Neural Networks Optimization

EE698V - Machine Learning for Signal Processing

Vipul Arora

Optimization Schemes

References:

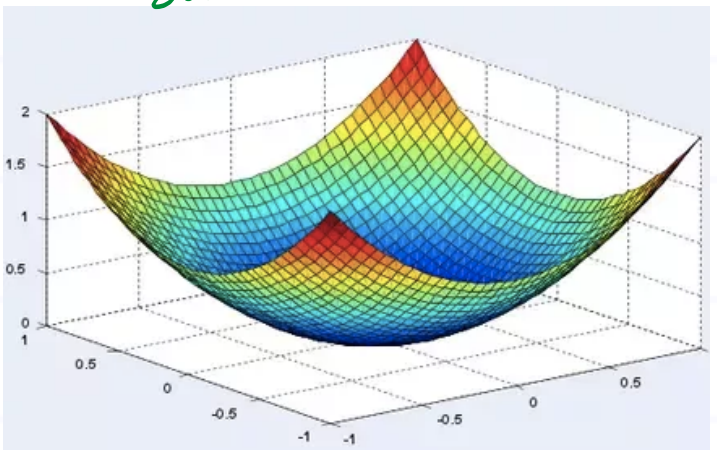
- Hinton's slides: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
- Hinton's video lectures: 6a, 6b, 6c, 6d, 6e

