Обратное распространение ошибки

Виктор Китов
victorkitov.github.io

Настройка весов сети

Итерация градиентной оптимизации:

$$w := w - \varepsilon \nabla \mathcal{L}(w)$$

$$\nabla \mathcal{L}(w) = \left[\frac{\partial \mathcal{L}(w)}{\partial w_1}; \frac{\partial \mathcal{L}(w)}{\partial w_2}; ... \frac{\partial \mathcal{L}(w)}{\partial w_K} \right]$$

- $\mathcal{L}(w)$ - по минибатчу объектов.
- $w = [w_1, w_2, ...w_K]$ обновляются все синхронно.
- Как эффективно вичислить $\nabla \mathcal{L}(w)$?

Численная аппроксимация

- Вычисление $\nabla \mathcal{L}(w)$ через разностную аппроксимацию ($\delta_i = [0, ...0, \delta, 0, ...0]$)

$$\frac{\partial \mathcal{L}}{\partial w_i} \approx \frac{\mathcal{L}(w + \delta_i) - \mathcal{L}(w)}{\delta}$$

Численная аппроксимация

- Вычисление $\nabla \mathcal{L}(w)$ через разностную аппроксимацию ($\delta_i = [0, ...0, \delta, 0, ...0]$)

$$\frac{\partial \mathcal{L}}{\partial w_i} \approx \frac{\mathcal{L}(w + \delta_i) - \mathcal{L}(w)}{\delta}$$

  $\oplus$: автоматический метод, нет риска ошибки

  $\ominus$: приближённая. а не точная оценка

  $\ominus$: имеет вычислительную сложность $O(K^2)$

- нужно посчитать K производных
- сложность вычисления каждой: $O(K)$

Вычисление напрямую

- Вычисление напрямую
  - $\oplus$: точная производная
  - $\ominus$: громоздкие вычисления, риск ошибки

Вычисление напрямую

- Вычисление напрямую
  - $\oplus$: точная производная
  - $\ominus$: громоздкие вычисления, риск ошибки
- Библиотеки символьного дифференцирования
  - $\oplus$: точная производная
  - $\oplus$: автоматический метод, нет риска ошибки

Вычисление напрямую

- Вычисление напрямую
  $\oplus$: точная производная
  $\ominus$: громоздкие вычисления, риск ошибки
- Библиотеки символьного дифференцирования
  $\oplus$: точная производная
  $\oplus$: автоматический метод, нет риска ошибки

- Оба метода вычислительно неэффективны - повторное вычисление одинаковых слагаемых:

$$[A(w)B(w)C(w)]'$$
$$= A'(w)B(w)C(w) + A(w)B'(w)C(w) + A(w)B(w)C'(w)$$

Автоматическое дифференцирование

- Библиотеки автоматического дифференцирования вычисляют значение градиента в точке
  $\oplus$: автоматически, без риска ошибки
  $\oplus$: вычисляют точное значение
  $\oplus$: эффективно за $O(K)$
- Используют метод обратного распространения ошибки (backpropagation).
- Основные библиотеки: PyTorch, Tensorflow, JAX.

Дифференцирование сложной функции

$$\mathcal{L}(w) = A(B(w))$$

Дифференцирование сложной функции

$$\mathcal{L}(w) = A(B(w))$$

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \frac{\partial A(B)}{\partial B}\frac{\partial B}{\partial w}$$

$$\mathcal{L}(w) = \mathcal{L}(A_1(w), A_2(w), A_3(w))$$

Дифференцирование сложной функции

$$\mathcal{L}(w) = A(B(w))$$

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \frac{\partial A(B)}{\partial B}\frac{\partial B}{\partial w}$$

$$\mathcal{L}(w) = \mathcal{L}(A_1(w), A_2(w), A_3(w))$$

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \frac{\partial \mathcal{L}}{\partial A_1}\frac{\partial A_1}{\partial w} + \frac{\partial \mathcal{L}}{\partial A_2}\frac{\partial A_2}{\partial w} + \frac{\partial \mathcal{L}}{\partial A_3}\frac{\partial A_3}{\partial w}$$

