CS4501: Introduction to Computer Vision

Max-Margin Classifier, Regularization, Generalization, Momentum, Regression, Multi-label Classification / Tagging

Previous Class

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

Previous Class

- Softmax Classifier
  - Inference vs Training
  - Gradient Descent (GD)
  - Stochastic Gradient Descent (SGD)
  - mini-batch Stochastic Gradient Descent (SGD)
- Generalization
- Regularization / Momentum
- Max-Margin Classifier
- Regression / Tagging

(mini-batch) Stochastic Gradient Descent (SGD)

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

For Softmax Classifier

$\lambda = 0.01$

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

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

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 0 0] $\hat{y}_1 =$ [4.3 -1.3 1.1]

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

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

.
.
.

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

Linear – Max Margin Classifier - Inference

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $\quad y_i =$ [1 0 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?

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

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?

Regression vs Classification

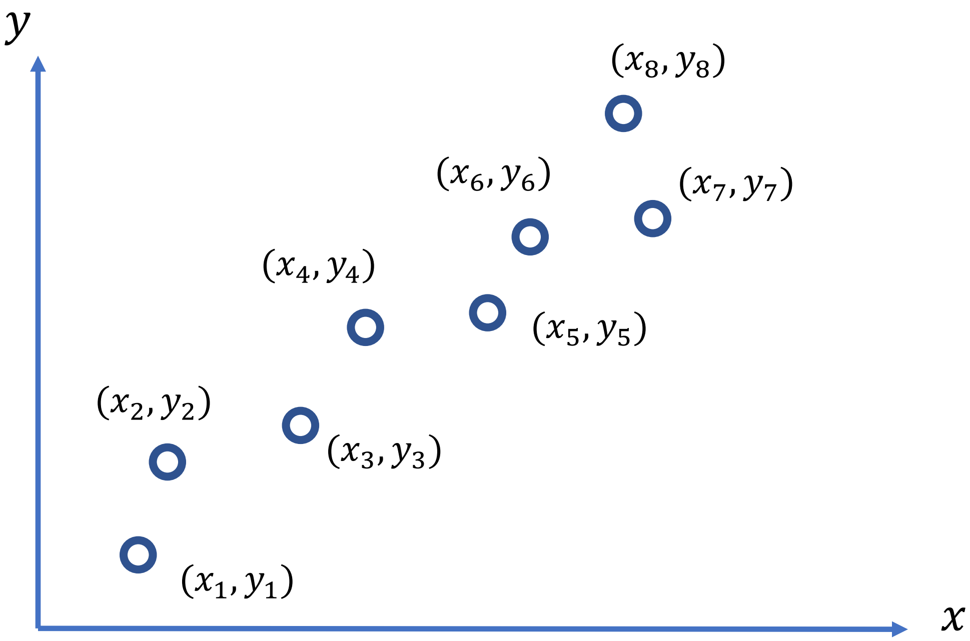
Regression

- Labels are continuous variables – e.g. distance.
- Losses: Distance-based losses, e.g. sum of distances to true values.
- Evaluation: Mean distances, correlation coefficients, etc.

