CSC 411 Lecture 8: Linear Classification II

Roger Grosse, Amir-massoud Farahmand, and Juan Carrasquilla

University of Toronto

Today's Agenda

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:

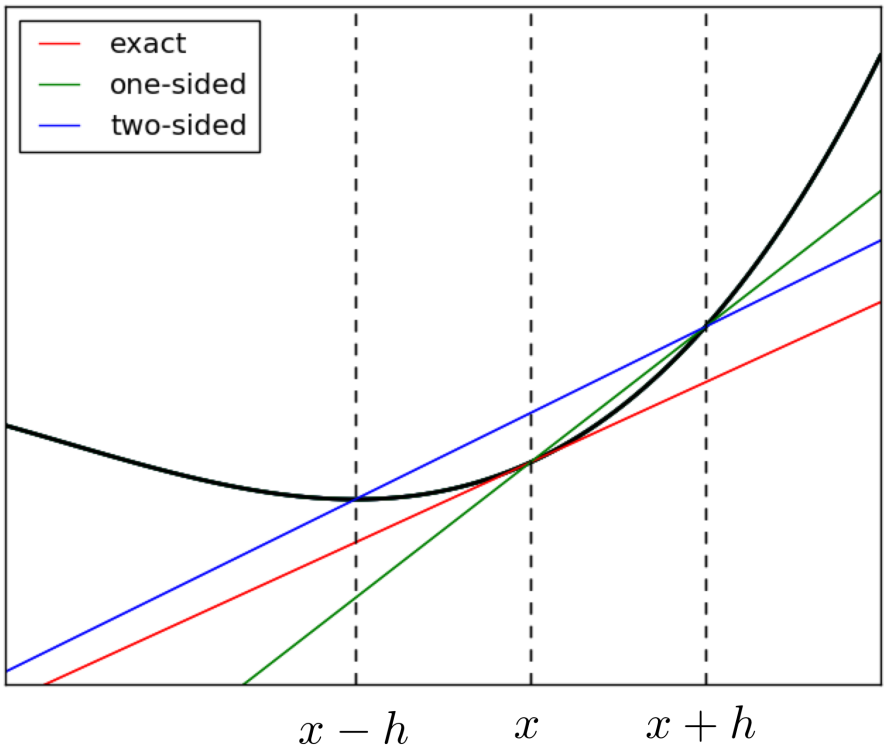
$$\frac{\partial}{\partial x_i} f(x_1, \ldots, x_N) = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_N) - f(x_1, \ldots, x_i, \ldots, x_N)}{h}$$

- 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:

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

- 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.

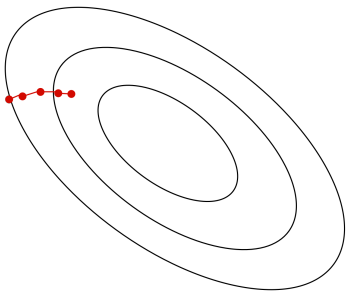
Today's Agenda

Today's agenda:

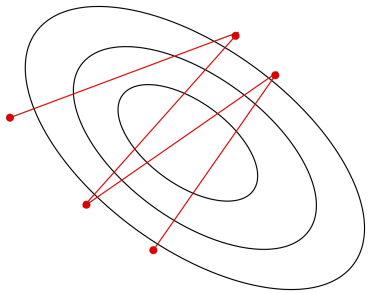
- Gradient checking with finite differences
- **Learning rates**
- Stochastic gradient descent
- Convexity
- Multiclass classification and softmax regression
- Limits of linear classification

Learning Rate

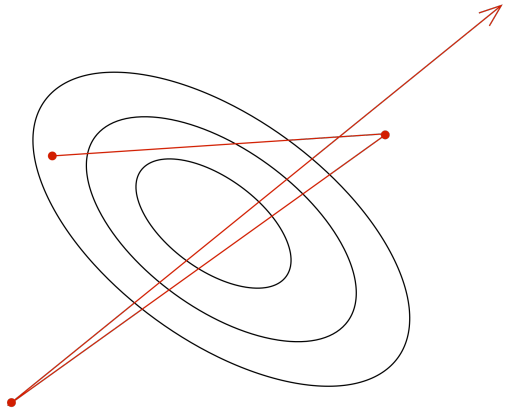
- In gradient descent, the learning rate α is a hyperparameter we need to tune. Here are some things that can go wrong:



