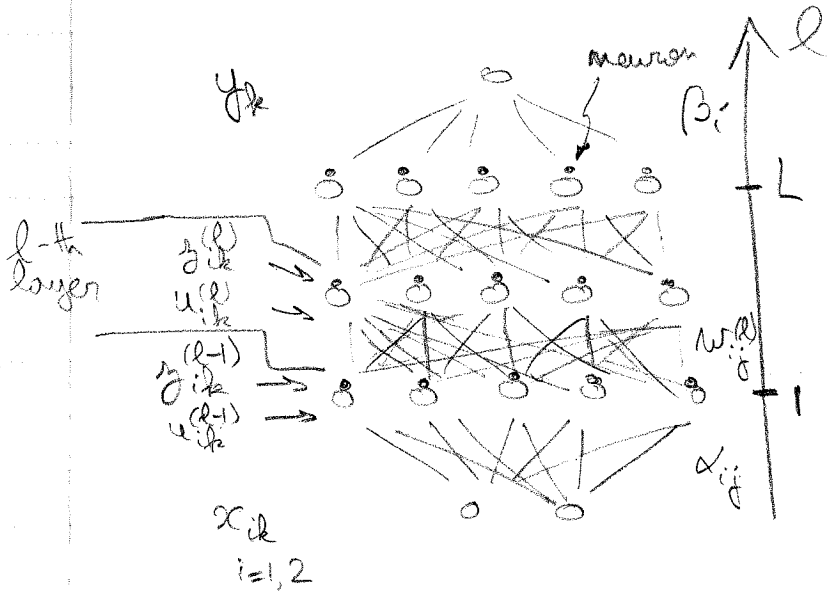
03/22/2018 Gradient descent

Ref. Nocedal, Wright, Numerical Optimization

$\min f(x), \quad f \in C^1$:

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

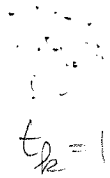
ex Neural network classifier



$k = 1, \dots, K$
sample index
$i = 1, \dots, N$
width

x_{ik}: input
$z_{ik}^{(l)}$: neuron output
y_k: output / class label
α, w, β: weights

y_k should match t_k.
→ tune α, w, β.



$t_k = 0$.

(K points overall)

$$\begin{cases} y_k = \sum_{i=1}^{m} \beta_i z_{ik}^{(L)} \\ z_{ik}^{(l)} = \varphi\left(\sum_{j=1}^{m/2} w_{ij}^{(l)} z_{jk}^{(l-1)} \right) \\ z_{ik}^{(1)} = \varphi\left(\sum_{j=1}^{N} \alpha_{ij} x_{jk} \right) \end{cases}$$

← the layers are linear, not affine

with $\varphi(x) = \dfrac{1}{1+e^{-x}}$ activation function

$$\varphi'(x) = \varphi(x)(1-\varphi(x)) =: \psi(x)$$

$E = \frac{1}{2}\sum_k (y_k - t_k)^2$ → min over α, w, β

→ need $\frac{\partial E}{\partial \beta_i}$, $\frac{\partial E}{\partial w_{ij}^{(l)}}$, $\frac{\partial E}{\partial \alpha_{ij}}$

Backpropagation: $l = L : -1 : 1$
(know sensitivities for all the neurons above you)

$$\frac{\partial E}{\partial y_k} = y_k - t_k.$$

$$\frac{\partial E}{\partial \beta_i} = \sum_k \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial \beta_i} = \sum_k \frac{\partial E}{\partial y_k} z_{ik}^{(L)}$$

$$\frac{\partial E}{\partial z_{ik}^{(l)}} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_{ik}^{(l)}} = \frac{\partial E}{\partial y_k}\beta_i$$

$w_{ij}^{(l)}$: i → j out

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = \sum_k \frac{\partial E}{\partial z_{jk}^{(l)}} \frac{\partial z_{jk}^{(l)}}{\partial u_{jk}^{(l)}} \frac{\partial u_{jk}^{(l)}}{\partial w_{ij}^{(l)}}$$

$\frac{\partial u_{jk}^{(l)}}{\partial w_{ij}^{(l)}} \to z_{ik}^{(l-1)}$

$\frac{\partial z_{jk}^{(l)}}{\partial u_{jk}^{(l)}} \to \psi(u_{jk}^{(l)})$

$$\frac{\partial E}{\partial z_{ik}^{(l-1)}} = \sum_{j=1}^{m} \frac{\partial E}{\partial z_{jk}^{(l)}} \frac{\partial z_{jk}^{(l)}}{\partial z_{ik}^{(l-1)}}$$

$\frac{\partial z_{jk}^{(l)}}{\partial z_{ik}^{(l-1)}} \to \psi(u_{ik}^{(l)})\, w_{ij}^{(l)}$

