CS4501: Introduction to Computer Vision

Softmax Classifier

Various slides from previous courses by:

D.A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL). S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook) . J. M. Frahm (UNC), V. Ordonez (UVA), Steve Seitz (UW).

Previous Class

- Introduction to Machine Learning
- Ethics in Machine Learning: Choosing Problems in ML
- Supervised vs Unsupervised ML
- K-Nearest Neighbors Classification
- K-Means Clustering

Today's Class

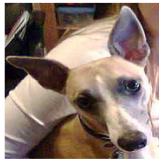
- Softmax Classifier
  - Inference vs Training
  - Gradient Descent (GD)
  - Stochastic Gradient Descent (SGD)
  - mini-batch Stochastic Gradient Descent (SGD)
- Regularization
- Max-Margin Classifier

Supervised Learning - Classification

Training Data

cat

dog

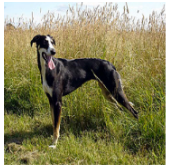


bear

cat

cat

dog

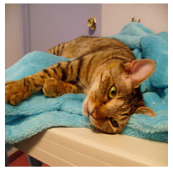


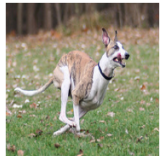
bear

dog

bear

Test Data





Supervised Learning - Classification

Training Data

cat

dog

cat

.
.
.

bear

Test Data

.
.
.

Supervised Learning - Classification

Training Data

$x_1 = [$ $]$ $y_1 = [\text{cat} \quad]$

$x_2 = [$ $]$ $y_2 = [\text{dog} \quad]$

$x_3 = [$ $]$ $y_3 = [\text{cat} \quad]$

.
.
.

$x_n = [$ $]$ $y_n = [\text{bear} \quad]$

Supervised Learning - Classification

Training Data

inputs

targets / labels / ground truth

predictions

$x_1 = [x_{11} \quad x_{12} \quad x_{13} \quad x_{14}]$ $y_1 = 1$ $\hat{y}_1 = 1$

$x_2 = [x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]$ $y_2 = 2$ $\hat{y}_2 = 2$

$x_3 = [x_{31} \quad x_{32} \quad x_{33} \quad x_{34}]$ $y_3 = 1$ $\hat{y}_3 = 2$

.
.
.

$x_n = [x_{n1} \quad x_{n2} \quad x_{n3} \quad x_{n4}]$ $y_n = 3$ $\hat{y}_n = 1$

We need to find a function that maps *x* and *y* for any of them.

$$\hat{y}_i = f(x_i; \theta)$$

How do we ”learn” the parameters of this function?

We choose ones that makes the following quantity small:

$$\sum_{i=1}^{n} Cost(\hat{y}_i, y_i)$$

Supervised Learning – Linear Softmax

Training Data

inputs

targets / labels / ground truth

$x_1 = [x_{11} \quad x_{12} \quad x_{13} \quad x_{14}]$ $y_1 = 1$

$x_2 = [x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]$ $y_2 = 2$

$x_3 = [x_{31} \quad x_{32} \quad x_{33} \quad x_{34}]$ $y_3 = 1$

.
.
.

$x_n = [x_{n1} \quad x_{n2} \quad x_{n3} \quad x_{n4}]$ $y_n = 3$

Supervised Learning – Linear Softmax

Training Data

inputs

targets / labels / ground truth

predictions

$x_1 = [x_{11} \quad x_{12} \quad x_{13} \quad x_{14}]$ $y_1 = [1 \quad 0 \quad 0]$ $\hat{y}_1 = [0.85 \quad 0.10 \quad 0.05]$

$x_2 = [x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]$ $y_2 = [0 \quad 1 \quad 0]$ $\hat{y}_2 = [0.20 \quad 0.70 \quad 0.10]$

$x_3 = [x_{31} \quad x_{32} \quad x_{33} \quad x_{34}]$ $y_3 = [1 \quad 0 \quad 0]$ $\hat{y}_3 = [0.40 \quad 0.45 \quad 0.15]$

.
.
.

$x_n = [x_{n1} \quad x_{n2} \quad x_{n3} \quad x_{n4}]$ $y_n = [0 \quad 0 \quad 1]$ $\hat{y}_n = [0.40 \quad 0.25 \quad 0.35]$

Supervised Learning – Linear Softmax

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $\quad y_i = [1 \quad 0 \quad 0]$ $\quad \hat{y}_i = [f_c \quad f_d \quad f_b]$

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$
$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$
$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$f_c = e^{g_c}/(e^{g_c}+e^{g_d} + e^{g_b})$$
$$f_d = e^{g_d}/(e^{g_c}+e^{g_d} + e^{g_b})$$
$$f_b = e^{g_b}/(e^{g_c}+e^{g_d} + e^{g_b})$$

How do we find a good w and b?

