective is to $\min\limits_{w,b} L(w,b)$

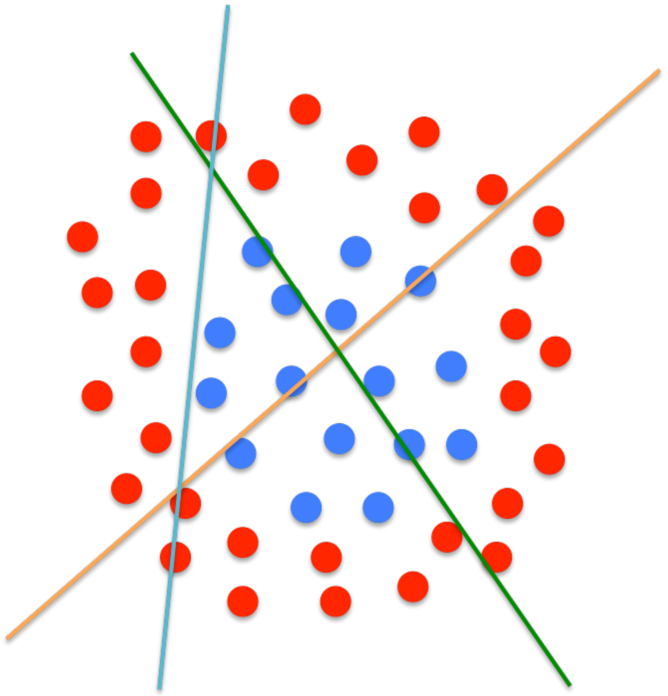
- Repeat until converge:

$$w_j := w_j - \alpha\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})x_j^{(i)} \quad j = 1,\ldots,n$$

$$b := b - \alpha\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})$$

Is linear enough?





- Pros: they can be fit efficiently and reliably with convex optimization
- Cons: they do NOT know how to classify high-dimensional, complex data that are *nonlinear*

Nonlinear

- “Nonlinear” definition
  - The change of the output is not proportional to the input
- Nonlinear function:
  - Polynomial, Gaussian, …

Features and polynomial regression

- Go back to linear regression with one variable:

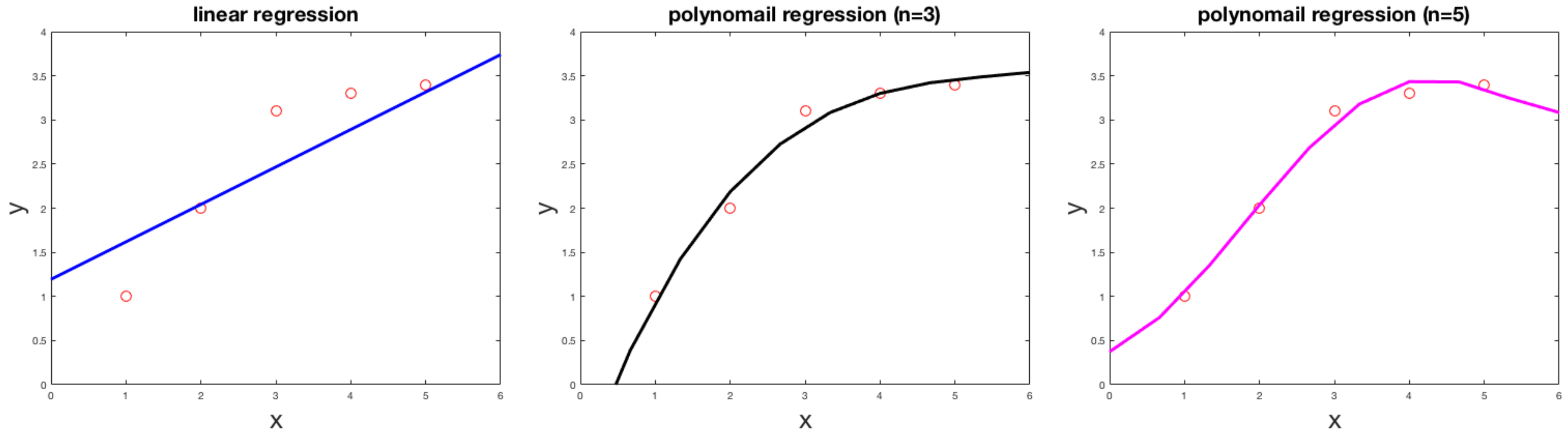
$$h(x) = wx + b$$

- To improve model, we can add more "artificial" features:

$$h(x) = w_1 x + w_2 x^2 + \ldots + w_n x^n + b = \sum_{i=0}^{n} w_i x^i + b$$

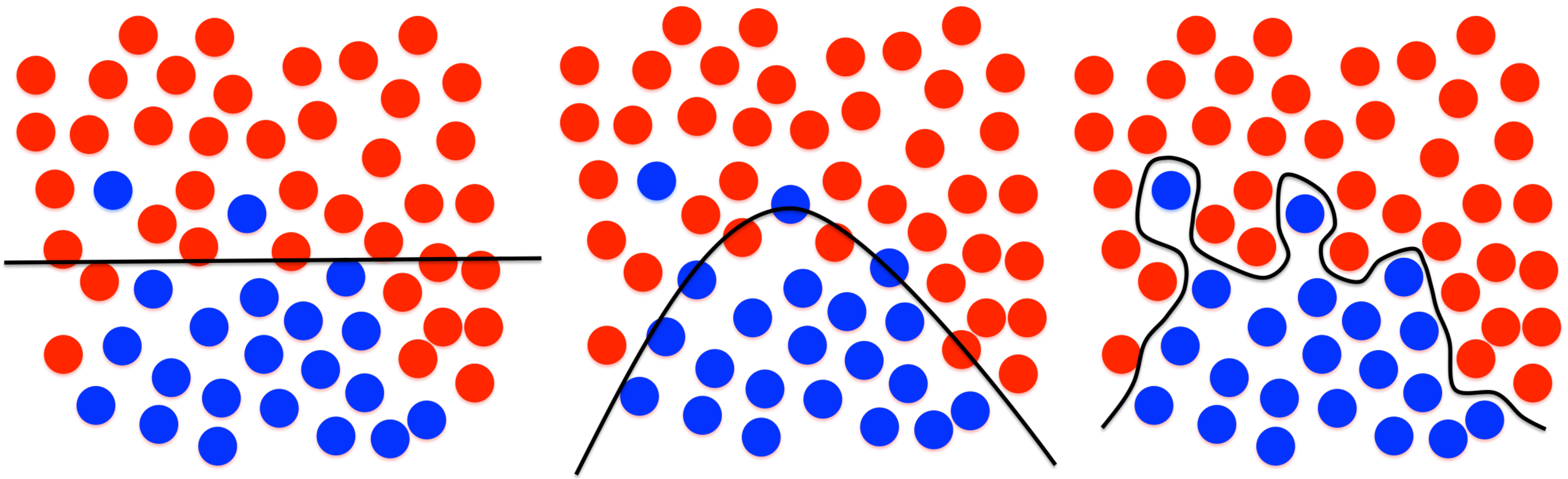
- The hypothesis becomes a polynomial function, therefore, we could call it "polynomial regression" instead