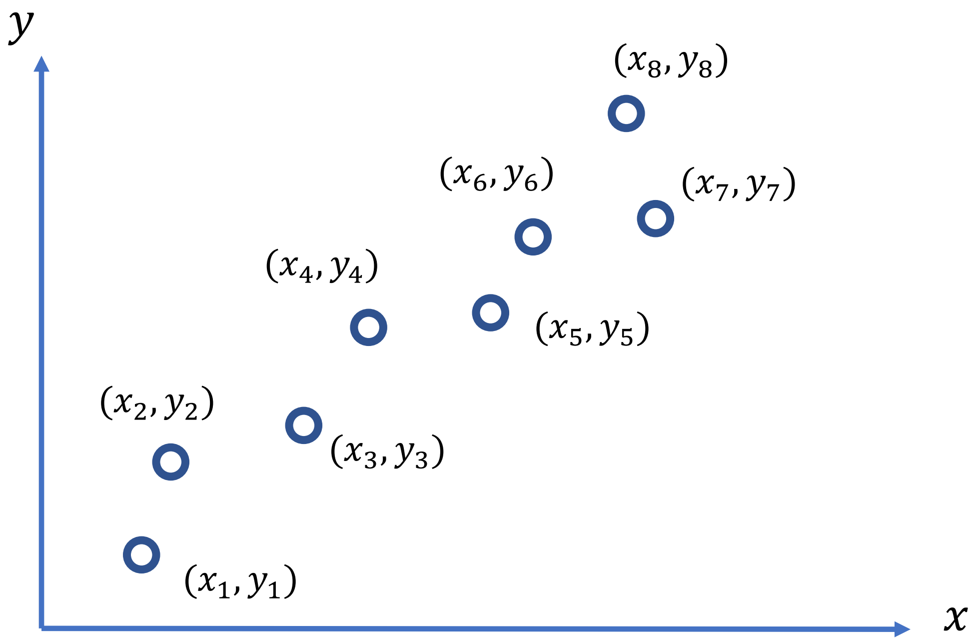
Classification

- Labels are discrete variables (1 out of K categories)
- Losses: Cross-entropy loss, margin losses, logistic regression (binary cross entropy)
- Evaluation: Classification accuracy, etc.