α too small:
slow progress



α too large:
oscillations

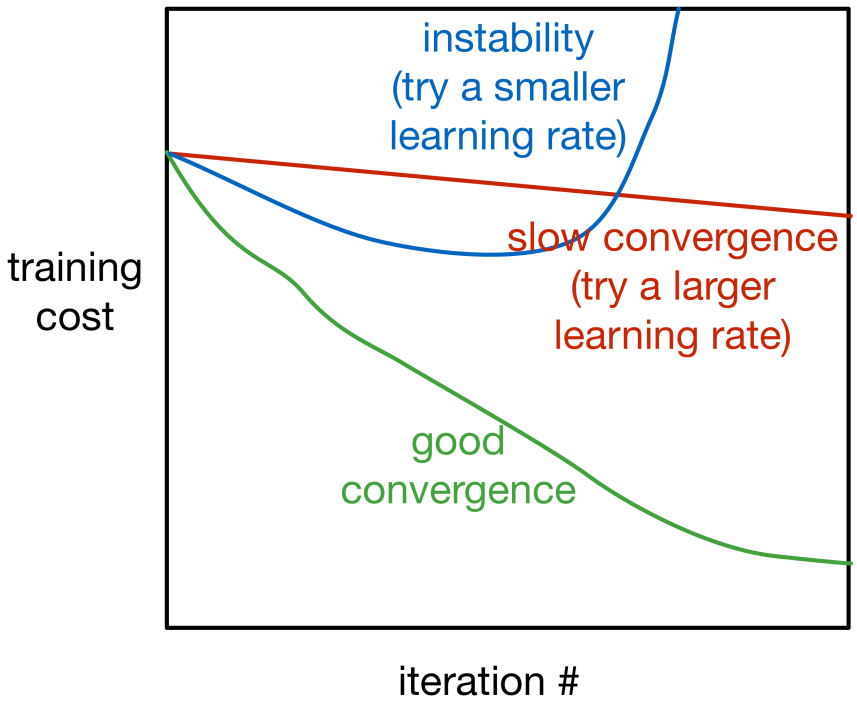


α much too large:
instability

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

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.

Today's Agenda

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 $\mathcal{J}$ 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.
- Batch training is impractical if you have a large dataset (e.g. millions of training examples)!

Stochastic Gradient Descent

- Stochastic gradient descent (SGD): update the parameters based on the gradient for a single training example:

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

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

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

- Problem: if we only look at one training example at a time, we can't exploit efficient vectorized operations.

Stochastic Gradient Descent

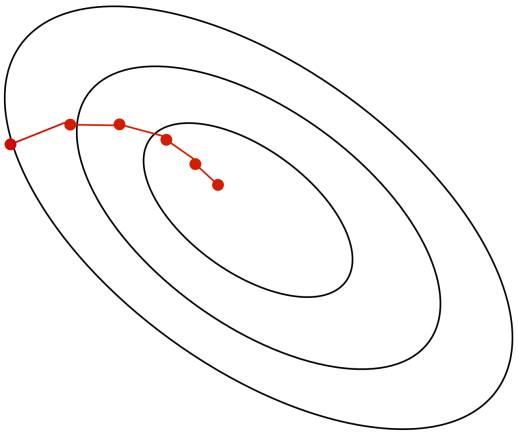
- Compromise approach: compute the gradients on a medium-sized set of training examples, called a mini-batch.
- Each entire pass over the dataset is called an epoch.
- Stochastic gradients computed on larger mini-batches have smaller variance:

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

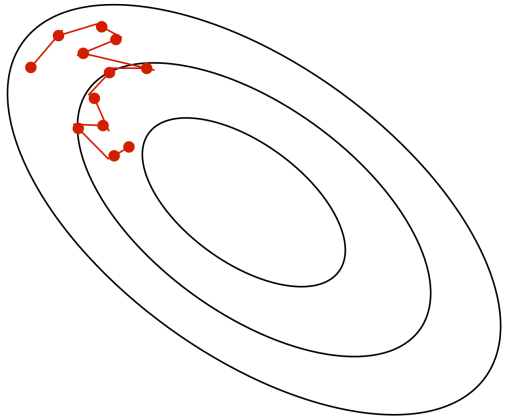
- The mini-batch size S is a hyperparameter that needs to be set.
  - Too large: takes more memory to store the activations, and longer to compute each gradient update
  - Too small: can't exploit vectorization
  - A reasonable value might be $S = 100$.