Example: linear / polynomial regression

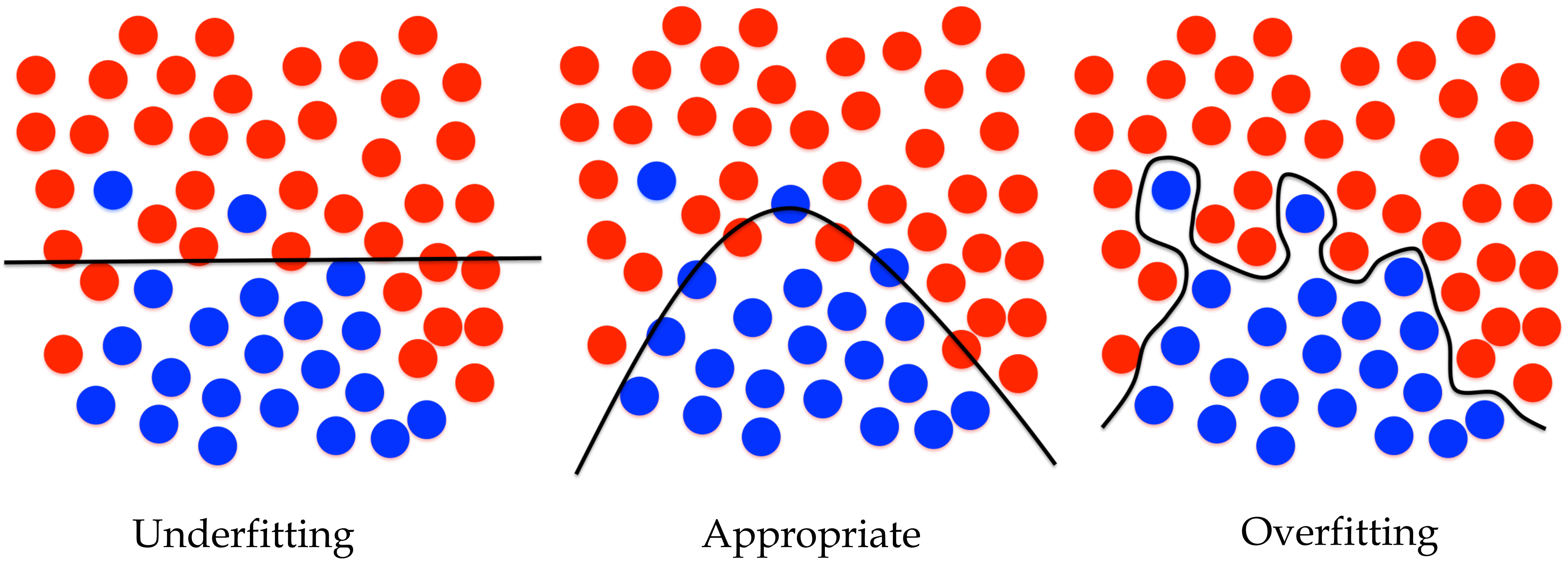


Q: the more parameters / features, the better?

Question: which binary classifier performs best?



Question: which binary classifier performs best?



Capacity, underfitting and overfitting

- Generalization: the ability to perform well on previously unobserved inputs
- Training error: error measure, e.g., accuracy, MSE, on the training set
- Test/generalization error: error measure on the test set
- Based on the assumption:
  - The examples in each dataset are independent from each other
  - The training set and the test set are identically distributed, drawn from the same probability distribution as each others
- We want a machine learning algorithm (differ from optimization):
  - The training error small
  - The gap between training and test error small

Capacity, underfitting and overfitting

- Capacity
  - A model's ability to fit a wide variety of functions.
- Underfitting (high bias)
  - A phenomenon that a ML model maps poorly to the trend of the data
  - Occurs when your hypothesis is too simple or use too few features
- Overfitting (high variance)
  - A phenomenon that a ML model fit the training set very well, but fail to generalize to test set. In other words, an overfitting model performs well in training set, but quit bad in test set
  - Occurs when your hypothesis is excessively complex, e.g, too many parameters or too many features

Address underfitting and overfitting

- Address underfitting:
  - Add more features
  - Use a more complex hypothesis
- Address overfitting:
  - Reduce number of features
  - Use a less complex hypothesis
  - Regularization
    - Keep all the features, but reduce values of parameters
    - Regularization works well when we have lots of slightly useful features w_j
  - Early stopping, dropout in deep learning

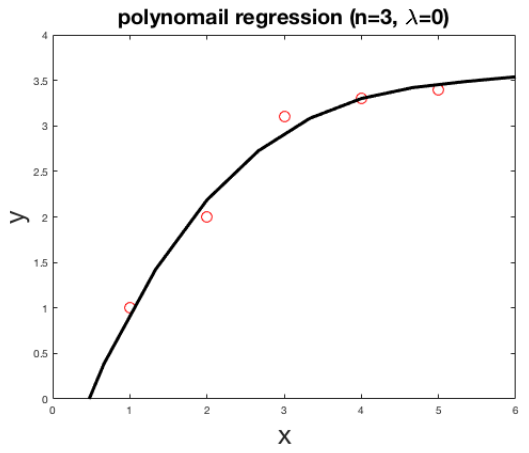
Regularization

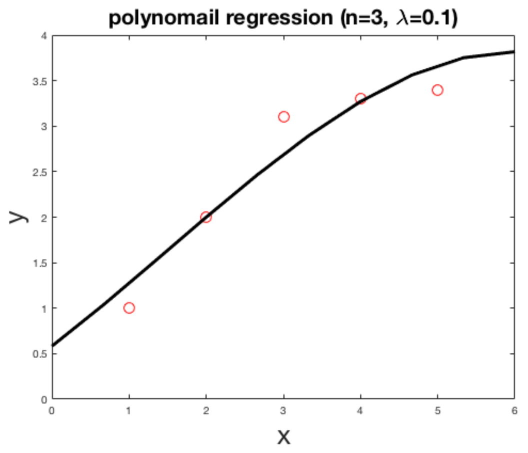
- A hyper-parameter is determined outside the learning algorithm, e.g., learning rate α in gradient descent
- Small values for parameters $w_1, w_2, \ldots, w_n$
  - "Simpler" hypothesis, thus less prone to overfitting
- Regularization for linear regression

