Batch normalization

ISTD 50.035

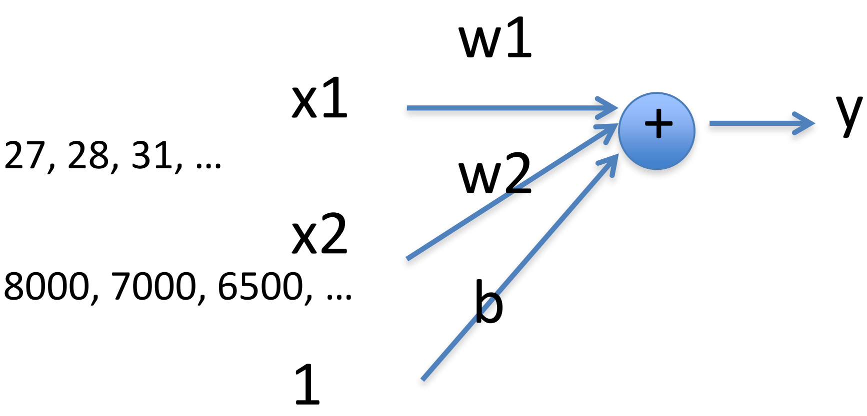
Computer Vision

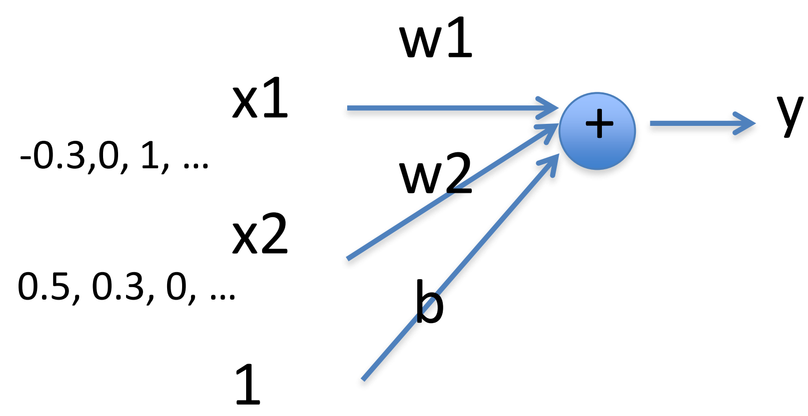
Acknowledgement: Some images are from various sources: UCF, Stanford cs231n, etc.

Batch normalization

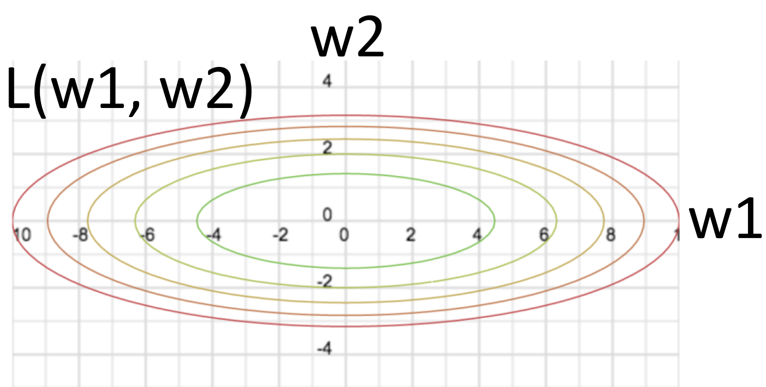
- Batch normalization (Sergey Ioffe, Christian Szegedy; 2015) enables the use of higher learning rates and accelerates the learning process
  - Converges with only 7% of the training steps compared to previous work
- Not an optimization
- Other normalization techniques have proposed subsequently