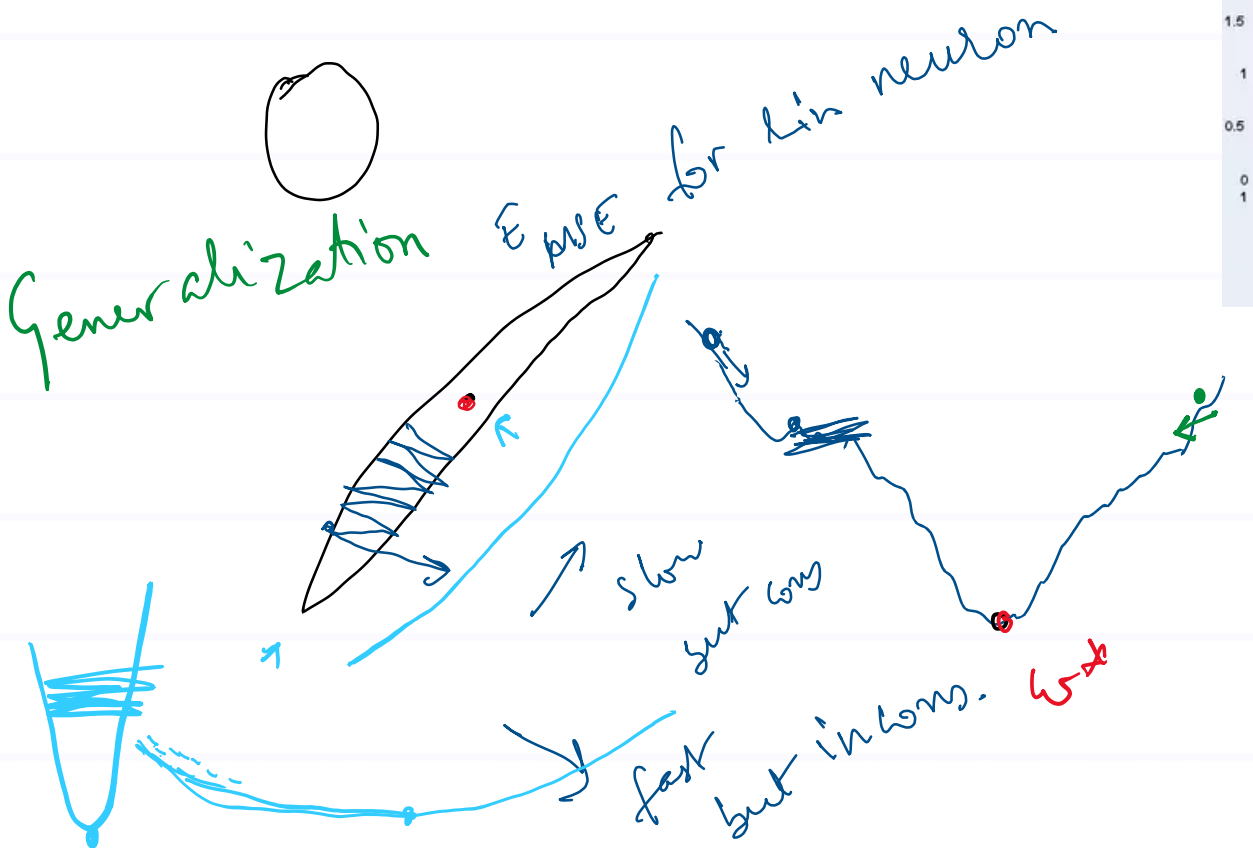
Optimization Schemes

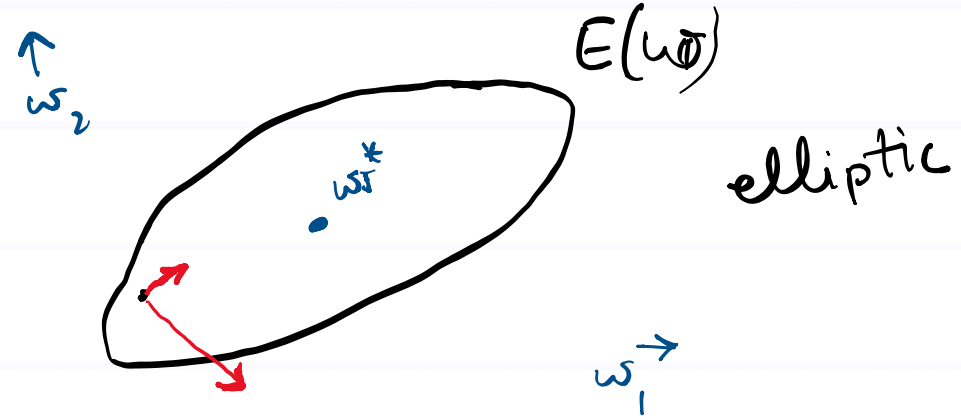
$$E = \|t - y(x, w)\|^2$$
$$= 2(t, y(x, w))$$

- Gradient Descent

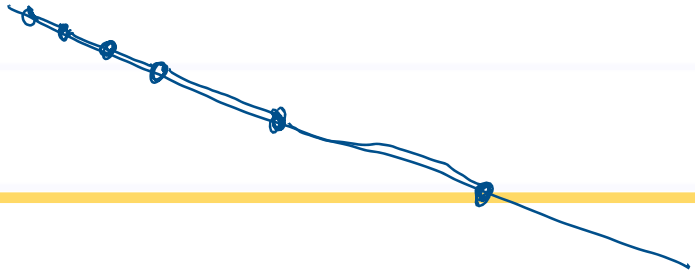
$$w \leftarrow w - \eta \frac{\partial E}{\partial w}$$



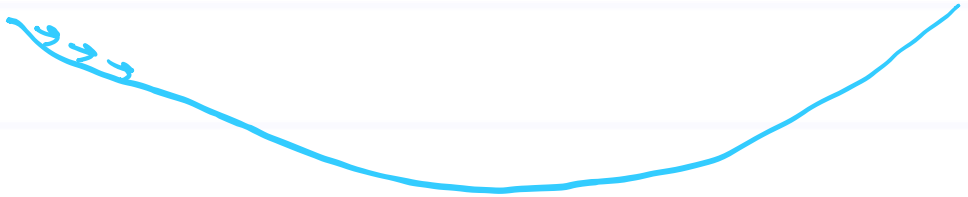




Learning Rate



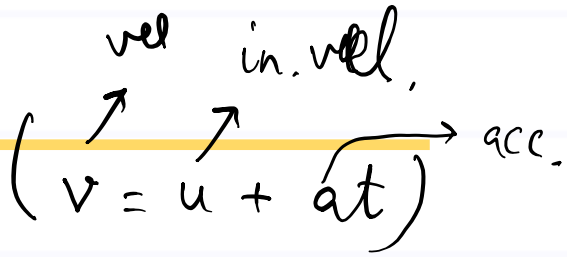
- Move quickly in directions with small but consistent gradients.