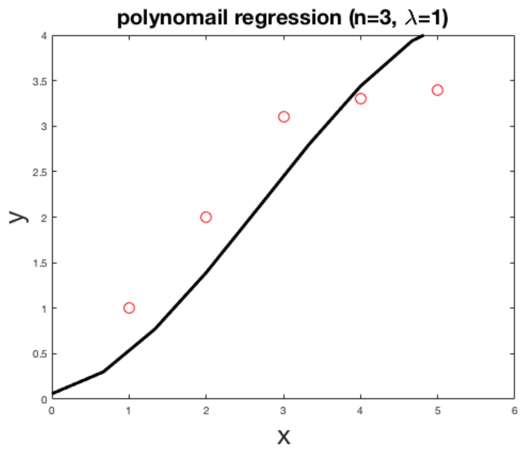
$$L(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} w_j^2\right]$$

- The regularization parameter λ is a hyper-parameter, control the tradeoff between two different goals: fitting the training set well and keeping the parameter small

Example



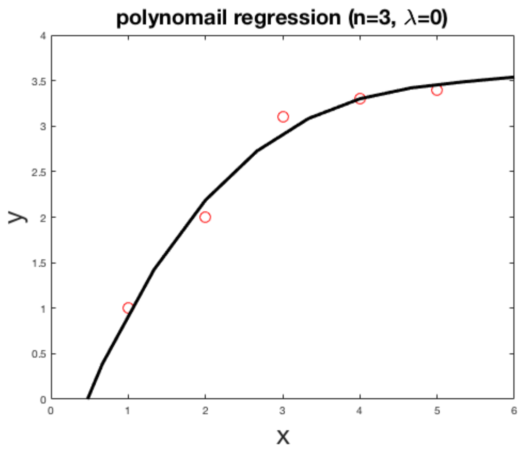


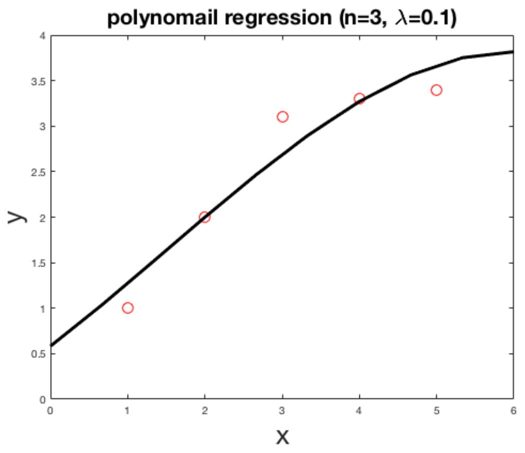


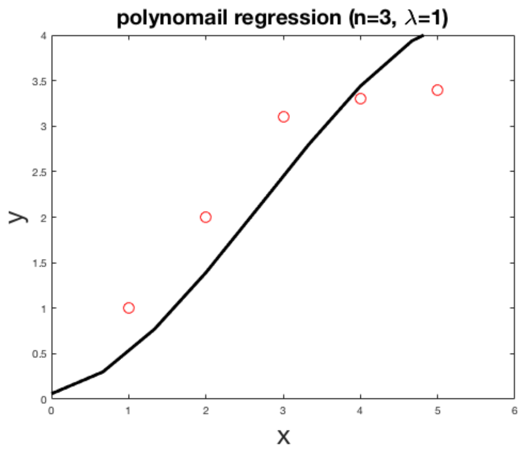
Question: you would like to apply regularization in your ML model, what will happen:

- if λ is too small?
- if λ is too large?

Example







Question: you would like to apply regularization in your ML model, what will happen:

- if λ is too small? Overfitting
- if λ is too large? Underfitting

Gradient descent for regularized linear regression

- Repeat until converge:

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} L(w,b) \quad j = 1,\ldots,n$$

$$\Downarrow$$

$$L(w,b) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} w_j^2\right]$$

Gradient descent for regularized linear regression

- Repeat until converge:

$$w_j := w_j - \alpha\left[\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}w_j\right] \quad j = 1,2,,n$$

$$b := b - \alpha\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})$$

Gradient descent for regularized linear regression

- Repeat until converge:

$$w_j := w_j - \alpha\left[\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}w_j\right] \quad j = 1,2,,n$$

$$b := b - \alpha\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})$$

Regularization term

Regularized logistic regression

- Without regularization:

$$L(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\Big(y\log(h(x)) + (1-y)\log(1-h(x))\Big)$$

- With regularization:

$$L(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\Big(y\log(h(x)) + (1-y)\log(1-h(x))\Big) + \underbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2}_{\text{Regularization term}}$$

Gradient descent for regularized logistic regression

- Repeat until converge:

$$b := b - \alpha \frac{1}{m} \sum_{i=0}^{m} (h(x^{(i)}) - y^{(i)})$$

$$w_j := w_j - \alpha \left[\frac{1}{m} \sum_{i=0}^{m} (h(x^{(i)}) - y^{(i)}) x_j^{(i)} + \underbrace{\frac{\lambda}{m} w_j}_{\text{Regularization term}} \right] \quad j = 1, 2, , n$$

Multiclass classification

- In the previous work we assume the labels in logistic regression were binary:

