기계학습 (2022년도 2학기)

Linear Classification I

전북대학교 컴퓨터공학부

Overview

- Classification: predicting a discrete-valued target
  - Binary classification: predicting a binary-valued target
- Examples
  - predict whether a patient has a disease, given the presence or absence of various symptoms
  - classify e-mails as spam or non-spam
  - predict whether a financial transaction is fraudulent

Overview

- **Binary linear classification**
  - **classification**: predict a discrete-valued target
  - **binary**: predict a binary target $t \in \{0,1\}$
    - Training examples with t = 1 are called positive examples, and training examples with t = 0 are called negative examples.
  - **linear**: model is a linear function of x, followed by a threshold

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

Some simplifications

Eliminating the threshold

- We can assume without loss of generality that the threshold $r = 0$:

$$\mathbf{w}^T\mathbf{x} + b \geq r \quad \Longleftrightarrow \quad \mathbf{w}^T\mathbf{x} + \underbrace{b - r}_{\triangleq b'} \geq 0$$

Eliminating the bias

- Add a dummy feature x_0 which always takes the value 1. The weight w_0 is equivalent to a bias (i.e. $w_0 \equiv b$) ($\mathbf{x}$에 x_0를 추가해서 차원을 확장)

Simplified model

$$z = \mathbf{w}^T\mathbf{x}$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Examples

- Let's consider some simple examples to examine the properties of our model
- Forget about generalization and suppose we just want to learn Boolean functions

Examples

- This is our "training set"

NOT

x_0	x_1	t
1	0	1
1	1	0

- What conditions are needed on w_0, w_1 to classify all examples?
  - When $x_1 = 0$, need: $w_0 x_0 + w_1 x_1 > 0 \iff w_0 > 0$
  - When $x_1 = 1$, need: $w_0 x_0 + w_1 x_1 < 0 \iff w_0 + w_1 < 0$
- Example solution: $w_0 = 1, w_1 = -2$
- Is this the only solution?

Examples

AND

x_0	x_1	x_2	t
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

need: $w_0 < 0$

need: $w_0 + w_2 < 0$

need: $w_0 + w_1 < 0$

need: $w_0 + w_1 + w_2 > 0$

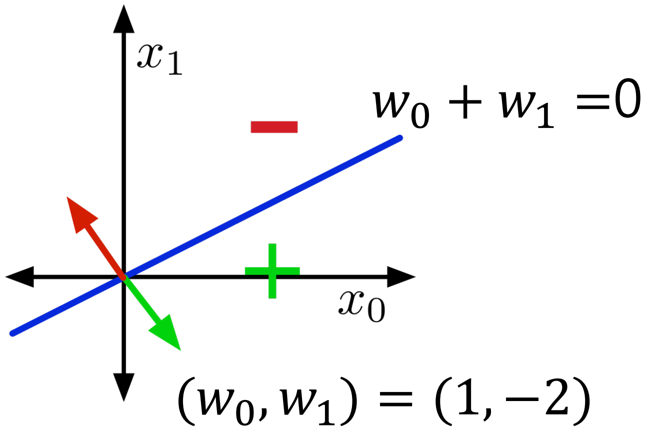
Example solution: $w_0 = -1.5,\ w_1 = 1,\ w_2 = 1$

The Geometric Picture

Input Space, or Data Space for NOT example

x_0	x_1	t
1	0	1
1	1	0

$$w_0 x_0 + w_1 x_1 > 0 \iff w_0 > 0$$
$$w_0 x_0 + w_1 x_1 < 0 \iff w_0 + w_1 < 0$$



- Training examples are points
- Hypotheses w can be represented by half-spaces

$$H_+ = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} \geq 0\},\ H_- = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} < 0\}$$

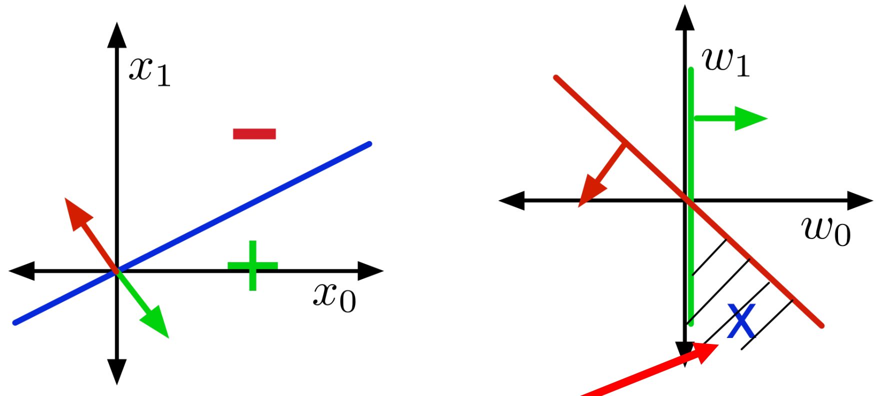
  - The boundaries of these half-spaces pass through the origin (why?)

