CS 333: Natural Language Processing

Fall 2025

Prof. Carolyn Anderson
Wellesley College

Reminders

- HW 5 will be released today:
  - Regression with hand-crafted features to classify song lyrics by artist
- Quiz 6 on Tuesday (J&M Chapter 6)
- My next help hours: Monday 4-5:30

PyTorch !!

EAST ASIAN LANGUAGES AND CULTURES &
COGNITIVE AND LINGUISTIC SCIENCES DEPARTMENT PRESENT

AI AND HUMANITIES:

Human and Computational Language Models Can Help Each Other - Focusing on Linguistic Testing in Korean

Dr. Sanghoun Song, Associate Professor of Linguistics and Director of the Research Institute of Language and Information at Korea University

Wednesday, October 22

4:30 - 5:30 pm

Science Center - H401

Regression

Classification in logistic regression: summary

Given:

- a set of classes: (+ sentiment,- sentiment)
- a vector **x** of features `[x1, x2, …, xn]`
  - x1= count("awesome")
  - x2 = log(number of words in review)
- A vector **w** of weights `[w1, w2, …, wn]`
  - w_i for each feature f_i

z

$$P(y=1) = \sigma(w \cdot x + b)$$

P(y=0) = 1 - P(y=1)

$$= \frac{1}{1+\exp(-(w \cdot x + b))}$$

The two phases of logistic regression

Training: we learn weights w and b using **stochastic gradient descent** and **cross-entropy loss**.

Test: Given a test example x we compute $p(y|x)$ using learned weights w and b, and return whichever label ($y = 1$ or $y = 0$) is higher probability

How Does Learning Work?

Learning in Supervised Classification

Supervised classification:

- We know the correct label $\boldsymbol{y}$ (either 0 or 1) for each x.
- But what the system produces is an estimate, $\hat{y}$

We want to set w and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $\boldsymbol{y}^{(i)}$.

- We need a distance estimator: a **loss function** or a **cost function**
- We need an optimization algorithm to update w and b to minimize the loss.

Learning components

A loss function:

cross-entropy loss

An optimization algorithm:

stochastic gradient descent

The distance between $\hat{y}$ and y

We want to know how far is the classifier output:

$$\hat{y} = \sigma(w \cdot x + b)$$

from the true output:

y [either $\overset{-}{0}$ or $\overset{+}{1}$]

We'll call this difference the *loss*: $L(\hat{y}, y) =$ how far off $\hat{y}$ is from y.

Intuition of negative log likelihood loss = cross-entropy loss

A case of conditional maximum likelihood estimation

We choose the parameters w,b that maximize

- the log probability
- of the true y labels in the training data
- given the observations x

Deriving cross-entropy loss for a single observation x

Goal: maximize probability of the correct label $p(y|x)$

Since there are only 2 discrete outcomes (0 or 1) we can express the probability $p(y|x)$ from our classifier as:

$$p(y|x) = \hat{y}^{y}(1-\hat{y})^{1-y}$$

If $y=1$, this simplifies to $\hat{y}$

If $y=0$, this simplifies to $1-\hat{y}$

Deriving cross-entropy loss for a single observation x

Goal: maximize probability of the correct label $p(y|x)$

Maximize: $p(y|x) = \hat{y}^{y}(1-\hat{y})^{1-y}$

$$\log p(y|x) = \log\left[\hat{y}^{y}(1-\hat{y})^{1-y}\right]$$
$$= y\log\hat{y} + (1-y)\log(1-\hat{y})$$

Deriving cross-entropy loss for a single observation x

Goal: maximize probability of the correct label $p(y|x)$

Minimize the cross-entropy loss

Minimize: $L_{\mathrm{CE}}(\hat{y}, y) = -\log p(y|x)$

$$= -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$$

$$= -[y \log \sigma(wx+b) + (1-y)\log(1-\sigma(wx+b))]$$

Does this work for our sentiment example?

We want loss to be:

- smaller if the model estimate is close to correct
- bigger if model is confused

Let's first suppose the true label of this is $y=0$ (negative)

CATS was a marvelous disaster, with witty charm and emotion throughout, cheeky charisma and crying no doubt... I personally went in expecting the worst movie I had ever seen - and it was far more awful and disappointing that I expected.

