기계학습 (2022년도 2학기)

Linear Classification II

전북대학교 컴퓨터공학부

Today's Agenda

- **Gradient checking with finite differences**
- Learning rates
- Stochastic gradient descent
- Convexity
- Multiclass classification and softmax regression
- Limits of linear classification

Gradient Checking

- We've derived a lot of gradients so far. How do we know if they're correct?
- Recall the definition of the partial derivative:

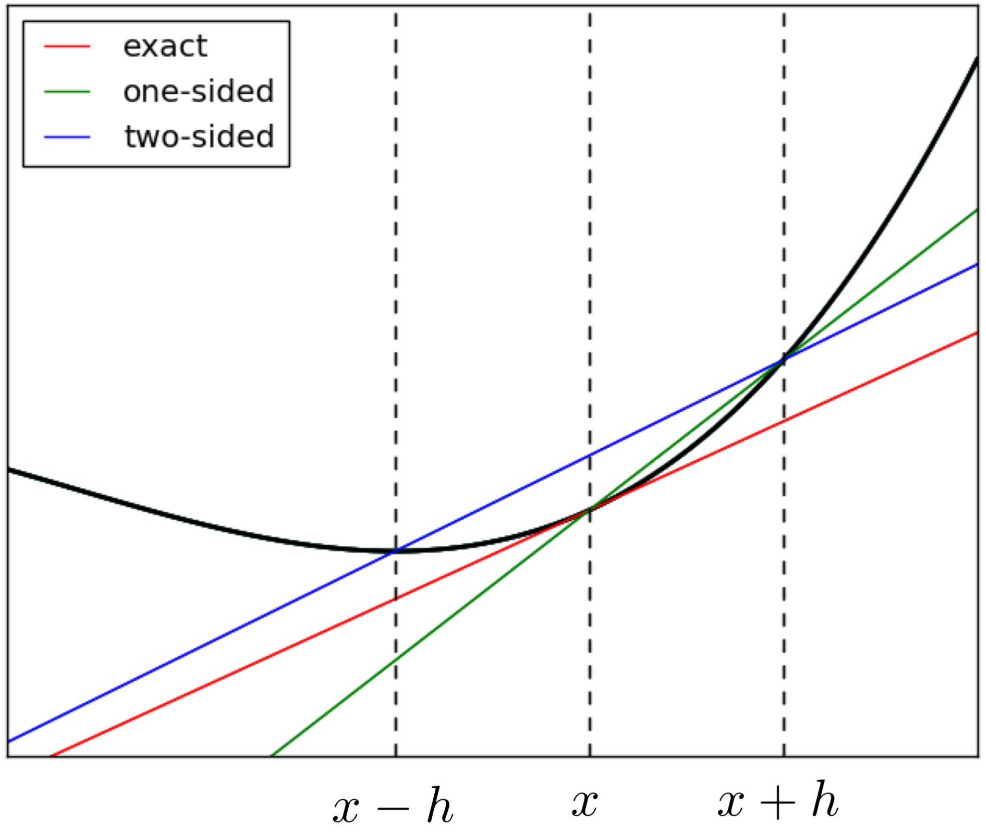
$$\frac{\partial}{\partial x_i} f(x_1, \ldots, x_N) = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_N) - f(x_1, \ldots, x_i, \ldots, x_N)}{h}$$

- Check your derivatives numerically by plugging in a small value of h, e.g. 10^{-10}. This is known as finite differences.

Gradient Checking

- Even better: the two-sided definition

$$\frac{\partial}{\partial x_i} f(x_1, \ldots, x_N) = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_N) - f(x_1, \ldots, x_i - h, \ldots, x_N)}{2h}$$



Gradient Checking

- Run gradient checks on small, randomly chosen inputs
- Use double precision floats (not the default for TensorFlow, PyTorch, etc.!)
- Compute the relative error between derived gradient and finite difference approximation:

$$\frac{|a - b|}{|a| + |b|}$$

- The relative error should be very small, e.g. 10^{-6}

Gradient Checking

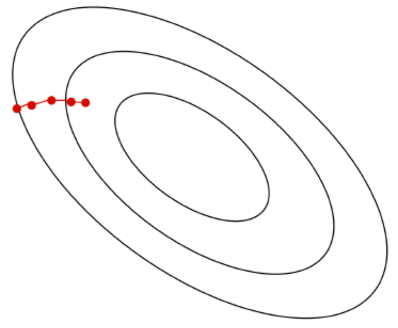
- Gradient checking is really important!
- Learning algorithms often appear to work even if the math is wrong.
- **But:**
  - They might work much better if the derivatives are correct.
  - Wrong derivatives might lead you on a wild goose chase.
- If you implement derivatives by hand, gradient checking is the single most important thing you need to do to get your algorithm to work well.

Today's Agenda

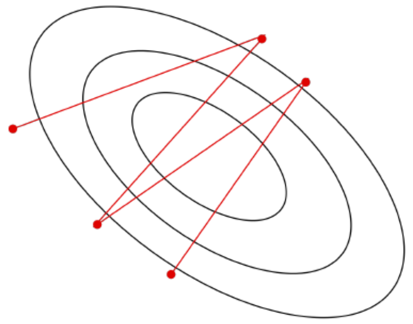
- Gradient checking with finite differences
- **Learning rates**
- Stochastic gradient descent
- Convexity
- Multiclass classification and softmax regression
- Limits of linear classification

Learning Rate

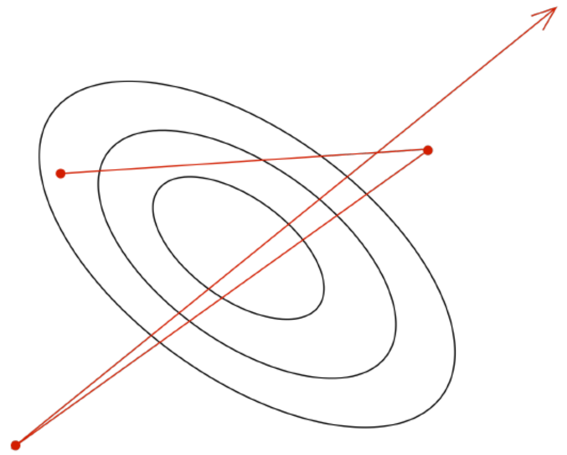
- In gradient descent, the learning rate α is a hyperparameter we need to tune. Here are some things that can go wrong:



α too small: slow progress



α too large: oscillations

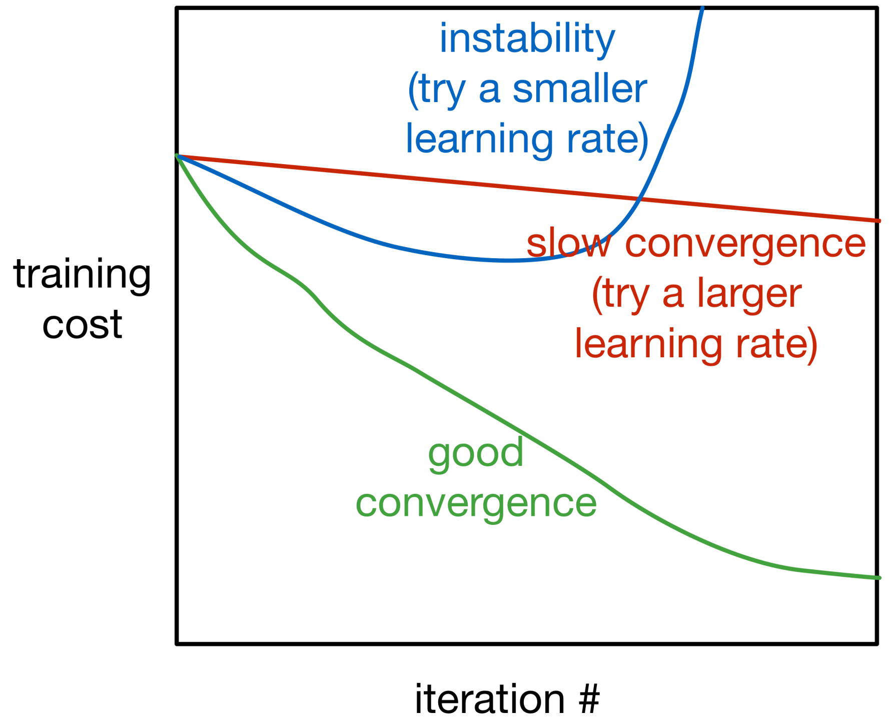


α much too large: instability

- Good values are typically between 0.001 and 0.1. You should do a grid search if you want good performance (i.e. try 0.1, 0.03, 0.01,…).

Training Curves

- To diagnose optimization problems, it's useful to look at training curves: plot the training cost as a function of iteration.



- Warning: it's very hard to tell from the training curves whether an optimizer has converged. They can reveal major problems, but they can't guarantee convergence.
  (일반적으로 Training curve와 validation curve의 모양이 다름.학습 수렴 여부는 보통 validation curve로 판단함)

Today's Agenda

- Gradient checking with finite differences
- Learning rates
- **Stochastic gradient descent**
- Convexity
- Multiclass classification and softmax regression
- Limits of linear classification

Stochastic Gradient Descent

- So far, the cost function *J* has been the average loss over the training examples:

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}^{(i)} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y(\mathbf{x}^{(i)},\boldsymbol{\theta}),t^{(i)})$$

- By linearity,

$$\frac{\partial\mathcal{J}}{\partial\boldsymbol{\theta}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial\mathcal{L}^{(i)}}{\partial\boldsymbol{\theta}}$$

- Computing the gradient requires summing over all of the training examples. This is known as batch training.

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

- Batch training is impractical if you have a large dataset (e.g. millions of training examples)!
  - Weight를 한번 update할때 마다 전체 training data의 loss를 계산해야 함!

Stochastic Gradient Descent

- Stochastic gradient descent (SGD): update the parameters based on the gradient for a single training example, chosen uniformly at random:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{\theta}}$$

- SGD can make significant progress before it has even looked at all the data!
- Mathematical justification: if you sample a training example uniformly at random, the stochastic gradient is an unbiased estimate of the batch gradient:

$$\mathbb{E}\left[\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{\theta}}\right] = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}}$$

- Problems:
  - Variance in this estimate may be high
  - If we only look at one training example at a time, we can't exploit efficient vectorized operations. (많은 기계학습 라이브러리들은 행렬과 벡터 연산에 최적화 되어 있음)

Stochastic Gradient Descent

- Compromise approach: compute the gradients on a randomly chosen medium-sized set of training examples $\mathcal{M} \subset \{1, \ldots, N\}$, called a mini-batch.
- Stochastic gradients computed on larger mini-batches have smaller variance:
  (Bagging ensemble 방법의 효과!)

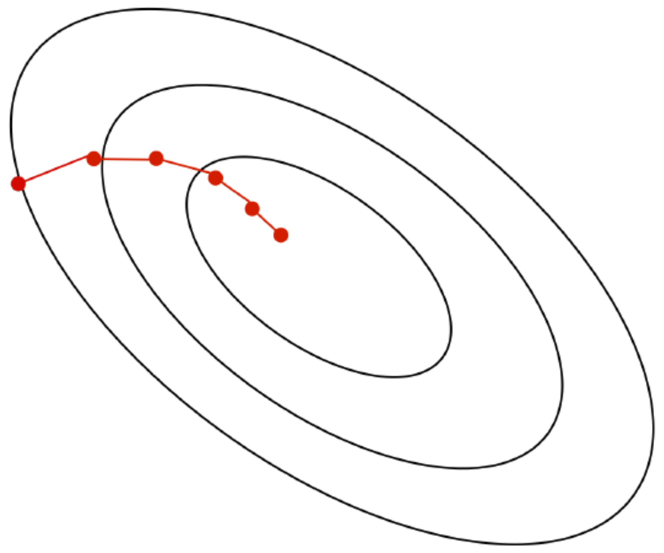
$$\mathrm{Var}\left[\frac{1}{|\mathcal{M}|}\sum_{i\in\mathcal{M}}\frac{\partial\mathcal{L}^{(i)}}{\partial\theta_j}\right] = \frac{1}{|\mathcal{M}|^2}\sum_{i\in\mathcal{M}}\mathrm{Var}\left[\frac{\partial\mathcal{L}^{(i)}}{\partial\theta_j}\right] = \frac{1}{|\mathcal{M}|}\mathrm{Var}\left[\frac{\partial\mathcal{L}^{(i)}}{\partial\theta_j}\right]$$

- The mini-batch size $|\mathcal{M}|$ is a hyperparameter that needs to be set.
  - Too large: takes more memory to store the activations, and longer to compute each gradient update
  - Too small: can't exploit vectorization
  - A reasonable value might be $|\mathcal{M}| = 100$

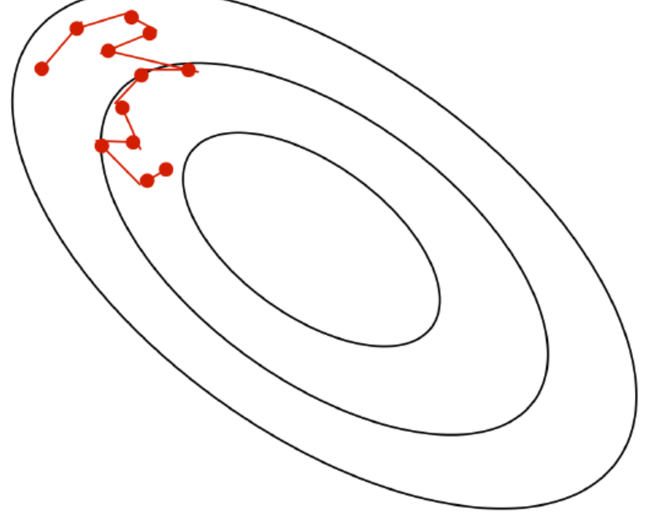
(모델의 크기, GPU 메모리 사이즈 등에 의해 결정되고,
deep learning에서는 일반적으로 2의 배수 크기의 batch size를 많이 사용)

Stochastic Gradient Descent

- Batch gradient descent moves directly downhill. SGD takes steps in a noisy direction, but moves downhill on average.



