Южный федеральный университет

@SINECOR

Функции ошибки и графы вычислений



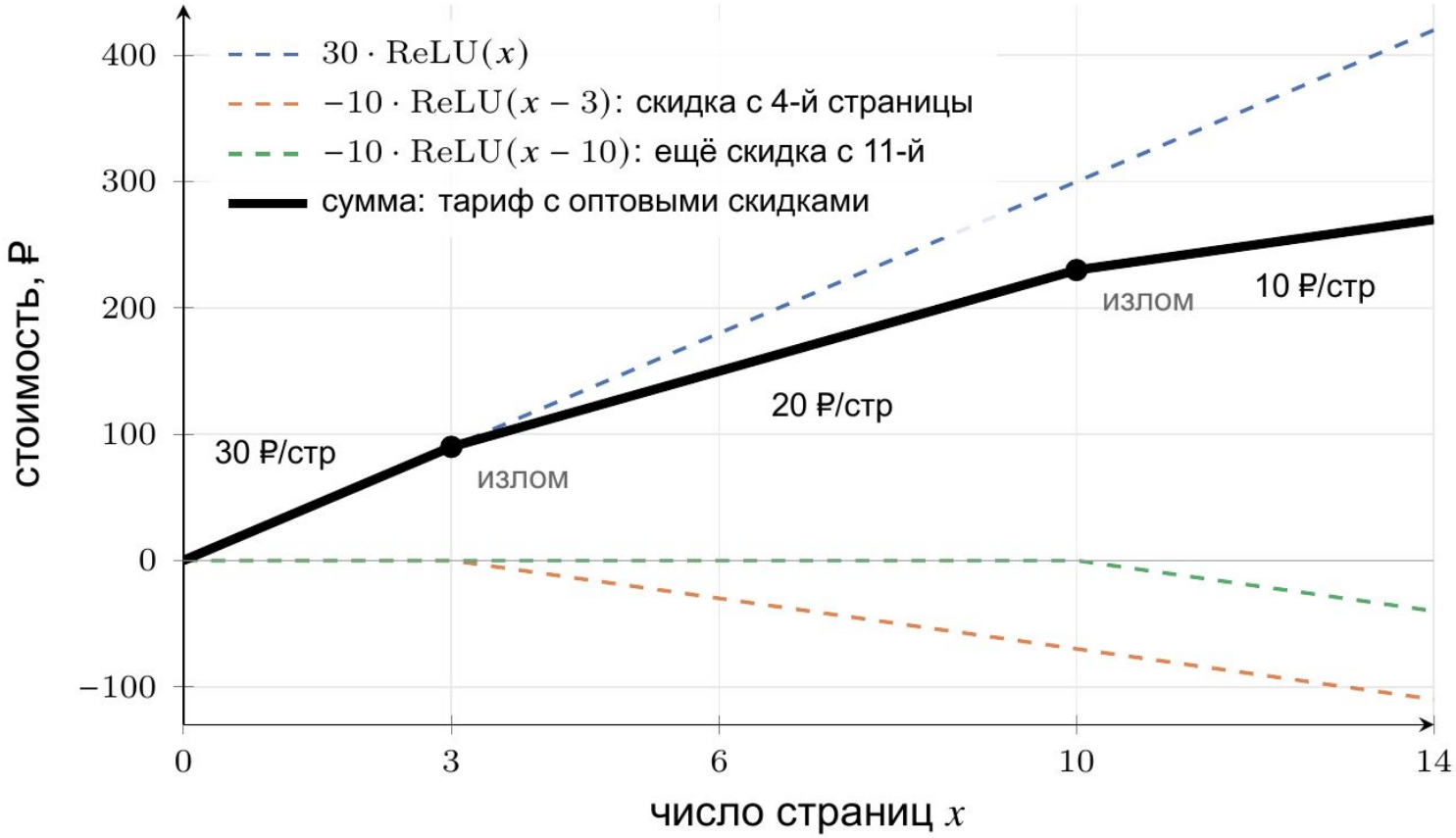
Сергей Николенко
19 октября 2026 г.

