CS6501: Deep Learning for Visual Recognition

Stochastic Gradient Descent (SGD)

Today's Class

Stochastic Gradient Descent (SGD)

- SGD Recap
- Regression vs Classification
- Generalization / Overfitting / Underfitting
- Regularization
- Momentum Updates / ADAM Updates

Our function L(w)

$$L(w) = 3 + (w - 4)^2$$

Our function L(w)

$$L(w) = 3 + (w - 4)^2$$

Easy way to find minimum (and max): Find where $\frac{\partial}{\partial w} L(w) = 0$

$$\frac{\partial}{\partial w} L(w) = 2(w - 4)$$

This is zero when: $w = 4$

Our function L(w)

$$L(w) = 3 + (w - 4)^2$$

But this is not easy for complex functions:

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

Our function L(w)

$$L(w) = 3 + (w - 4)^2$$

Or even for simpler functions:

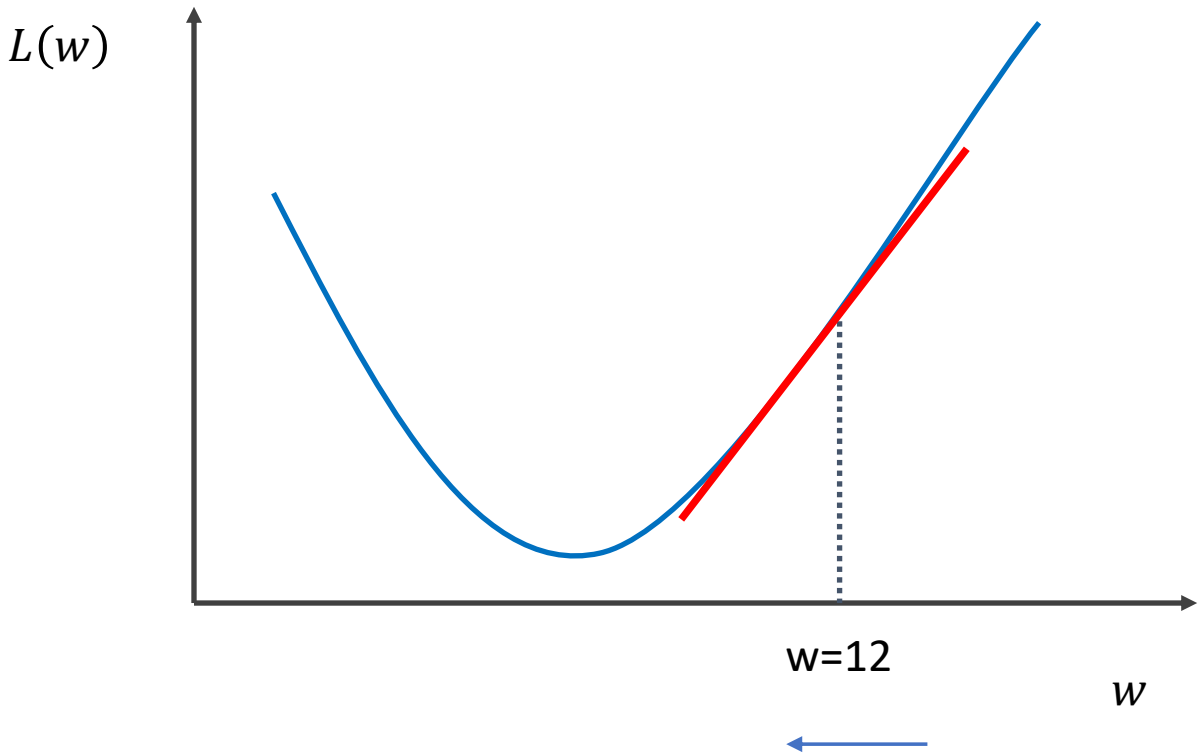
$$L(w) = e^{-x} + x^2$$

$$\frac{\partial L(w)}{\partial w} = -e^{-x} + 2x$$

$$0 = -e^{-x} + 2x$$

How do you find x?