왜 아래쪽 영역이 + 클래스인가?
→ 아래 영역의 임의의 점과 (w0,w1)의 내적을 구하면?

- The boundary is the decision boundary: $\{\mathbf{x} : \mathbf{w}^T\mathbf{x} = 0\}$
  - In 2-D, it's a line, but think of it as a hyperplane
- If the training examples can be separated by a linear decision rule, they are linearly separable.

The Geometric Picture

Weight Space



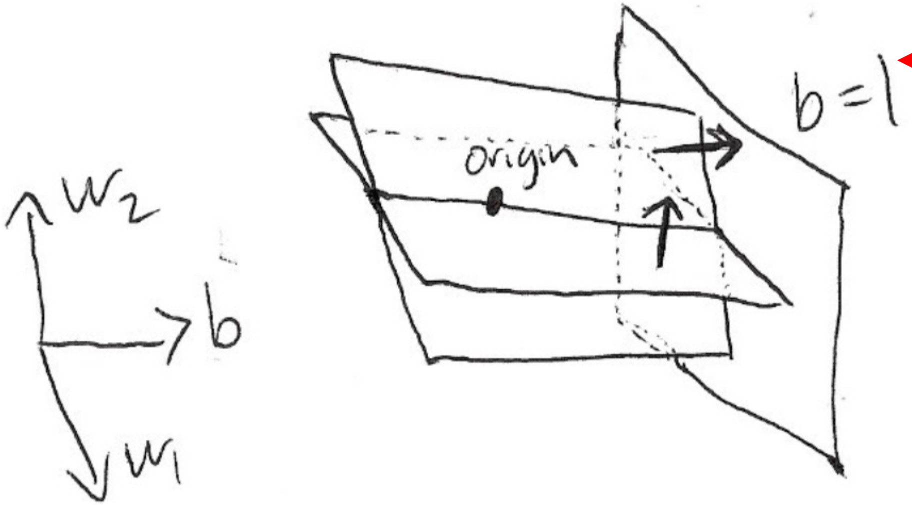
$$w_0 > 0$$

$$w_0 + w_1 < 0$$

- Hypotheses $\boldsymbol{w}$ are points
- Each training example x specifies a half-space $\boldsymbol{w}$ must lie in to be correctly classified
- For NOT example:
  - $x_0 = 1, x_1 = 0, t = 1 \implies (w_0, w_1) \in \{\mathbf{w} : w_0 > 0\}$
  - $x_0 = 1, x_1 = 1, t = 0 \implies (w_0, w_1) \in \{\mathbf{w} : w_0 + w_1 < 0\}$
- The region satisfying all the constraints is the feasible region; if this region is nonempty, the problem is feasible

The Geometric Picture

- The **AND** example requires three dimensions, including the dummy one.
- To visualize data space and weight space for a 3-D example, we can look at a 2-D slice:



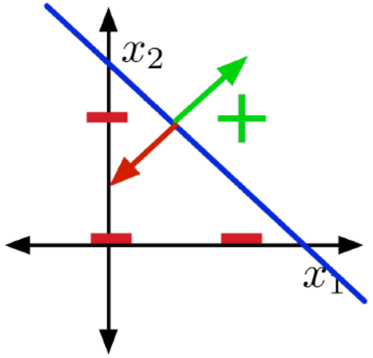
x_0	x_1	x_2	t	
1	0	0	0	need: $w_0 < 0$
1	0	1	0	need: $w_0 + w_2 < 0$
1	1	0	0	need: $w_0 + w_1 < 0$
1	1	1	1	need: $w_0 + w_1 + w_2 > 0$

- The visualizations are similar, except that the decision boundaries and the constraints need not pass through the origin.
  - The origin in our visualization may not have all coordinates set to 0!