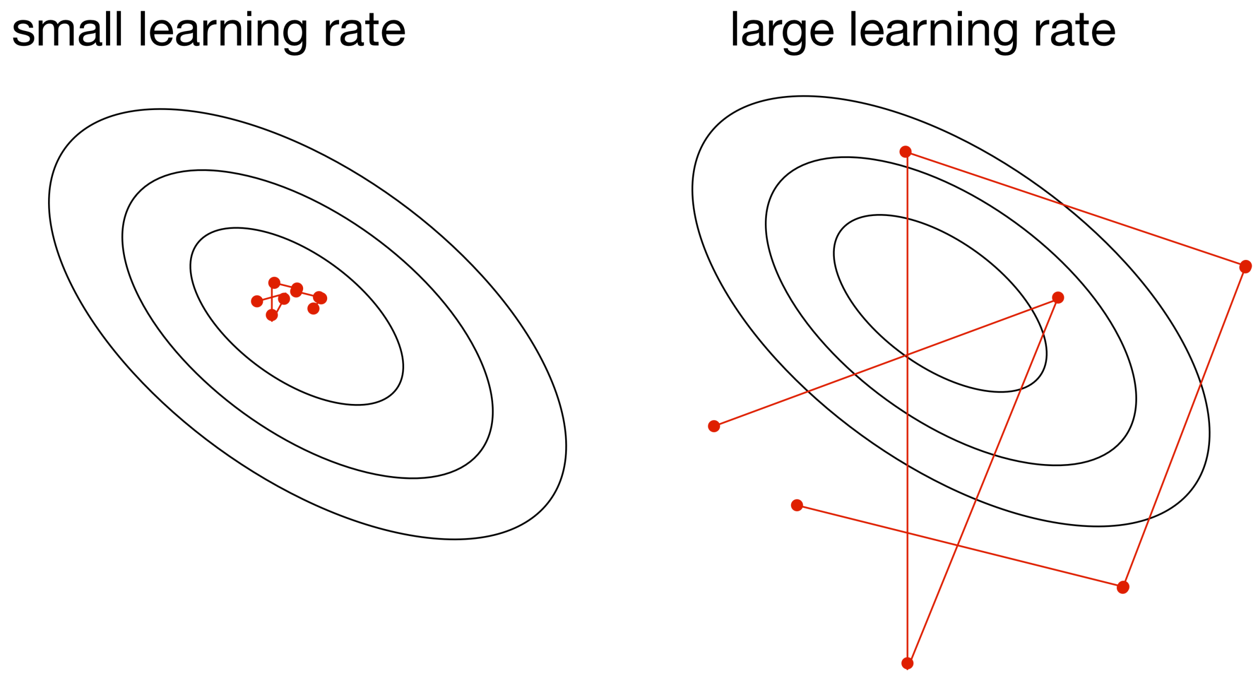
batch gradient descent



stochastic gradient descent

SGD Learning Rate

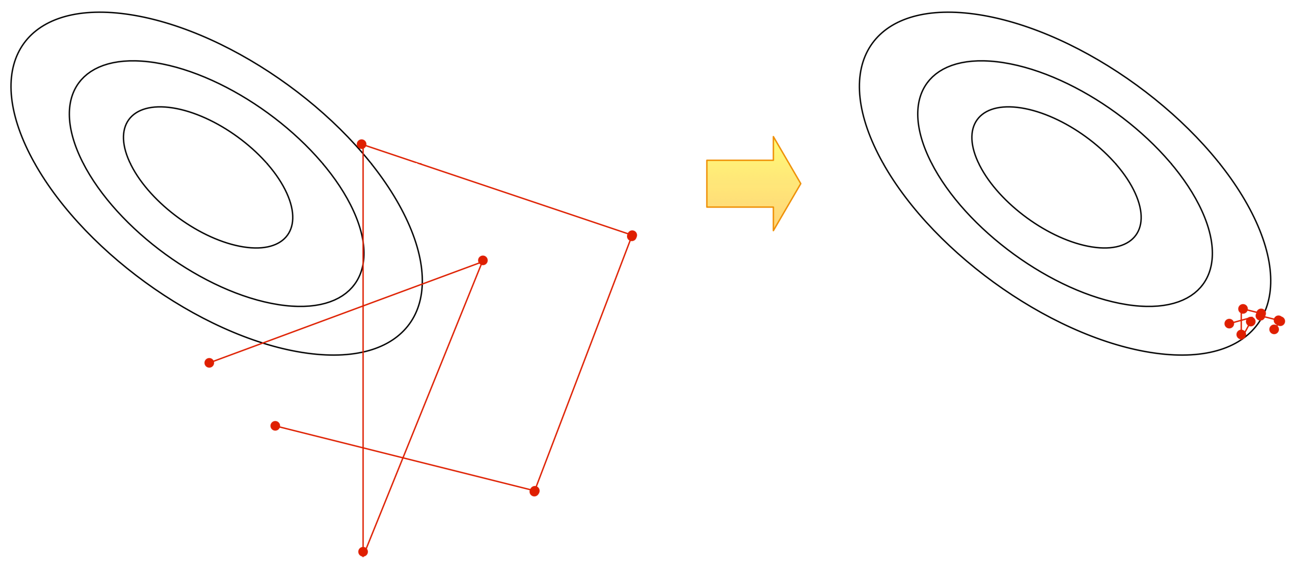
- In stochastic training, the learning rate also influences the fluctuations due to the stochasticity of the gradients.