$$\mathcal{L}(w) = \mathcal{L}(A(B(C(w))))$$

Дифференцирование сложной функции

$$\mathcal{L}(w) = A(B(w))$$

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \frac{\partial A(B)}{\partial B}\frac{\partial B}{\partial w}$$

$$\mathcal{L}(w) = \mathcal{L}(A_1(w), A_2(w), A_3(w))$$

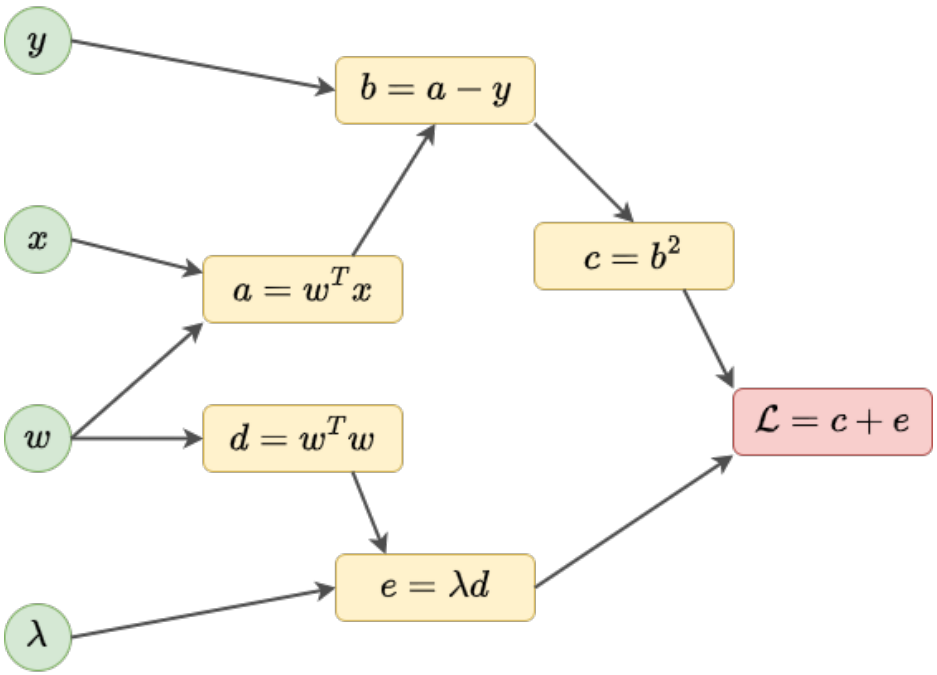
$$\frac{\partial \mathcal{L}(w)}{\partial w} = \frac{\partial \mathcal{L}}{\partial A_1}\frac{\partial A_1}{\partial w} + \frac{\partial \mathcal{L}}{\partial A_2}\frac{\partial A_2}{\partial w} + \frac{\partial \mathcal{L}}{\partial A_3}\frac{\partial A_3}{\partial w}$$

$$\mathcal{L}(w) = \mathcal{L}(A(B(C(w))))$$

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \frac{\partial \mathcal{L}}{\partial A}\frac{\partial A}{\partial B}\frac{\partial B}{\partial C}\frac{\partial C}{\partial w}$$