Gradient Descent (GD) (idea)



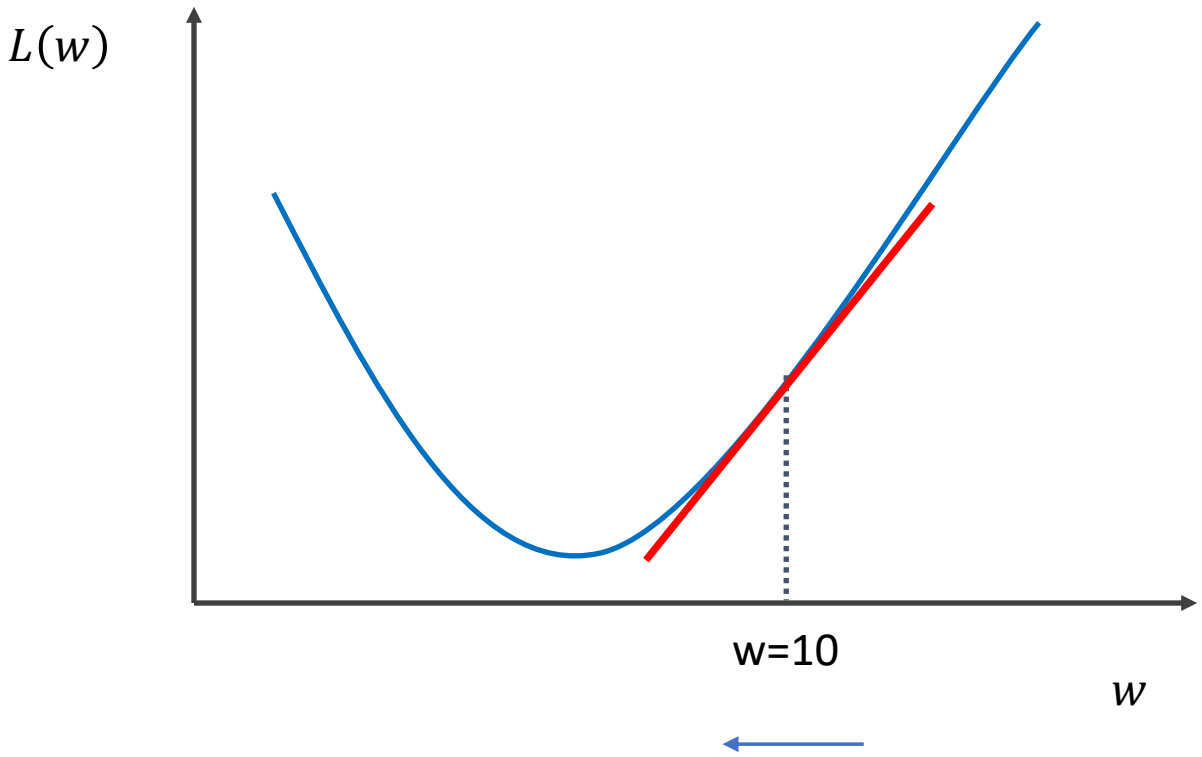
1. Start with a random value of w (e.g. w = 12)

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)