Технологии и фронтиры науки о данных

Напоминание о сетях с ReLU-нейронами



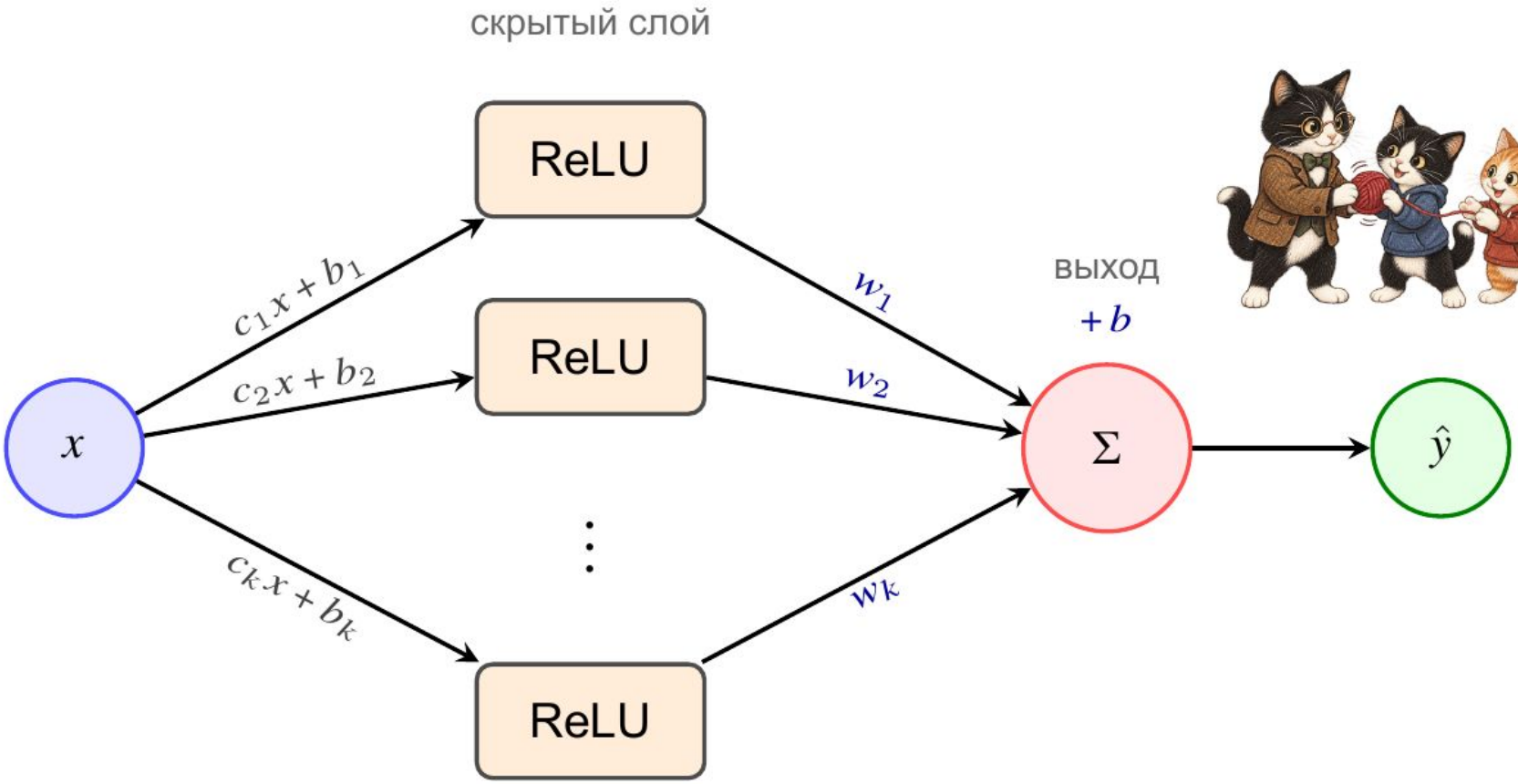
Напоминание: регрессия с ReLU





- В прошлый раз мы изучали регрессию

Напоминание: регрессия с ReLU



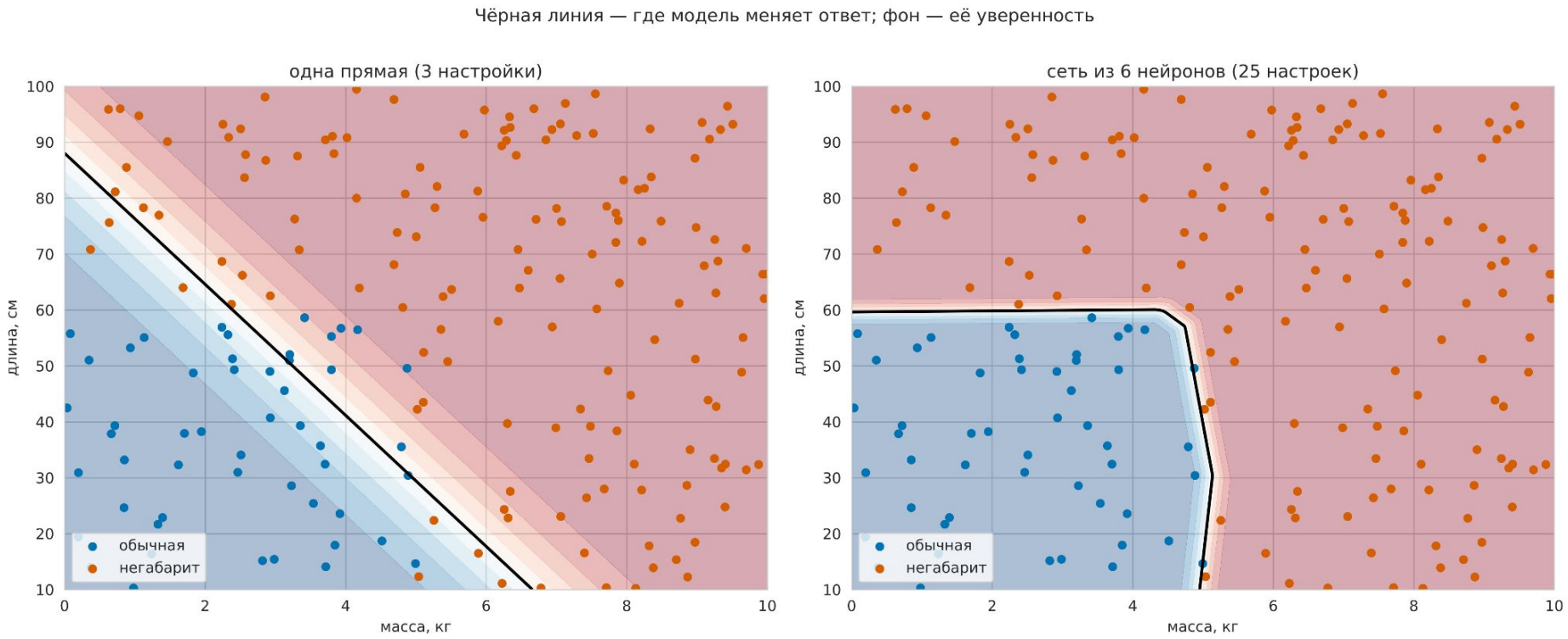
$$\hat{y} = b + \sum_{i=1}^{k} w_i \,\mathrm{ReLU}(c_i x + b_i), \qquad \mathrm{ReLU}(z) = \max(0, z)$$