Let's see if this works for our sentiment example

True value is y=0. How well is our model doing?

$$p(+|x) = P(y=1|x) = \sigma(wx+b)$$

$$\sigma\left(\overset{w}{[2.5, -5, 0.5, 2, 0.7]} * \overset{x}{[0, 4, 1, 0, 3.74]} + 0.1\right)$$

$$= \sigma(-1.78)$$

$$= 0.14 \leftarrow \hat{y}$$

$$p(-|x) = 1 - 0.11 = 0.86 \leftarrow 1 - \hat{y}$$

Learning components

A loss function:

- **cross-entropy loss**

How wrong is our model?

An optimization algorithm:

- **stochastic gradient descent**

Stochastic Gradient Descent

Slides borrowed from Jurafsky & Martin Edition 3

Our goal: minimize the loss

Let's make explicit that the loss function is parameterized by weights Θ=(w,b)

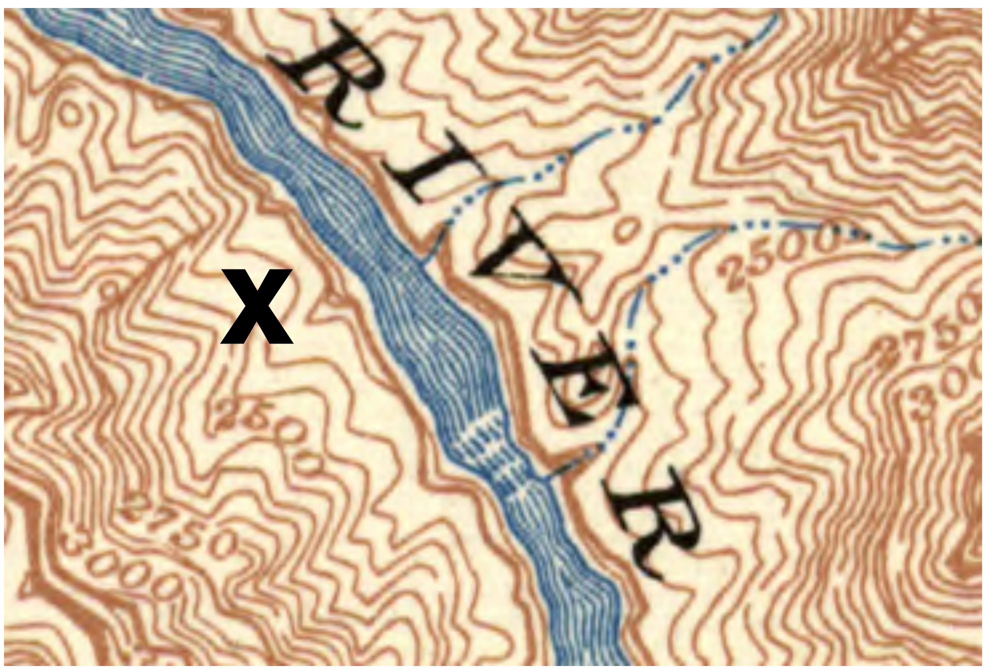
We'll represent $\hat{y}$ as f (x; θ) to make the dependence on θ more obvious

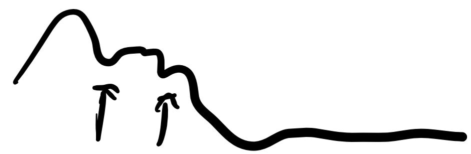
We want the weights that minimize the loss, averaged over all examples:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(f(x^{(i)}; \theta), y^{(i)})$$

Intuition of gradient descent

How do I get to the bottom of this river canyon?





