Artificial Intelligence II (CS4442 & CS9542)

Classification: Logistic Regression

Boyu Wang
Department of Computer Science
University of Western Ontario

Recall: tumor example

- Thirty real-valued variables per tumor.
- Two variables that can be predicted:
  - Outcome (R=recurrence, N=non-recurrence)
  - Time (until recurrence, for R, time healthy, for N).

tumor size	texture	perimeter	...	outcome	time
18.02	27.6	117.5		N	31
17.99	10.38	122.8		N	61
20.29	14.34	135.1		R	27
...					

Slide credit: Doina Precup

Recall: supervised learning

- A training example i has the form: (x_i, y_i), where $x_i \in \mathbb{R}^n$ it the number of features (feature dimension). If $y \in \mathbb{R}$, this is the regression problem.
- If $y \in \{0, 1\}$, this is the binary classification problem.
- If $y \in \{1, \dots, C\}$ (i.e., y can take more than two values), this is the multi-class classification problem.
- Most binary classification algorithms can be extended to multi-class classification algorithms.

Linear model for classification

- As in linear regression, we consider a linear model h_w for classification:

$$h_w(x) = w^\top x$$

where we already augmented the data: $x \to [x; 1]$.

- Rules for binary classification: $h_w(x) \geq 0 \Rightarrow y = 1$; $h_w(x) < 0 \Rightarrow y = 0$

- How to choose w?

Error (cost) function for classification

- Recall: for regression, we use the sum-of-squared errors:

$$J(w) = \frac{1}{2}\sum_{i=1}^{m}(h_w(x_i) - y_i)^2$$

Error (cost) function for classification

- Recall: for regression, we use the sum-of-squared errors:

$$J(w) = \frac{1}{2}\sum_{i=1}^{m}(h_w(x_i) - y_i)^2$$

- Can we use it for classification?

Error (cost) function for classification

- Recall: for regression, we use the sum-of-squared errors:

$$J(w) = \frac{1}{2}\sum_{i=1}^{m}(h_w(x_i) - y_i)^2$$

- Can we use it for classification?

- We could, but it is not the best choice

  - If $y = 1$, we want $h_w(x) > 0$ as much as possible, which reflects our confidence of classification
  - recall the connection between linear regression and maximum likelihood with Gaussian assumption

Probabilistic view for classification

- We would like a way of learning that is more suitable for the problem

Probabilistic view for classification

- We would like a way of learning that is more suitable for the problem

- Classification can be formulated as the following question: given a data point x, what is the probability $p(y|x)$?

Probabilistic view for classification

- We would like a way of learning that is more suitable for the problem
- Classification can be formulated as the following question: given a data point x, what is the probability $p(y|x)$?
- Intuition: we want find the conditional probability $p(y|x; w)$ parameterized by w in a way such that

$$h_w(x) = w^\top x \to +\infty \Rightarrow p(y = 1|x) \to 1$$
$$h_w(x) = w^\top x \to -\infty \Rightarrow p(y = 1|x) \to 0 \quad \text{(i.e., } p(y = 0|x) \to 1\text{)}$$
$$h_w(x) = w^\top x = 0 \Rightarrow p(y = 1|x) = 0.5$$

Sigmoid function

Consider the following function:

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

- $a \to +\infty \Rightarrow \sigma(a) \to 1$
- $a \to -\infty \Rightarrow \sigma(a) \to 0$
- $a = 0 \Rightarrow \sigma(a) = 0.5$

Plug $h_w(x)$ into $\sigma(\cdot)$:

$$p(y = 1|x; w) \triangleq \sigma(h_w(x)) = \frac{1}{1+e^{-w^\top x}},$$

which is exactly what we are looking for!

1D sigmoid function

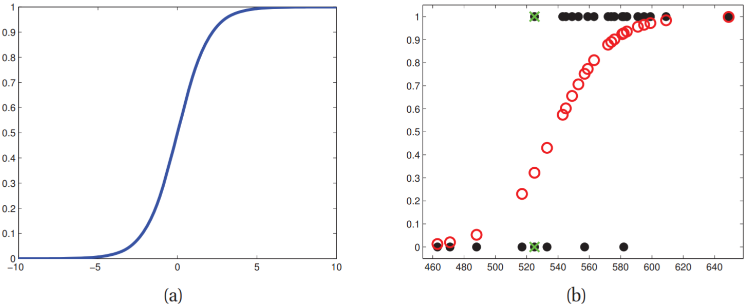


Figure: The sigmoid function and the predicted probabilities.