каждый шарнир — две настройки: наклон c_i и сдвиг b_i (он задаёт, где излом), и ещё вес w_i, с которым шарнир входит в сумму

Пример задачи классификации



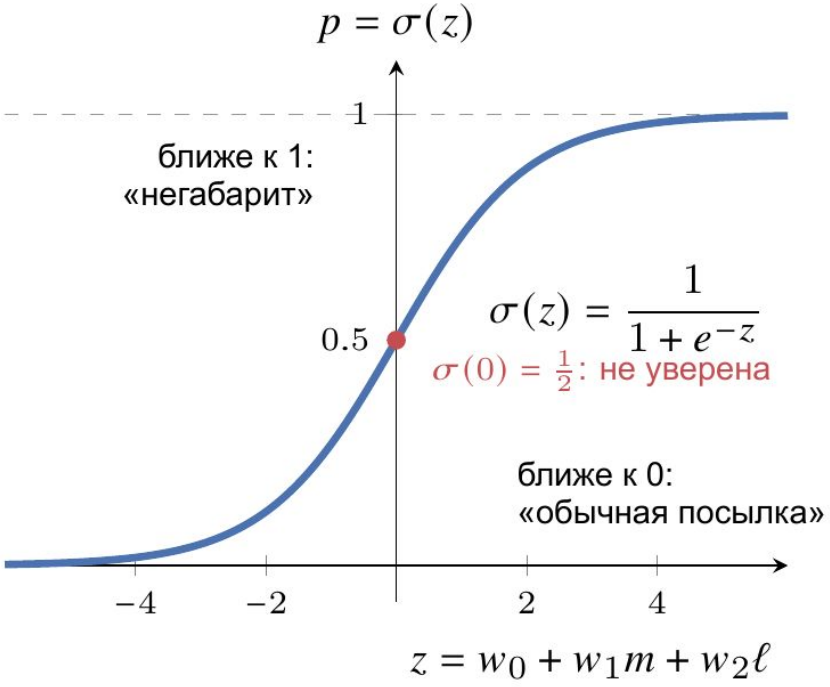
Классификация



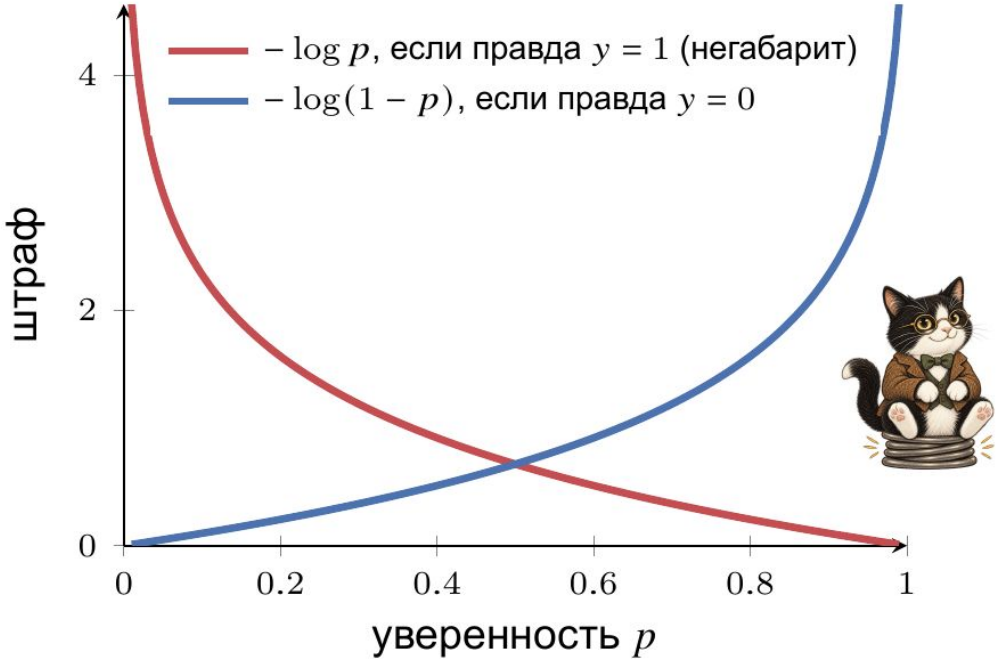
- То же можно провернуть и с классификацией
- Здесь, например, просто надо вырезать прямоугольничек
- Прямая не может этого сделать, а нейросеть со скрытым слоем легко
- Но как её обучать?