Look around me 360°

Find the direction of steepest slope down

Go that way

Our goal: minimize the loss

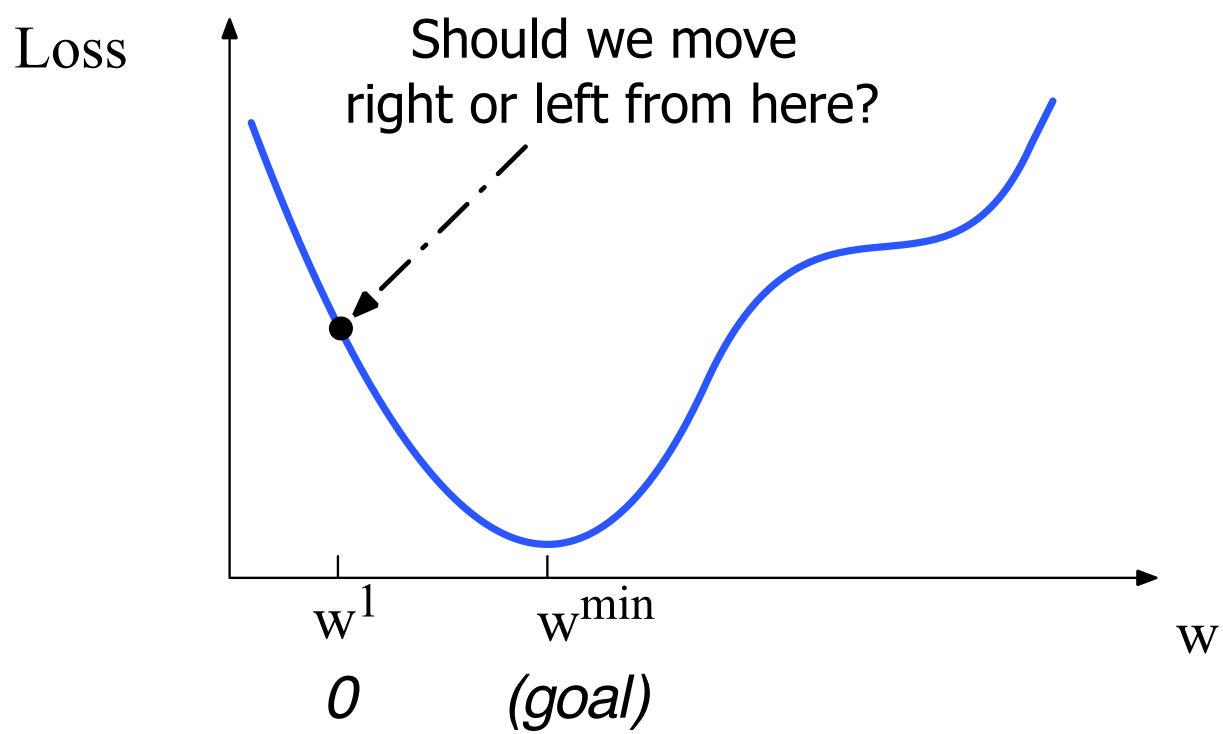
For logistic regression, loss function is **convex**

- A convex function has just one minimum
- Gradient descent starting from any point is guaranteed to find the minimum
    - (Loss for neural networks is non-convex)

Let's first visualize for a single scalar w

Q: Given current w, should we make it bigger or smaller?