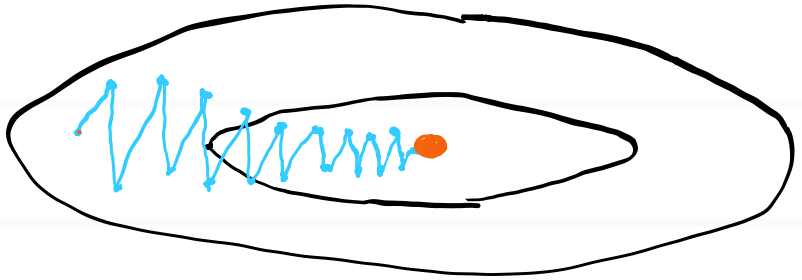
- Move slowly in directions with big but inconsistent gradients.

- Towards the end of learning, reduce the learning rate

Momentum in Gradient Descent



$$v \Leftarrow \beta v + (1-\beta)\,\Delta w$$



$$W = W - \underbrace{\alpha v}_{\Delta w}$$

$$s = ut + \frac{1}{2}at^2$$

RmsProp

- DIY
- References:
  - https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b
  - Hinton's slides: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
  - Hinton's video lectures: 6a, 6b, 6c, 6d, 6e

leaky ReLU $\alpha << 1$ $y=\alpha x$ $y=x$

Hyperparameter Tuning

- model arch. - layers
  - neurons/l
  - η
  - λ

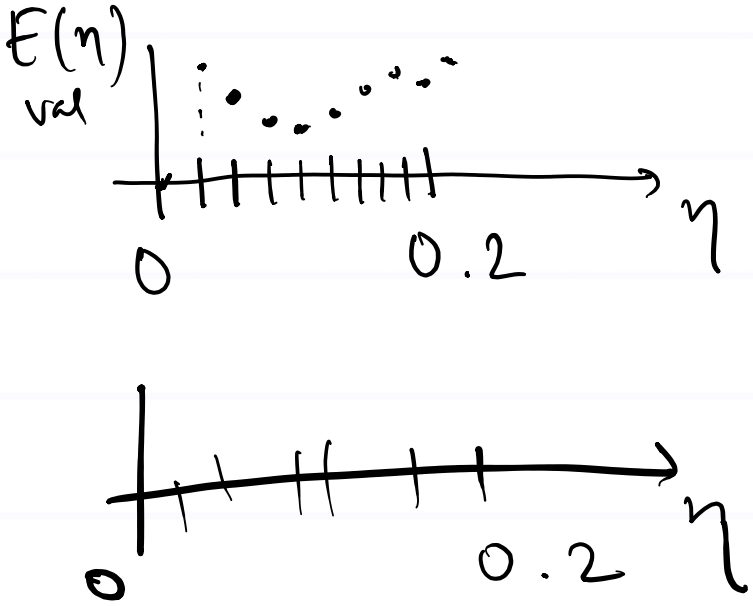
any hyperparam.

- How to find the best learning rate?
- Do not optimize it on the test set and do not overfit the training set
- Thus, use validation set
- **Reference**:
  - DeepLearningBook – Section 7.8

E_{val}	hyp.	E_{tr}	E_{test}
0.49	λ_1	0.3	0.5
0.61	λ_2	0.4	0.6
0.57	λ_3	0.2	0.52

Hyperparameter Search

- Grid search vs random search



Grid search

$\eta \sim \text{unif}(0, 0.2)$

Validation

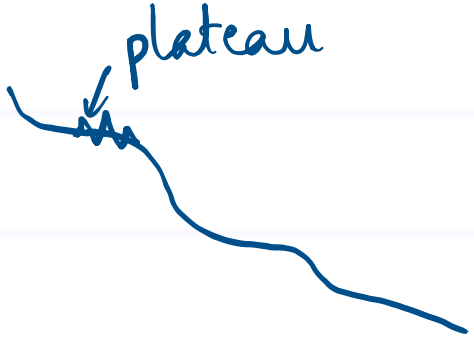
- For efficient implementation
  - use separate machine
  - reduce data size
  - reduce frequency of checks