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $\quad y_i = [1 \quad 0 \quad 0]$ $\quad \hat{y}_i = [f_c(w,b) \quad f_d(w,b) \quad f_b(w,b)]$

We need to find w, and b that minimize the following:

$$L(w,b) = \sum_{i=1}^{n}\sum_{j=1}^{3} -y_{i,j}\log(\hat{y}_{i,j}) = \sum_{i=1}^{n} -\log(\hat{y}_{i,label}) = \sum_{i=1}^{n} -\log f_{i,label}(w,b)$$

Why?

Gradient Descent (GD)

$\lambda = 0.01$

Initialize w and b randomly

$$L(w,b) = \sum_{i=1}^{n} -\log f_{i,label}(w,b)$$

for e = 0, num_epochs **do**

Compute: $dL(w,b)/dw$ and $dL(w,b)/db$

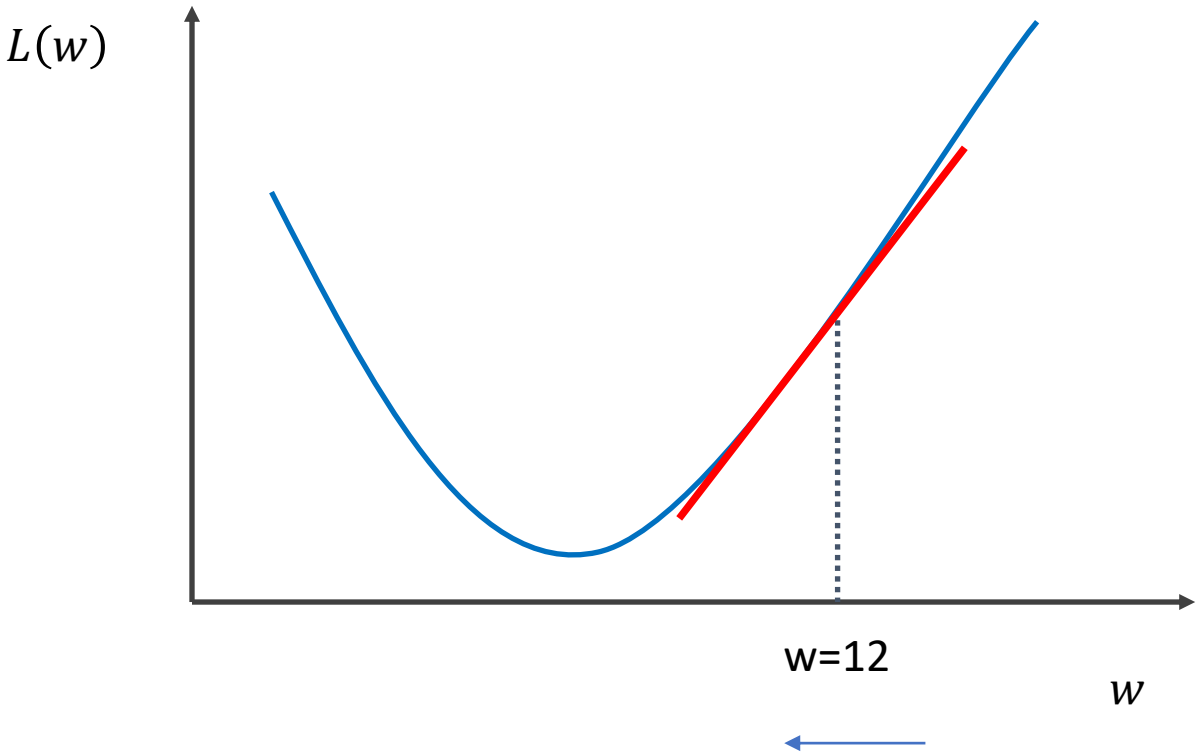
Update w: $w = w - \lambda\, dL(w,b)/dw$

Update b: $b = b - \lambda\, dL(w,b)/db$

Print: $L(w,b)$ // Useful to see if this is becoming smaller or not.

end

Gradient Descent (GD) (idea)