α_{ij}: i → j out

$$\frac{\partial E}{\partial \alpha_{ij}} = \sum_k \frac{\partial E}{\partial z_{jk}^{(l)}} \frac{\partial z_{jk}^{(l)}}{\partial u_{jk}^{(l)}} \frac{\partial u_{jk}^{(l)}}{\partial \alpha_{ij}}$$

$\frac{\partial u_{jk}^{(l)}}{\partial \alpha_{ij}} \to x_{ik}$

$\frac{\partial z_{jk}^{(l)}}{\partial u_{jk}^{(l)}} \to \psi(u_{jk}^{(l)})$

GD: $$\begin{bmatrix}\alpha \\ w \\ \beta\end{bmatrix} \leftarrow \begin{bmatrix}\alpha \\ w \\ \beta\end{bmatrix} - \eta \nabla_{(\alpha, w, \beta)} E\left(\begin{bmatrix}\alpha \\ w \\ \beta\end{bmatrix}\right)$$

for some adequate η
OK if converges to a local minimizer.

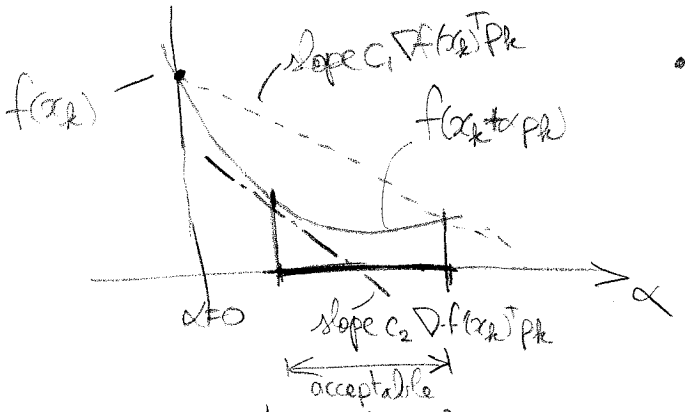
SGD: instead of $k = 1, \dots, K$
pick $k \in S$
↳ drawn at random
called minibatch.
with or without replacement.
cheaper by a factor $K/|S|$, ideally
→ can do more iterations.
stochastic interpretation:
$\mathbb{E}_S \nabla E|_S = \nabla E$.

also called KACZMARZ method.

Step length for GD: Wolfe conditions,
line search method

$\min f(x)$, have p_k descent direction
(e.g. $p_k = -\nabla f(x_k)$)

Consider $f(x_k + \alpha p_k)$.



- Descent: slope
$\frac{\partial f}{\partial \alpha}(x_k + \alpha p_k)\Big|_{\alpha=0} =$
$\nabla f(x_k)^T p_k < 0$
→ need acute angle between p_k and $-\nabla f(x_k)$

- Don't go too far: stay below dashed line.
$f(x_k + \alpha p_k) \le f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k$ (1)
for some $c_1 \in (0,1)$
(Armijo condition, sufficient decrease

- Don't stay too close:

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k \qquad (2)$$

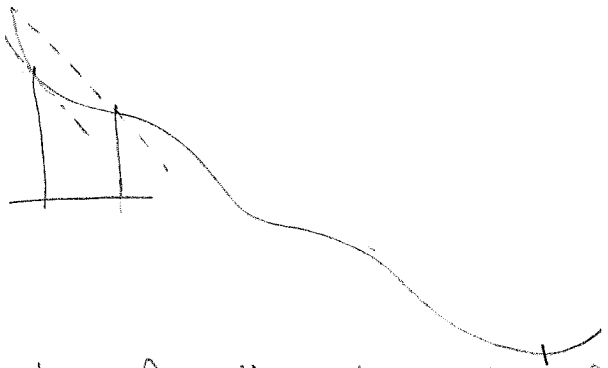
for some $c_2 \in (c_1, 1)$

→ greater negative slope at α than initially, i.e., flatter graph.

(curvature condition)

(1), (2): Wolfe conditions

no guarantee to be close to a minimum



- Another option for "don't stay too close": Goldstein condition

$$f(x_k) + (1-c_1)\alpha \nabla f(x_k)^T p_k \leq f(x_k + \alpha p_k)$$

with $c_1 \in (0, \frac{1}{2})$.

Step length selection:

- start with a large α like $\alpha_{prev} \frac{\nabla f(x_{k-1})^T p_{k-1}}{\nabla f(x_k)^T p_k}$

(so the first-order change in x is the same as that obtained at the previous iteration)

- decrease slowly until (1) is satisfied, e.g. form a quadratic from $f(x_{k-1})$, $f(x_k)$, and $\nabla f(x_k)^T p_k$, then minimize it.