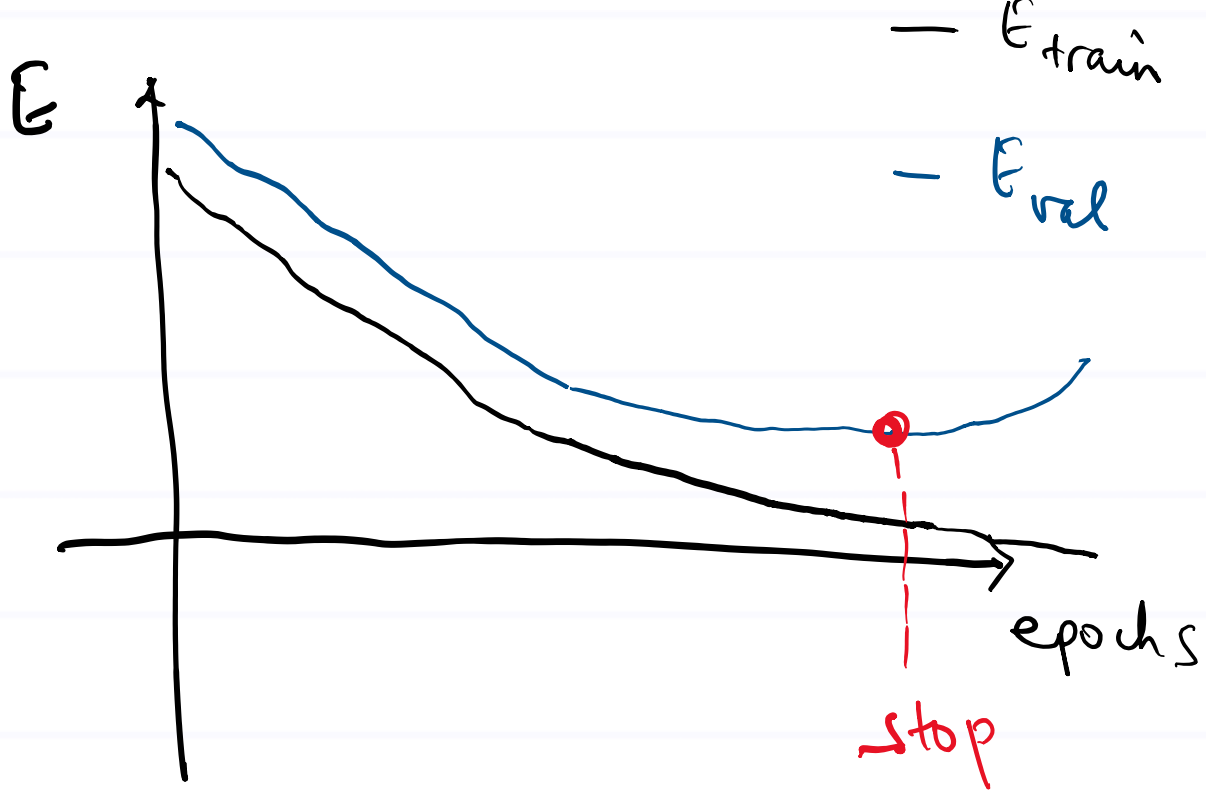
Epoch → seeing all training samples once

Stopping criterion

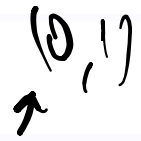
- When training error stops decreasing (threshold)
- Monitor N epochs

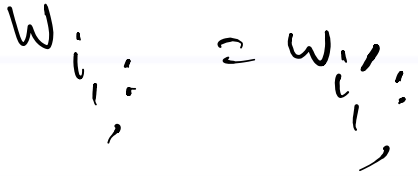
$$|\Delta(\varepsilon_{val})| < \theta$$

