Gradient Descent

ISTD 50.035

Computer Vision

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

Linear Classifier

$$s = f(x; W, b) = Wx + b$$

- **Testing**: W, b are fixed, x is the input
- **Training**: Given N training samples (x_i, y_i), y_i takes value in [1,...,K], learn W and b

Training: (x_i, y_i) are given and fixed; W, b are the variables to be determined

Learn W using loss function L(W)

- Try different W (randomly), choose the one with the min loss function

$$L = \frac{1}{N}\sum_i L_i + \lambda R(W)$$

N training samples

- W is very large: Kx(D+1)
- Even larger in deep neural network

- Start from a random W, iteratively improve W (reduce L(W)): Gradient descent

- Note: $L(W) = L(W; (x_1,y_1), (x_2,y_2), \ldots (x_i,y_i) \ldots (x_N,y_N))$

Learn W by gradient descent

- Update W by W+ΔW, using the gradient

$$L(W) = L(w_1, w_2, ...w_l...)$$

$$W' = W - \gamma \nabla L$$

$$\nabla L = [\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, ...\frac{\partial L}{\partial w_l}, ...]^T$$

$$w_l' = w_l - \gamma \frac{\partial L}{\partial w_l}$$

Gradient descent

γ: step size (learning rate), a hyperparameter

Learn W by gradient descent

- Update W by W+ΔW, using the gradient

$$w_l' = w_l - \gamma \frac{\partial L}{\partial w_l}$$

$$L = \frac{1}{N} \sum_i L_i + \lambda R(W)$$

$$\frac{\partial L}{\partial w_l} = \frac{1}{N} \sum_i \frac{\partial L_i}{\partial w_l} + \lambda \frac{\partial R(W)}{\partial w_l}$$

-Sum gradients for all (partial) training samples for one w_l
-Make one update of W once we have the whole gradient vector (dim: Kx(D+1))

Learn W by gradient descent

- Update W by W+ΔW, using the gradient

$$\Delta W = -\gamma \left[\qquad \right]$$

$$\Delta W = -\gamma \nabla L$$

$$\frac{\partial L}{\partial w_l} = \frac{1}{N} \sum_i \frac{\partial L_i}{\partial w_l} + \lambda \frac{\partial R(W)}{\partial w_l}$$

-Sum gradients for all (partial) training samples for one w_l
-Make one update of W once we have the whole gradient vector (dim: Kx(D+1))

Gradient of one training sample: SVM loss

- SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + d)$$

$$L_i = \sum_{j \neq y_i} \max(0, \mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d)$$

- $\mathbf{w}_j$: j-th row of W
- Loss function of one training sample: $L_i(W; (x_i, y_i))$

Gradient of one training sample

- SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, \mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d)$$

- For row j, $\mathbf{w_j}$, j = y_i

$$\nabla_{\mathbf{w}_j} L_i = - \left[\sum_{j \neq y_i} \mathbf{I}(\mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d > 0) \right] x_i$$

- For row j, $\mathbf{w_j}$, j <> y_i

$$\nabla_{\mathbf{w}_j} L_i = \mathbf{I}(\mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d > 0) x_i$$

$\mathbf{I}$(cond) = 1 if cond is true, 0 otherwise

Gradient of one training sample

- SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, \mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d)$$

- Justification:

For a specific j, only 1 term between w_j and w_{yi}; if non-zero, gradient = x_i

- For row j, $\mathbf{w_j}$, j <> y_i

$$\nabla_{\mathbf{w}_j} L_i = \mathbf{I}(\mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d > 0) x_i$$

$\mathbf{I}$(cond) = 1 if cond is true, 0 otherwise

Gradient of one training sample

- SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, \mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d)$$

- For row j, $\mathbf{w_j}$, j = y_i

$$\nabla_{\mathbf{w}_j} L_i = -\left[\sum_{j \neq y_i} \mathbf{I}(\mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d > 0)\right] x_i$$

- Justification:

All the (K-1) terms of L_i involve w_{yi}; some are zero; for non-zero one, grad = $-x_i$

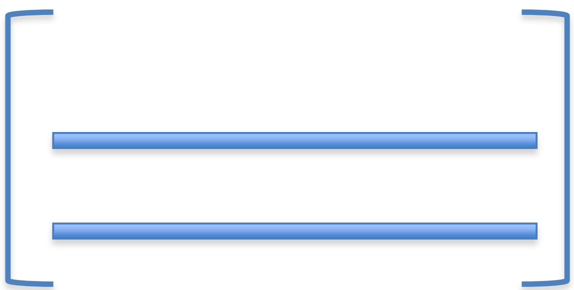
$\mathbf{I}$(cond) = 1 if cond is true, 0 otherwise

Learn W by gradient descent

- Update W by W+ΔW, using the gradient

$$\nabla_{\mathbf{w}_j} L_i = -\left[\sum_{j \neq y_i} \mathbf{I}(\mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d > 0)\right] x_i$$

dim: D+1

ΔW=-γ 

i.e. $-c\, x_i$

c is the number of terms with loss

$$\nabla_{\mathbf{w}_j} L_i = \mathbf{I}(\mathbf{w}_j^T x_i - \mathbf{w}_{y_i}^T x_i + d > 0) x_i$$

dim: D+1

i.e. 0 (if no loss due to w_j) or x_i

$$\frac{\partial L}{\partial w_l} = \frac{1}{N} \sum_i \frac{\partial L_i}{\partial w_l} + \lambda \frac{\partial R(W)}{\partial w_l}$$

Learn W by gradient descent

- Mini-batch gradient descent / stochastic gradient descent: use small batch (64, 128, 256) for one update of W

$$\frac{\partial L}{\partial w_l} = \frac{1}{N_{batch}} \sum_i \frac{\partial L_i}{\partial w_l} + \lambda \frac{\partial R(W)}{\partial w_l}$$

- Random sampling without replacement

- Mini-batch: average for each update of W
- An epoch: go through the entire training dataset (multiple updates of W)

Gradient of one training sample: Cross-entropy loss

- Cross-entropy loss

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

- Or, p_m is the probability of the m-th class (output of softmax function)

$$p_m = \frac{e^{f_m}}{\sum_{j=1}^{K} e^{f_j}}$$

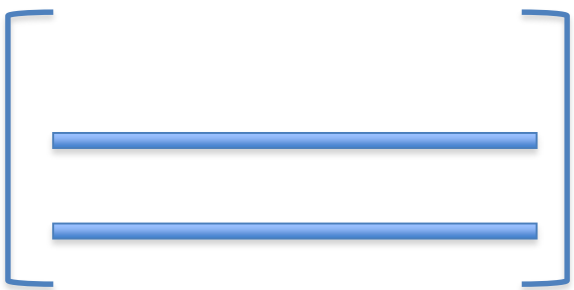
- Then

$$L_i = -\log p_{y_i}$$

Gradient (linear classifier, cross-entropy loss)

- Gradient matrix (for updating W):

$$\nabla_W L_i =$$



$$\frac{\partial L_i}{\partial \mathbf{w}_m} = p_m x_i \qquad m \neq y_i$$

dim: D+1

$$\frac{\partial L_i}{\partial \mathbf{w}_m} = (p_{y_i} - 1) x_i \quad m = y_i$$

dim: D+1

See derivation