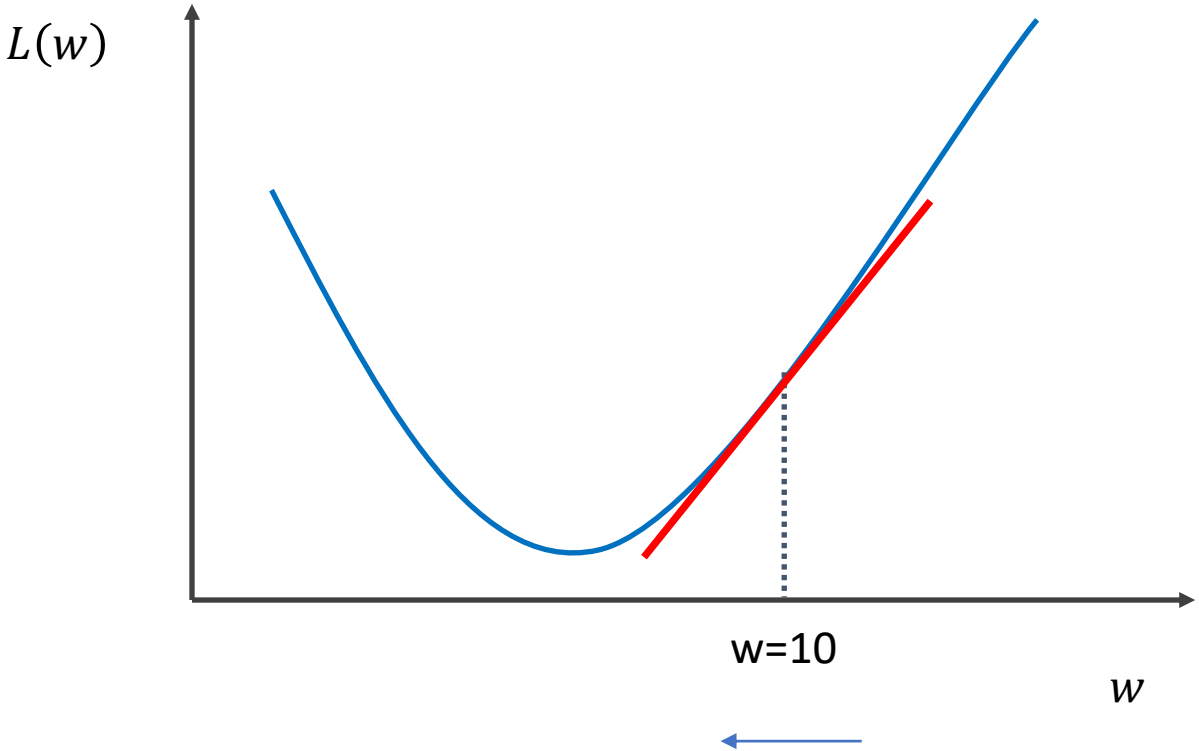
1. Start with a random value of w (e.g. w = 12)

2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

3. Recompute w as:

w = w – lambda * (dL / dw)

Gradient Descent (GD) (idea)

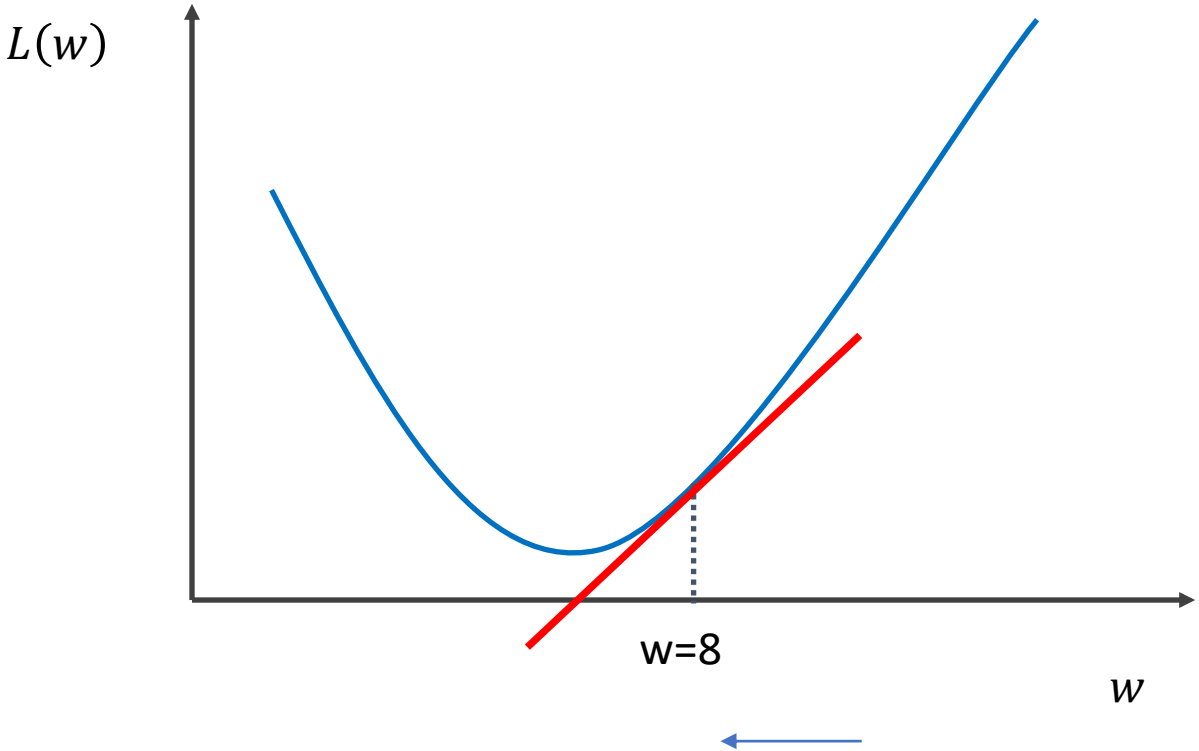


2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

3. Recompute w as:

w = w – lambda * (dL / dw)