Обратное распространение ошибки

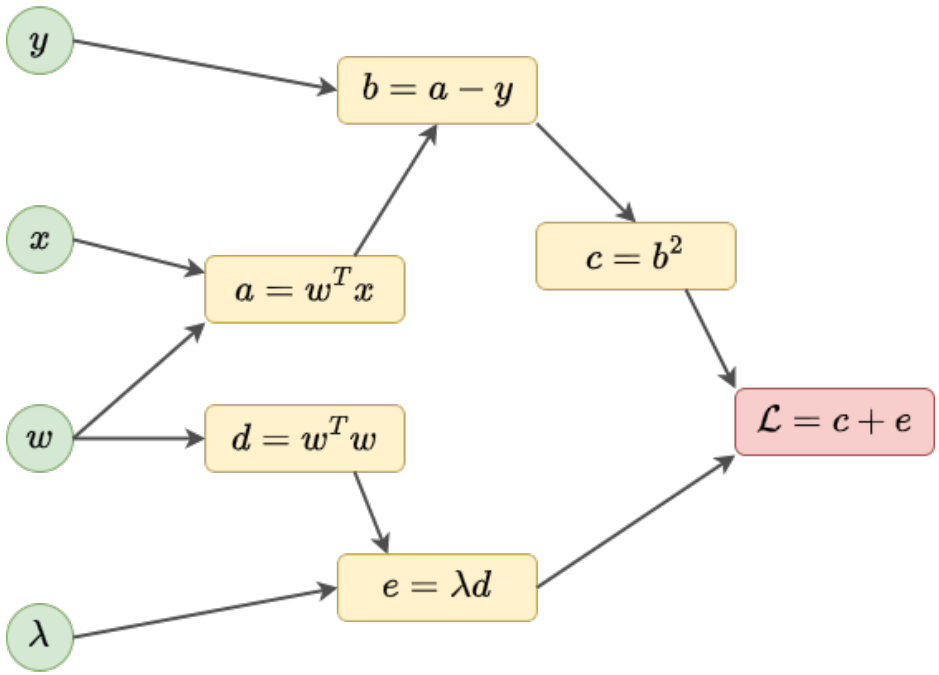
- Вычисление функции потерь $\mathcal{L}(w)$ декомпозируется в вычислительный граф из элементарных преобразований
  - по которым мы можем посчитать производную.
  - +, -, *, : , exp, log, sin, cos, ...
- Пример: $\hat{y} = \mathbf{w}^T\mathbf{x}, \quad \mathcal{L}(\mathbf{w}) = (\mathbf{w}^T\mathbf{x} - y)^2 + \lambda\mathbf{w}^T\mathbf{w}$



Проходы вперёд и назад

- Вычисление $\nabla \mathcal{L}(w)$: 1) проход вперёд 2) проход назад.
- Проход вперёд (forward pass): итеративно слева-направо
  - вычисляются все промежуточные переменные
  - запоминаются промежуточные переменные и их функциональные преобразования.
- Проход назад (backward pass): итеративно справа-налево
  - вычисляются производные итога от предыдущих переменных графа как функции.
  - подстановкой переменных получаем численные значения производных.
  - после получения чисел функции уже не нужны.

Проход вперёд: $\mathcal{L}(\mathbf{w}) = (\mathbf{w}^T\mathbf{x} - y)^2 + \lambda\mathbf{w}^T\mathbf{w}$