Stochastic Gradient Descent

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



batch gradient descent

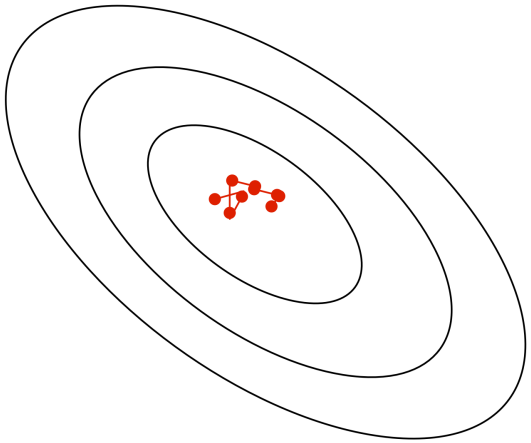


stochastic gradient descent

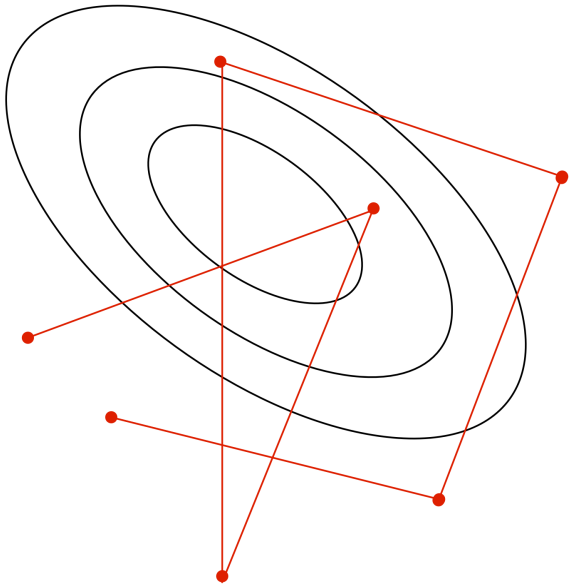
SGD Learning Rate

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

small learning rate



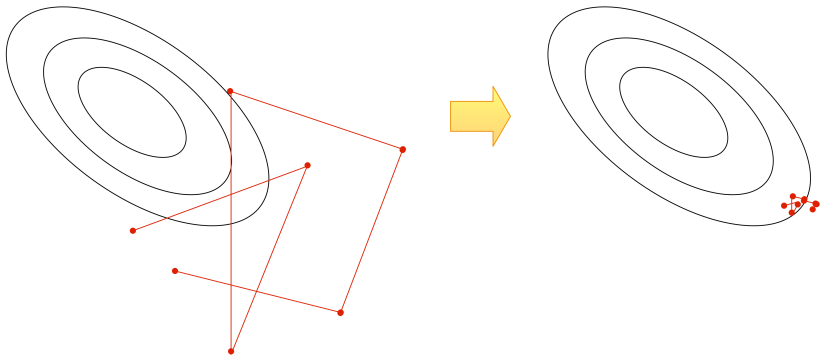
large learning rate

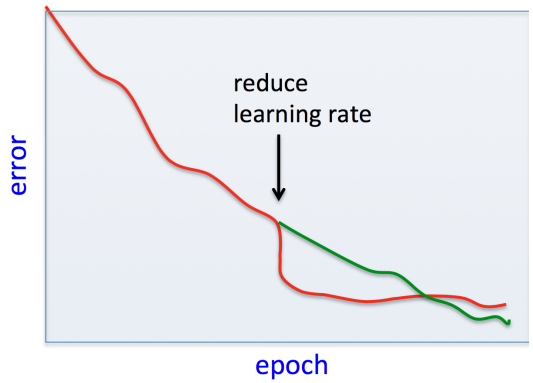


- Typical strategy:
  - Use a large learning rate early in training so you can get close to the optimum
  - Gradually decay the learning rate to reduce the fluctuations

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.