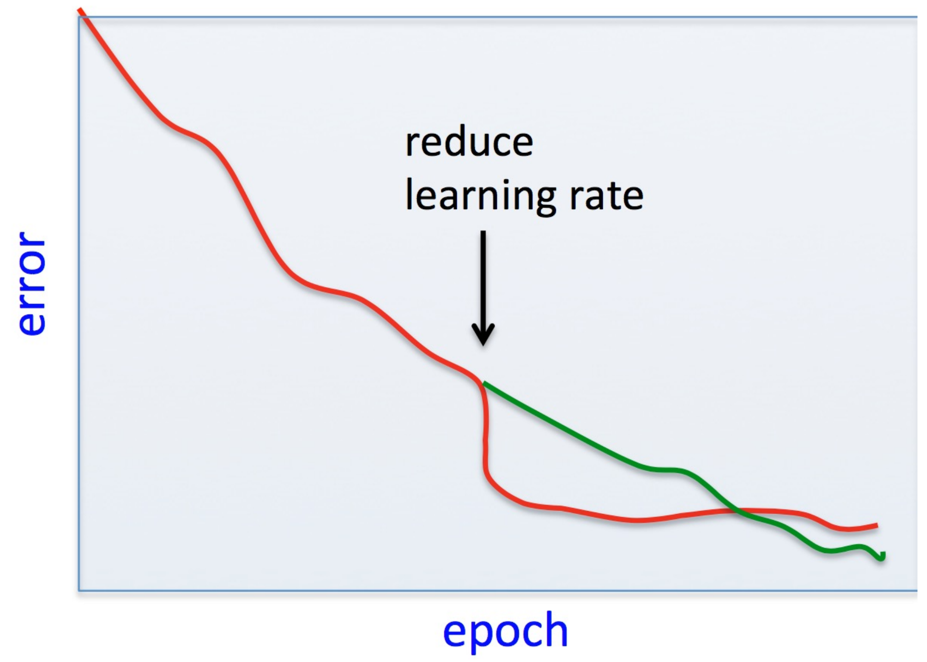


- Typical strategy:
  - Use a large learning rate early in training so you can get close to the optimum
  - Gradually decay the learning rate to reduce the fluctuations

SGD Learning Rate

- Warning: by reducing the learning rate, you reduce the fluctuations, which can appear to make the loss drop suddenly. But this can come at the expense of long-run performance. (학습 시간과 모델 성능의 trade-off)



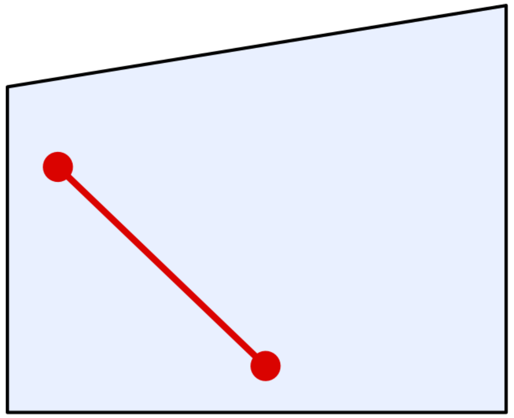


Today's Agenda

- Gradient checking with finite differences
- Learning rates
- Stochastic gradient descent
- **Convexity**
- Multiclass classification and softmax regression
- Limits of linear classification

Convex Sets

(기계학습의 loss (또는 cost) function들이 최적화를 통해 수렴하기 위해 가져야 할 특징)



- A set ***S*** is convex if any line segment connecting points in ***S*** lies entirely within ***S***. Mathematically,

$$\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S} \quad \Longrightarrow \quad \lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in \mathcal{S} \quad \text{for } 0 \leq \lambda \leq 1$$

- A simple inductive argument shows that for $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{S}$, weighted averages, or convex combinations, lie within the set:

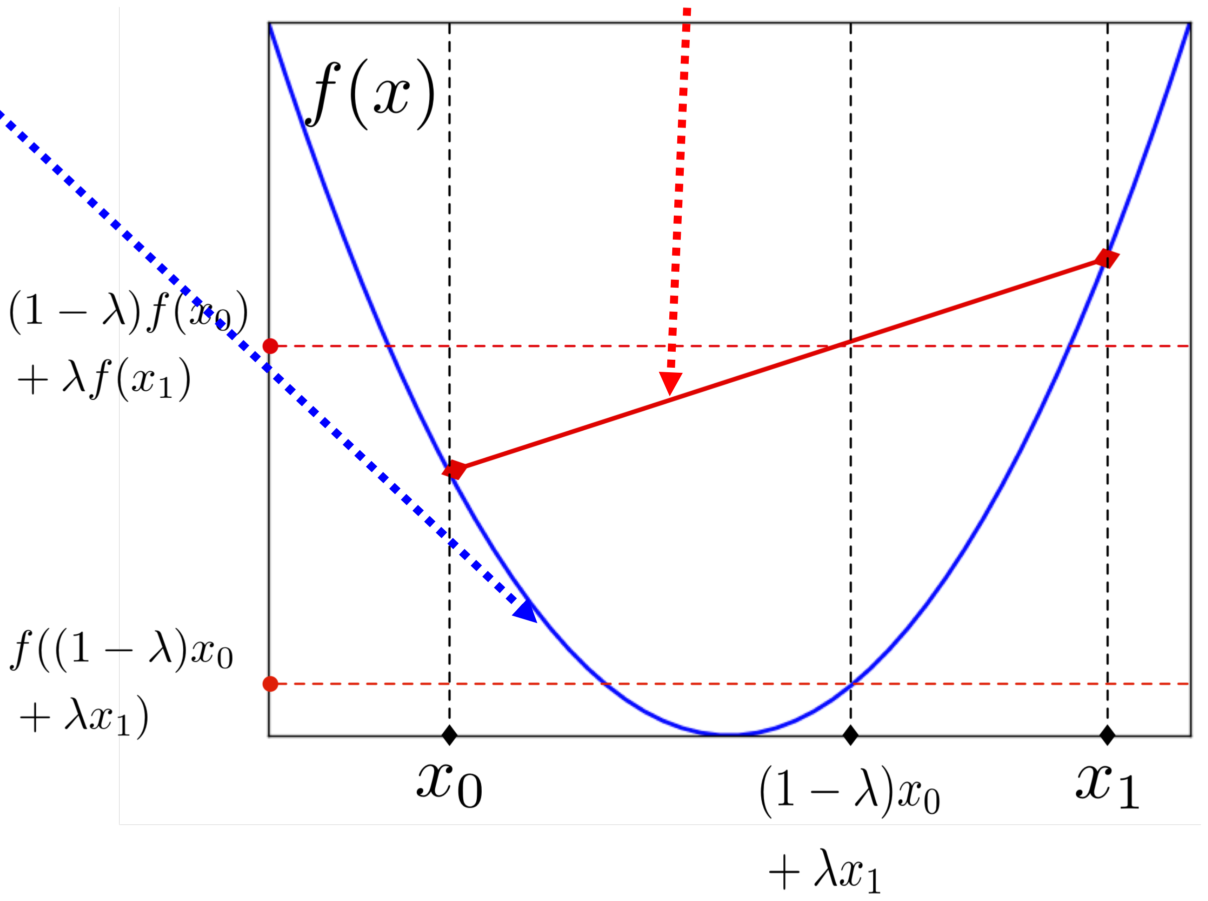
$$\lambda_1 \mathbf{x}_1 + \cdots + \lambda_N \mathbf{x}_N \in \mathcal{S} \quad \text{for } \lambda_i > 0,\ \lambda_1 + \cdots \lambda_N = 1$$

Convex Functions

- A function f is convex if for any $\mathbf{x}_0, \mathbf{x}_1$ in the domain of f,