Gradient Descent (GD) (idea)



2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

3. Recompute w as:

w = w – lambda * (dL / dw)

Our function L(w)

$$L(w) = 3 + (1 - w)^2$$

Our function L(w)

$$L(w) = 3 + (1 - w)^2$$

$$L(W, b) = \sum_{i=1}^{n} -\log f_{i,label}(W, b)$$

Our function L(w)

$$L(w) = 3 + (1 - w)^2$$

$$\mathrm{L}(w_1, w_2, .., w_{12}) = -logsoftmax\big(g(w_1, w_2, .., w_{12}, x_1)_{label_1}\big)$$
$$-logsoftmax\big(g(w_1, w_2, .., w_{12}, x_2)_{label_2}\big)$$
$$\dots$$
$$-logsoftmax\big(g(w_1, w_2, .., w_{12}, x_n)_{label_n}\big)$$

Gradient Descent (GD)

$\lambda = 0.01$

Initialize w and b randomly

for e = 0, num_epochs **do**

expensive

$$L(w,b) = \sum_{i=1}^{n} -\log f_{i,label}(w,b)$$

Compute: $dL(w,b)/dw$ and $dL(w,b)/db$

Update w: $w = w - \lambda\, dL(w,b)/dw$

Update b: $b = b - \lambda\, dL(w,b)/db$

Print: $L(w,b)$ // Useful to see if this is becoming smaller or not.

end

(mini-batch) Stochastic Gradient Descent (SGD)

$\lambda = 0.01$

Initialize w and b randomly

$$l(w,b) = \sum_{i \in B} -\log f_{i,label}(w,b)$$

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dl(w,b)/dw$ and $dl(w,b)/db$

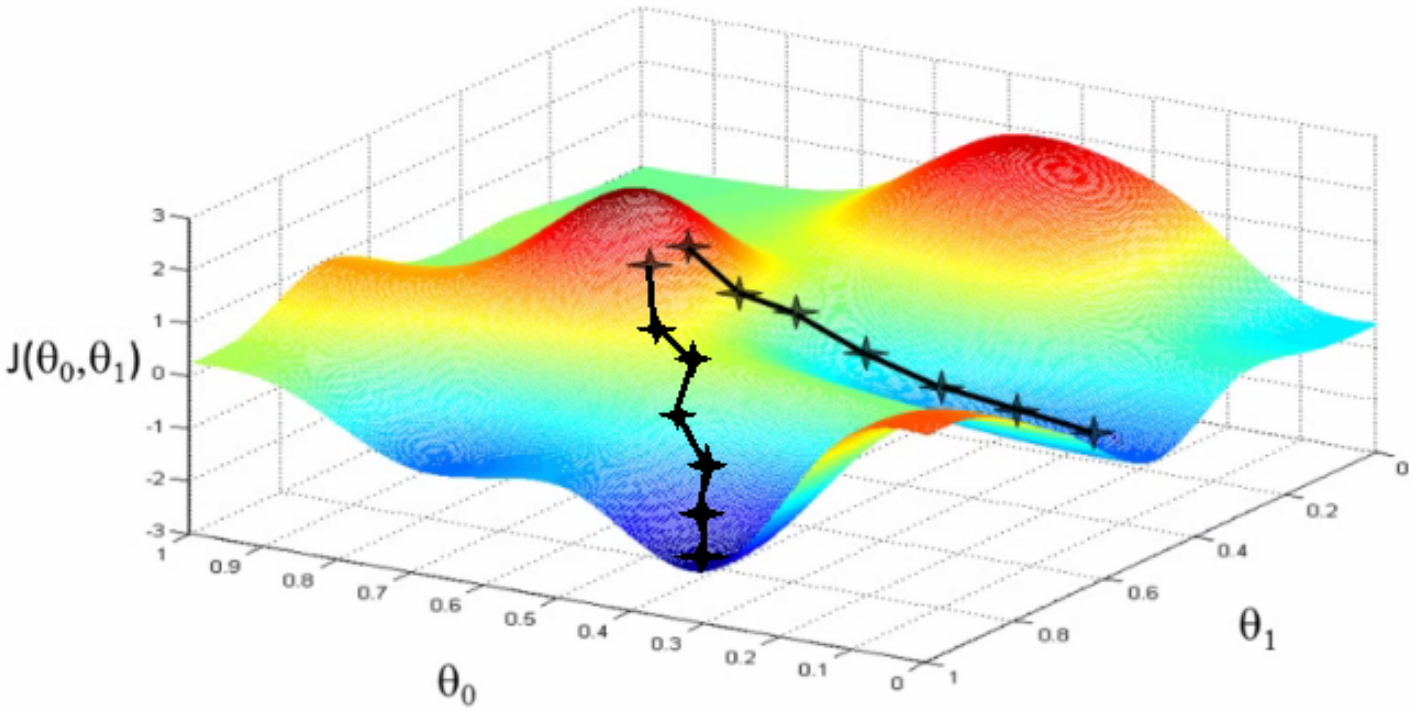
Update w: $w = w - \lambda\, dl(w,b)/dw$