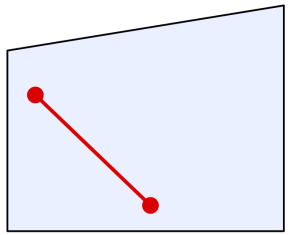


Today's Agenda

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



- A set $\mathcal{S}$ is convex if any line segment connecting points in $\mathcal{S}$ lies entirely within $\mathcal{S}$. 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:

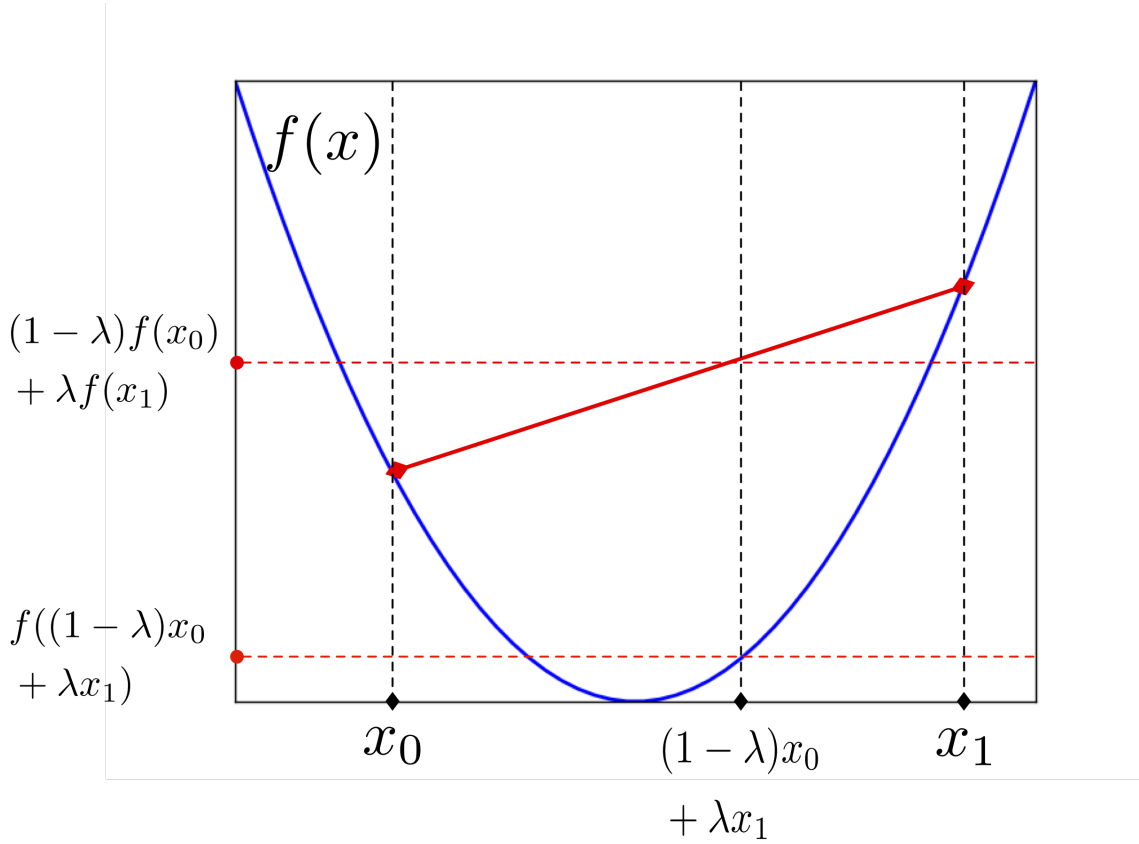
$$\lambda_1 \mathbf{x}_1 + \cdots + \lambda_N \mathbf{x}_N \in \mathcal{S} \quad \text{for } \lambda_i > 0, \ \lambda_1 + \cdots \lambda_N = 1.$$