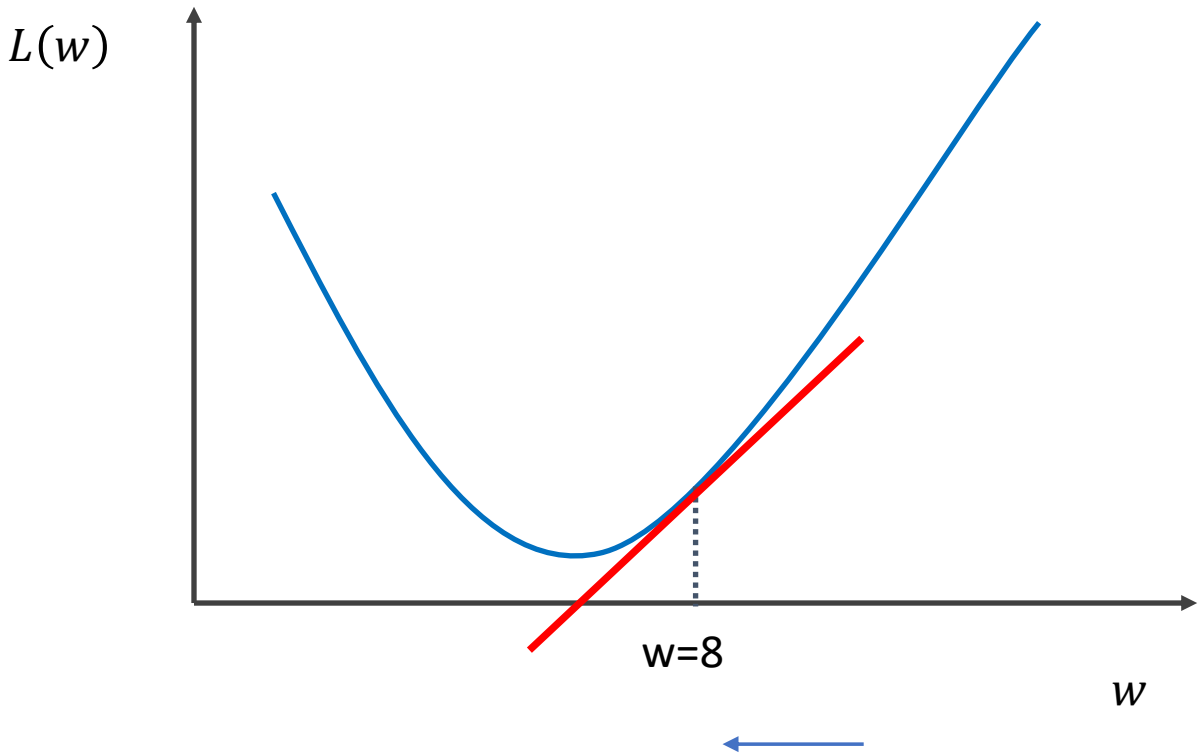


2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

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)

Gradient Descent (GD)

$\lambda = 0.01$

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

Initialize w and b randomly

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)

expensive

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

$\lambda = 0.01$

Initialize w and b randomly

for e = 0, num_epochs **do**

Compute: $\underbrace{dL(w,b)/dw}$ and $\underbrace{dL(w,b)/db}$

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)

$\lambda = 0.01$

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

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

(mini-batch) Stochastic Gradient Descent (SGD)