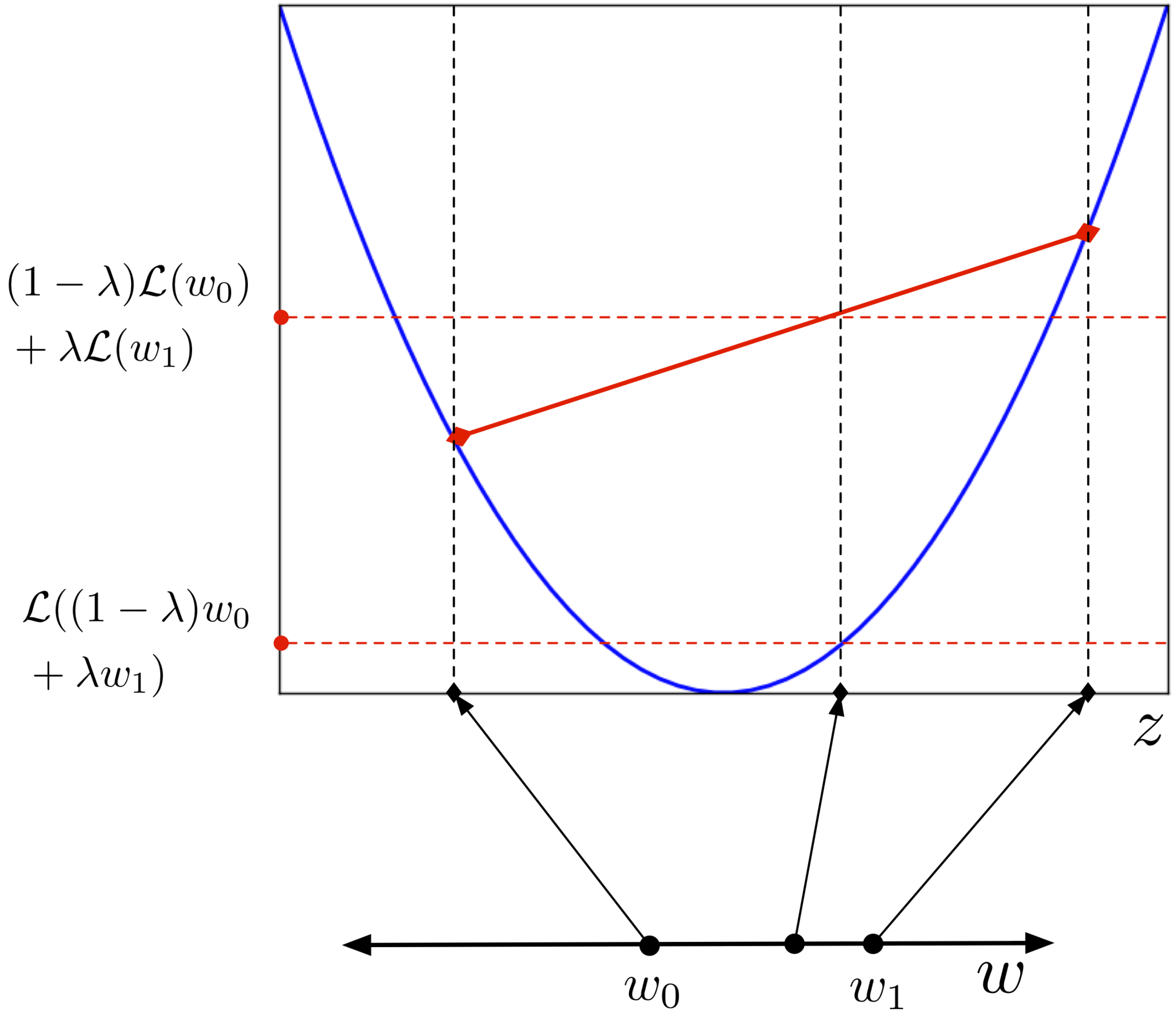
$$\boxed{f((1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}_1)} \leq \boxed{(1-\lambda)f(\mathbf{x}_0) + \lambda f(\mathbf{x}_1)}$$

- Equivalently, the set of points lying above the graph of f is convex.
- Intuitively: the function is bowl-shaped