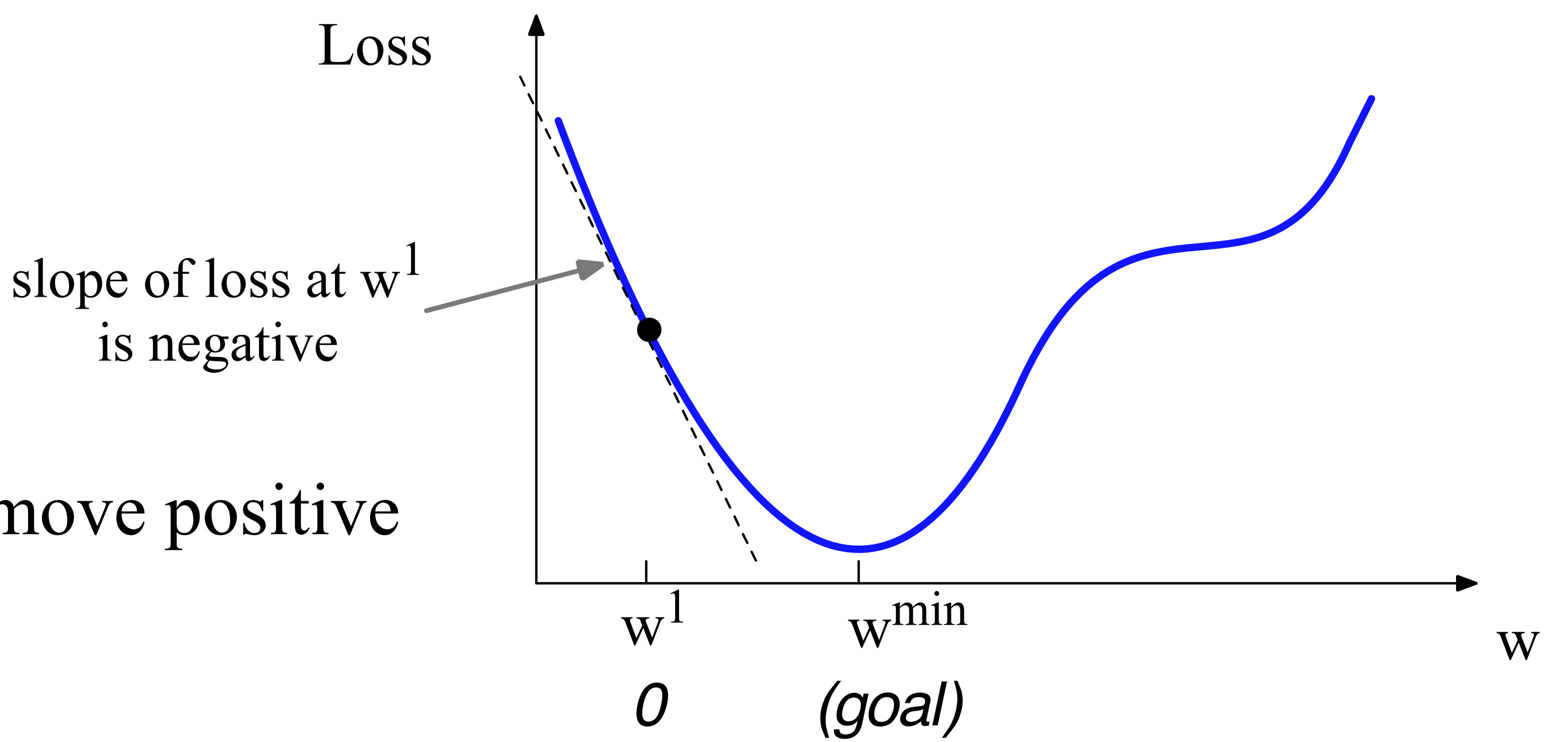
A: Move *w* in the reverse direction from the slope of the function



Let's first visualize for a single scalar w

Q: Given current w, should we make it bigger or smaller?