Update b: $b = b - \lambda\, dl(w,b)/db$

Print: $l(w,b)$ // Useful to see if this is becoming smaller or not.

end

end



Source: Andrew Ng

(mini-batch) Stochastic Gradient Descent (SGD)

$\lambda = 0.01$

Initialize w and b randomly

$$l(w,b) = \sum_{i \in B} -\log f_{i,label}(w,b)$$

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dl(w,b)/dw$ and $dl(w,b)/db$ for |B| = 1

Update w: $w = w - \lambda\, dl(w,b)/dw$

Update b: $b = b - \lambda\, dl(w,b)/db$

Print: $l(w,b)$ // Useful to see if this is becoming smaller or not.

end

end

Computing Analytic Gradients

This is what we have:

$$\ell(W, b) = -\log(\hat{y}_{label}(W, b)) = -\log\left(\frac{\exp(a_{label}(W, b))}{\sum_{k=1}^{10} \exp(a_k(W, b))}\right)$$

Computing Analytic Gradients

This is what we have:

$$\ell(W,b) = -\log(\hat{y}_{label}(W,b)) = -\log\left(\frac{\exp(a_{label}(W,b))}{\sum_{k=1}^{10}\exp(a_k(W,b))}\right)$$

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^{10}\exp(a_k)}\right)$$

Reminder: $a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^{10}\exp(a_k)}\right)$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^{10}\exp(a_k)}\right)$$

This is what we need:

$\frac{\partial \ell}{\partial w_{ij}}$ for each w_{ij} $\frac{\partial \ell}{\partial b_i}$ for each b_i

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^{10}\exp(a_k)}\right)$$

Step 1: Chain Rule of Calculus

$$\frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i}\frac{\partial a_i}{\partial w_{ij}} \qquad \frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i}\frac{\partial a_i}{\partial b_i}$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^{10}\exp(a_k)}\right)$$

Step 1: Chain Rule of Calculus

Let's do these first

$$\frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i}\boxed{\frac{\partial a_i}{\partial w_{ij}}}$$

$$\frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i}\boxed{\frac{\partial a_i}{\partial b_i}}$$

Computing Analytic Gradients

$$\frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial a_i}{\partial b_i}$$

$$a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial w_{i,3}} = \frac{\partial}{\partial w_{i,3}}(w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial w_{i,3}} = x_3$$

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

Computing Analytic Gradients

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

$$\frac{\partial a_i}{\partial b_i}$$

$$a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial b_i} = \frac{\partial}{\partial b_i}(w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial b_i} = 1$$

Computing Analytic Gradients

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

$$\frac{\partial a_i}{\partial b_i} = 1$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^{10}\exp(a_k)}\right)$$

Step 1: Chain Rule of Calculus

Now let's do this one (same for both!)

$$\frac{\partial \ell}{\partial w_{ij}} = \boxed{\frac{\partial \ell}{\partial a_i}}\frac{\partial a_i}{\partial w_{ij}} \qquad \frac{\partial \ell}{\partial b_i} = \boxed{\frac{\partial \ell}{\partial a_i}}\frac{\partial a_i}{\partial b_i}$$

Computing Analytic Gradients

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i}\left[-\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^{10}\exp(a_k)}\right)\right]$$

$$= \frac{\partial}{\partial a_i}\left[\log\left(\sum_{k=1}^{10}\exp(a_k)\right) - a_{label}\right]$$

In our cat, dog, bear classification example: i = {0, 1, 2}

Computing Analytic Gradients

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i}\left[-\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^{10} \exp(a_k)} \right) \right]$$

$$= \frac{\partial}{\partial a_i}\left[\log\left(\sum_{k=1}^{10} \exp(a_k) \right) - a_{label} \right]$$

In our cat, dog, bear classification example: i = {0, 1, 2}

Let's say: label = 1

We need: $\frac{\partial \ell}{\partial a_0}$ $\frac{\partial \ell}{\partial a_1}$ $\frac{\partial \ell}{\partial a_2}$