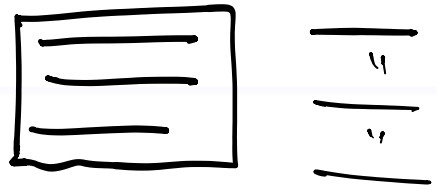




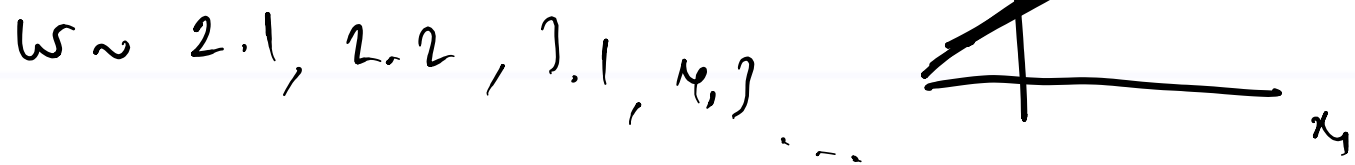
Weight Initialization







- Random weights
  - Same weights will update in the same way and will remain same
- Small weights
  - more fan-in => overshoot in training

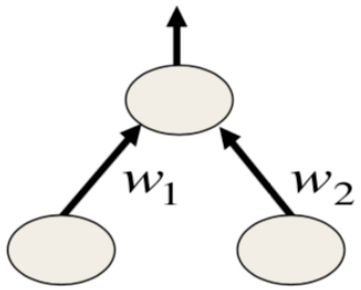


Data Normalization

- **Shifting** of data helps
  - make it zero mean
  - $x_{i,s} \leftarrow x_{i,s} - \mu_i$

$w_1 101 + w_2 101 = 2$

$w_1 101 + w_2 99 = 0$



color indicates training case

x t

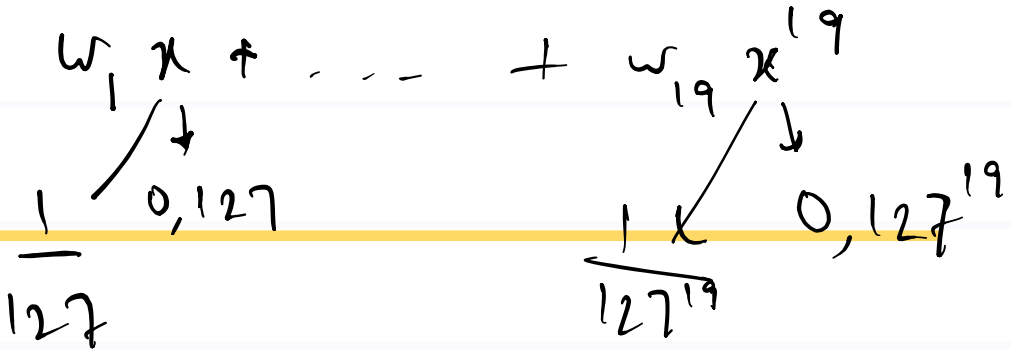
101, 101 → 2 gives error
101, 99 → 0 surface

−∞ −∞

1, 1 → 2 gives error
1, -1 → 0 surface

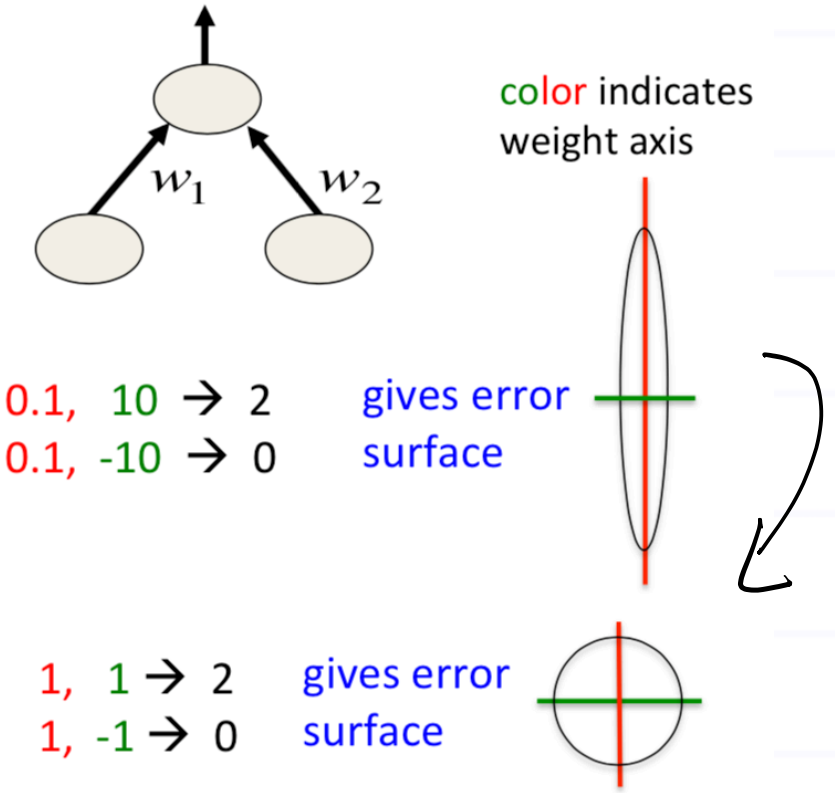
slopes are slightly diff.