Figure credit: Kevin Murphy

2D sigmoid function

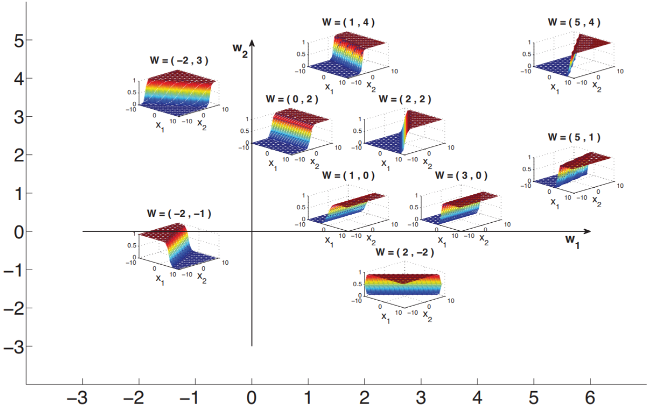


Figure: Plots of $\sigma(w_1 x_1 + w_2 x_2)$

Figure credit: Kevin Murphy

Error (cost) function for classification – revisited

- Maximum likelihood classification assumes that we will find the hypothesis that maximizes the (log) likelihood of the training data:

$$\arg\max_h \log p(\{x_i, y_i\}_{i=1}^m | h) = \arg\max_h \sum_{i=1}^{m} \log p(x_i, y_i | h)$$

  (using the i.i.d. assumption)

- If we ignore the marginal distribution $p(x)$, we may maximize the conditional probability of the labels, given the inputs and the hypothesis h:

$$\arg\max_h \sum_{i=1}^{m} \log p(y_i | x_i; h)$$

The cross-entropy loss function for logistic regression

- Recall that for any data point (x_i, y_i), the conditional probability can be represented by the sigmoid function:

$$p(y_i = 1 | x_i; h) = \sigma(h_w(x_i))$$

- Then the log-likelihood of a hypothesis h_w is

$$\log L(w) = \sum_{i=1}^{m} \log p(y_i | x_i; h_w) = \sum_{i=1}^{m} \begin{cases} \log \sigma(h_w(x_i)), & \text{if } y_i = 1 \\ \log(1 - \sigma(h_w(x_i))), & \text{if } y_i = 0 \end{cases}$$
$$= \sum_{i=1}^{m} \left(y_i \log t_i + (1 - y_i) \log(1 - t_i) \right),$$

where $t_i = \sigma(h_w(x_i))$.

- The cross-entropy loss function is the negative of $\log L(w)$.

Linear regression vs. logistic regression

Both use linear model: $h_w(x) = w^\top x$

- Conditional probability:
  - linear regression: $p(y_i|x_i; w) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y_i - w^\top x_i}{\sigma}\right)^2}$
  - logistic regression: $p(y_i|x_i; w) = \begin{cases} \frac{1}{1+e^{-w^\top x_i}}, & \text{if } y_i = 1 \\ 1 - \frac{1}{1+e^{-w^\top x_i}}, & \text{if } y_i = 0 \end{cases}$
- Log-likelihood function:
  - linear regression: $\log L(w) = \sum_{i=1}^m \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y_i - w^\top x_i}{\sigma}\right)^2} \right)$
  - logistic regression:
    $\log L(w) = \sum_{i=1}^m (y_i \log t_i + (1 - y_i) \log(1 - t_i))$, where
    $t_i = \sigma(h_w(x_i))$.
- Solution:
  - linear regression: $w = (x^\top X)^{-1} X^\top y$
  - logistic regression: No analytical solution

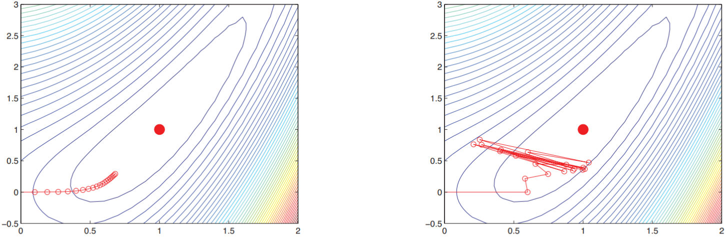
Optimization procedure: gradient descent

Objective: minimize a loss function $J(w)$

Gradient descent for function minimization

Input: number of iterations: N, learning rate: α
Initialize $w_{(0)}$
for $n = 1$ to N **do**
  Compute the gradient: $g_{(n)} = \nabla J(w_{(n)})$
  $w_{(n+1)} = w_{(n)} - \alpha g_{(n)}$
  if converges (e.g, $|w_{(n+1)} - w_{(n)}| \leq \epsilon$) **then**
    Stop
  end if
end for

Effect of the learning rate



(a) (b)

Figure: Gradient descent on a simple function, starting from $(0, 0)$, for 20 steps using a fixed learning rate α. (a) $\alpha = 0.1$. (b) $\alpha = 0.6$