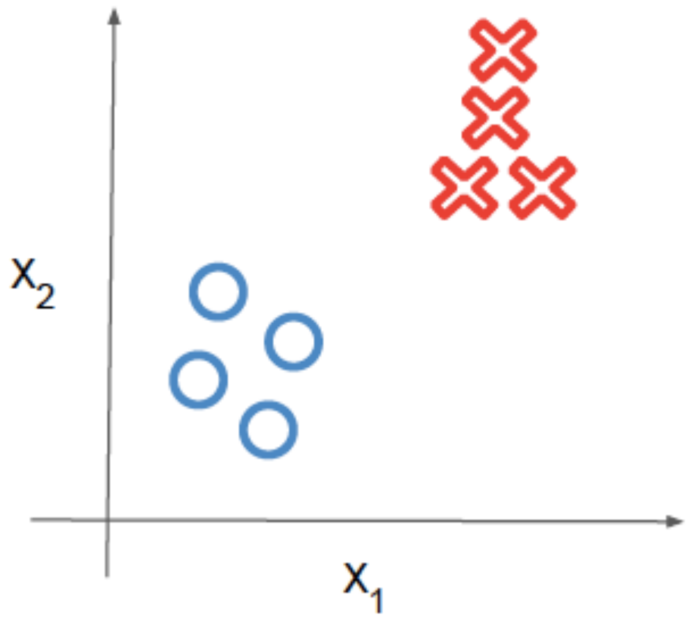
$$y \in \{0,1\}$$

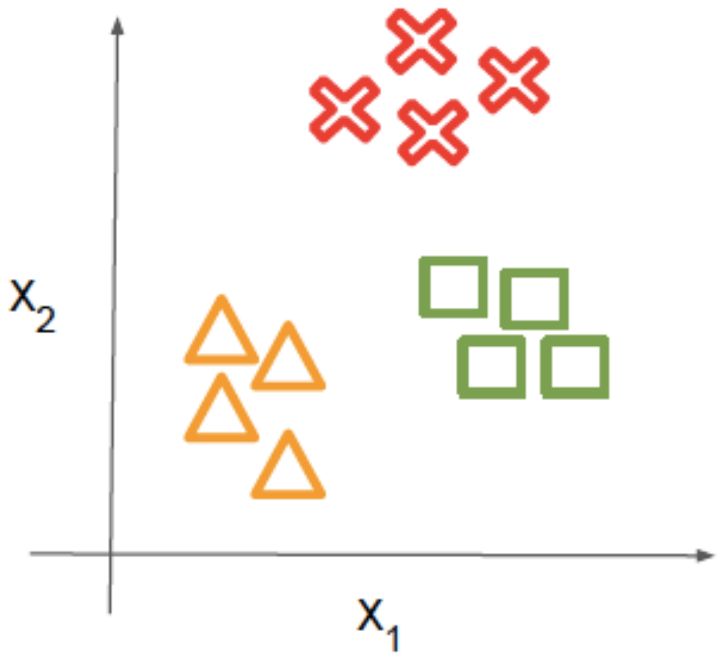
- In multiclass classification, we expand our definition so that:

$$y \in \{1,2,\ldots,K\}$$

- Examples:
  - Animal categorization: cat, dog, chicken, …
  - Weather: sunny, cloudy, rain, snow, …
  - Face recognition: person A, person B, …
  - Handwritten digits recognition: 0(1), 1(2), …, 9(10)

Example: binary vs. multiclass



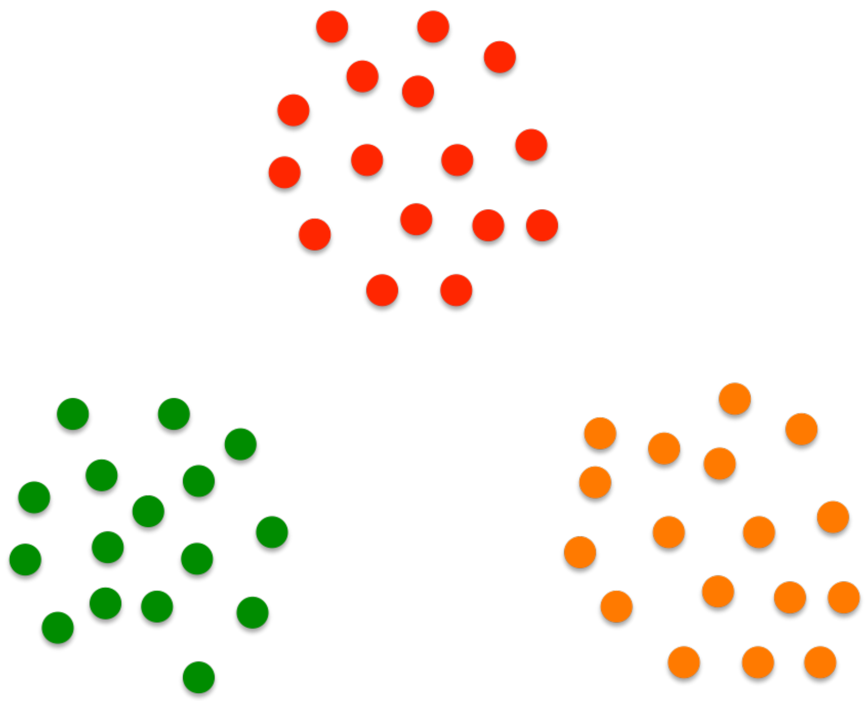


Solutions for multiclass classification

- One-vs-all (one-vs-rest):
  - Divide a multiclass classification problem into K binary classification problems
  - To make a prediction on a new data, pick the class that has the maximum output
  - Impractical when K is very large, why?
- Multiclass classification:
  - Softmax regression ("multinomial logistic regression")
  - Instead of learning a binary classifier for each class, softmax regression learns a multiclass classifier simultaneously