Computing Analytic Gradients

$$= \frac{\partial}{\partial a_i}\left[\log\left(\sum_{k=1}^{10}\exp(a_k)\right) - a_{label}\right]$$

$\frac{\partial \ell}{\partial a_0}$ $\frac{\partial \ell}{\partial a_2}$

when $i \neq label$:

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i}\left[\log\left(\sum_{k=1}^{10}\exp(a_k)\right) - a_{label}\right]$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i}\log\left(\sum_{k=1}^{10}\exp(a_k)\right)$$

$$\frac{\partial \ell}{\partial a_i} = \left(\frac{1}{\sum_{k=1}^{10}\exp(a_k)}\right)\left(\frac{\partial}{\partial a_i}\sum_{k=1}^{10}\exp(a_k)\right)$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\exp(a_i)}{\sum_{k=1}^{10}\exp(a_k)} = \hat{y}_i$$

Remember this slide?

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $\quad y_i = [1 \quad 0 \quad 0]$ $\quad \hat{y}_i = [f_c \quad f_d \quad f_b]$

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$
$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$
$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$f_c = e^{g_c}/(e^{g_c}+e^{g_d} + e^{g_b})$$
$$f_d = e^{g_d}/(e^{g_c}+e^{g_d} + e^{g_b})$$
$$f_b = e^{g_b}/(e^{g_c}+e^{g_d} + e^{g_b})$$

Computing Analytic Gradients

$$= \frac{\partial}{\partial a_i}\left[\log\left(\sum_{k=1}^{10}\exp(a_k)\right) - a_{label}\right]$$

$\frac{\partial \ell}{\partial a_1}$

when $i = label$:

$$\frac{\partial \ell}{\partial a_{label}} = \frac{\partial}{\partial a_{label}}\left[\log\left(\sum_{k=1}^{10}\exp(a_k) - a_{label}\right)\right]$$

$$\frac{\partial \ell}{\partial a_{label}} = \frac{\partial}{\partial a_{label}}\log\left(\sum_{k=1}^{10}\exp(a_k)\right) - 1$$

$$\frac{\partial \ell}{\partial a_{label}} = \left(\frac{1}{\sum_{k=1}^{10}\exp(a_k)}\right)\left(\frac{\partial}{\partial a_{label}}\sum_{k=1}^{10}\exp(a_k)\right) - 1$$

$$\frac{\partial \ell}{\partial a_{label}} = \frac{\exp(a_{label})}{\sum_{k=1}^{10}\exp(a_k)} - 1 \qquad = \hat{y}_i - 1$$

Computing Analytic Gradients

label = 1

$$\frac{\partial \ell}{\partial a_0} = \hat{y}_0 \qquad \frac{\partial \ell}{\partial a_1} = \hat{y}_1 - 1 \qquad \frac{\partial \ell}{\partial a_1} = \hat{y}_2$$

$$\frac{\partial \ell}{\partial a} = \begin{bmatrix} \frac{\partial \ell}{\partial a_0} \\ \frac{\partial \ell}{\partial a_1} \\ \frac{\partial \ell}{\partial a_2} \end{bmatrix} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 - 1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \hat{y} - y$$

$$\frac{\partial \ell}{\partial a_i} = \hat{y}_i - y_i$$

Computing Analytic Gradients

$$\frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial b_i}$$

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

$$\frac{\partial a_i}{\partial b_i} = 1$$

$$\frac{\partial \ell}{\partial a_i} = \hat{y}_i - y_i$$

$$\boxed{\frac{\partial \ell}{\partial w_{i,j}} = (\hat{y}_i - y_i)x_j}$$

$$\boxed{\frac{\partial \ell}{\partial b_i} = (\hat{y}_i - y_i)}$$

Supervised Learning –Softmax Classifier

Extract features

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$

Run features through classifier

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$

$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$

$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$f_c = e^{g_c}/(e^{g_c}+e^{g_d} + e^{g_b})$$

$$f_d = e^{g_d}/(e^{g_c}+e^{g_d} + e^{g_b})$$

$$f_b = e^{g_b}/(e^{g_c}+e^{g_d} + e^{g_b})$$

Get predictions

$\hat{y}_i = [f_c \quad f_d \quad f_b]$

Linear Max Margin-Classifier

Training Data

inputs

targets / labels / ground truth

predictions

$x_1 = [x_{11} \quad x_{12} \quad x_{13} \quad x_{14}]$ $y_1 = [1 \quad 0 \quad 0]$ $\hat{y}_1 = [4.3 \quad -1.3 \quad 1.1]$

$x_2 = [x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]$ $y_2 = [0 \quad 1 \quad 0]$ $\hat{y}_2 = [0.5 \quad 5.6 \quad -4.2]$

$x_3 = [x_{31} \quad x_{32} \quad x_{33} \quad x_{34}]$ $y_3 = [1 \quad 0 \quad 0]$ $\hat{y}_3 = [3.3 \quad 3.5 \quad 1.1]$

.
.
.