$\lambda = 0.01$

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

Initialize w and b randomly

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

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

Regression vs Classification

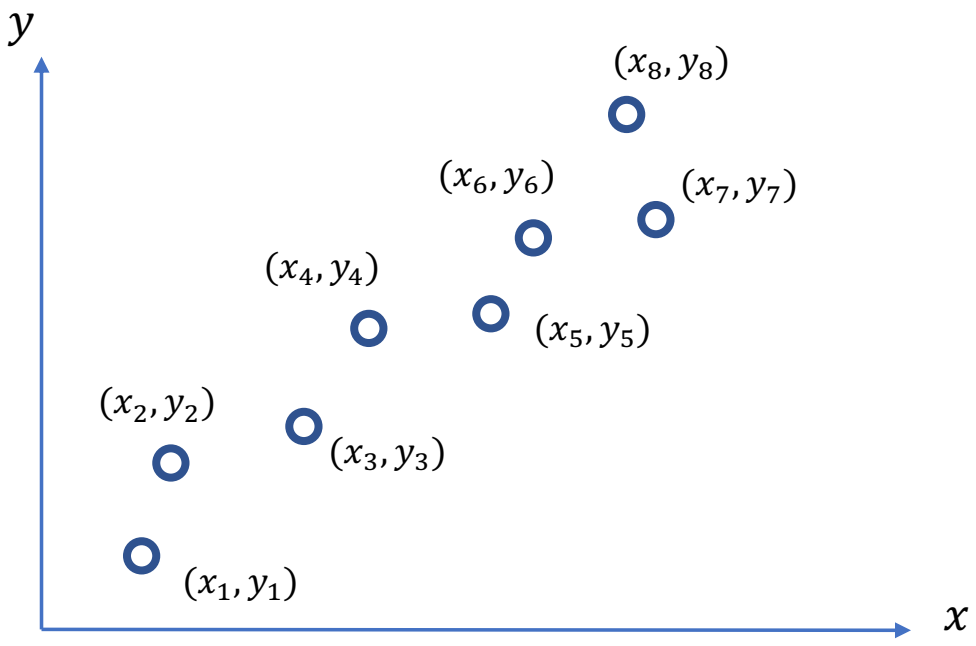
Regression

- 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.