The Geometric Picture

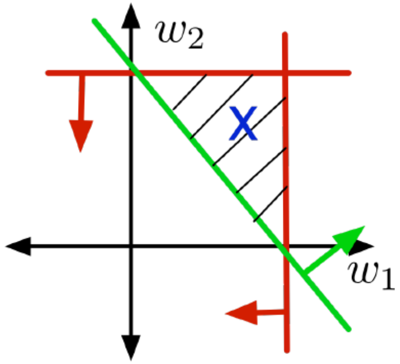
- Visualizations of the **AND** example

Data Space



Slice for $x_0 = 1$

Weight Space

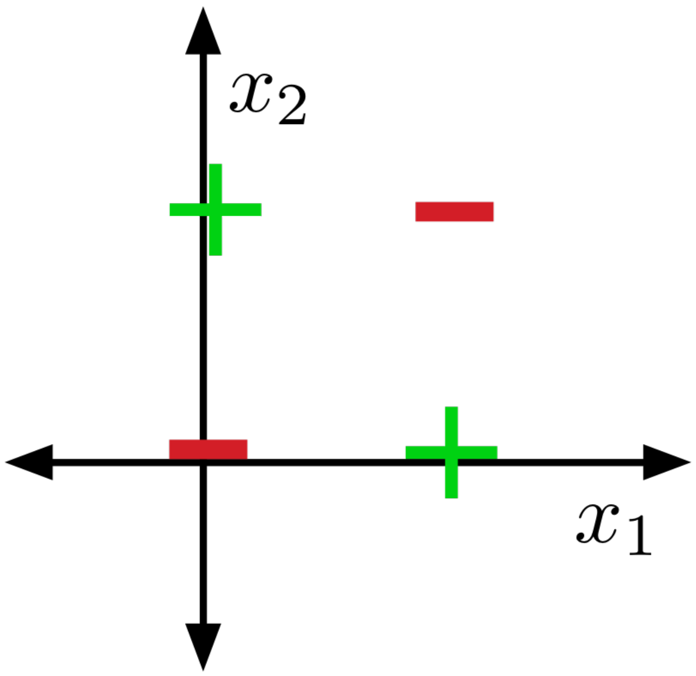


Slice for $w_0 = -1$

- Recall constraints:
  - $w_0 < 0$
  - $w_0 + w_2 < 0$
  - $w_0 + w_1 < 0$
  - $w_0 + w_1 + w_2 > 0$
- Why are only 3 constraints shown?

The Geometric Picture

- Some datasets are not linearly separable, e.g. **XOR**



Proof coming next lecture...

Overview

- **Recall: binary linear classifiers.** Targets $t \in \{0,1\}$

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

- How can we find good values for $\boldsymbol{w}, b$?
- If training set is separable, we can solve for $\boldsymbol{w}, b$ using linear programming
- If it's not separable, the problem is harder

Loss functions

- Instead: define loss function then try to minimize the resulting cost function
  - Recall: cost is loss averaged over the training set
- Seemingly obvious loss function: 0-1 loss

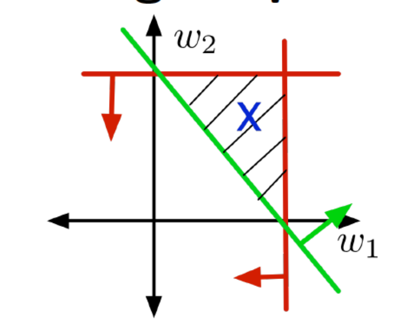
$$
\begin{aligned}
\mathcal{L}_{0-1}(y, t) &= \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases} \\
&= \mathbb{I}[y \neq t]
\end{aligned}
$$

Attempt 1: 0-1 loss

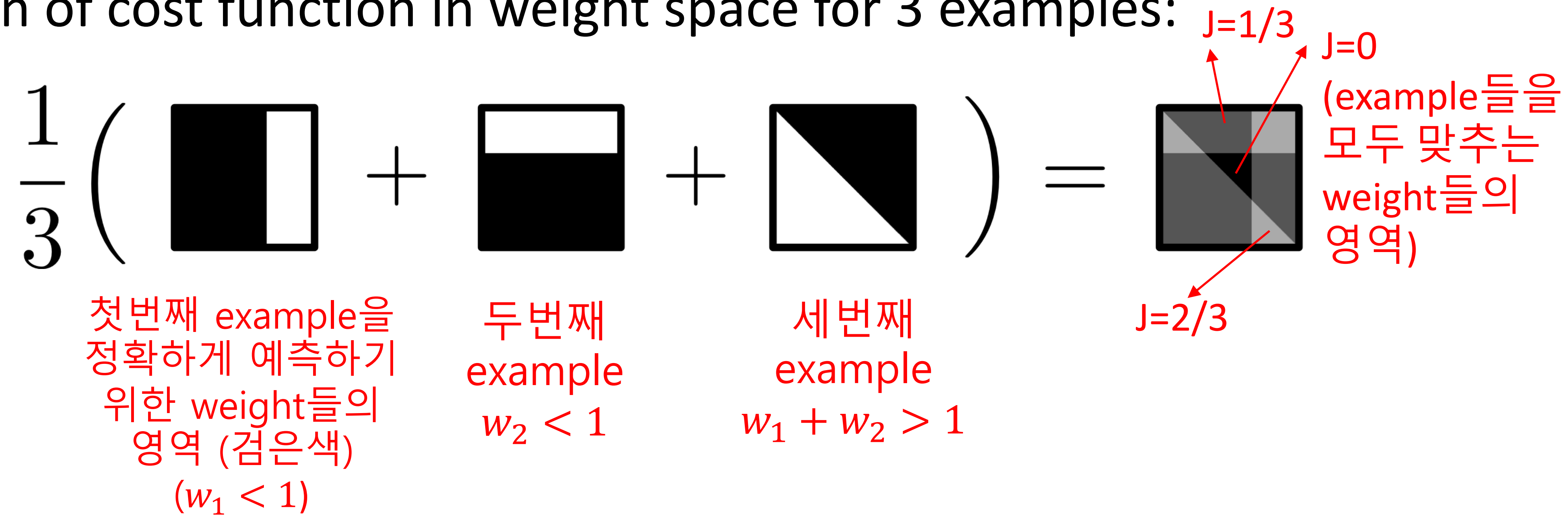
- As always, the cost J is the average loss over training examples; for 0-1 loss, this is the error rate:

$$\mathcal{J} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[y^{(i)} \neq t^{(i)}]$$

- Visualization of cost function in weight space for 3 examples:



Slice for $w_0 = -1$



Attempt 1: 0-1 loss

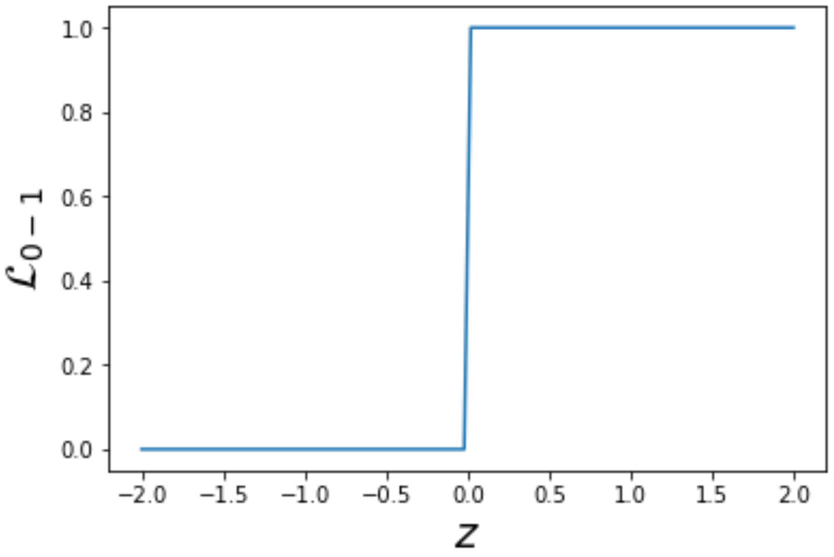
- Problem: how to optimize? In general, a hard problem
- (Guruswami and Raghavendra)
  "For arbitrary $\epsilon, \sigma > 0$, we prove that given a set of examples-label pairs from the hypercube a fraction $(1 - \epsilon)$ of which can be explained by a halfspace, it is NP-hard to find a halfspace that correctly labels a fraction $\left(\frac{1}{2} + \delta\right)$ of the examples."

Attempt 1: 0-1 loss

- Let's try the one optimization tool in our arsenal: gradient descent
- Chain rule:

$$\frac{\partial \mathcal{L}_{0-1}}{\partial w_j} = \frac{\partial \mathcal{L}_{0-1}}{\partial z} \frac{\partial z}{\partial w_j}$$

- But $\frac{\partial L_{0-1}}{\partial z}$ is zero everywhere it's defined!
  - $\frac{\partial L_{0-1}}{\partial z} = 0$ means that changing the weights by a very small amount probably has no effect on the loss.
  - The gradient descent update is a no-op.