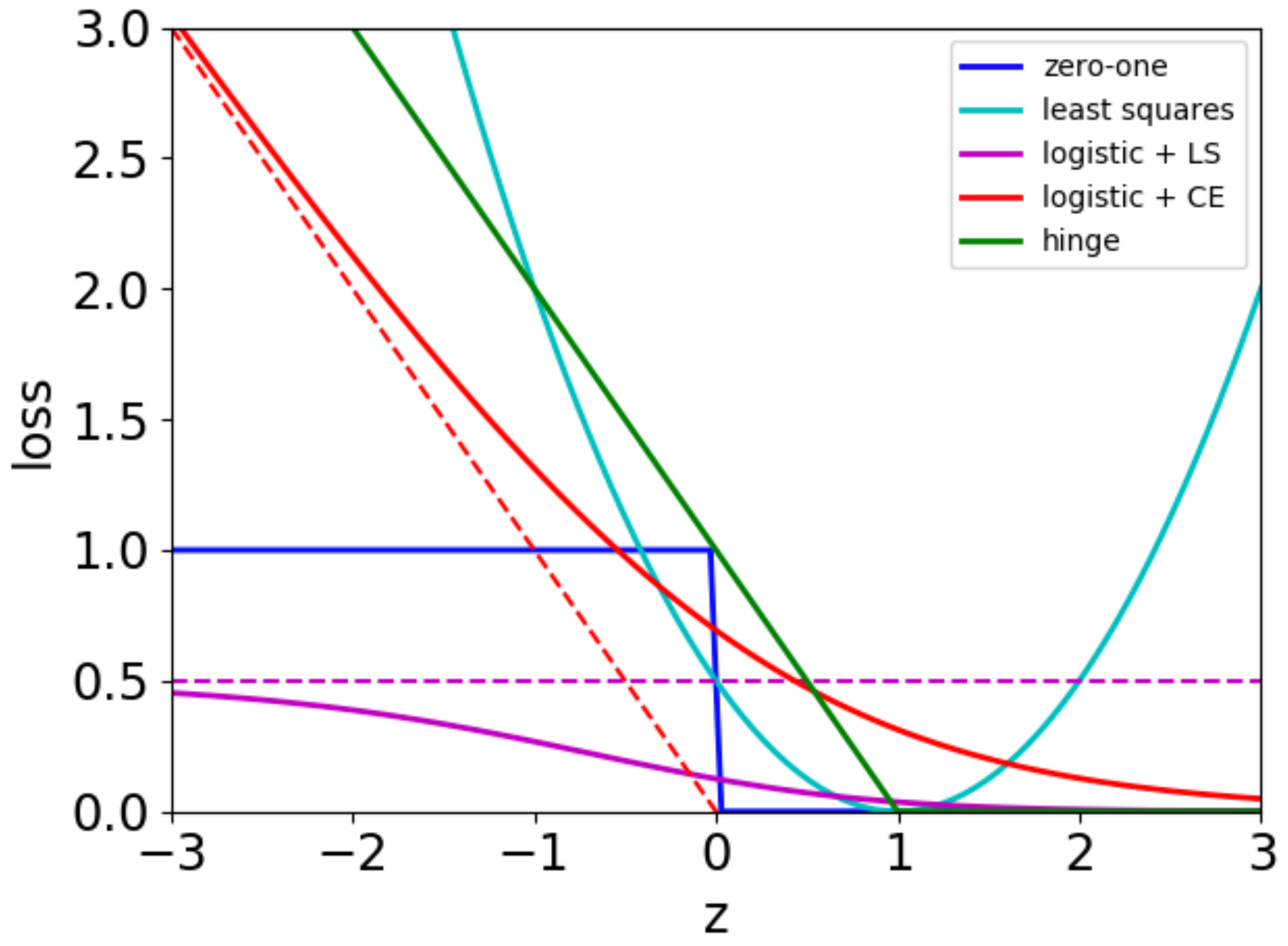
Convex Functions

- We just saw that the least-squares loss function $\frac{1}{2}(y-t)^2$ is convex as a function of y
- For a linear model, $z = \mathbf{w}^T\mathbf{x} + b$ is a linear function of $\mathbf{w}$ and b. If the loss function is convex as a function of z, then it is convex as a function of $\mathbf{w}$ and b.


$(1-\lambda)\mathcal{L}(w_0)$
$+\lambda\mathcal{L}(w_1)$
$\mathcal{L}((1-\lambda)w_0$
$+\lambda w_1)$
z
w_0
w_1
w

Convex Functions

- Which loss functions are convex?



Convex Functions

Why we care about convexity

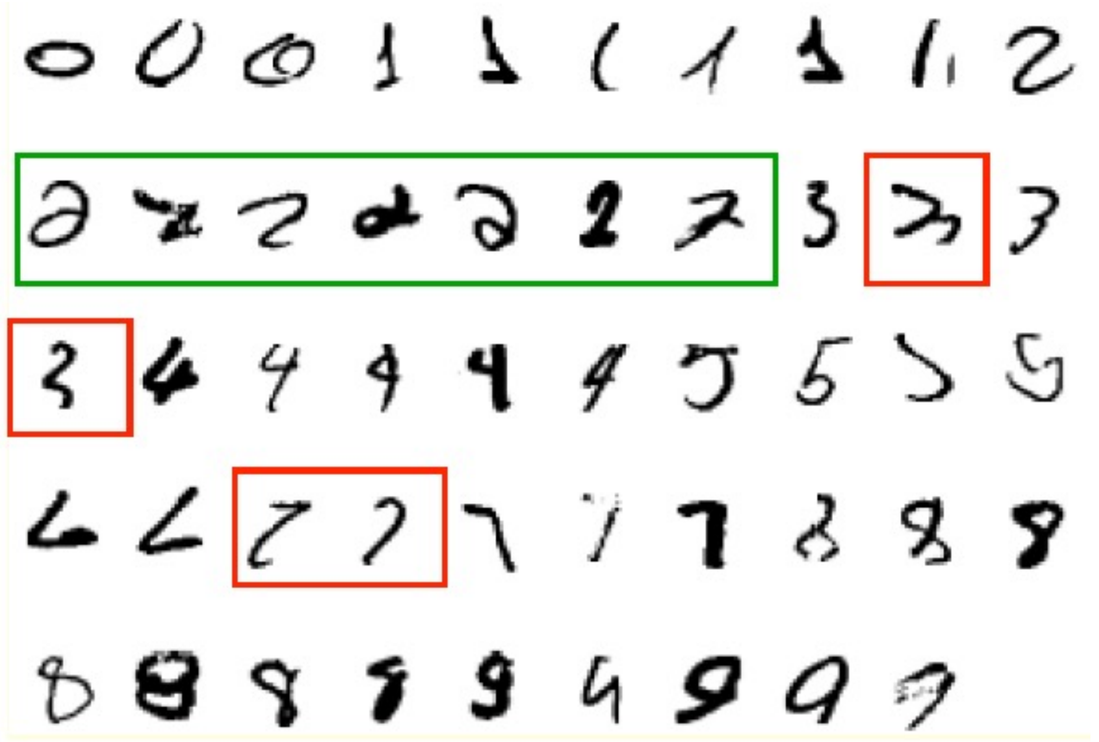
- All critical points are minima
- Gradient descent finds the optimal solution (more on this in a later lecture)