Data Normalization



- **Scaling** of data helps
  - different variance of different dimensions
  - make each dim unit variance
  - $x_{i,s} \leftarrow x_{i,s}/\sigma_i$

Overall, $x_{i,s} \leftarrow (x_{i,s} - \mu_i)/\sigma_i$

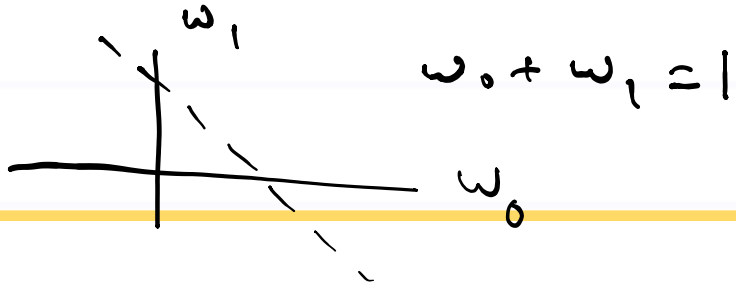


Regularization

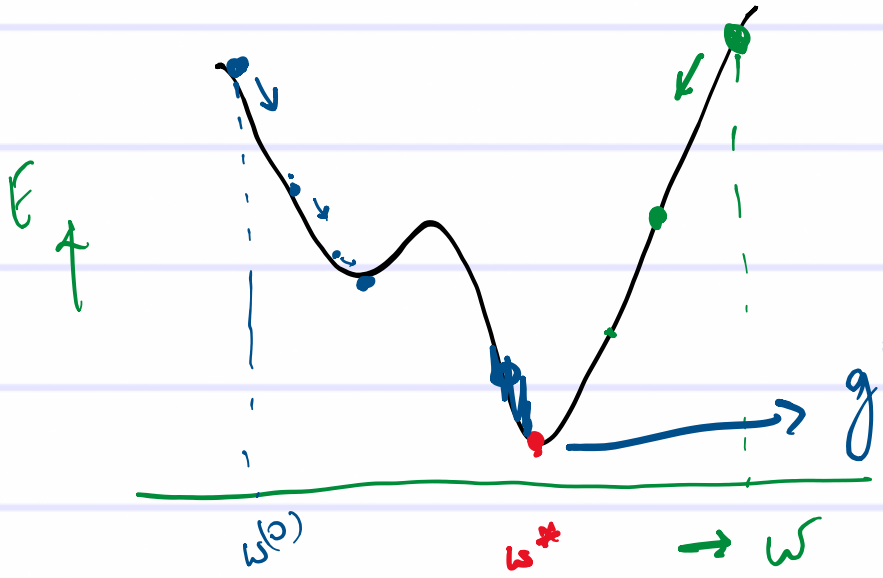
References:

- Deep Learning Book, Chapter 7

Regularization



- Constrain weights
- Prior knowledge injection
- Prefer simpler model classes
- Make underdetermined problems determined
- Ensemble methods



$\frac{\partial E}{\partial w} = 0$ local min.