Attempt 2: Linear Regression

- Sometimes we can replace the loss function we care about with one which is easier to optimize. This is known as a surrogate loss function.
  - 0-1 loss 는 문제를 가장 정확하게 설명하지만 SGD기반 optimization이 불가능하기 때문에 0-1 loss를 근사하는 연속함수를 고려
- One problem with L_{0-1}: defined in terms of final prediction, which inherently involves a discontinuity
- Instead, define loss in terms of $\boldsymbol{w}^T\boldsymbol{x} + b$ directly
  - Redo notation for convenience: $y = \boldsymbol{w}^T\boldsymbol{x} + b$

Attempt 2: Linear Regression

- We already know how to fit a linear regression model. Can we use this instead?
  - 0-1 loss의 surrogate loss로 squared error loss를 고려

$$y = \mathbf{w}^\top \mathbf{x} + b$$

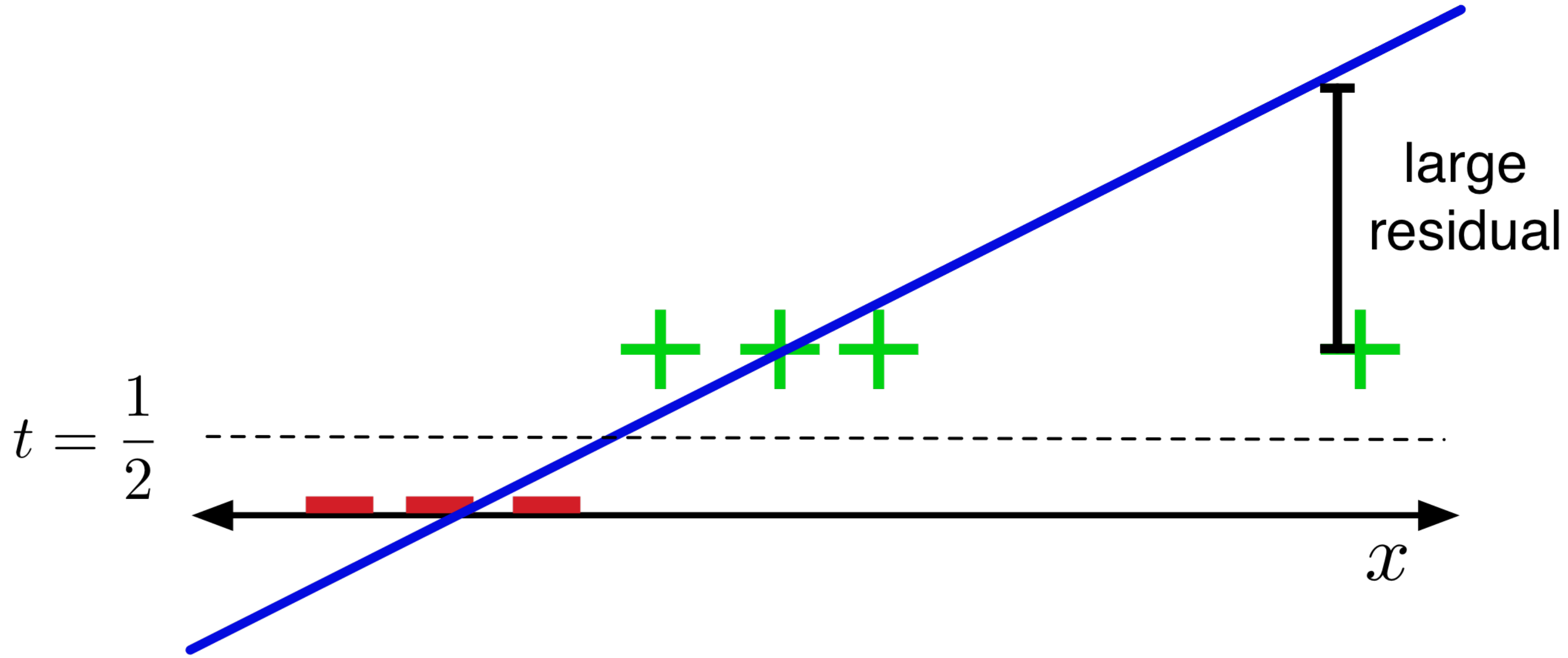
$$\mathcal{L}_{\mathrm{SE}}(y, t) = \frac{1}{2}(y - t)^2$$

- Doesn't matter that the targets are actually binary.
- For this loss function, it makes sense to make final predictions by thresholding y at $\frac{1}{2}$ (why?)