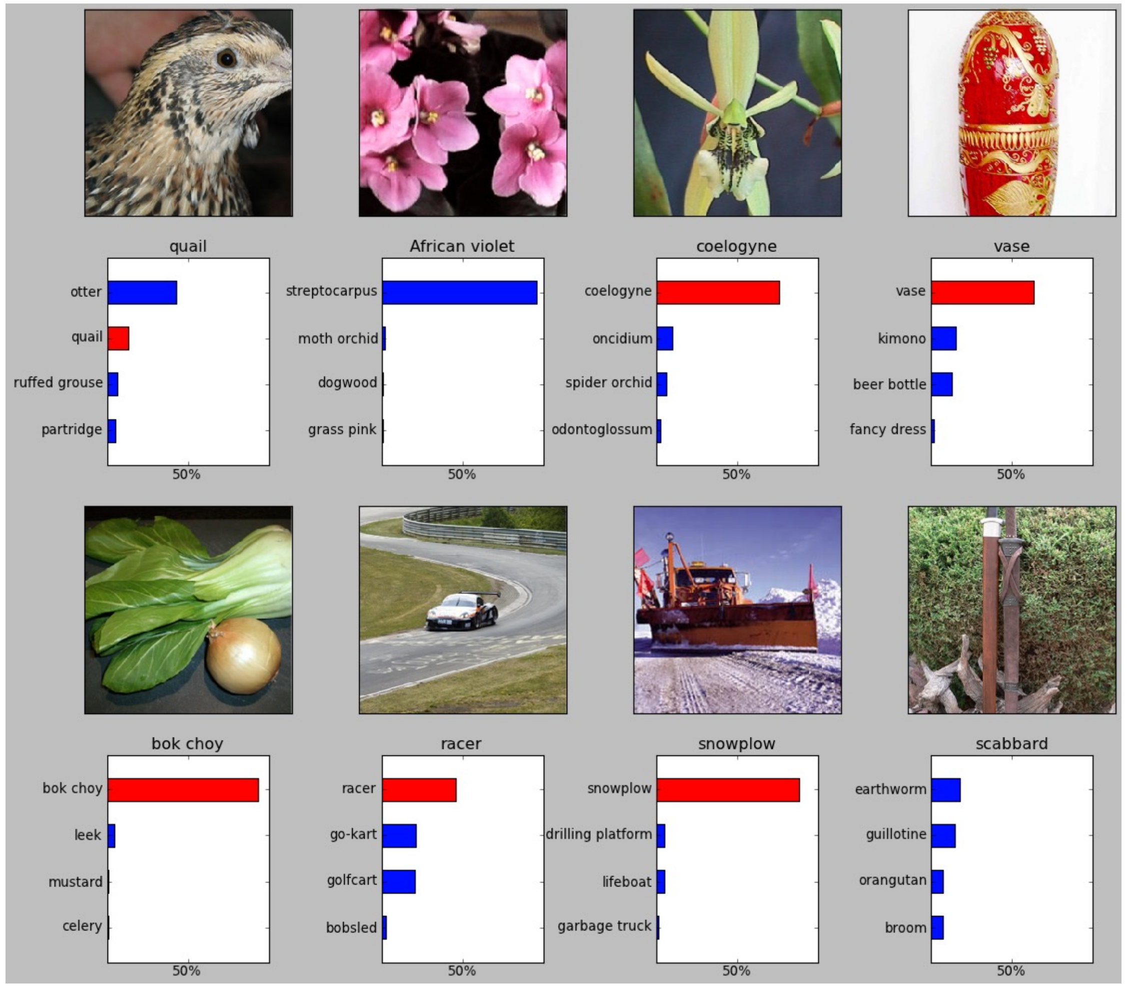
Today's Agenda

- Gradient checking with finite differences
- Learning rates
- Stochastic gradient descent
- Convexity
- **Multiclass classification and softmax regression**
- Limits of linear classification

Multiclass Classification

- What about classification tasks with more than two categories?





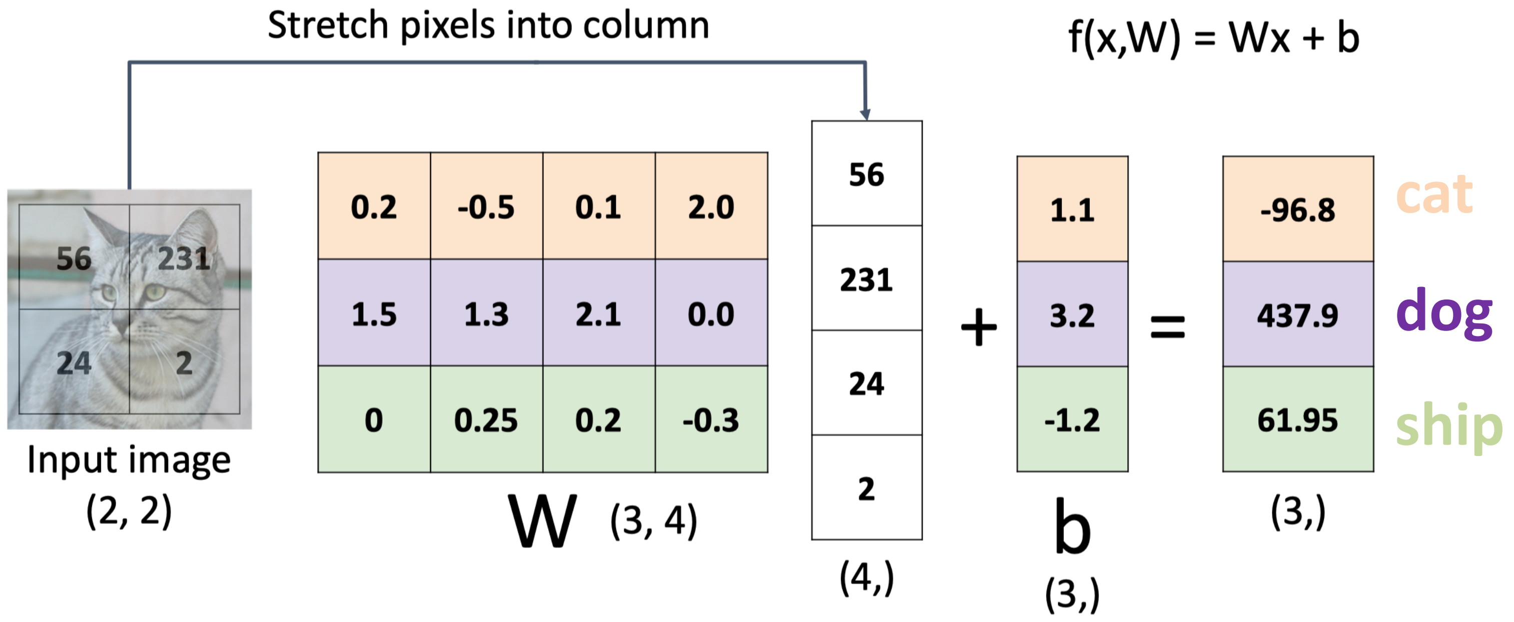
Multiclass Classification

- Targets form a discrete set $\{1, \ldots, K\}$.
- It's often more convenient to represent them as one-hot vectors, or a one-of-K encoding:

$$\mathbf{t} = \underbrace{(0, \ldots, 0, 1, 0, \ldots, 0)}_{\text{entry } k \text{ is } 1}$$

Multiclass Classification

- Now there are D input dimensions and K output dimensions, so we need $K \times D$ weights, which we arrange as a weight matrix $\boldsymbol{w}$.
- Also, we have a K-dimensional vector $\boldsymbol{b}$ of biases.
- Linear predictions: $$z_k = \sum_j w_{kj} x_j + b_k \qquad \mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b} \text{ (Vectorized)}$$
- Examples for 2x2 images, 3 classes (cat/dog/ship) (즉, D=2x2=4, K=3)



$$\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}$$

Multiclass Classification

- A natural activation function to use is the softmax function, a multivariable generalization of the logistic function:

$$y_k = \mathrm{softmax}(z_1, \ldots, z_K)_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$$

- The inputs z_k are called the logits.
- Properties:
  - Outputs are positive and sum to 1 (so they can be interpreted as probabilities)
  - If one of the z_k's is much larger than the others, $\mathrm{softmax}(\boldsymbol{z})$ is approximately the argmax. (So really it's more like “soft-argmax".)
  - **Exercise**: how does the case of $K = 2$ relate to the logistic function?

  (logistic function을 K=2인 softmax function으로 나타낼 수 있음)
- Note: sometimes $\sigma(\boldsymbol{z})$ is used to denote the softmax function; in this class, it will denote the logistic function applied elementwise.

Multiclass Classification

- If a model outputs a vector of class probabilities, we can use cross-entropy as the loss function:

$$\begin{aligned}\mathcal{L}_{\mathrm{CE}}(\mathbf{y}, \mathbf{t}) &= -\sum_{k=1}^{K} t_k \log y_k \\ &= -\mathbf{t}^\top (\log \mathbf{y}),\end{aligned}$$

  where the log is applied elementwise. (**t**는 one-hot vector)

- Just like with logistic regression, we typically combine the softmax and cross-entropy into a softmax-cross-entropy function.