$x_n = [x_{n1} \quad x_{n2} \quad x_{n3} \quad x_{n4}]$ $y_n = [0 \quad 0 \quad 1]$ $\hat{y}_n = [1.1 \quad -5.3 \quad -9.4]$

Linear – Max Margin Classifier - Inference

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $\quad y_i = [1 \quad 0 \quad 0]$ $\quad \hat{y}_i = [f_c \quad f_d \quad f_b]$

$$f_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$
$$f_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$
$$f_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

Training: How do we find a good w and b?

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $\quad y_i = [1 \quad 0 \quad 0]$ $\quad \hat{y}_i = [f_c(w,b) \quad f_d(w,b) \quad f_b(w,b)]$

We need to find w, and b that minimize the following:

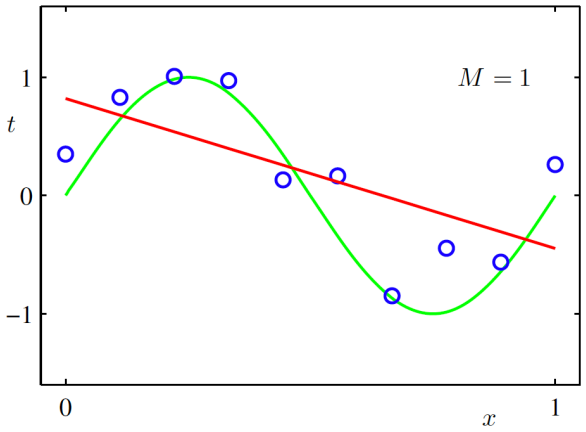
$$L(w,b) = \sum_{i=1}^{n} \sum_{j \neq label} \max(0, \hat{y}_{ij} - \hat{y}_{i,label} + \Delta)$$

Why this might be good compared to softmax?

Taken from Christopher Bishop's Machine Learning and Pattern Recognition Book.

Overfitting

f is linear

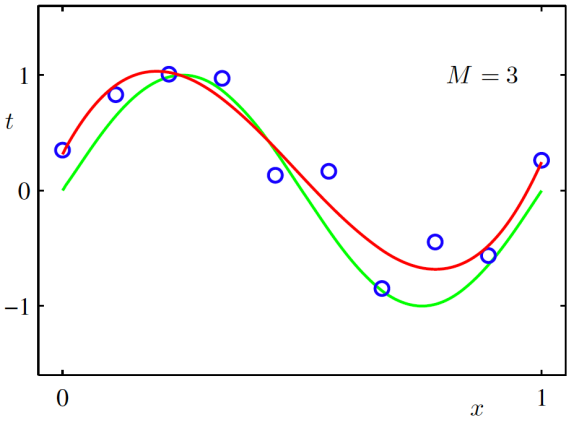


$Loss(w)$ is high

Underfitting

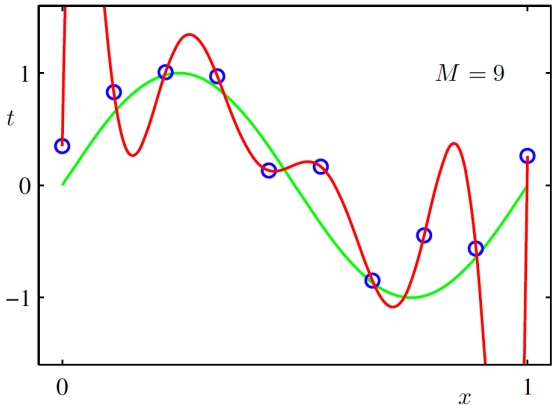
High Bias

f is cubic



$Loss(w)$ is low

f is a polynomial of degree 9



$Loss(w)$ is zero!

Overfitting

High Variance

Detecting Overfitting

- Look at the values of the weights in the polynomial

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Recommended Reading

- http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf

Print and Read Chapter 1
(at minimum)

More …

- Regularization
- Momentum updates

Regularization

- Large weights lead to large variance. i.e. model fits to the training data too strongly.

- Solution: Minimize the loss but also try to keep the weight values small by doing the following:

minimize $L(w, b) + \sum_i |w_i|^2$

Regularization

- Large weights lead to large variance. i.e. model fits to the training data too strongly.