Сигмоид и логарифмическая ошибка

(а) уверенность модели



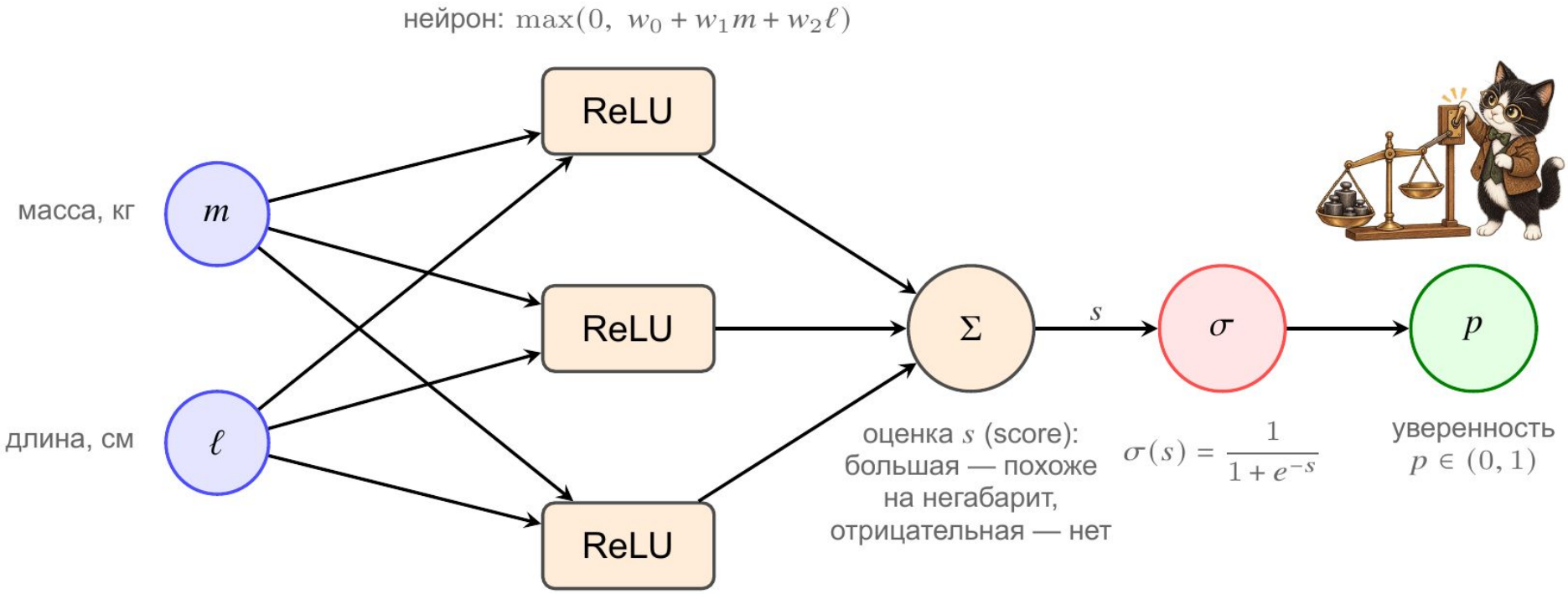
(б) логарифмическая ошибка



$$\ell(\boldsymbol{w}) = -y \log p - (1-y)\log(1-p), \qquad L(\boldsymbol{w}) = \sum_{n=1}^{N} \ell(\boldsymbol{w}; \boldsymbol{x}_n, y_n) \;\to\; \min_{\boldsymbol{w}}$$

чем увереннее модель в неправильном ответе, тем больше штраф; за уверенный правильный ответ штраф почти ноль