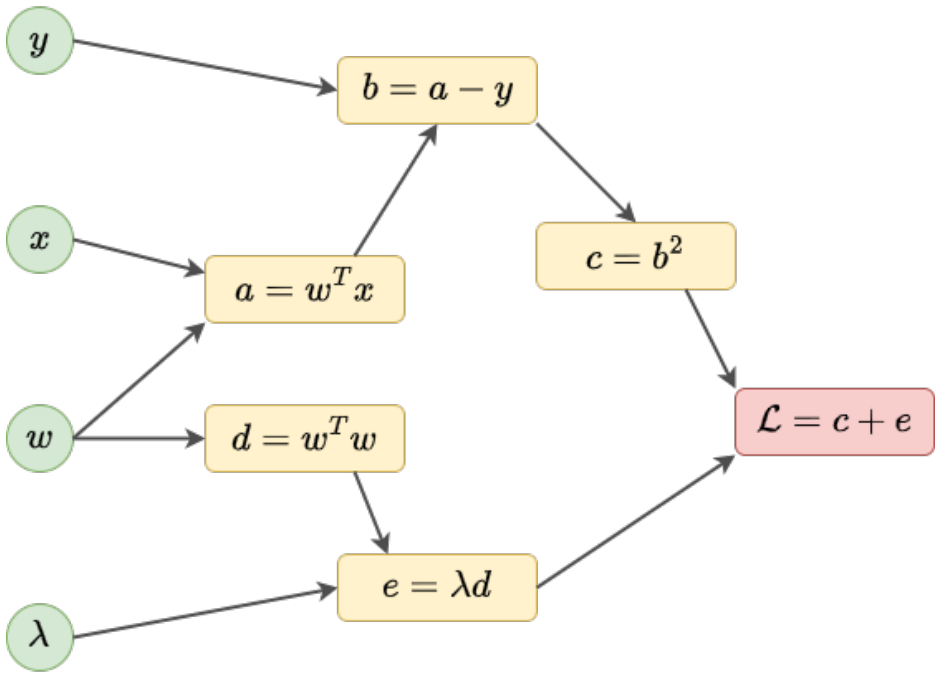
$$y = 0,\ \mathbf{x} = [1, 2],\ \mathbf{w} = [3, 4],\ \lambda = 5. \quad \nabla\mathcal{L}(w) - ?$$

$$a = 1 \cdot 3 + 2 \cdot 4 = 3 + 8 = 11; \quad b = a - y = 11$$

$$c = b^2 = 11^2 = 121; \quad d = 3 \cdot 3 + 4 \cdot 4 = 9 + 16 = 25$$

$$e = \lambda d = 125; \quad \mathcal{L} = c + e = 121 + 125 = 246$$

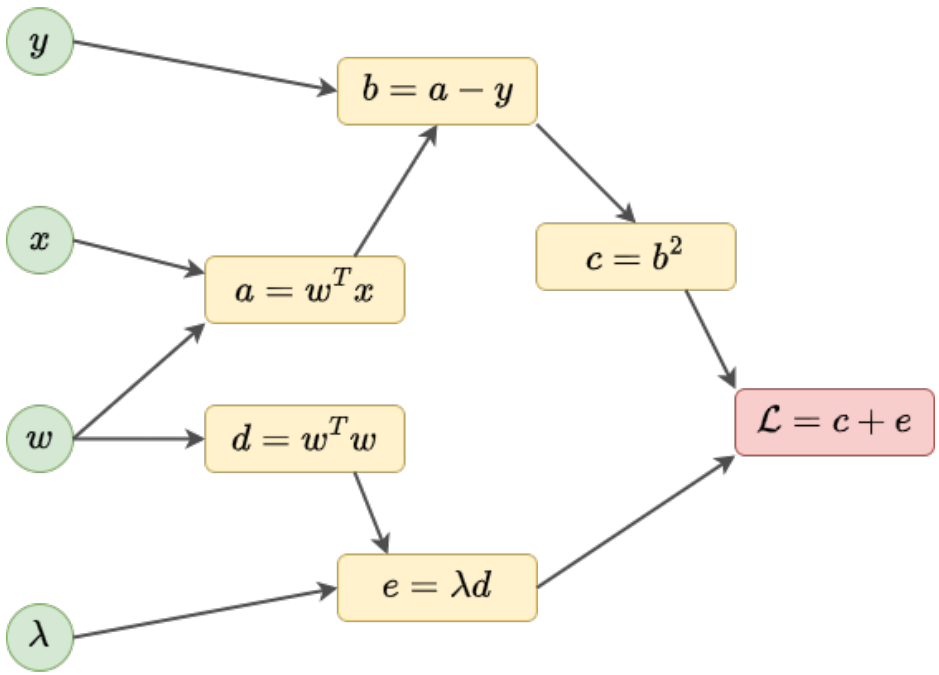
Проход назад: $\mathcal{L}(\mathbf{w}) = (\mathbf{w}^T\mathbf{x} - y)^2 + \lambda\mathbf{w}^T\mathbf{w}$



$$a = 11; \quad b = 11; \quad c = 121; \quad d = 25; \quad e = 125$$

$$\frac{\partial \mathcal{L}}{\partial c} = 1, \quad \frac{\partial \mathcal{L}}{\partial e} = 1$$