Linear Regression – 1 output, 1 input

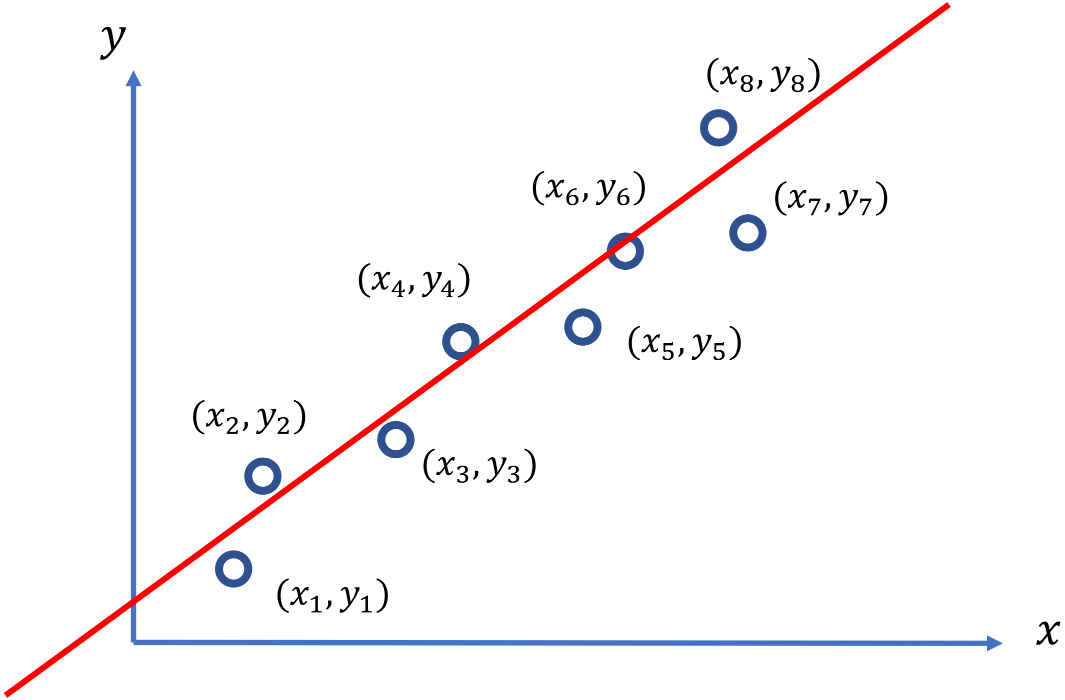


Linear Regression – 1 output, 1 input



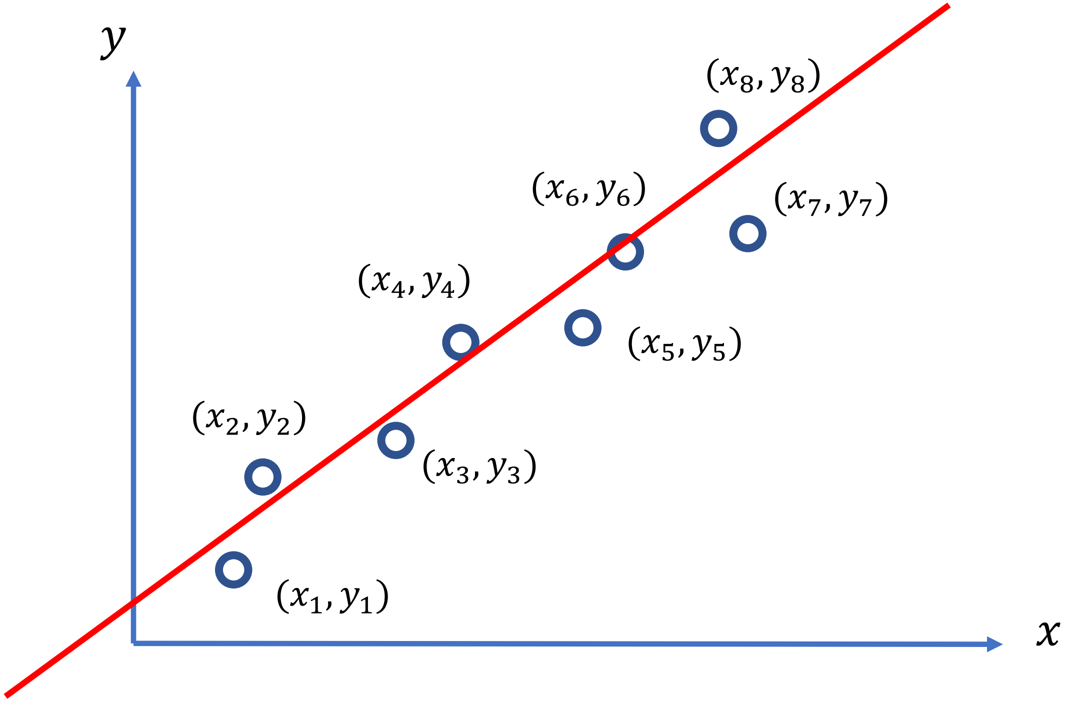
Model: $\hat{y} = wx + b$

Linear Regression – 1 output, 1 input



Model: $\hat{y} = wx + b$

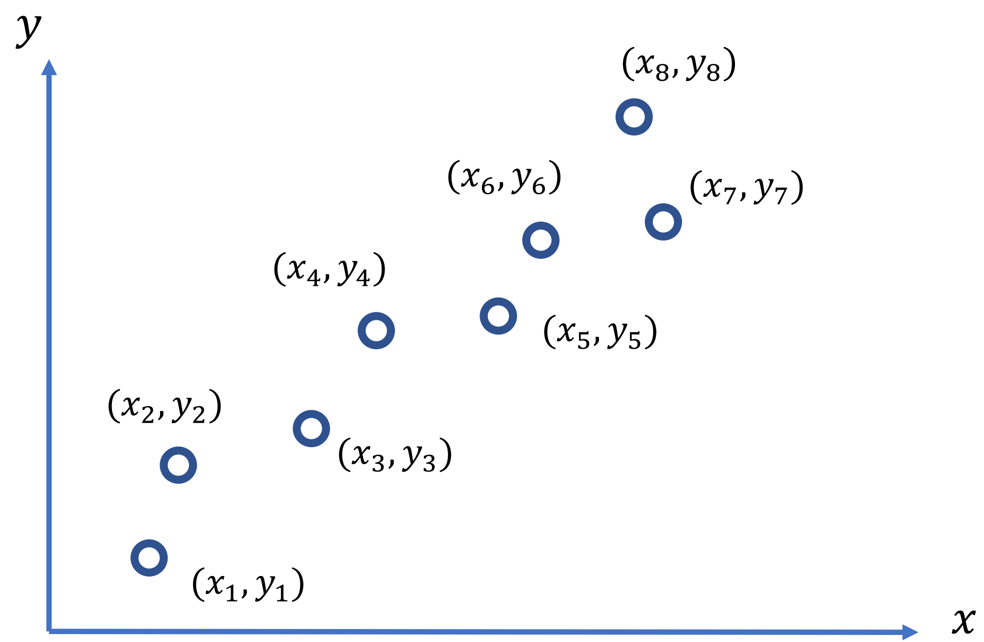
Linear Regression – 1 output, 1 input



Model: $\hat{y} = wx + b$