- $w^{(0)}$
- 10 samples - find Δw
- $w^{(1)} \leftarrow w^{(0)} + \Delta w$

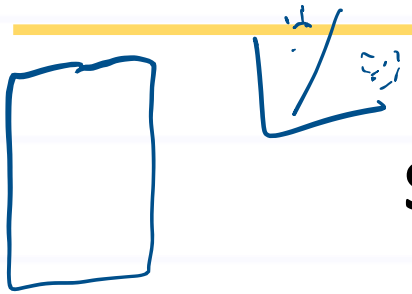
- $w^{(0)}$
- 5 samples randomly, find Δw
- $w^{(0,1)} \leftarrow w^{(0)} + \Delta w$
- 5 samples

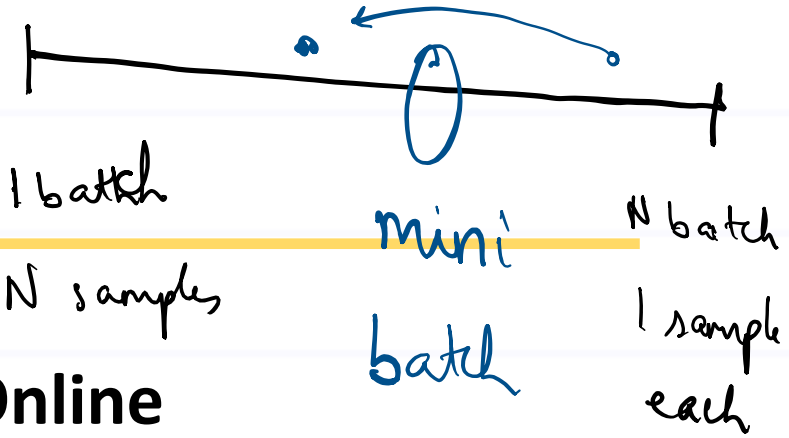
10 samples

5 → $w^{(1)} \leftarrow w^{(0,1)} + \Delta w$

5

Stochastic Gradient Descent





Single batch

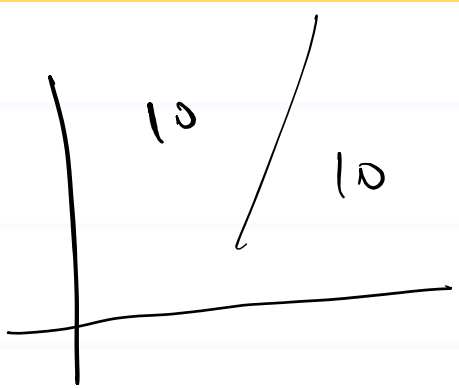
- Computationally efficient (matrix, GPU) and takes less time for one epoch
- Redundancy: less improvement in one epoch
- Stable at the minima

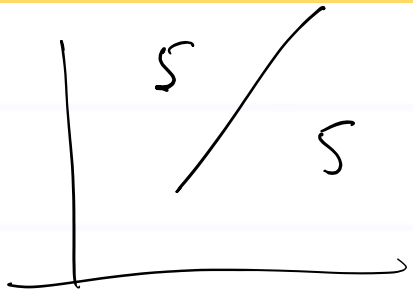
Online

- Computationally inefficient and time consuming for one epoch
- Each sample improves weights
- Gets too noisy near the minima

Thus, use randomly sampled **mini-batches**

One strategy: Start with small batch size and increase it with epochs

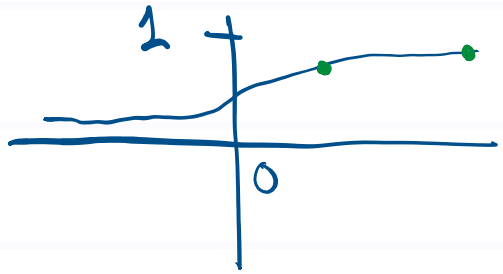




Online

Adding Noise improves Generalization

- Output Noise
  - $\{0,1\} \rightarrow \left\{\frac{\epsilon}{1-K}, 1-\epsilon\right\}$ Label smoothing