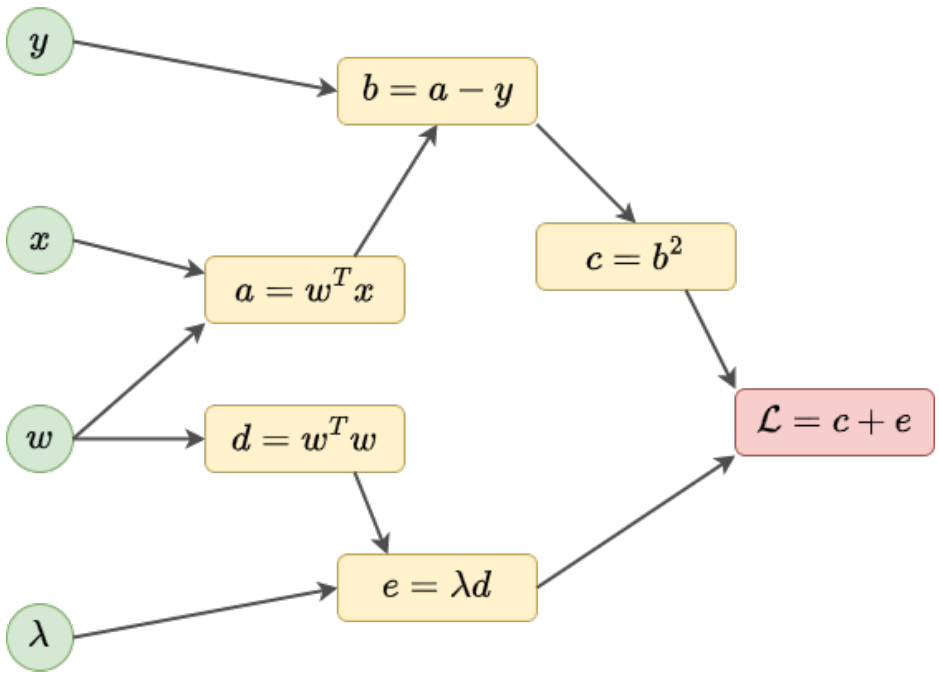
Проход назад: $\mathcal{L}(\mathbf{w}) = (\mathbf{w}^T\mathbf{x} - y)^2 + \lambda\mathbf{w}^T\mathbf{w}$



$$a = 11; \quad b = 11; \quad c = 121; \quad d = 25; \quad e = 125$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}(c)}{\partial b} = \frac{\partial \mathcal{L}}{\partial c}\frac{\partial c}{\partial b} = 1 \cdot 2b = 22; \; \frac{\partial \mathcal{L}}{\partial d} = \frac{\partial \mathcal{L}(e)}{\partial d} = \frac{\partial \mathcal{L}}{\partial e}\frac{\partial e}{\partial d} = 1 \cdot \lambda = 5$$