Multiclass Classification

- Softmax regression:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$
$$\mathbf{y} = \mathrm{softmax}(\mathbf{z})$$
$$\mathcal{L}_{\mathrm{CE}} = -\mathbf{t}^{\top}(\log \mathbf{y})$$

- Gradient descent updates are derived in the readings:
(여기의 식(18) 참고, 반드시 직접 풀어볼 것!)

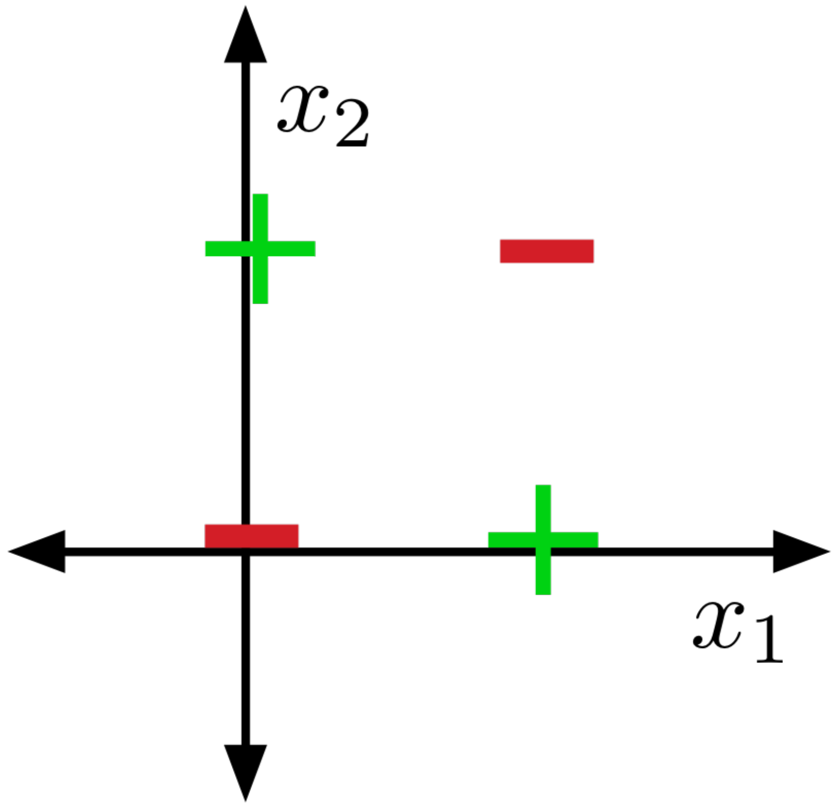
$$\frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial \mathbf{z}} = \mathbf{y} - \mathbf{t}$$

Today's Agenda

- Gradient checking with finite differences
- Learning rates
- Stochastic gradient descent
- Convexity
- Multiclass classification and softmax regression
- **Limits of linear classification**

Limits of Linear Classification

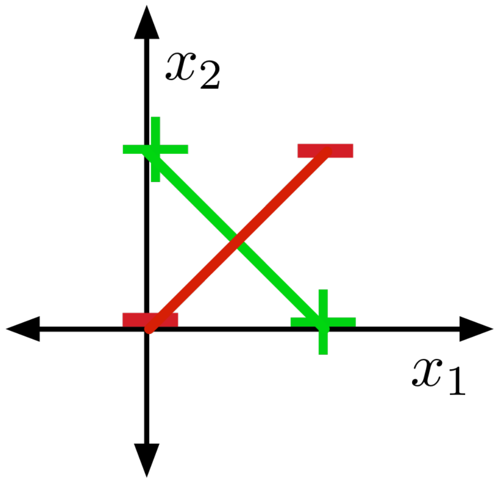
- Visually, it's obvious that **XOR** is not linearly separable. But how to show this?



Limits of Linear Classification

Showing that XOR is not linearly separable

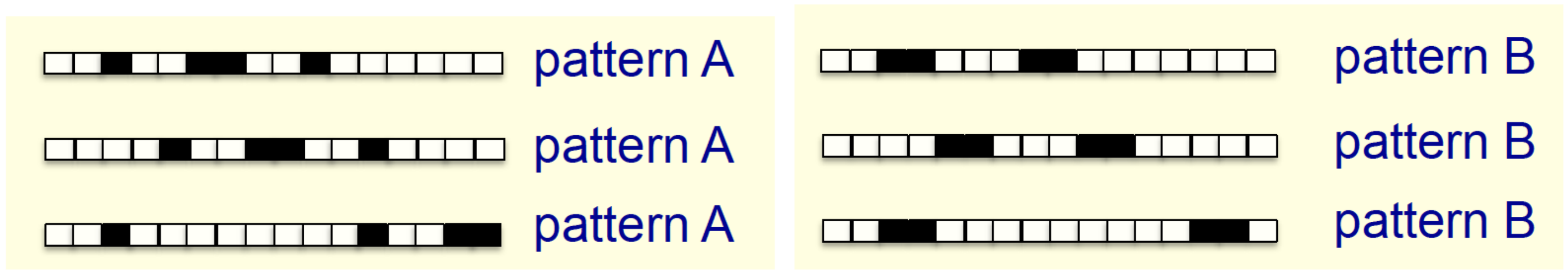
- Half-spaces are obviously convex.(직선으로 분리되는 half-space는 convex)
- Suppose there were some feasible hypothesis. If the positive examples are in the positive half-space, then the green line segment must be as well.
- Similarly, the red line segment must line within the negative half-space.



- But the intersection can't lie in both half-spaces. Contradiction!

Limits of Linear Classification

A more troubling example



- These images represent 16-dimensional vectors. White = 0, black = 1.
- Want to distinguish patterns A and B in all possible translations (with wrap-around)
- Translation invariance is commonly desired in vision!
- Suppose there's a feasible solution. The average of all translations of A is the vector (0.25, 0.25,..., 0.25). Therefore, this point must be classified as A.

$$\lambda_1 \mathbf{x}_1 + \cdots + \lambda_N \mathbf{x}_N \in \mathcal{S} \quad \text{for } \lambda_i > 0,\ \lambda_1 + \cdots \lambda_N = 1$$

- Similarly, the average of all translations of B is also (0.25, 0.25,..., 0.25). Therefore, it must be classified as B. Contradiction!

Limits of Linear Classification

- Sometimes we can overcome this limitation using feature maps, just like for linear regression. E.g., for **XOR**:

$$\psi(\mathbf{x}) = \begin{pmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{pmatrix}$$

x_1	x_2	$\psi_1(\mathbf{x})$	$\psi_2(\mathbf{x})$	$\psi_3(\mathbf{x})$	t
0	0	0	0	0	0
0	1	0	1	0	1
1	0	1	0	0	1
1	1	1	1	1	0

- This is linearly separable. (반드시 직접 해볼 것!)
- Not a general solution: it can be hard to pick good basis functions. Instead, we'll use neural nets to learn nonlinear hypotheses directly.