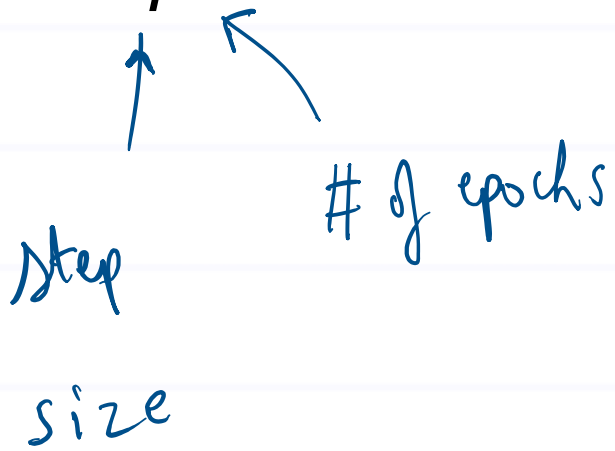


Adding Noise improves Generalization

- Noise in weights
  - Add a gaussian noise to weights
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} + \epsilon_{\boldsymbol{w}}$ where $\epsilon_{\boldsymbol{w}} \sim \mathcal{N}(0, \sigma^2 I)$
  - This is equivalent to adding a regularization term to the loss function

Early Stopping

- Limits the search region volume to $\propto \eta\tau$



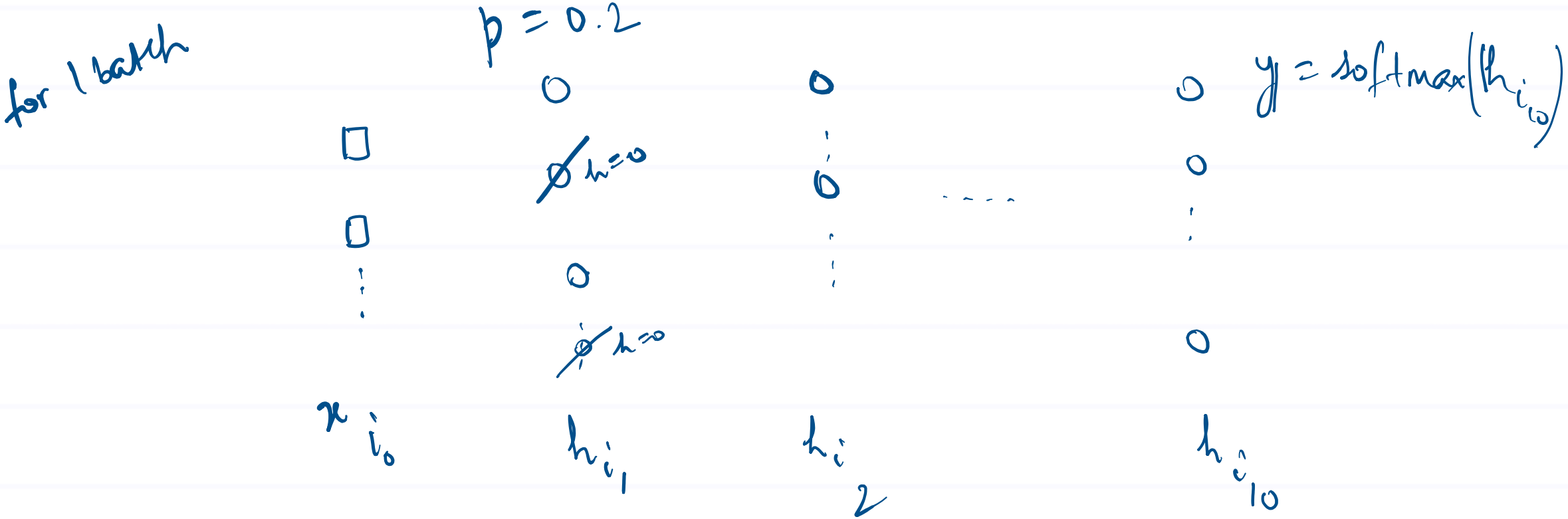
Ensemble models or Bagging

- Final output is the average of many models
- Not all models make same errors on the test set

Dropout

forward pass

- Mask neurons randomly



Other References

- https://www.deeplearningbook.org/contents/optimization.html

Keras library for Python

- https://keras.io/
- https://keras.io/#getting-started-30-seconds-to-keras

Announcements

- Assignment 3 deadline extended by 1 day
- Quiz 2 next week (Fri or Sat?)
- Quiz 1 marks today
  - Copying cases caught
    - will deduct 5% marks from total
    - more severe penalty next time
- Mid sem exam will be written as well as coding based