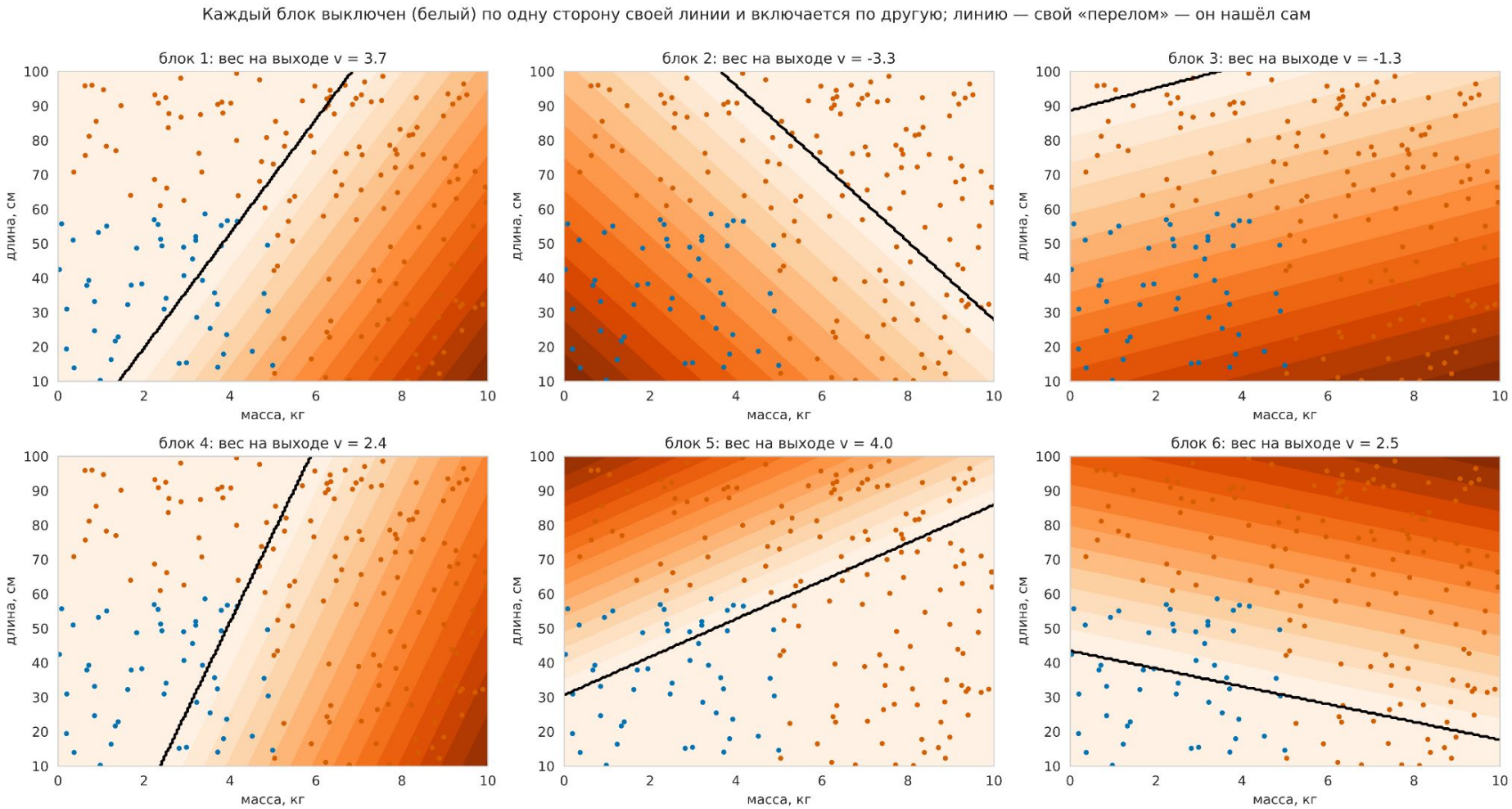
Сеть с ReLU для классификации



та же конструкция, что для тарифа: линейная функция, потом ReLU; только входов два, а на выходе сигмоида, чтобы получилась вероятность; учим всё тем же градиентным спуском по логарифмической ошибке

- Получается такая же сеть, но в конце линейная оценка пропускается через сигмоид, чтобы получилось что-то, похожее на вероятность