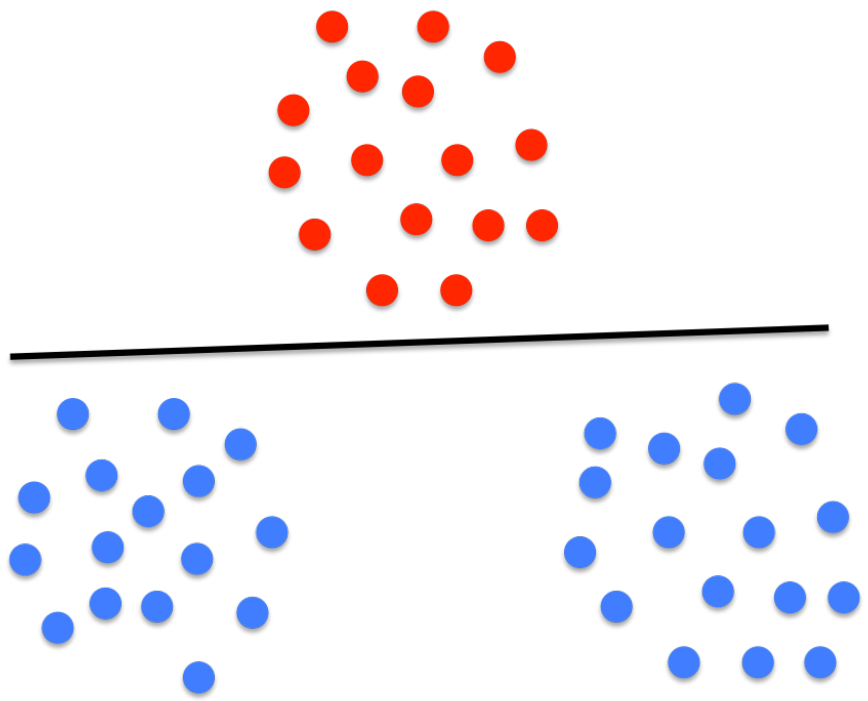
One-vs-all: example

- Suppose we have three classes colored in red, green, and orange respectively



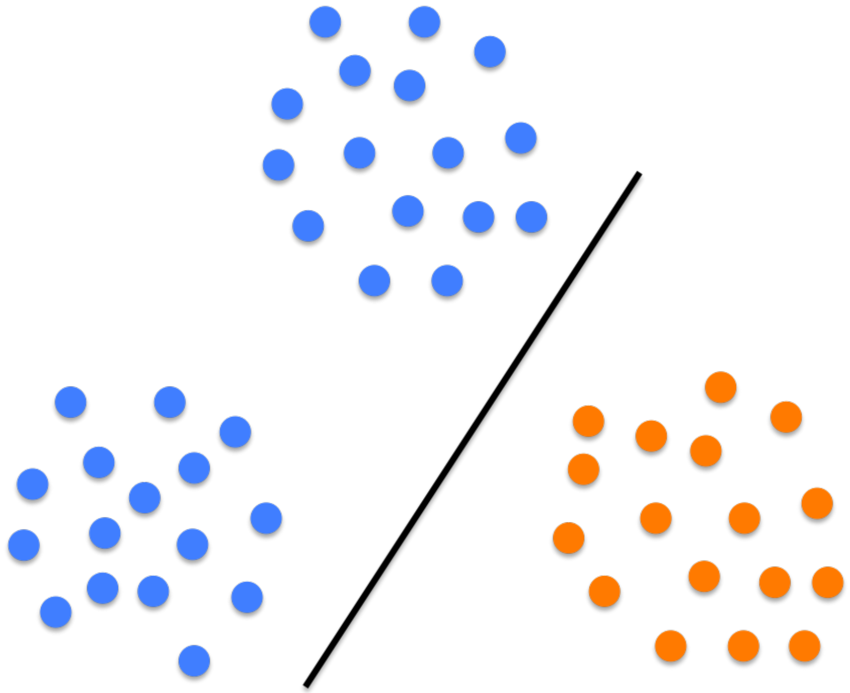
One-vs-all: example

$$h_{w,b}^{(1)}(x) = P(y = 1 \mid x; w, b)$$



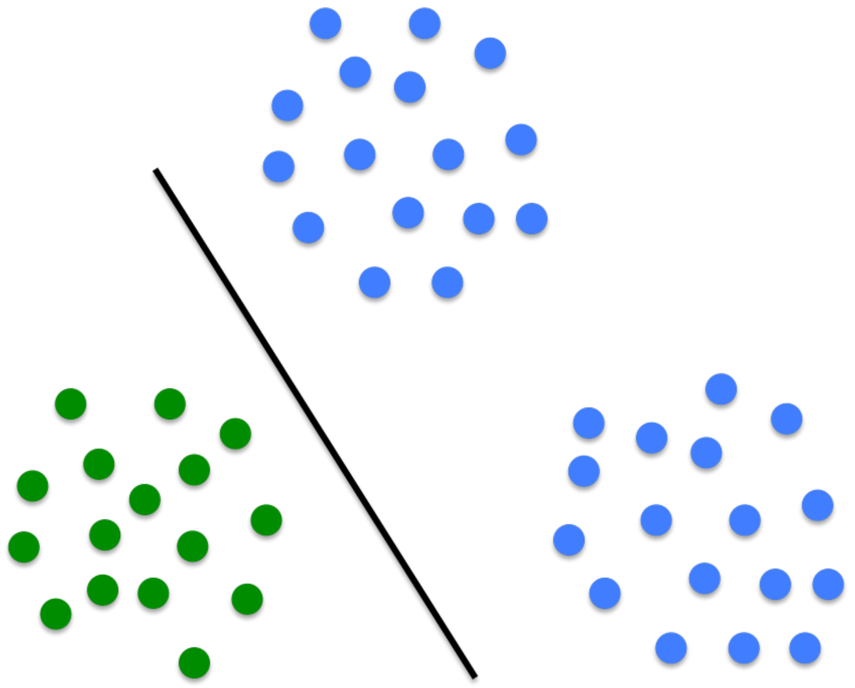
One-vs-all: example

$$h_{w,b}^{(2)}(x) = P(y = 2 \mid x; w, b)$$



One-vs-all: example

$$h_{w,b}^{(3)}(x) = P(y = 3 \mid x; w, b)$$



One-vs-all: example

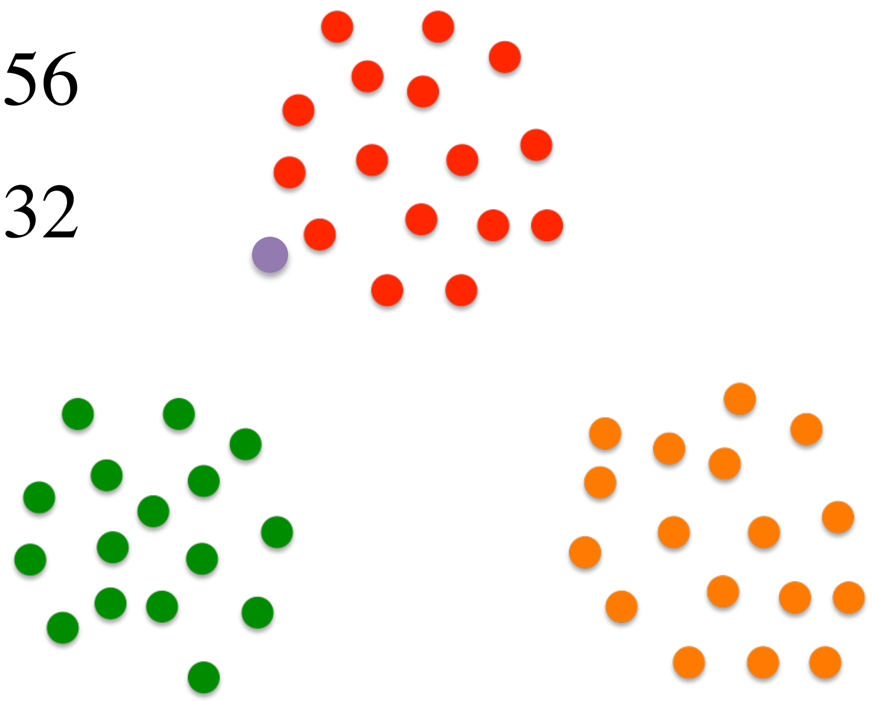
$$\text{Prediction} = \underset{k}{\operatorname{argmax}}\, h_{w,b}^{(k)}(x^{(\text{new})}) \quad k = 1,2,3$$