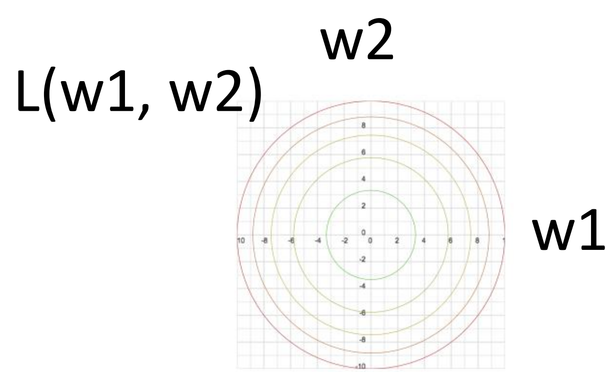
Feature scaling





Example: x1=age; x2=salary; y=apartment size
Desire to scale the feature to the same range

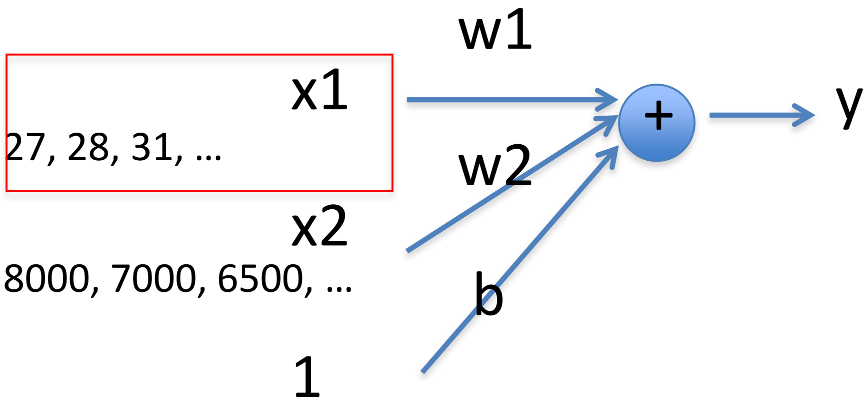


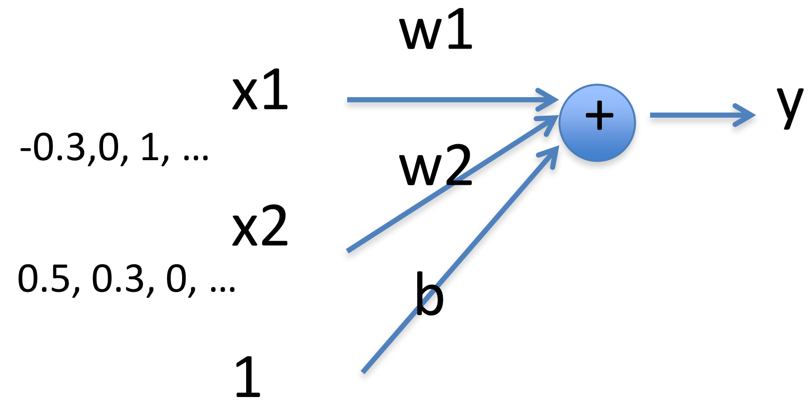


If x2 is large, then small change in w2 will have large change in y, and hence L

Larger step size can be used here

Feature scaling





For each feature (dimension), compute mean and standard deviation

$$x_i^{(r)} := \frac{x_i^{(r)} - \mu_i}{\sigma_i}$$

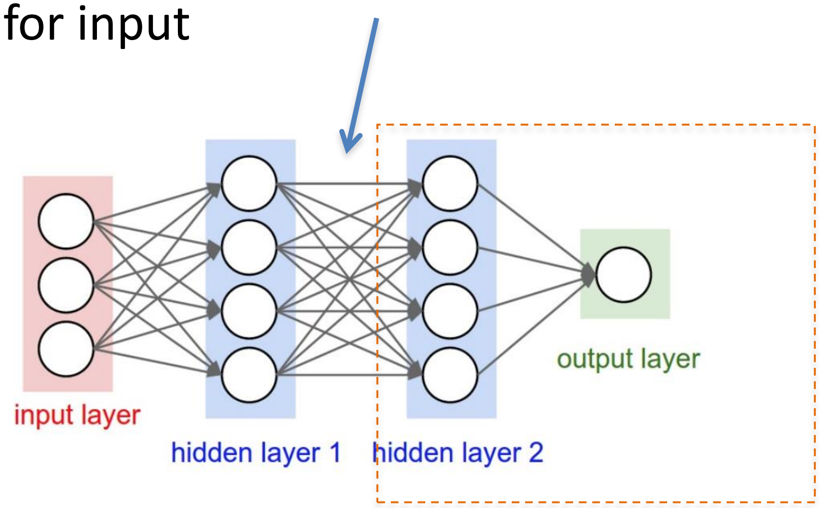
For i-th dim

The means of all dimensions become 0, variances become 1

Feature scaling in DNN

Feature scaling for input

Feature scaling for input into layer 2, i.e. output of layer 1



Batch Norm

m values of an activation in the mini-batch

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

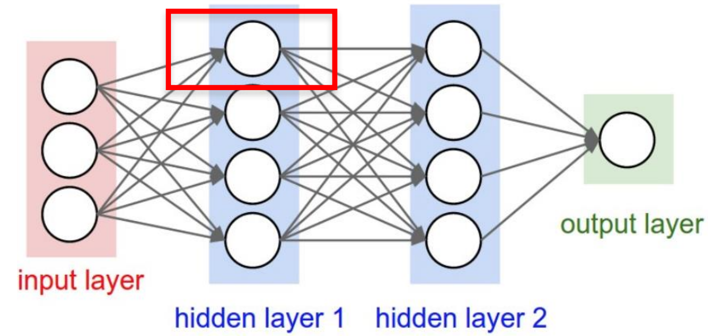
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



average of a mini-batch, i.e. average across training examples

Apply to each activation (input) independently

Batch Norm

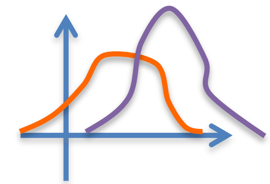
- Apply BN before non-linear, usually