Figure credit: Kevin Murphy

Optimization procedure: Newton's method

Objective: minimize a loss function $J(w)$

Newton's method for function minimization

Input: number of iterations: N

Initialize $w_{(0)}$

for $n = 1$ to N **do**

  Compute the gradient: $g_{(n)} = \nabla J(w_{(n)})$

  Compute the Hessian: $H_{(n)} = \nabla^2 J(w_{(n)})$

  $w_{(n+1)} = w_{(n)} - H_{(n)}^{-1} g_{(n)}$

  if converges (e.g, $|w_{(n+1)} - w_{(n)}| \leq \epsilon$) **then**

    Stop

  end if

end for

Gradient descent for logistic regression

Objective: minimize the cross-entropy loss function($-\log L(w)$):

$$J(w) \triangleq -\log L(w) = -\sum_{i=1}^{m} \left(y_i \log t_i + (1 - y_i)\log(1 - t_i)\right)$$

$$\nabla J(w) = \sum_{i=1}^{m} (t_i - y_i)x_i, \quad t_i = \sigma(h_w(x_i)) = \frac{1}{1 + e^{-w^\top x_i}}$$

Gradient descent for logistic regression

1: **Input:** number of iterations: N, learning rate: α

2: Initialize $w_{(0)}$

3: **for** $n = 1$ to N **do**

4: Compute the gradient: $g_{(n)} = \nabla J(w_{(n)}) = \sum_{i=1}^{m}(t_i - y_i)x_i$

5: $w_{(n+1)} = w_{(n)} - \alpha g_{(n)}$

6: **if** converges (e.g, $|w_{(n+1)} - w_{(n)}| \leq \epsilon$) **then**

7: Stop

8: **end if**

9: **end for**

Newton's method for logistic regression

Objective: minimize the cross-entropy loss function($-\log L(w)$):

$$J(w) \triangleq -\log L(w) = -\sum_{i=1}^{m} (y_i \log t_i + (1-y_i)\log(1-t_i))$$

$$\nabla J(w) = \sum_{i=1}^{m} (t_i - y_i)x_i, \quad H(w) = \nabla^2 J(w) = \sum_{i=1}^{m} t_i(1-t_i)x_i^\top x_i = X^\top S X$$

where $S \in \mathbb{R}^{m\times m}$ is a diagonal matrix with elements $S_{ii} = t_i(1-t_i)$,
$t_i = \sigma(h_w(x_i)) = \frac{1}{1+e^{-w^\top x_i}}$.
Then, the update rule becomes:

$$w_{(n+1)} = w_{(n)} - H(w_{(n)})^{-1}\nabla J(w_{(n)})$$

Gradient descent vs. Newton's method

- Newtons method usually requires significantly fewer iterations than gradient descent
- Computing the Hessian and its inverse is expensive
- Approximation algorithms exist which help to compute the product of the inverse Hessian with gradient without explicitly computing H

Cross-entropy loss vs. squared loss

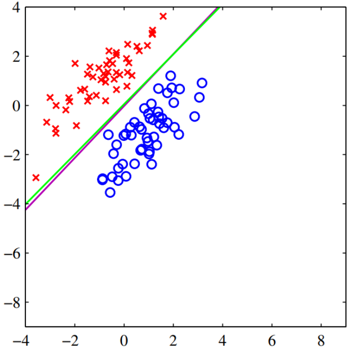


Figure: Decision boundaries obtained by minimizing squared loss (magenta line) and cross-entropy loss (green line)

Figure credit: Christopher Bishop

Regularized logistic regression

One can do regularization for logistic regression just like in the case of linear regression.

- ℓ_2-regularized logistic regression

$$J(w) \triangleq -\log L(w) + \frac{\lambda}{2}||w||_2^2$$

2. ℓ_1-regularized logistic regression

$$J(w) \triangleq -\log L(w) + \lambda||w||_1$$

Multi-class logistic regression

- For 2 classes:

$$p(y = 1|x; w) = \frac{1}{1 + e^{-w^\top x}} = \frac{e^{w^\top x}}{1 + e^{w^\top x}}$$

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

$$p(y = c|x; w_1, \ldots, w_C) = \frac{e^{w_c^\top x}}{\sum_{c=1}^{C} e^{w_c^\top x}}$$

– called the softmax function

Multi-class logistic regression

Gradient descent for multi-class logistic regression

```
Input: number of iterations: N, learning rate: α
Initialize C vectors: w_{1,(0)}, ..., w_{C,(0)}
for n = 1 to N do
    for c = 1 to C do
        Compute the gradient with respect to w_{c,(n)}: g_{c,(n)}
        w_{c,(n+1)} = w_{c,(n)} − α g_{c,(n)}
    end for
    if converges (e.g, |w_{c,(n+1)} − w_{c,(n)}| ≤ ϵ for all classes) then
        Stop
    end if
end for
```

Multi-class logistic regression

- After training, the probability of $y = c$ is given by

$$p(y = c|x; w_1, \ldots, w_C) \triangleq \sigma_c(x) = \frac{e^{w_c^\top x}}{\sum_{c=1}^{C} e^{w_c^\top x}}$$

- Predict class label as the most probable label:

$$y = \arg\max_c \sigma_c(x)$$

Summary

- Logistic regression for classification
- Cross-entropy loss for logistic regression
- No analytical solution that minimizes the cross-entropy loss/maximizes the log-likelihood
- Use gradient descent to find a solution
- Multi-class logistic regression