$$h_{w,b}^{(1)}(x^{(\text{purple})}) = 0.76$$

$$h_{w,b}^{(2)}(x^{(\text{purple})}) = 0.56$$

$$h_{w,b}^{(3)}(x^{(\text{purple})}) = 0.32$$

$$\text{Prediction} = 1$$



One-vs-all summary

- Train a binary classifier for each class to predict the output, e.g., probability
- To make a prediction on a new data, pick the class that has the maximum output

$$y \in \{1, 2, \ldots, K\}$$

$$h_{w,b}^{(1)}(x) = P(y = 1 \mid x; w, b)$$

$$h_{w,b}^{(2)}(x) = P(y = 2 \mid x; w, b)$$

$$\vdots$$

$$h_{w,b}^{(K)}(x) = P(y = K \mid x; w, b)$$

Q: what if a new data does not belong to any class?

$$\text{Prediction} = \underset{k}{\operatorname{argmax}}\, h_{w,b}^{(k)}(x^{(\text{new})})$$

Hypothesis of softmax regression

$$h_{W,b}(x) = \begin{bmatrix} P(y=1 \mid x; W, b) \\ P(y=2 \mid x; W, b) \\ \vdots \\ P(y=K \mid x; W, b) \end{bmatrix} = \frac{1}{\sum_{k=1}^{K} e^{w^{(k)}x+b^{(k)}}} \begin{bmatrix} e^{w^{(1)}x+b^{(1)}} \\ e^{w^{(2)}x+b^{(2)}} \\ \vdots \\ e^{w^{(K)}x+b^{(K)}} \end{bmatrix}$$

$\sum_{k=1}^{K} e^{w^{(k)}x+b^{(k)}}$ is the normalization term, squash the raw class scores into normalized positive values that sum to one.

$$W = \begin{bmatrix} w^{(1)} & w^{(2)} & \ldots & w^{(K)} \end{bmatrix}$$

Now W is a matrix instead of a vector.

Question: what is the size of parameters W, b?

Hypothesis of softmax regression

$$h_{W,b}(x) = \begin{bmatrix} P(y=1 \mid x; W, b) \\ P(y=2 \mid x; W, b) \\ \vdots \\ P(y=K \mid x; W, b) \end{bmatrix} = \frac{1}{\sum_{k=1}^{K} e^{w^{(k)}x+b^{(k)}}} \begin{bmatrix} e^{w^{(1)}x+b^{(1)}} \\ e^{w^{(2)}x+b^{(2)}} \\ \vdots \\ e^{w^{(K)}x+b^{(K)}} \end{bmatrix}$$

$\sum_{k=1}^{K} e^{w^{(k)}x+b^{(k)}}$ is the normalization term, squash the raw class scores into normalized positive values that sum to one.

$W = \begin{bmatrix} w^{(1)} & w^{(2)} & \ldots & w^{(K)} \end{bmatrix}$

Now W is a matrix instead of a vector.

Question: what is the size of parameters W, b? $(\mathrm{n} \times \mathrm{K})$, $(1 \times \mathrm{K})$

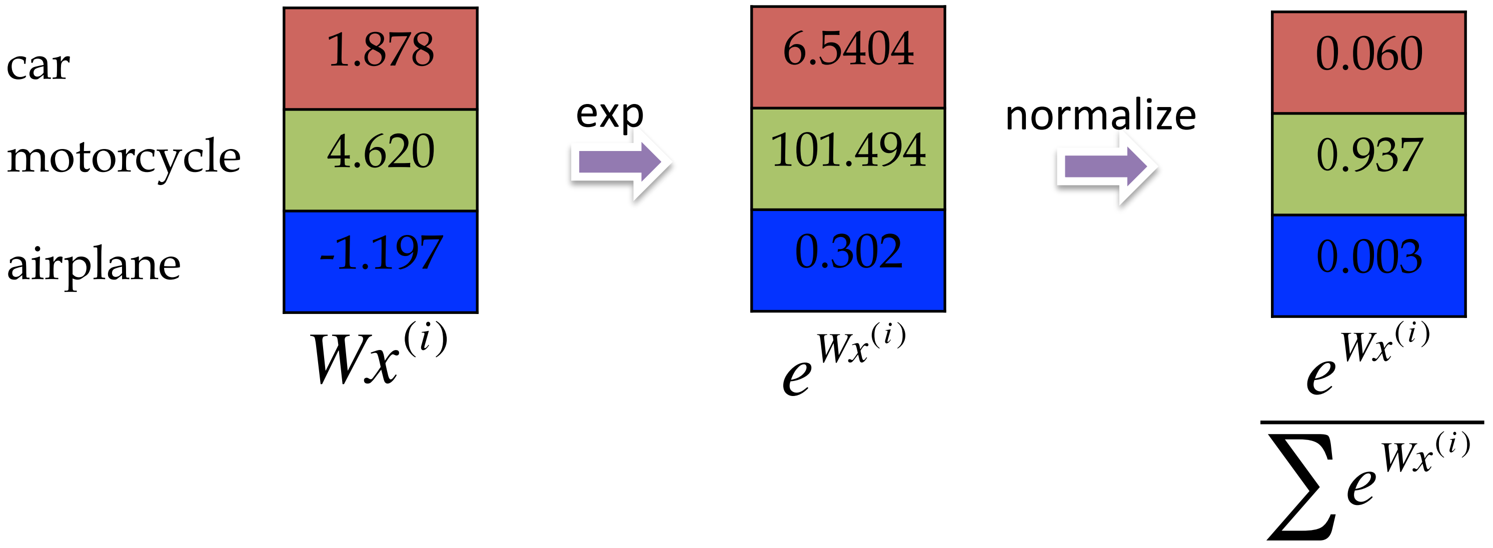
Example: softmax regression with 3 classes (car/motorcycle/airplane)