Convex Functions

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

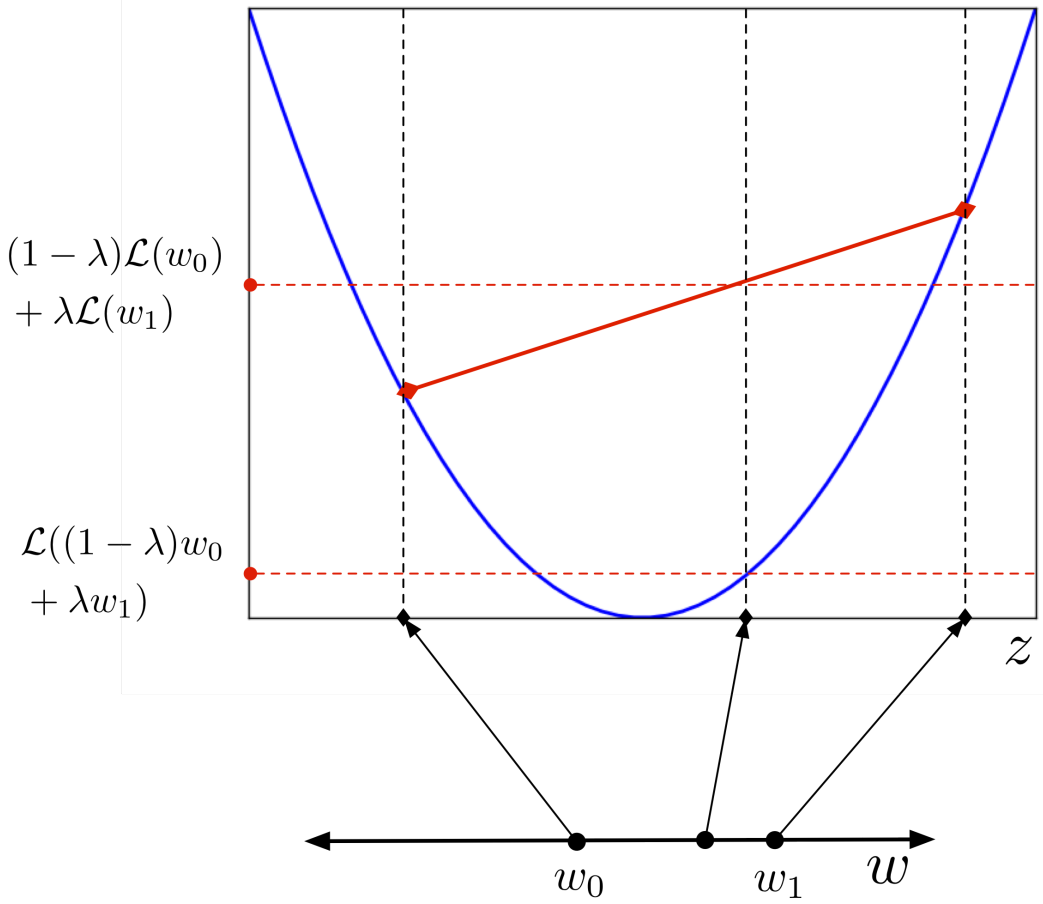
$$f((1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}_1) \leq (1-\lambda)f(\mathbf{x}_0) + \lambda f(\mathbf{x}_1)$$