$\rightarrow$ $t \in \{0,1\}$

Attempt 2: Linear Regression

The problem:

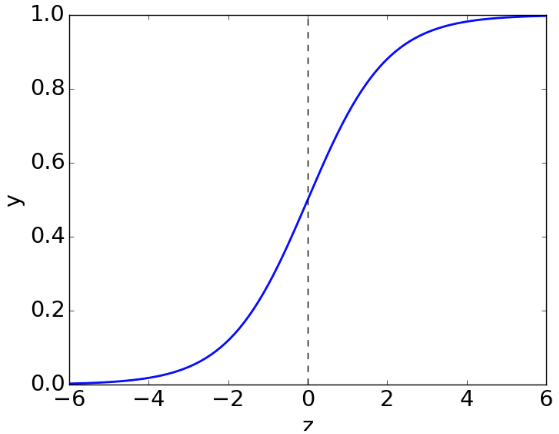


- The loss function hates when you make correct predictions with high confidence!
- If $t = 1$, it's more unhappy about $y = 10$ than $y = 0$. $\mathcal{L}_{\mathrm{SE}}(y, t) = \frac{1}{2}(y - t)^2$

Attempt 3: Logistic Activation Function

- There's obviously no reason to predict values outside [0, 1]. Let's squash y into this interval.
- The logistic function is a kind of sigmoidal, or S-shaped, function:

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



- A linear model with a logistic nonlinearity is known as log-linear:

$$z = \mathbf{w}^\top \mathbf{x} + b$$
$$y = \sigma(z)$$

$$\mathcal{L}_{\mathrm{SE}}(y, t) = \frac{1}{2}(y - t)^2$$

- Used in this way, σ is called an activation function, and z is called the logit.

Attempt 3: Logistic Activation Function

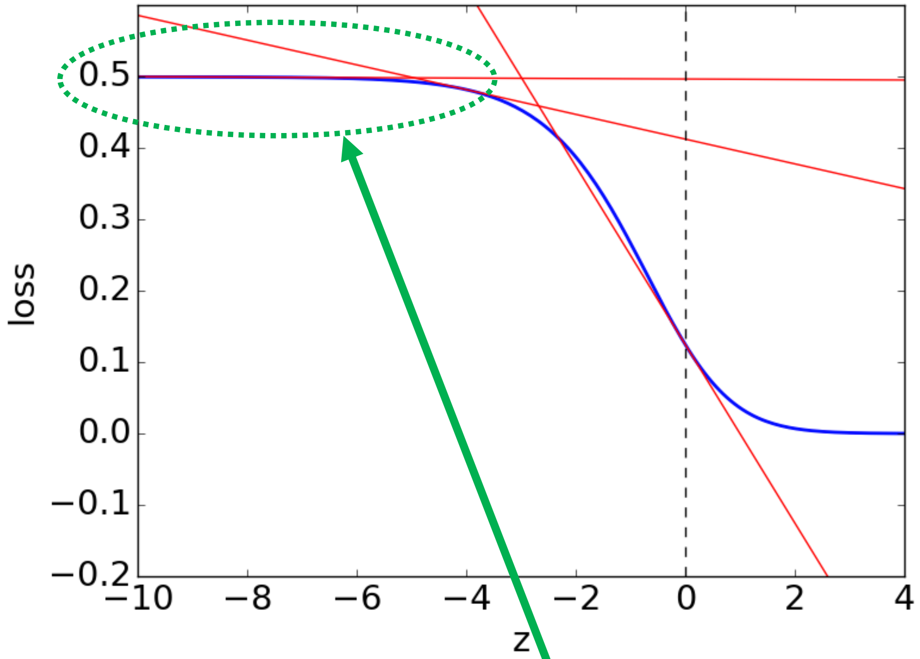
The problem:

(plot of L_{SE} as a function of z, assuming $t = 1$)

$$z = \mathbf{w}^\top \mathbf{x} + b$$

$$y = \sigma(z)$$

$$\mathcal{L}_{\mathrm{SE}}(y, t) = \frac{1}{2}(y - t)^2$$



$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial w_j}$$

$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{L}}{\partial w_j}$$

- For $z \ll 0$, $\frac{\partial \mathcal{L}}{\partial z} \approx 0$ (check!) $\implies \frac{\partial \mathcal{L}}{\partial w_j} \approx 0 \implies$ update to w_j is small
- If the prediction is really wrong, shouldn't you take a large step?