Stretch pixels into single column

$$\underbrace{\begin{bmatrix} 1.1 & 0.2 & -0.5 & 0.1 & 2.0 \\ 3.2 & 1.5 & 1.3 & 2.1 & 0.0 \\ -1.2 & 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 1 \\ 0.14 \\ -0.2 \\ 0.7 \\ 0.29 \end{bmatrix}}_{x^{(i)}} \Rightarrow \underbrace{\begin{bmatrix} 1.878 \\ 4.620 \\ -1.197 \end{bmatrix}}_{Wx^{(i)}} \begin{array}{l} \text{car score} \\ \text{motorcycle score} \\ \text{airplane score} \end{array}$$

Omit bias b for simplicity!

Example: softmax regression with 3 classes (car/motorcycle/airplane)



Indicator function

- Indicator function $\mathbf{1}\{\cdot\}$
  - $\mathbf{1}\{\text{a true statement}\} = 1$
  - $\mathbf{1}\{\text{a false statement}\} = 0$
- We can rewrite the loss function of binary logistic regression:

$$L(W,b) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=0}^{1}(\mathbf{1}\{y^{(i)} = k\}\log P(y^{(i)} = k \mid x^{(i)};W,b))$$

Loss function of softmax regression

- Extend loss function of binary logistic regression to softmax regression

$$L(W,b) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left(\mathbf{1}\{y^{(i)} = k\}\log\frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K}e^{w^{(j)}x^{(i)}+b^{(j)}}}\right)$$

where $$\frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K}e^{w^{(j)}x^{(i)}+b^{(j)}}} = P(y^{(i)} = k \mid x^{(i)};W,b)$$

One hot encoding

- One hot: “one-hot is a group of bits among which the legal combinations of values are only those with a single high (1) bit and all the others low (0)” – Wikipedia
- A one hot encoding is a representation of categorical variables as binary vectors
- Example: suppose you want to classify 3 categories objects: car, motorbike, airplane. You have three images that are labeled as car (1), motorbike (2), airplane (3), respectively. The corresponding one hot vectors are:

$$\text{car} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \text{motorbike} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \text{airplane} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Example: loss function of softmax regression

- Given the ground truth as one hot vector and their prediction, compute the loss:

Ground truth $\text{car} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $\text{motorbike} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ $\text{airplane} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Prediction $\text{car} = \begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix}$ $\text{motorbike} = \begin{bmatrix} 0.5 \\ 0.1 \\ 0.4 \end{bmatrix}$ $\text{airplane} = \begin{bmatrix} 0.2 \\ 0.7 \\ 0.1 \end{bmatrix}$

$$\begin{aligned} \text{loss} = -\frac{1}{3}(&1\log(0.2) + 0\log(0.5) + 0\log(0.3) \\ &+ 0\log(0.5) + 1\log(0.1) + 0\log(0.4) \\ &+ 0\log(0.2) + 0\log(0.7) + 1\log(0.1)) = 2.07 \end{aligned}$$

Example: loss function of softmax regression

- Given the ground truth as one hot vector and their prediction, compute the loss:

Ground truth $\text{car} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $\text{motorbike} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ $\text{airplane} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Prediction $\text{car} = \begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix}$ $\text{motorbike} = \begin{bmatrix} 0.5 \\ 0.1 \\ 0.4 \end{bmatrix}$ $\text{airplane} = \begin{bmatrix} 0.2 \\ 0.7 \\ 0.1 \end{bmatrix}$

$$\begin{aligned}\text{loss} = -\frac{1}{3}(&1\log(0.2) + 0\log(0.5) + 0\log(0.3) \\ &+ 0\log(0.5) + 1\log(0.1) + 0\log(0.4) \\ &+ 0\log(0.2) + 0\log(0.7) + 1\log(0.1)) = 2.07\end{aligned}$$

High loss

Example: loss function of softmax regression

- Given the ground truth as one hot vector and their prediction, compute the loss:

Ground truth $\text{car} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $\text{motorbike} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ $\text{airplane} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Prediction $\text{car} = \begin{bmatrix} 0.8 \\ 0.1 \\ 0.1 \end{bmatrix}$ $\text{motorbike} = \begin{bmatrix} 0.1 \\ 0.9 \\ 0 \end{bmatrix}$ $\text{airplane} = \begin{bmatrix} 0.1 \\ 0.05 \\ 0.85 \end{bmatrix}$

$$\begin{aligned} \text{loss} = -\frac{1}{3}(&1\log(0.8) + 0\log(0.1) + 0\log(0.1) \\ &+ 0\log(0.1) + 1\log(0.9) + 0\log(0) \\ &+ 0\log(0.1) + 0\log(0.05) + 1\log(0.85)) = 0.18 \end{aligned}$$

Example: loss function of softmax regression

- Given the ground truth as one hot vector and their prediction, compute the loss:

Ground truth $\text{car} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $\text{motorbike} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ $\text{airplane} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Prediction $\text{car} = \begin{bmatrix} 0.8 \\ 0.1 \\ 0.1 \end{bmatrix}$ $\text{motorbike} = \begin{bmatrix} 0.1 \\ 0.9 \\ 0 \end{bmatrix}$ $\text{airplane} = \begin{bmatrix} 0.1 \\ 0.05 \\ 0.85 \end{bmatrix}$

$$\begin{aligned}\text{loss} = -\frac{1}{3}(&1\log(0.8) + 0\log(0.1) + 0\log(0.1) \\ &+ 0\log(0.1) + 1\log(0.9) + 0\log(0) \\ &+ 0\log(0.1) + 0\log(0.05) + 1\log(0.85)) = 0.18\end{aligned}$$

Low loss

Softmax regression properties

- Softmax regression has a “redundant” set of parameters
  - Instead of training (n + 1)K parameters, you may only need to train (n + 1)(K - 1) parameters
  - When K = 2, softmax regression reduces to logistic regression