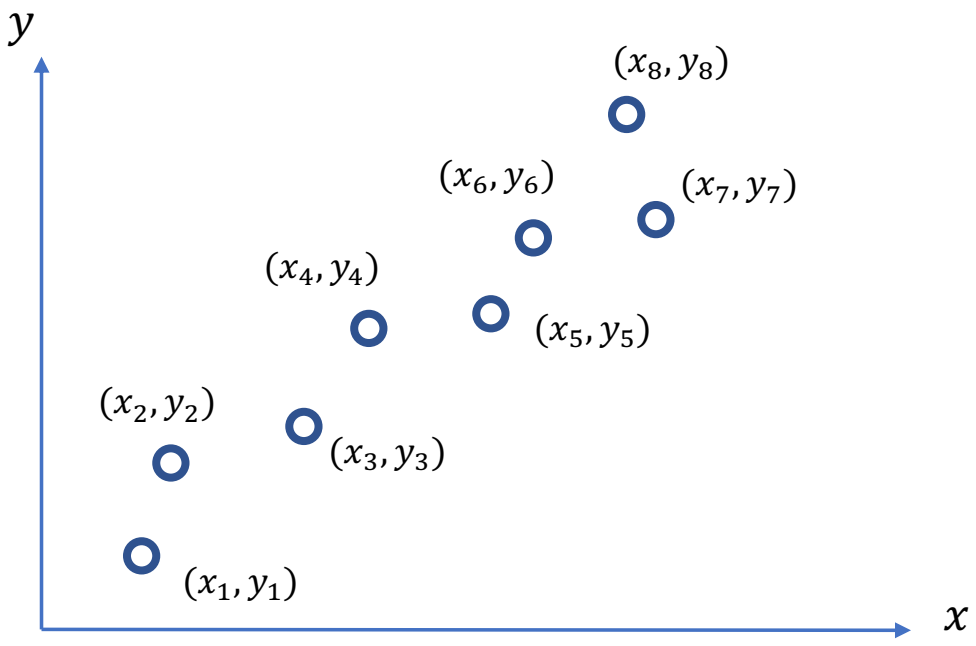
Classification

- 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

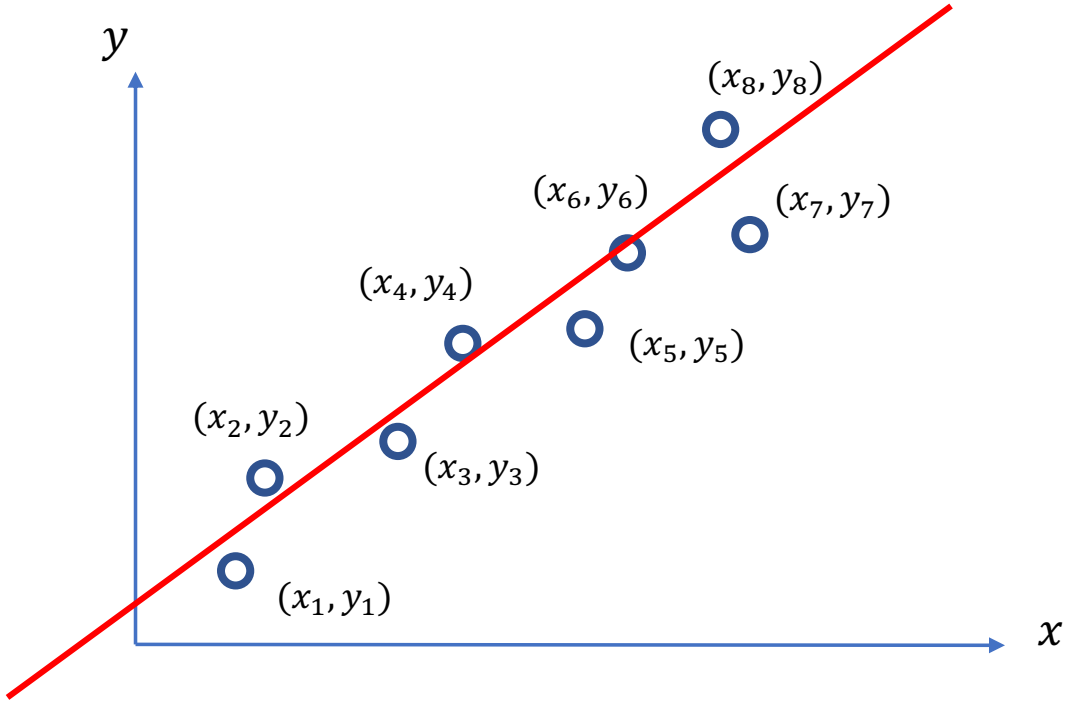


Linear Regression – 1 output, 1 input



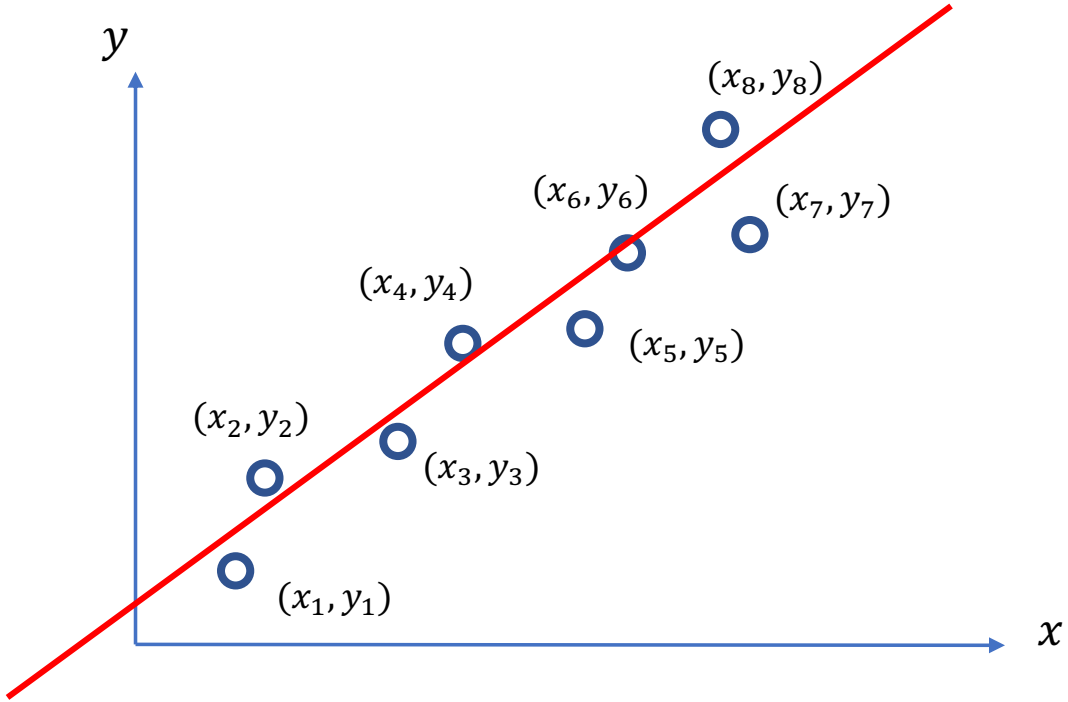
Model: $\hat{y} = wx + b$

Linear Regression – 1 output, 1 input



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

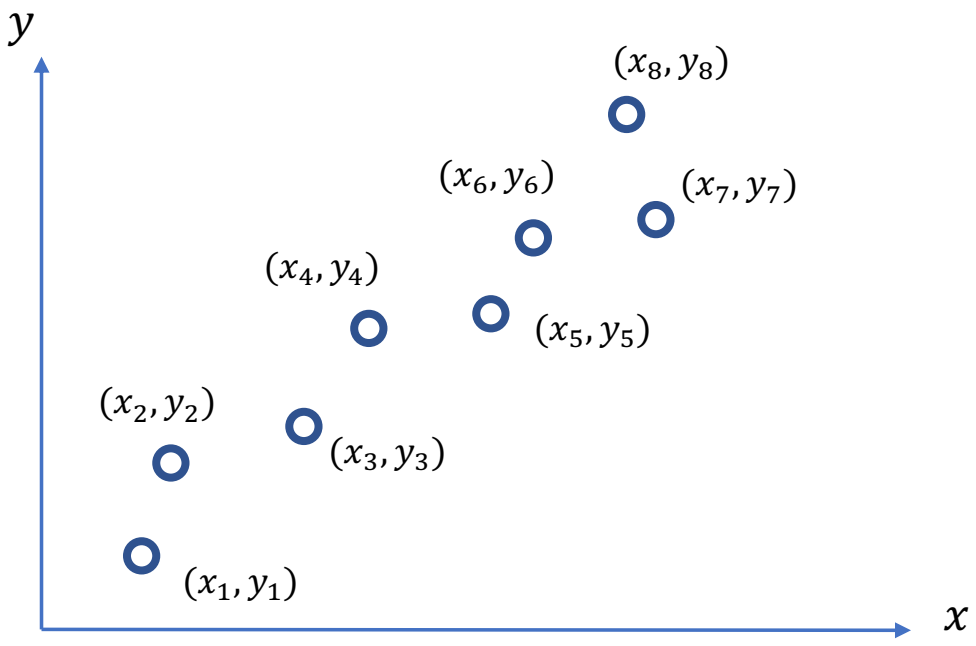
Linear Regression – 1 output, 1 input



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