Классификация



- ...её нейроны, правда, не так легко интерпретировать

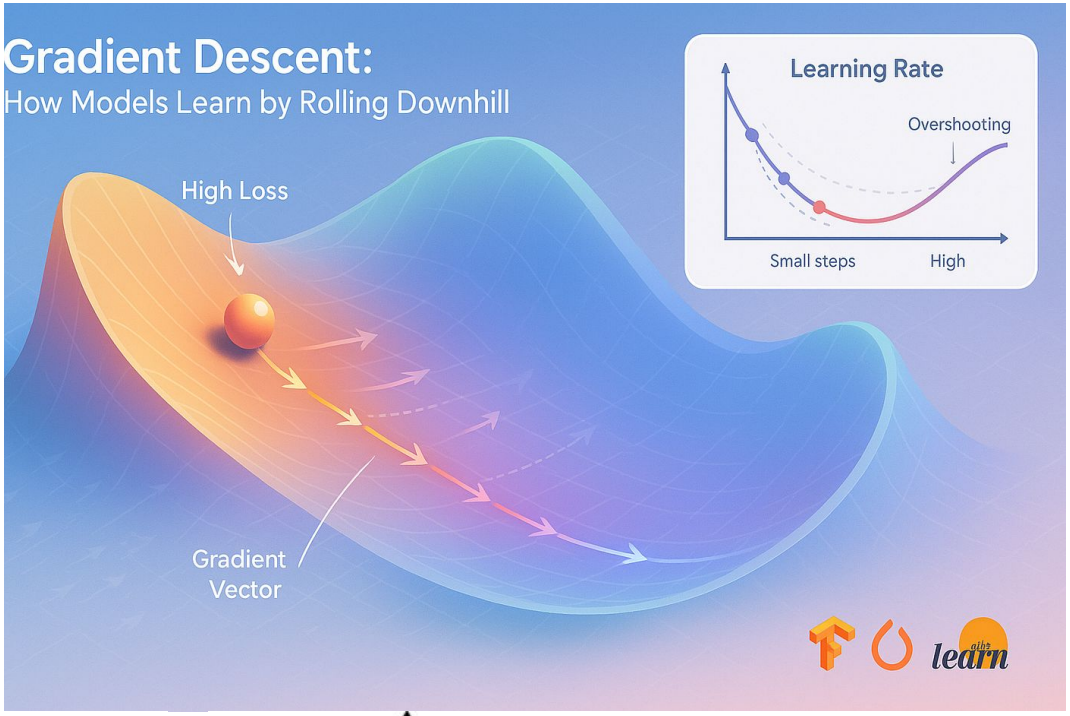
При чём здесь градиентный спуск

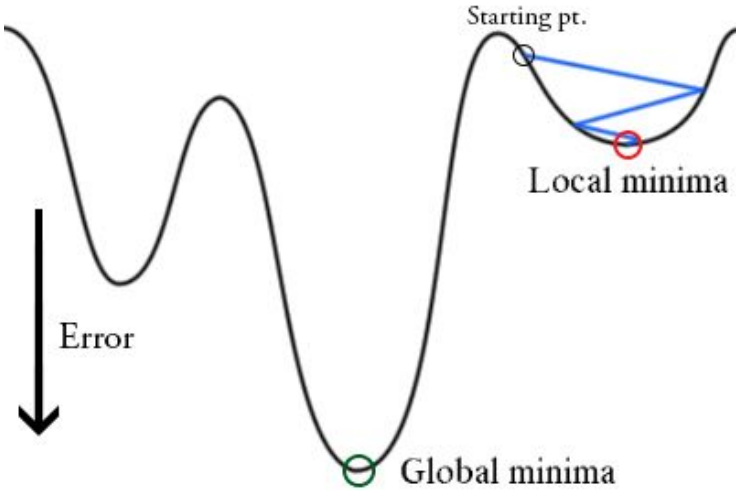


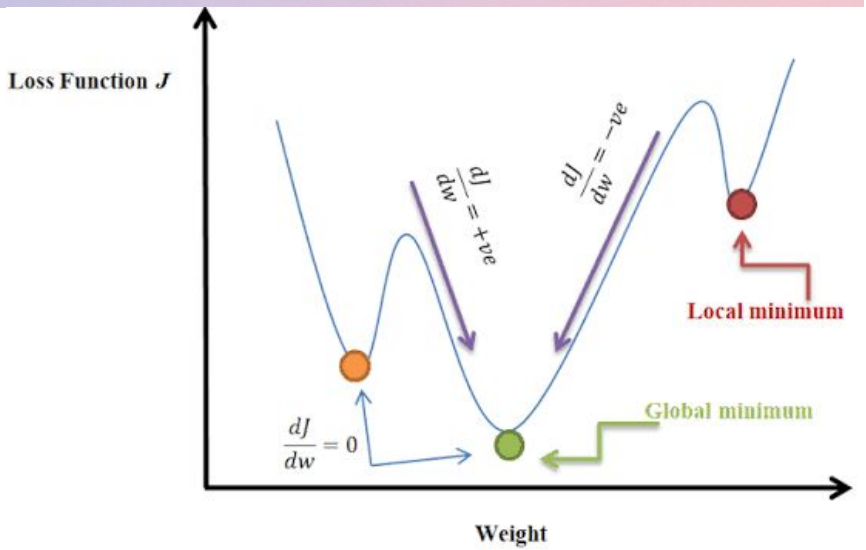
Градиентный спуск

- Самый главный метод оптимизации: нужно спускаться в сторону, обратную градиенту

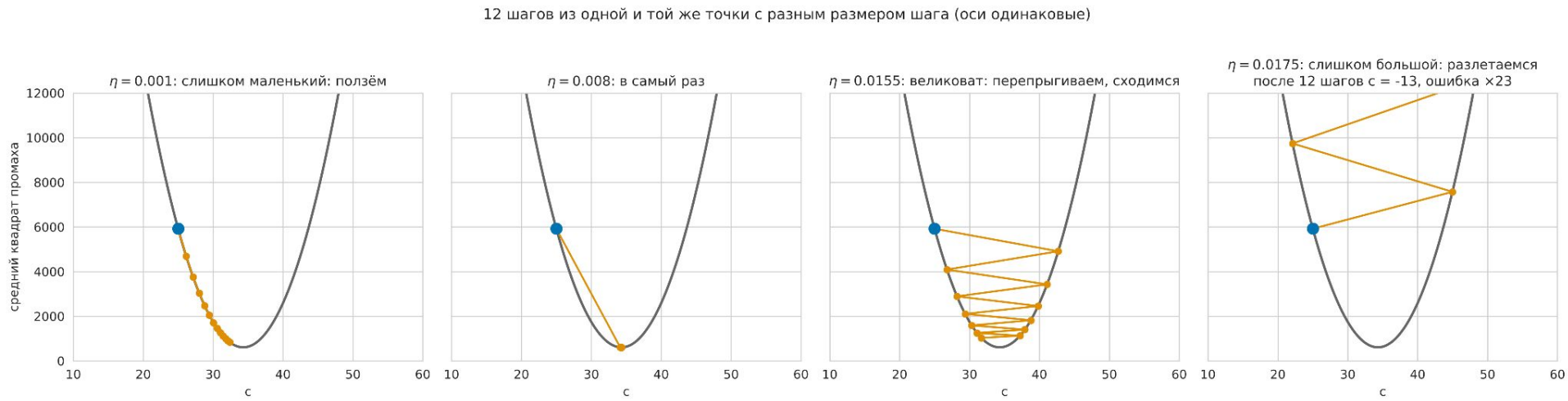
$$\mathbf{x}_k = \mathbf{x}_{k-1} - \alpha \nabla f(\mathbf{x}_k).$$