- Solution: Minimize the loss but also try to keep the weight values small by doing the following:

minimize $L(w, b) + \alpha \sum_i |w_i|^2$

Regularizer term
e.g. L2- regularizer

SGD with Regularization (L-2)

$\lambda = 0.01$

$$l(w,b) = l(w,b) + \alpha \sum_i |w_i|^2$$

Initialize w and b randomly

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dl(w,b)/dw$ and $dl(w,b)/db$

Update w: $w = w - \lambda\, dl(w,b)/dw - \lambda\alpha w$

Update b: $b = b - \lambda\, dl(w,b)/db - \lambda\alpha w$

Print: $l(w,b)$ // Useful to see if this is becoming smaller or not.

end

end

Revisiting Another Problem with SGD

$\lambda = 0.01$

$l(w, b) = l(w, b) + \alpha \sum_i |w_i|^2$

Initialize w and b randomly

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dl(w, b)/dw$ and $dl(w, b)/db$

Update w: $w = w - \lambda\, dl(w, b)/dw - \lambda\alpha w$

Update b: $b = b - \lambda\, dl(w, b)/db - \lambda\alpha w$

Print: $l(w, b)$ // Useful to see if this is becoming smaller or not.

end

end

These are only approximations to the true gradient with respect to $L(w, b)$

Revisiting Another Problem with SGD

$\lambda = 0.01$

$l(w,b) = l(w,b) + \alpha \sum_i |w_i|^2$

Initialize w and b randomly

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dl(w,b)/dw$ and $dl(w,b)/db$

Update w: $w = w - \lambda\, dl(w,b)/dw - \lambda\alpha w$

Update b: $b = b - \lambda\, dl(w,b)/db - \lambda\alpha w$

Print: $l(w,b)$ // Useful to see if this is becoming smaller or not.

end

end

This could lead to "un-learning" what has been learned in some previous steps of training.

Solution: Momentum Updates

$\lambda = 0.01$

$l(w,b) = l(w,b) + \alpha \sum_i |w_i|^2$

Initialize w and b randomly

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dl(w,b)/dw$ and $dl(w,b)/db$

Update w: $w = w - \lambda\, dl(w,b)/dw - \lambda\alpha w$

Update b: $b = b - \lambda\, dl(w,b)/db - \lambda\alpha w$

Print: $l(w,b)$ // Useful to see if this is becoming smaller or not.

end

end

Keep track of previous gradients in an accumulator variable! and use a weighted average with current gradient.

Solution: Momentum Updates

$\lambda = 0.01 \qquad \tau = 0.9$

Initialize w and b randomly

global v

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dl(w,b)/dw$

Compute: $v = \tau v + dl(w,b)/dw + \alpha w$

Update w: $w = w - \lambda v$

Print: $l(w,b)$ // Useful to see if this is becoming smaller or not.

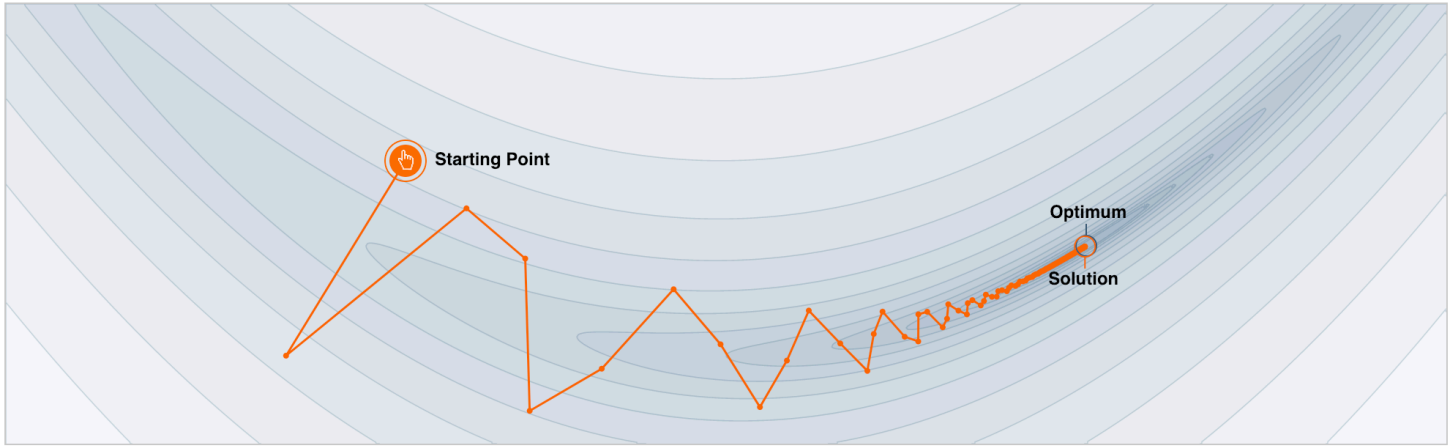
end

end

$$l(w,b) = l(w,b) + \alpha \sum_i |w_i|^2$$

Keep track of previous gradients in an accumulator variable! and use a weighted average with current gradient.

More on Momentum



Step-size α = 0.0050

0 0.003 0.006

Momentum β = 0.77

0.00 0.500 0.990

We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

https://distill.pub/2017/momentum/

Questions?