Logistic Regression

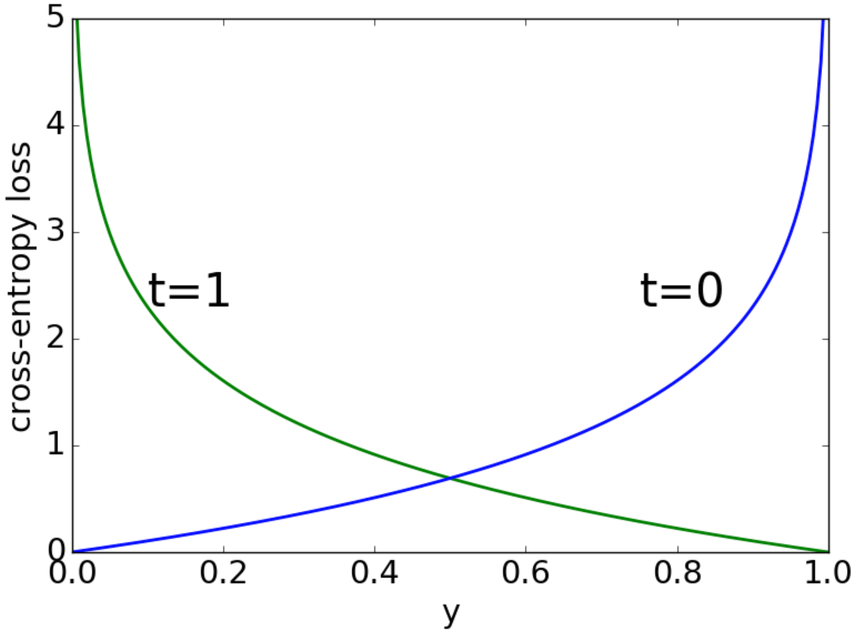
- Because $y \in [0, 1]$, we can interpret it as the estimated probability that $t = 1$.
- The pundits who were 99% confident Clinton would win were much more wrong than the ones who were only 90% confident.

  → 더 높은 confidence를 가진 예측에 대해 틀린 경우 더 많은 penalty를 부과
- Cross-entropy loss captures this intuition:

  (두 확률분포가 얼마나 유사한지를 측정하는 정도로 이해할 수 있음)

$$\mathcal{L}_{\mathrm{CE}}(y, t) = \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1-y) & \text{if } t = 0 \end{cases}$$

$$= -t \log y - (1-t)\log(1-y)$$

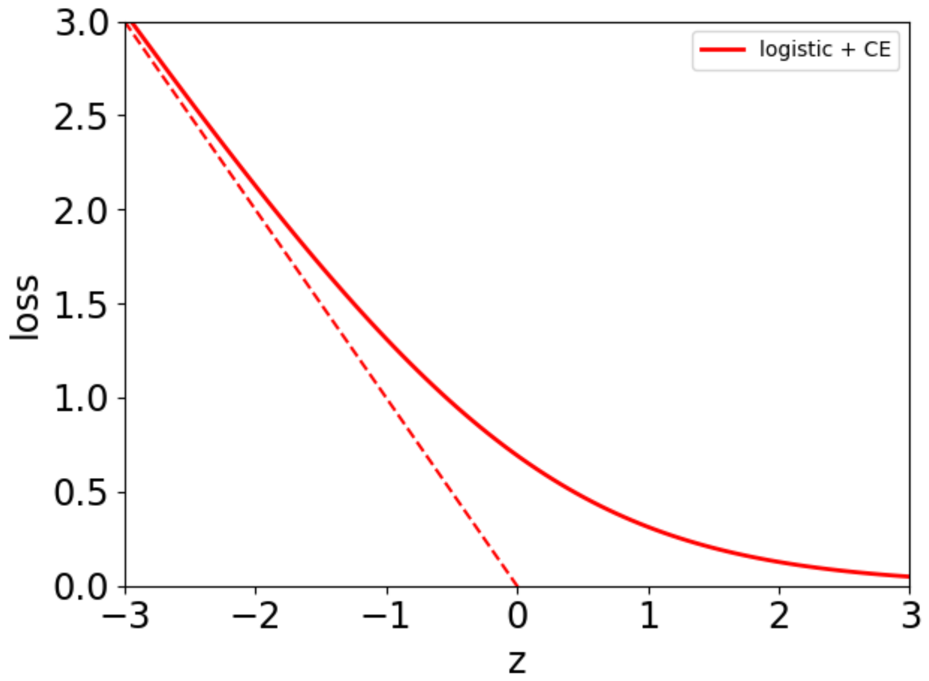


Logistic Regression

$$z = \mathbf{w}^\top \mathbf{x} + b$$

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

$$\mathcal{L}_{\mathrm{CE}} = -t \log y - (1 - t) \log(1 - y)$$



(L_{CE} 의 gradient 유도 과정은 여기의 pp. 5-7을 참고)

Logistic Regression

- Problem: what if $t = 1$ but you're really confident it's a negative example ($z \ll 0$)?
- If y is small enough, it may be numerically zero. This can cause very subtle and hard-to-find bugs.

$$y = \sigma(z) \qquad \Rightarrow y \approx 0$$

$$\mathcal{L}_{\text{CE}} = -t \log y - (1-t)\log(1-y) \qquad \Rightarrow \text{ computes } \log 0$$

- Instead, we combine the activation function and the loss into a single logistic-cross-entropy function.