Loss: $L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2$

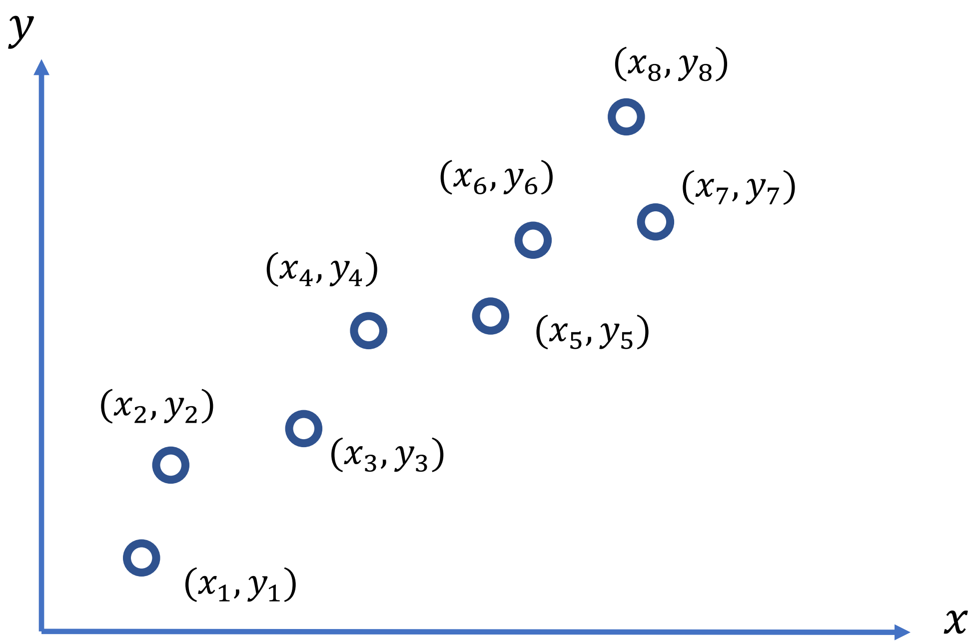
Quadratic Regression



Model: $\hat{y} = w_1 x^2 + w_2 x + b$

Loss: $L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2$

n-polynomial Regression



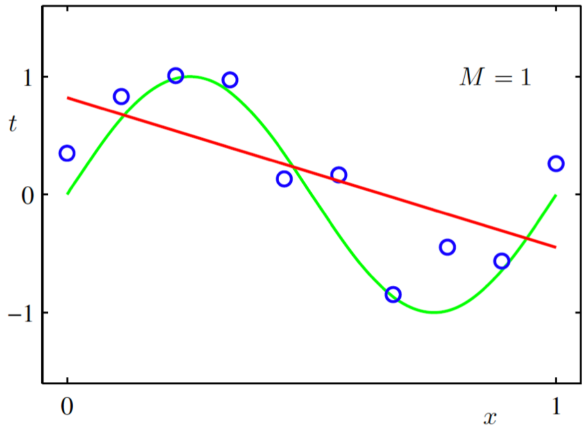
Model: $\hat{y} = w_n x^n + \cdots + w_1 x + b$