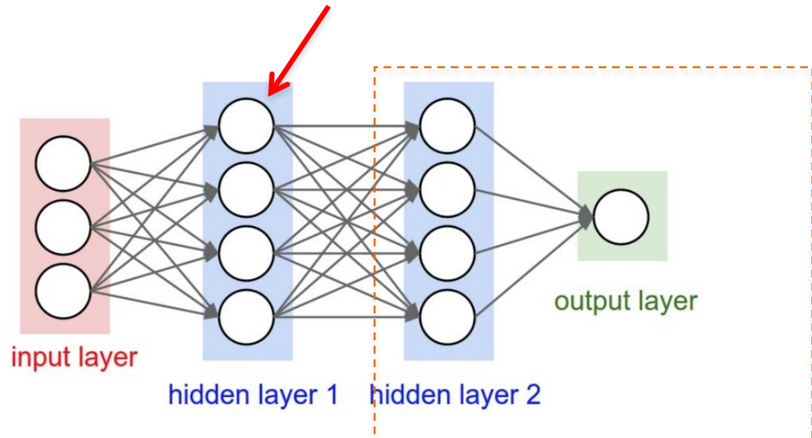
- During testing, replace mini-batch statistics with population statistics (computed by moving average during training)
  - Mini-batch means -> population means
  - Mini-batch variances -> population variances

- Therefore, means and variances are fixed during inference: normalization is a linear transform applied to each activation

BN reduces internal covariate shift

Distribution of this activation during training (without BN):

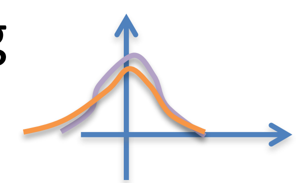




t-th mini-batch

$(t+1)$-th mini-batch

Distribution of this activation during training (with BN):



Covariate shift

- Covariate shift: Change in the distribution of the input values to a learning algorithm
- The learning algorithm may perform differently when the input distribution changes
- **Deep Neural Networks: Internal covariate shift**
  - Change in input distribution to the **inner hidden units** within the network
  - Training: Weights of each layer are updated -> activations of each layer change
  - As activations are inputs to the next layer -> input distribution to the inner hidden units changes with each step during training