Partial derivative of loss function

$$L(W,b) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}(\mathbf{1}\{y^{(i)} = k\}\log\frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K}e^{w^{(j)}x^{(i)}+b^{(j)}}})$$

$$\frac{\partial}{\partial w^{(k)}}L(W,b) = -\frac{1}{m}\sum_{i=1}^{m}(\mathbf{1}\{y^{(i)} = k\} - \frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K}e^{w^{(j)}x^{(i)}+b^{(j)}}})x^{(i)}$$

$$\frac{\partial}{\partial b^{(k)}}L(W,b) = -\frac{1}{m}\sum_{i=1}^{m}(\mathbf{1}\{y^{(i)} = k\} - \frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K}e^{w^{(j)}x^{(i)}+b^{(j)}}})$$

Regularized softmax regression

- Loss function

$$L(W,b) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}(\mathbf{1}\{y^{(i)} = k\}\log\frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K}e^{w^{(j)}x^{(i)}+b^{(j)}}}) + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{k=1}^{K}w_{jk}^{2}$$

- Partial derivative

$$\frac{\partial}{\partial w^{(k)}}L(W,b) = -\frac{1}{m}\sum_{i=1}^{m}(\mathbf{1}\{y^{(i)} = k\} - \frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K}e^{w^{(j)}x^{(i)}+b^{(j)}}})x^{(i)} + \lambda w^{(k)}$$

$$\frac{\partial}{\partial b^{(k)}}L(W,b) = -\frac{1}{m}\sum_{i=1}^{m}(\mathbf{1}\{y^{(i)} = k\} - \frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K}e^{w^{(j)}x^{(i)}+b^{(j)}}})$$

Softmax regression optimization

- Repeat until converge:

$$w^{(k)} := w^{(k)} - \alpha \frac{\partial}{\partial w^{(k)}} L(W,b)$$

$$\Rightarrow w^{(k)} := w^{(k)} + \alpha[\frac{1}{m}\sum_{i=1}^{m}(\mathbf{1}\{y^{(i)} = k\} - \frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K} e^{w^{(j)}x^{(i)}+b^{(j)}}})x^{(i)} - \lambda w^{(k)}]$$

$$\Rightarrow w^{(k)} := \underbrace{(1-\alpha\lambda)w^{(k)}}_{\text{weight decay}} + \alpha \frac{1}{m}\sum_{i=1}^{m}(\mathbf{1}\{y^{(i)} = k\} - \frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K} e^{w^{(j)}x^{(i)}+b^{(j)}}})x^{(i)}$$

$$b^{(k)} := b^{(k)} + \alpha[\frac{1}{m}\sum_{i=1}^{m}(\mathbf{1}\{y^{(i)} = k\} - \frac{e^{w^{(k)}x^{(i)}+b^{(k)}}}{\sum_{j=1}^{K} e^{w^{(j)}x^{(i)}+b^{(j)}}})]$$

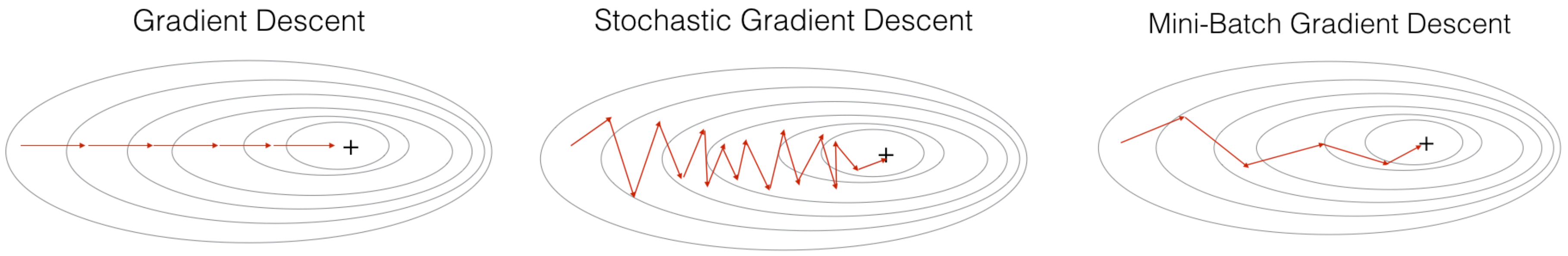
Batch and Mini-batch algorithms

- Motivation: computational cost and redundancy in the training set
- Batch/deterministic method:
  - process all the training example simultaneously in a large batch
  - fast to converge
  - computation is very expensive when your training set is very large
- Stochastic/online method:
  - only a single example at a time
  - suitable when examples are drawn from a stream of continually created examples
  - hard to converge
- Mini-batch/mini-batch stochastic:
  - use more than one but fewer than all the training examples

Mini-batch gradient descent

- Repeat until an approximate minimum is obtained:
  - Random shuffle examples in the training set
  - Split examples into N batches
  - For each batch do:

$$w := w - \alpha \frac{\partial L}{\partial w}$$



When N = m

One-vs-all or softmax regression

- Check the classes you want to divide is mutually exclusive or not.
  - If mutually exclusive, choose softmax regression
  - If not mutually exclusive, choose one-vs-all
- Example: Suppose you are working on a music classification application, and there are k types of music that you are trying to recognize.
  - if your four classes are classical, country, rock, and jazz, you should use softmax regression
  - If your categories are vocals, dance, soundtrack, pop, it is more appropriate to use one-vs-all

Question: one-vs-all or softmax regression

- Would you use softmax regression or three logistic regression classifiers in the following examples?
  - Suppose that your classes are indoor_scene, outdoor_urban_scene, and outdoor_wilderness_scene
  - Suppose your classes are indoor_scene, black_and_white_image, and image_has_people.

Question: one-vs-all or softmax regression

- Would you use softmax regression or three logistic regression classifiers in the following examples?
  - Suppose that your classes are indoor_scene, outdoor_urban_scene, and outdoor_wilderness_scene

    Softmax
  - Suppose your classes are indoor_scene, black_and_white_image, and image_has_people.

    One-vs-all

Summary

- Linear regression is a regression model, whereas logistic regression and softmax regression are classification models
- Linear regression and logistic regression have the same partial derivative although their loss functions are different
- When K=2, softmax regression reduces to logistic regression
- ML models overfit when hypothesis is excessively complex and underfit when hypothesis is too simple
- Regularization is an effective way to address overfitting, where the regularization parameter controls the tradeoff between two different goals: fitting the training set well and keeping the parameter small

Reading suggestion

- Chapter 5 in ‘Deep learning’
- Softmax regression: Sec 4.3.4 in ‘Pattern Recognition and Machine Learning’