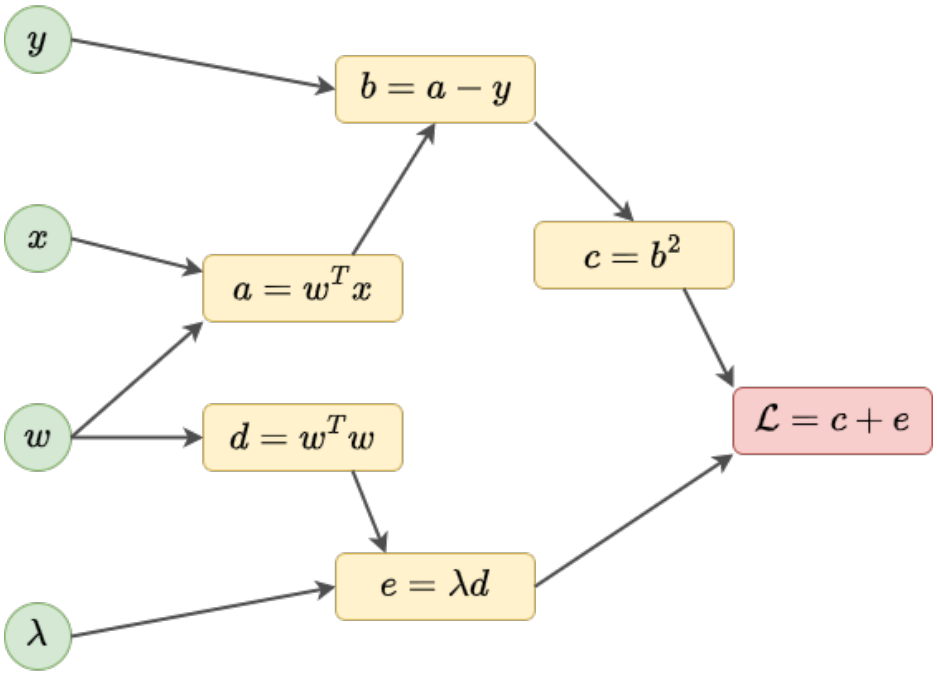
Проход назад: $\mathcal{L}(\mathbf{w}) = (\mathbf{w}^T\mathbf{x} - y)^2 + \lambda\mathbf{w}^T\mathbf{w}$



$$a = 11; \quad b = 11; \quad c = 121; \quad d = 25; \quad e = 125$$

$$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial \mathcal{L}(b)}{\partial a} = \frac{\partial \mathcal{L}}{\partial b}\frac{\partial b}{\partial a} = 22 \cdot 1 = 22$$

Проход назад: $\mathcal{L}(\mathbf{w}) = (\mathbf{w}^T\mathbf{x} - y)^2 + \lambda\mathbf{w}^T\mathbf{w}$