A: Move *w* in the reverse direction from the slope of the function

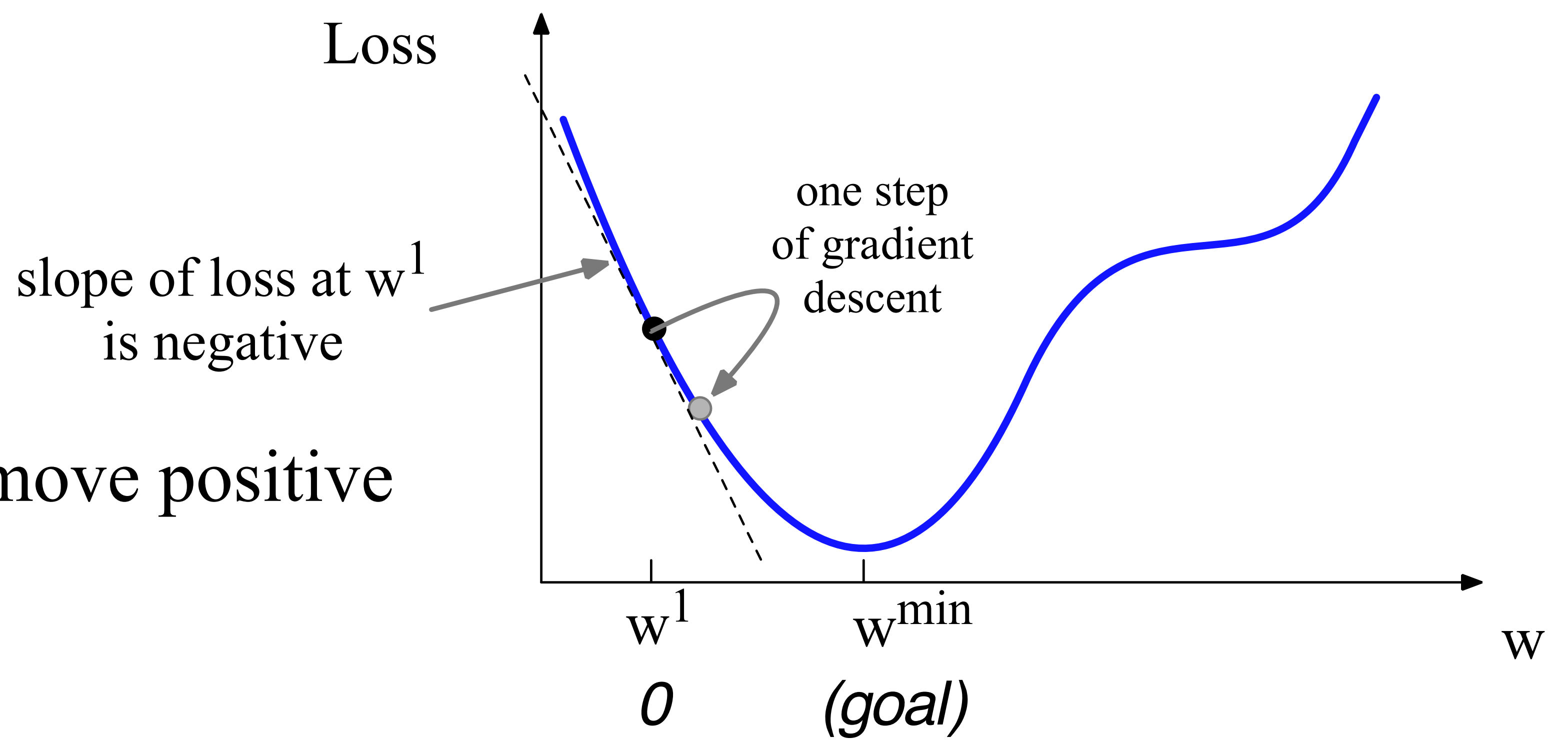


So we'll move positive

Let's first visualize for a single scalar w

Q: Given current w, should we make it bigger or smaller?

A: Move *w* in the reverse direction from the slope of the function



So we'll move positive

Gradients

The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function.

Gradient Descent: Find the gradient of the loss function at the current point and move in the **opposite** direction.

How much do we move in that direction ?

- The value of the gradient (slope in our example) $\frac{d}{dw}L(f(x;w),y)$ weighted by a **learning rate** η
- Higher learning rate means that we make bigger adjustments to the weights