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



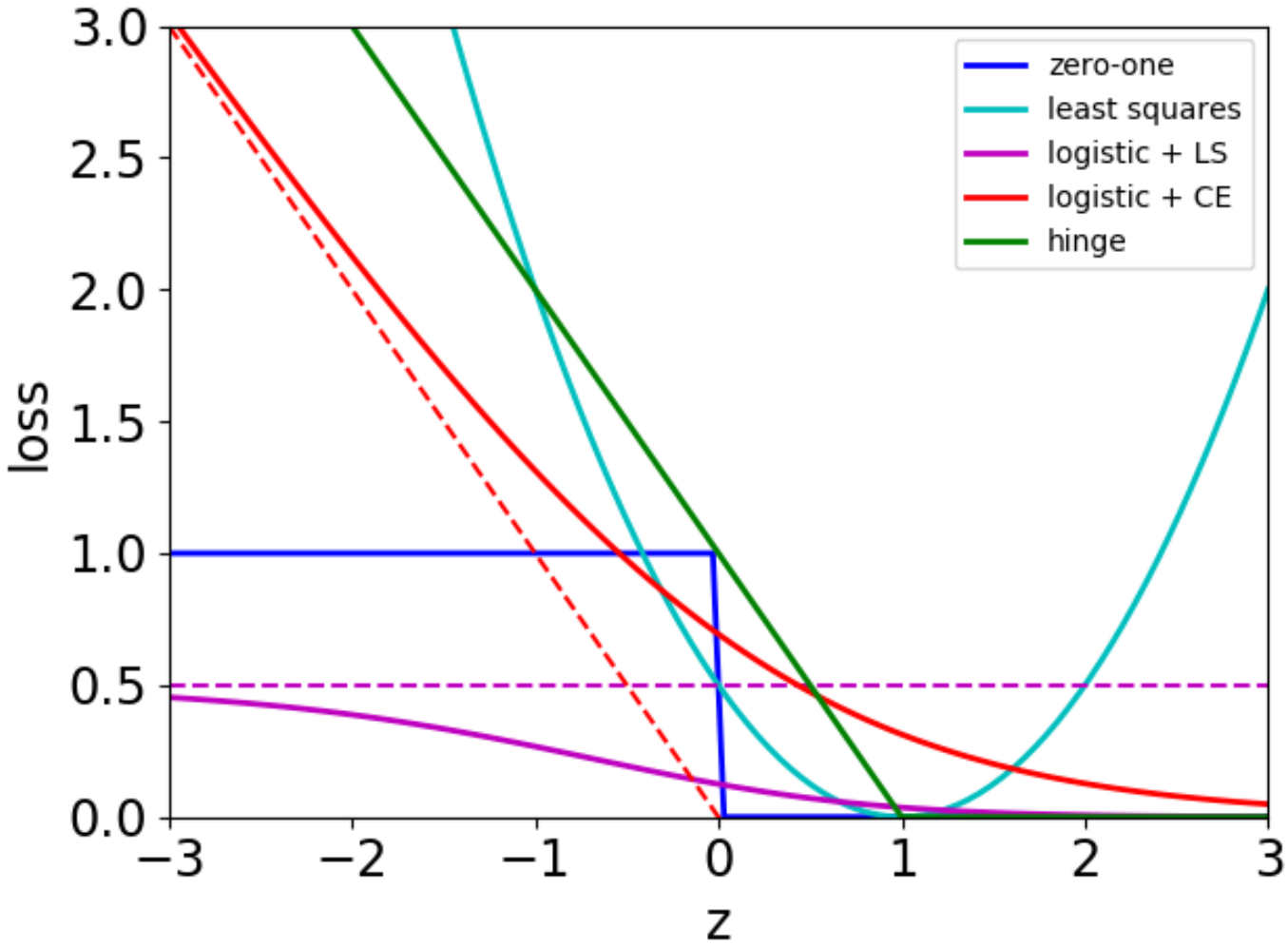
Convex Functions

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



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)

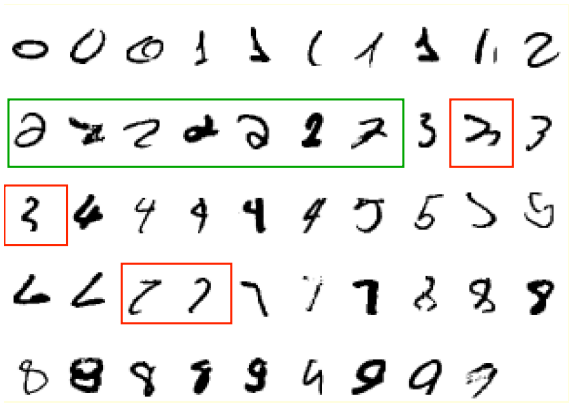
Today's Agenda

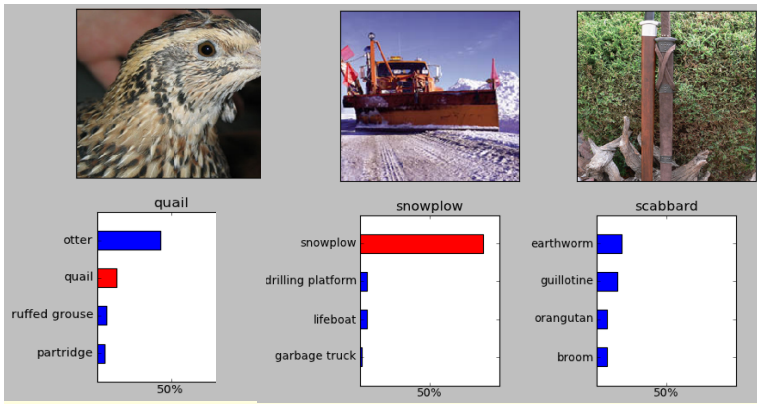
Today's agenda:

- 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 $\mathbf{W}$.
- Also, we have a K-dimensional vector $\mathbf{b}$ of biases.
- Linear predictions:

$$z_k = \sum_j w_{kj} x_j + b_k$$

- Vectorized:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Multiclass Classification

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

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

- 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, $\mathrm{softmax}(\mathbf{z})$ 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?
- Note: sometimes $\sigma(\mathbf{z})$ 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:

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

  where the log is applied elementwise.

- 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:

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

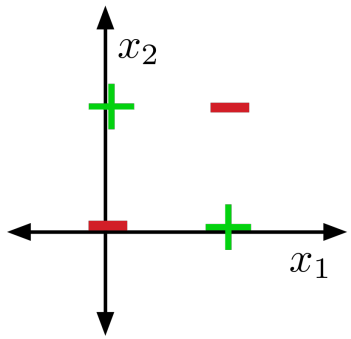
Today's Agenda

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

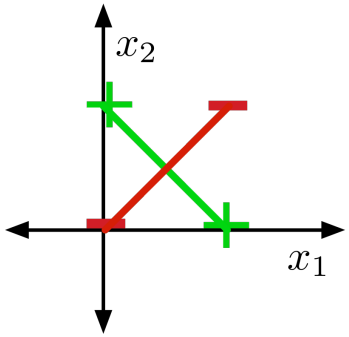
- Visually, it's obvious that **XOR** is not linearly separable. But how to show this?



Limits of Linear Classification

Showing that XOR is not linearly separable

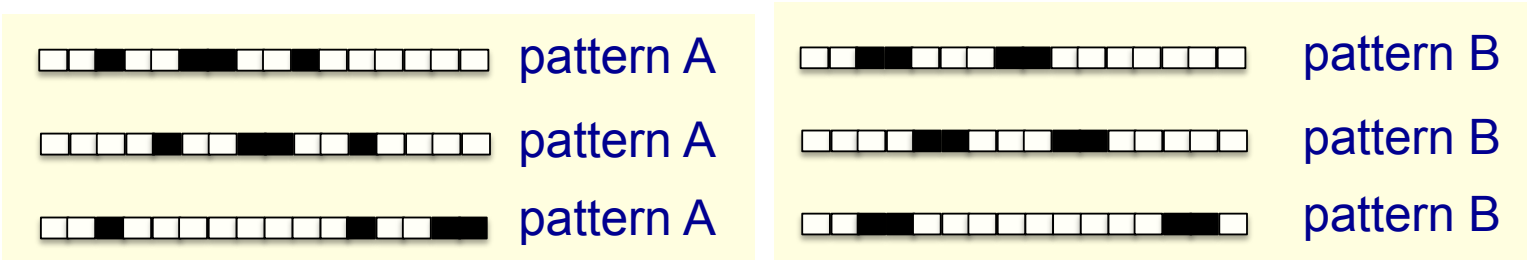
- Half-spaces are obviously convex.
- Suppose there were some feasible hypothesis. If the positive examples are in the positive half-space, then the green line segment must be as well.
- Similarly, the red line segment must line within the negative half-space.



- 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 $(0.25, 0.25, \ldots, 0.25)$. Therefore, this point must be classified as A.
- Similarly, the average of all translations of B is also $(0.25, 0.25, \ldots, 0.25)$. Therefore, it must be classified as B. Contradiction!

Credit: Geoffrey Hinton

Limits of Linear Classification

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

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

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. (Try it!)
- Not a general solution: it can be hard to pick good basis functions. Instead, we'll use neural nets to learn nonlinear hypotheses directly.