Loss: $L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2$

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

Overfitting

f is linear

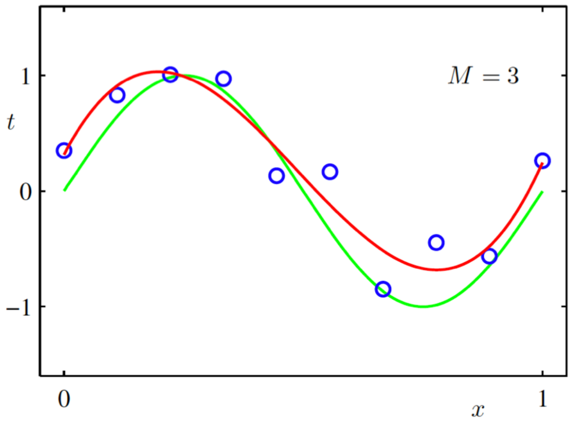


$Loss(w)$ is high

Underfitting

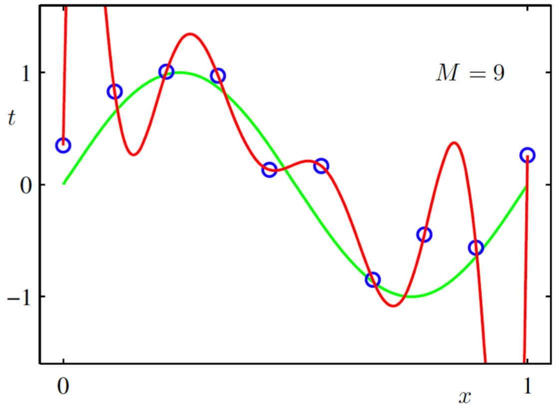
High Bias

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 $$L(w,b) + \boxed{\alpha \sum_i |w_i|^2}$$

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: $w = w - \lambda\, dl(w,b)/dw \boxed{- \lambda \alpha w}$

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

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: $\boxed{dl(w,b)/dw}$ and $\boxed{dl(w,b)/db}$

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: $\boxed{dl(w,b)/dw}$ and $\boxed{dl(w,b)/db}$

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: $\boxed{dl(w, b)/dw}$ and $\boxed{dl(w, b)/db}$

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

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

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

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

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

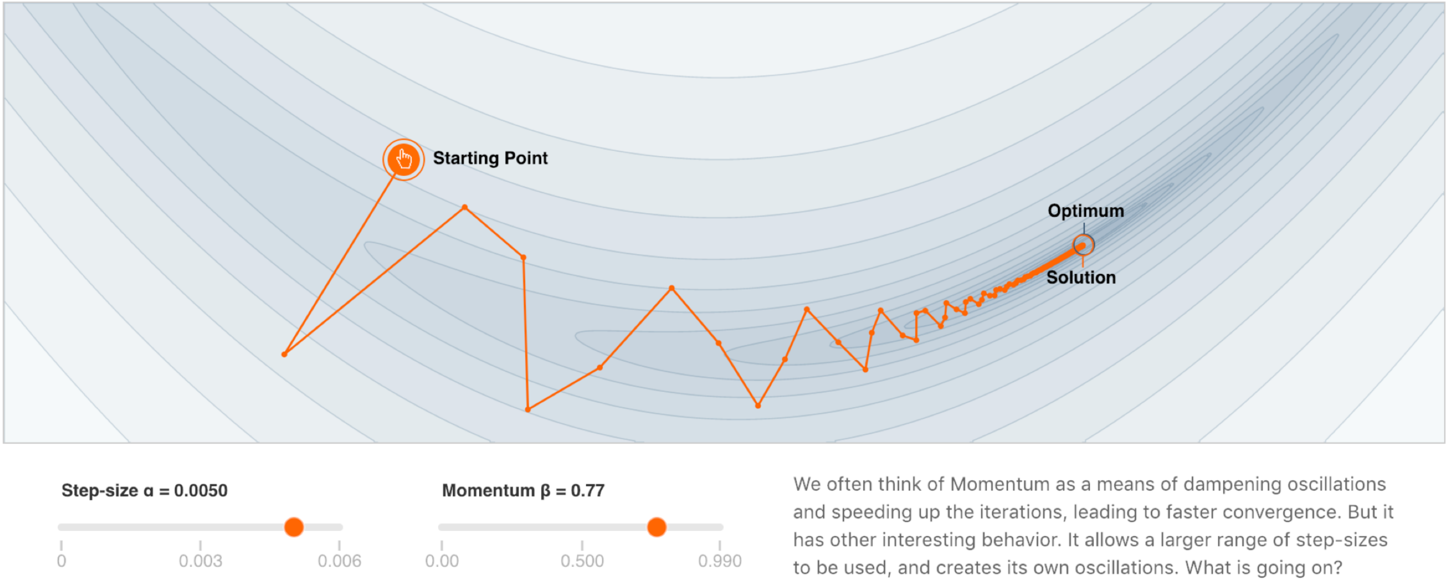
Print: $l(w, b)$ // Useful to see if this is becoming smaller or not.

end

end

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

More on Momentum



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?