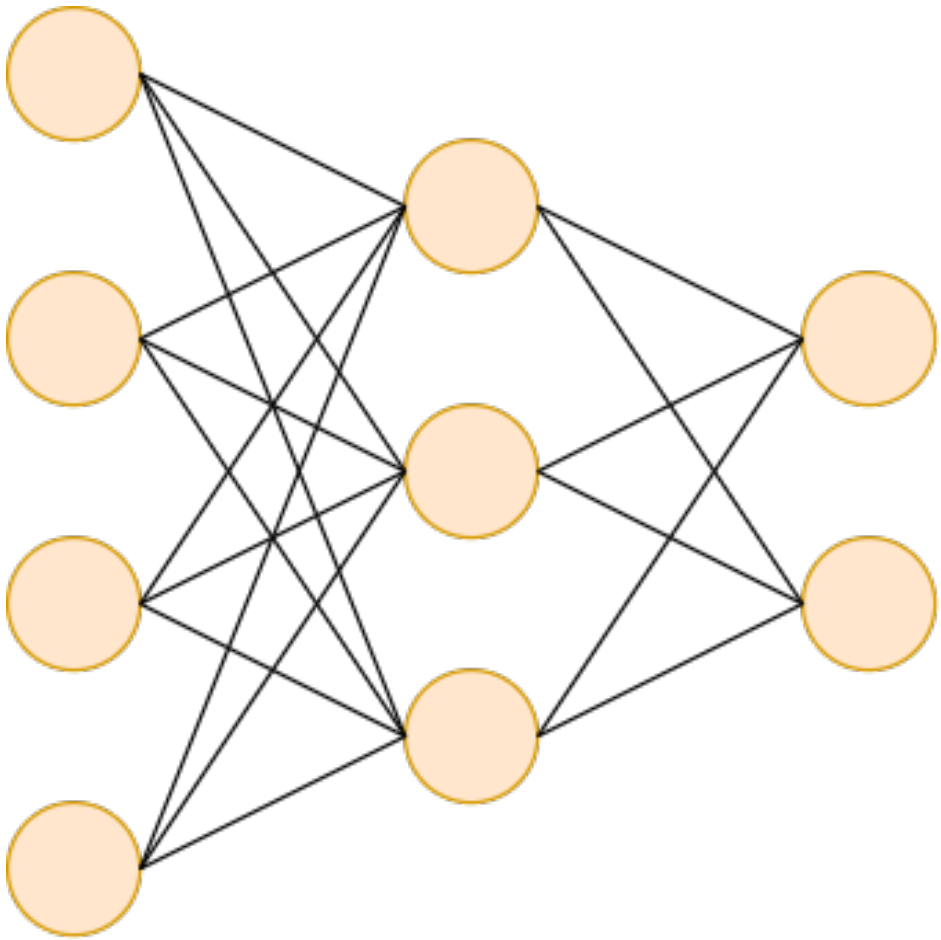


$$a = 11; \quad b = 11; \quad c = 121; \quad d = 25; \quad e = 125$$

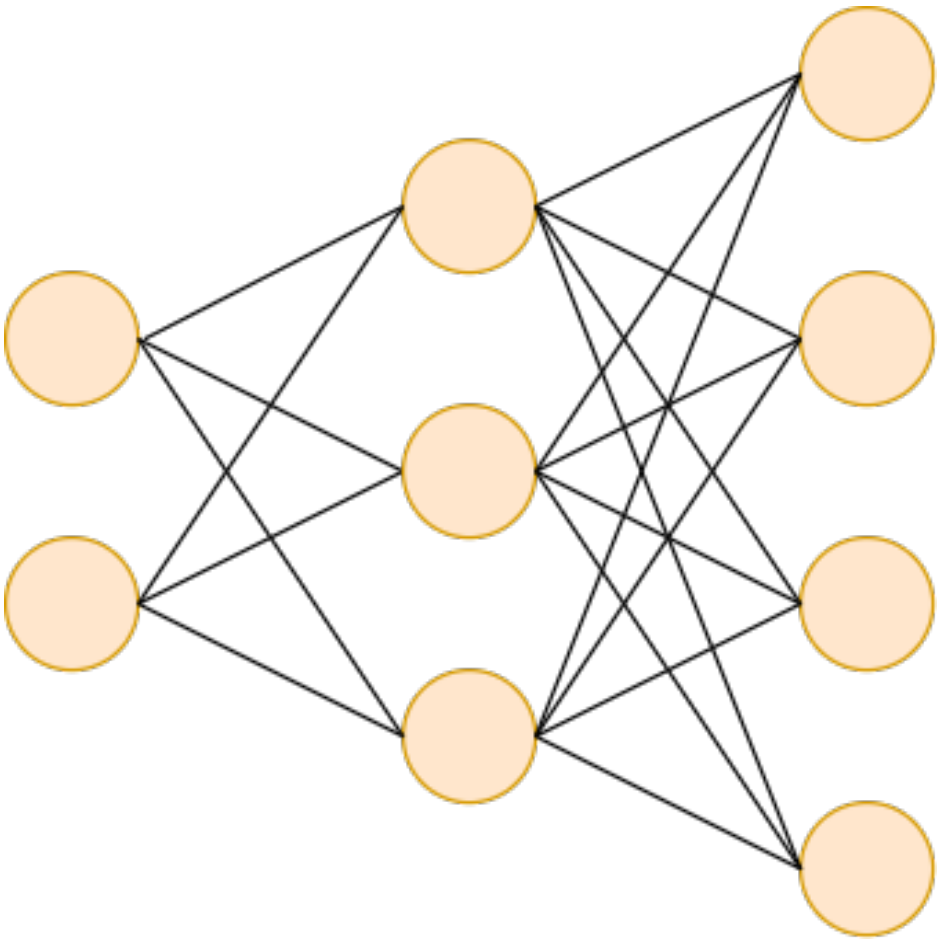
$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}(a,d)}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial \mathbf{w}} + \frac{\partial \mathcal{L}}{\partial d}\frac{\partial d}{\partial \mathbf{w}} = 22 \cdot \mathbf{x} + 5 \cdot 2\mathbf{w}$$