Скорость обучения

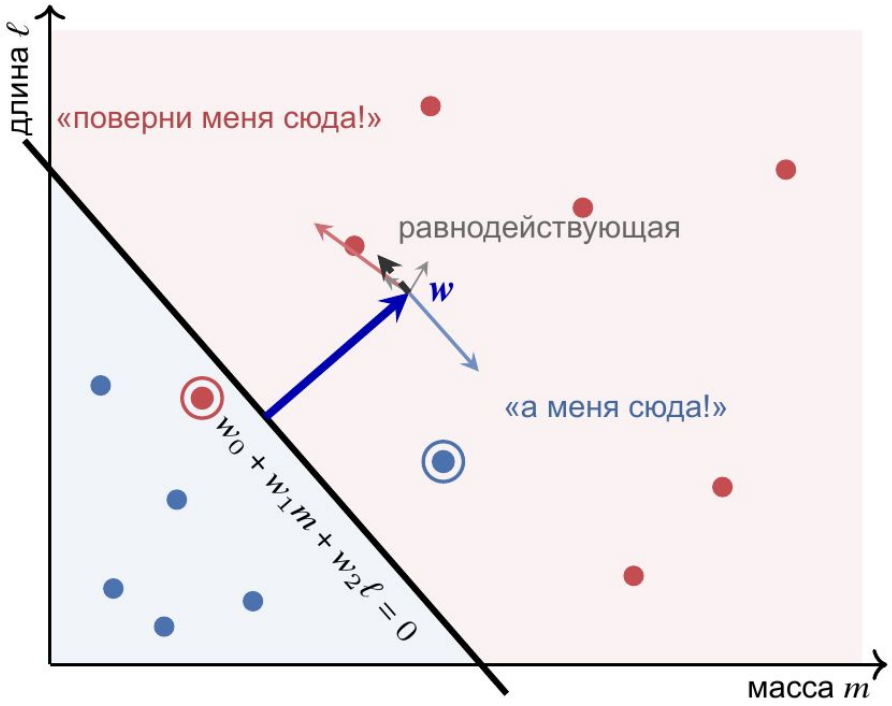


- В таких методах оптимизации (локальных, на градиентном спуске) многое зависит от скорости ошибки

Полный градиентный спуск не работает

- Самый главный метод оптимизации: нужно спускаться в сторону, обратную градиенту

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \alpha \nabla f(\mathbf{x}_k).$$



- Заметим, что в реальной жизни даже подсчитать функцию ошибки и её градиент практически невозможно! Это же сумма по данным!

$$L(\boldsymbol{w}) = \sum_{n=1}^{N} \ell(\boldsymbol{w}; \boldsymbol{x}_n, y_n)$$

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \sum_{n=1}^{N} \nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}; \boldsymbol{x}_n, y_n)$$

каждый пример — своя стрелка «куда повернуть веса, чтобы *мне* стало лучше»; шаг градиентного спуска — по их сумме.

Но N — это миллионы картинок или триллионы токенов: чтобы один раз посчитать $L(w)$, а тем более ∇L, нужно пройти по *всему* датасету. А шагов нужны тысячи…

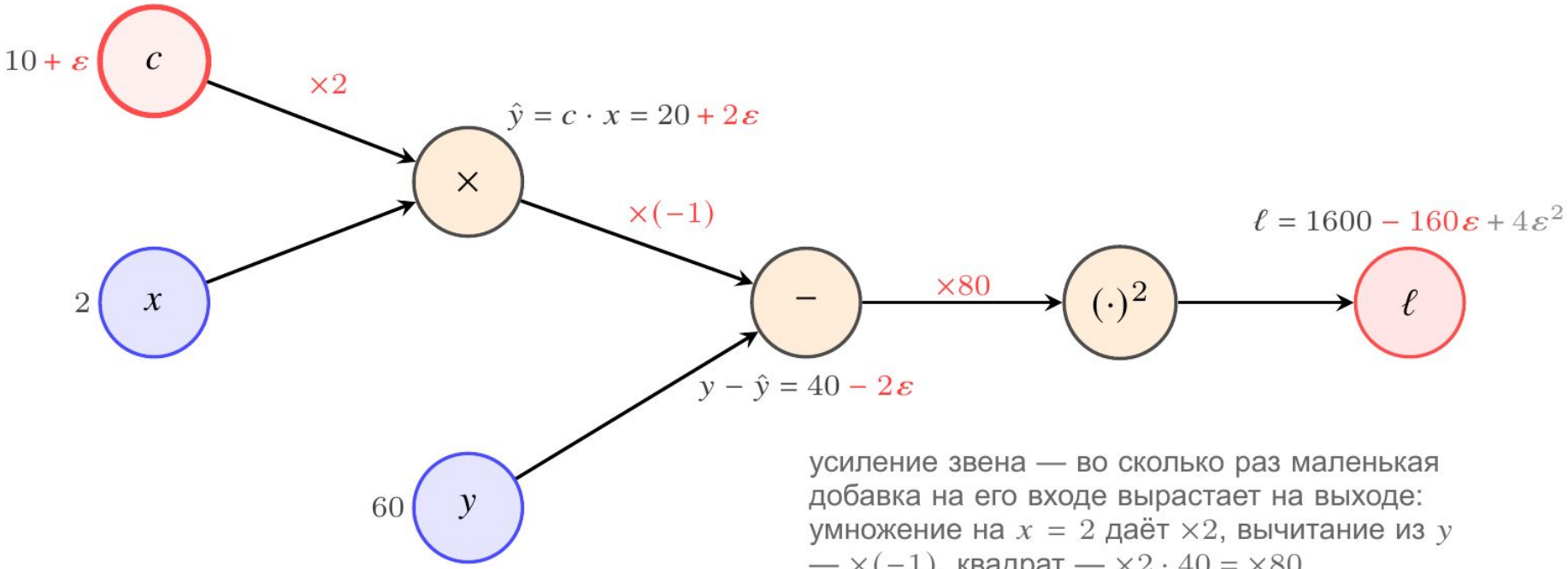


Как считать производные по графу



Дифференцирование сложной функции

- Но как подсчитать все эти производные?..
- Главный инструмент – правило дифференцирования сложной функции:



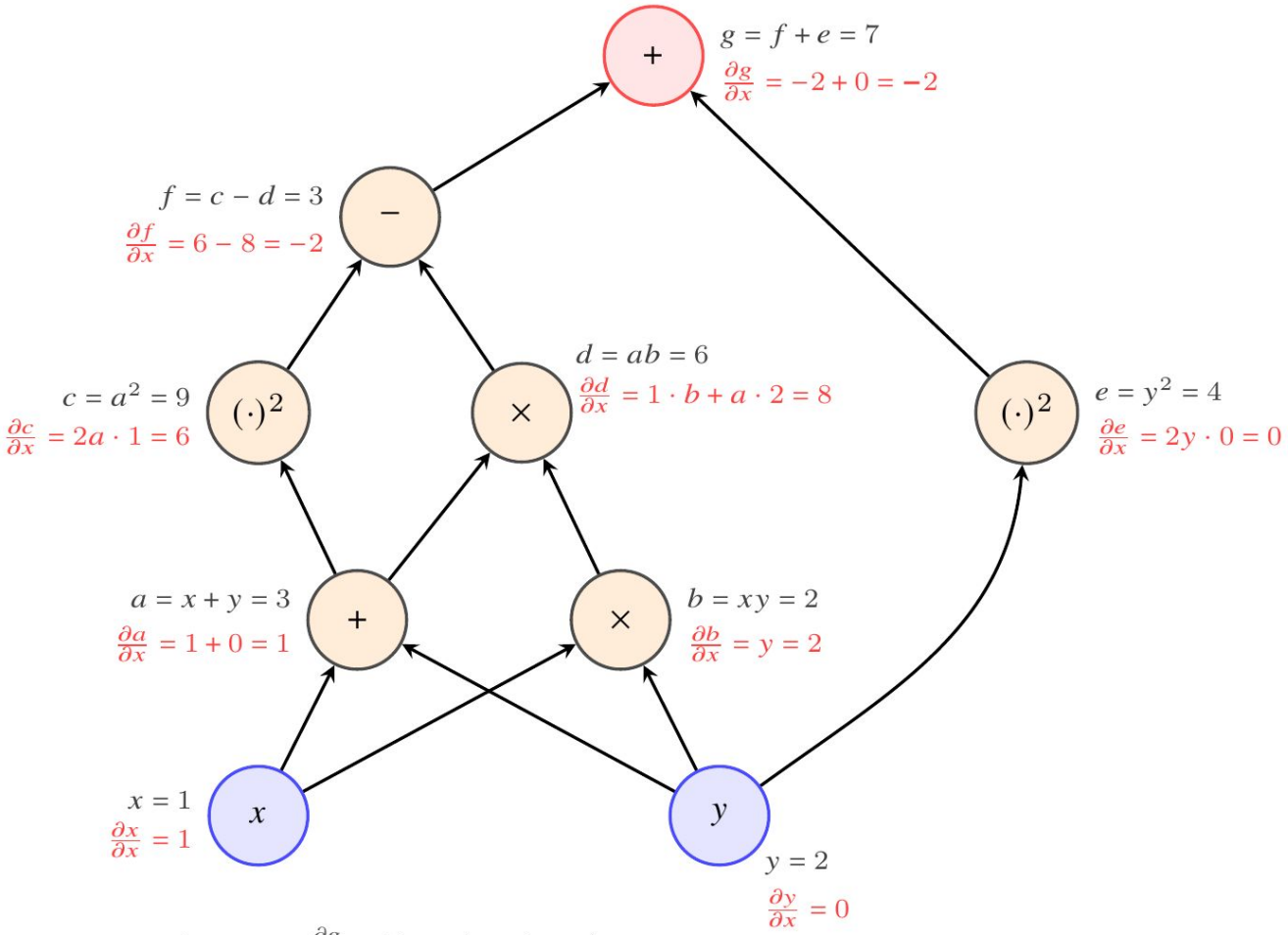
усиление звена — во сколько раз маленькая добавка на его входе вырастает на выходе: умножение на $x = 2$ даёт $\times 2$, вычитание из y — $\times(-1)$, квадрат — $\times 2 \cdot 40 = \times 80$

$$\frac{\partial \ell}{\partial c} = \frac{\partial \ell}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial c} = 2(\hat{y} - y) \cdot x = (-80) \cdot 2 = -160$$

усиления вдоль пути перемножаются — это и есть правило дифференцирования сложной функции; ε^2 ничтожно мало, производная — коэффициент при ε

Граф вычислений и прямой проход

$$g(x, y) = (x + y)^2 - xy(x + y) + y^2$$



проверка по формуле: $\frac{\partial g}{\partial x} = 2(x+y) - y(x+y) - xy = 6 - 6 - 2 = -2$.

Значение и производная считаются за один проход, но производная только по одному входу: для $\frac{\partial g}{\partial y}$ нужен ещё один проход

Южный федеральный университет



@SINECOR

Спасибо за внимание!