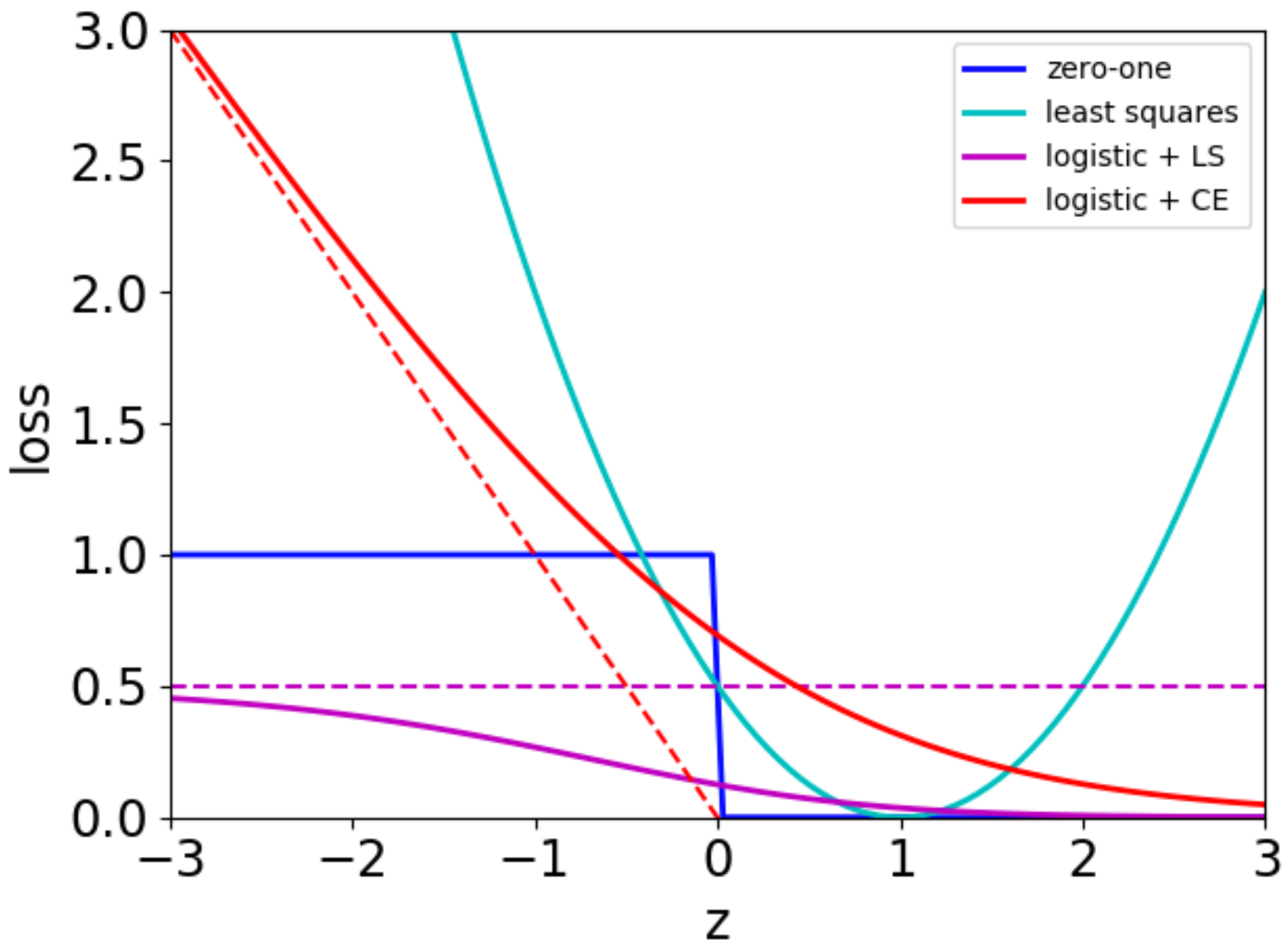
$$\mathcal{L}_{\text{LCE}}(z,t) = \mathcal{L}_{\text{CE}}(\sigma(z),t) = t\log(1+e^{-z}) + (1-t)\log(1+e^{z})$$

- Numerically stable computation:

```
E = t * np.logaddexp(0, -z) + (1-t) * np.logaddexp(0, z)
```

Logistic Regression

- Comparison of loss functions:



Logistic Regression

Comparison of gradient descent updates:

- Linear regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, \mathbf{x}^{(i)}$$

- Logistic regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, \mathbf{x}^{(i)}$$

- Not a coincidence! These are both examples of matching loss functions, but that's beyond the scope of this course.