$$= 22 \cdot [1,2] + 10 \cdot [3,4] = [22,44] + [30,40] = [52,84]$$

Эффективенее backward mode (backpropagation)

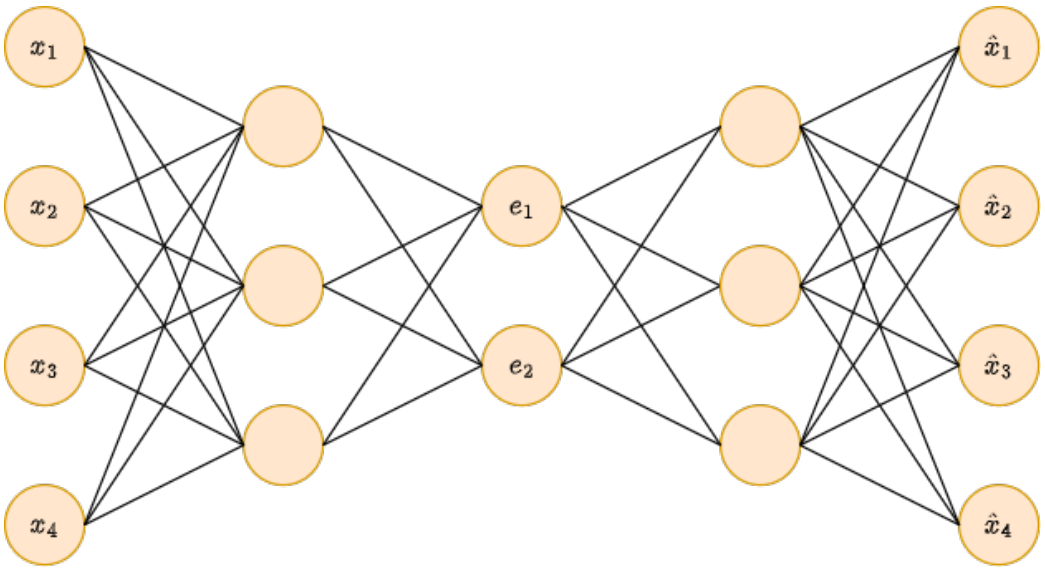


Эффективенее forward mode[1]



[1] См. https://deepmachinelearning.ru/docs/Neural-networks/Training/Backpropagation

Эффективнее комбинация



$$\frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{x}} = \frac{\partial \hat{\mathbf{x}}(\mathbf{e})}{\partial \mathbf{x}} = \frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial \mathbf{x}}$$

Эффективнее посчитать $\frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{e}}$ через backward mode, а $\frac{\partial \mathbf{e}}{\partial \mathbf{x}}$ - через forward mode.

Заключение

- Метод обратного распространение ошибки - эффективный метод расчёта $\nabla \mathcal{L}(w)$
  - именно он используется на практике, т.к. $\mathcal{L}(w)$ - скаляр.
  - вычисляет $\nabla \mathcal{L}(w)$ за $O(K)$.
- Использование метода на практике:
  - при проходе вперёд запоминаются производные, а не сами преобразования.
  - при проходе назад после использования переменных (справа) их можно удалить.
- PyTorch: requires_grad=True/False, grad_fn.