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w),y)$$

Now let's consider N dimensions

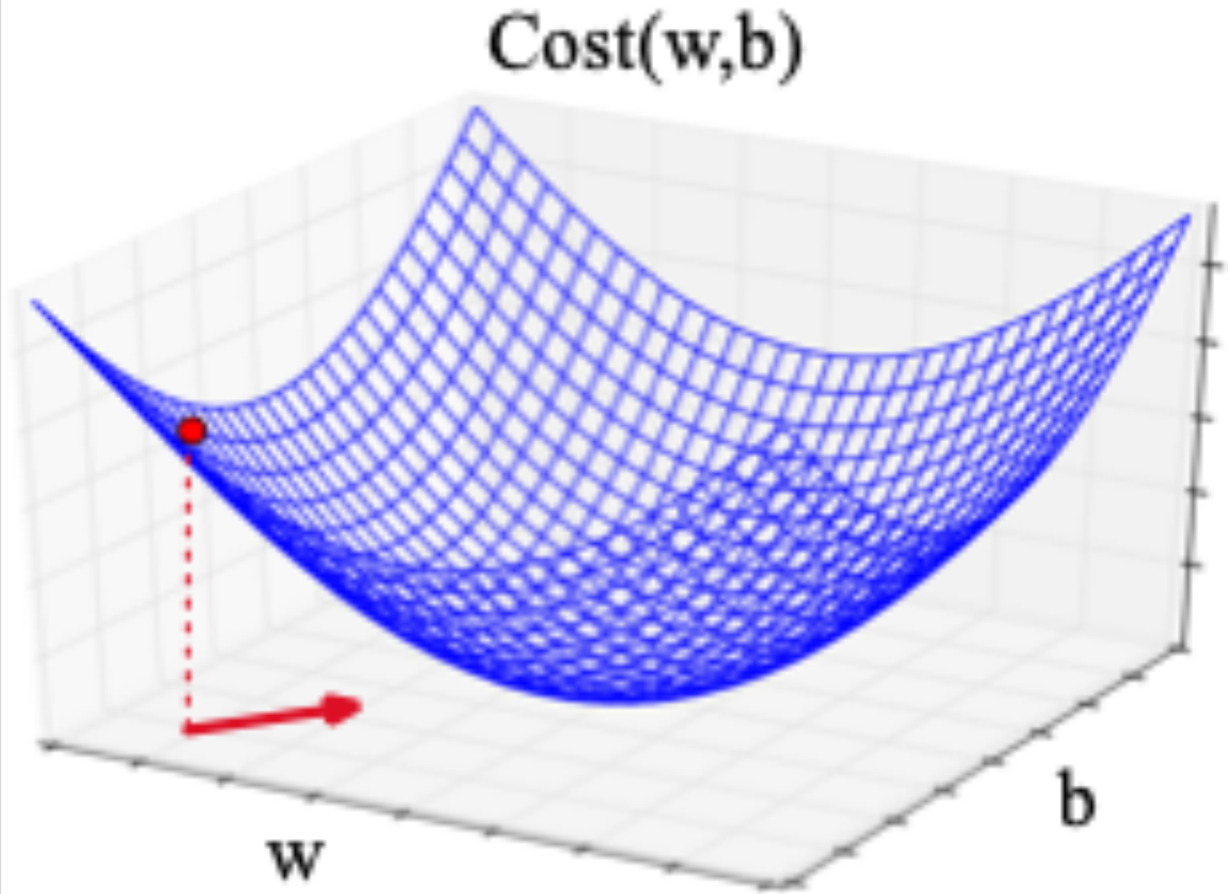
We want to know where in the N-dimensional space (of the *N* parameters that make up θ) we should move.

The gradient is just such a vector; it expresses the directional components of the sharpest slope along each of the *N* dimensions.

Imagine 2 dimensions, w and b

Visualizing the gradient vector at the red point

It has two dimensions shown in the x-y plane



function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

    # where: L is the loss function

    #   f is a function parameterized by θ

    #   x is the set of training inputs $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

    #   y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$

$\theta \leftarrow 0$

repeat til done   # see caption

  For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)

    1. Optional (for reporting):   # How are we doing on this tuple?

       Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output $\hat{y}$?

       Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?

    2. $g \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?

    3. $\theta \leftarrow \theta - \eta\, g$ # Go the other way instead

return θ

Hyperparameters

The learning rate η is a **hyperparameter**

- too high: the learner will take big steps and overshoot
- too low: the learner will take too long

Hyperparameters:

- Briefly, a special kind of parameter for an ML model
- Instead of being learned by algorithm from supervision (like regular parameters), they are chosen by algorithm designer.

How much do we move in that direction ?

- The value of the gradient (slope in our example) $\frac{d}{dw}L(f(x;w),y)$ weighted by a **learning rate** η
- Higher learning rate means that we make bigger adjustments to the weights

$$w^{t+1} = w^t - \eta \frac{d}{dw}L(f(x;w),y)$$

Partial Derivative for Logistic Regression

Cross-Entropy Loss $L_{\text{CE}}(\hat{y},y) = -[y\log\sigma(w\cdot x+b)+(1-y)\log(1-\sigma(w\cdot x+b))]$

Weight Update $w^{t+1} = w^t - \eta\frac{d}{dw}L(f(x;w),y)$

Chain Rule $\frac{df}{dx} = \frac{du}{dv}\cdot\frac{dv}{dx}$

Derivative of σ(x) $\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$

Derivative of ln(x) $\frac{d}{dx}\ln(x) = \frac{1}{x}$

Derivative of Loss $\frac{d}{dw}L(f(x;w),y) = \frac{\partial}{\partial \mathbf{w}_j} - [y\log\sigma(\mathbf{w}\cdot\mathbf{x}+b)+(1-y)\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))]$

Partial Derivative for Logistic Regression

Cross-Entropy Loss

$$L_{\text{CE}}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))]$$

Weight Update

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x; w), y)$$

Derivative of Cross-Entropy Loss

$$\frac{d}{dw} L(f(x; w), y) = [\sigma(w \cdot x + b) - y] x_j$$