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

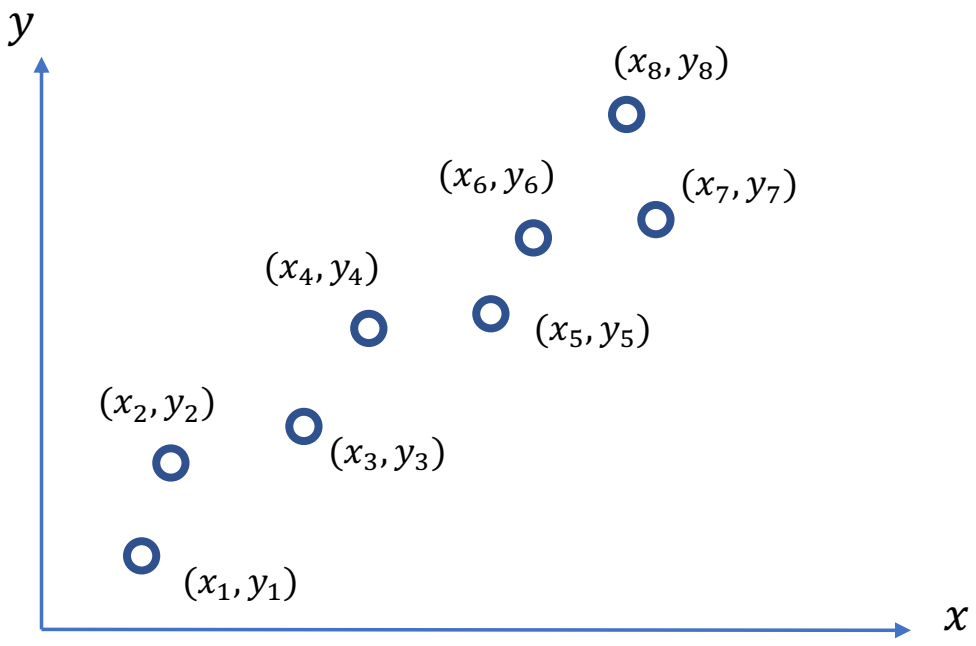
Quadratic Regression



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

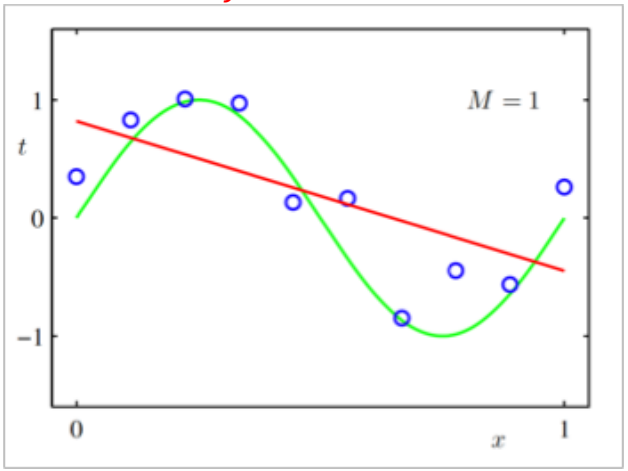


Model: $\hat{y} = w_n x^n + \cdots + w_1 x + b$

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

Overfitting

f is linear

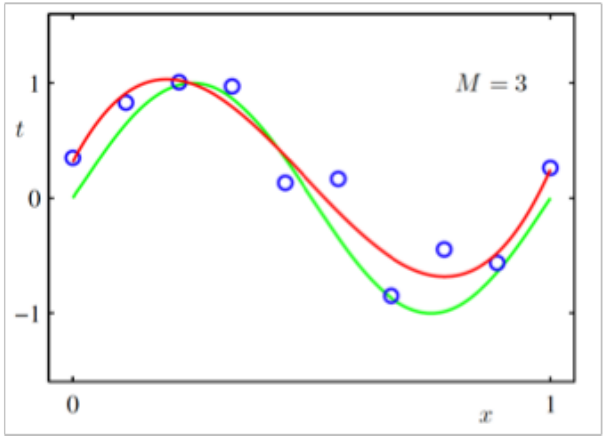


$Loss(w)$ is high

Underfitting

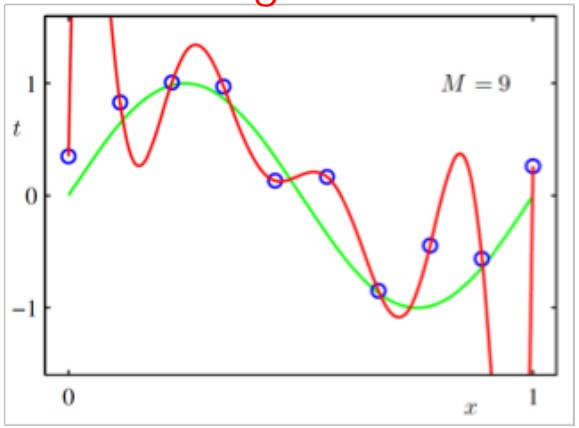
High Bias

f is cubic



$Loss(w)$ is low

f is a polynomial of degree 9



$Loss(w)$ is zero!

Overfitting

High Variance

Christopher M. Bishop – Pattern Recognition and Machine Learning

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

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

Initialize w and b randomly

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

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: $w = w - \lambda v$

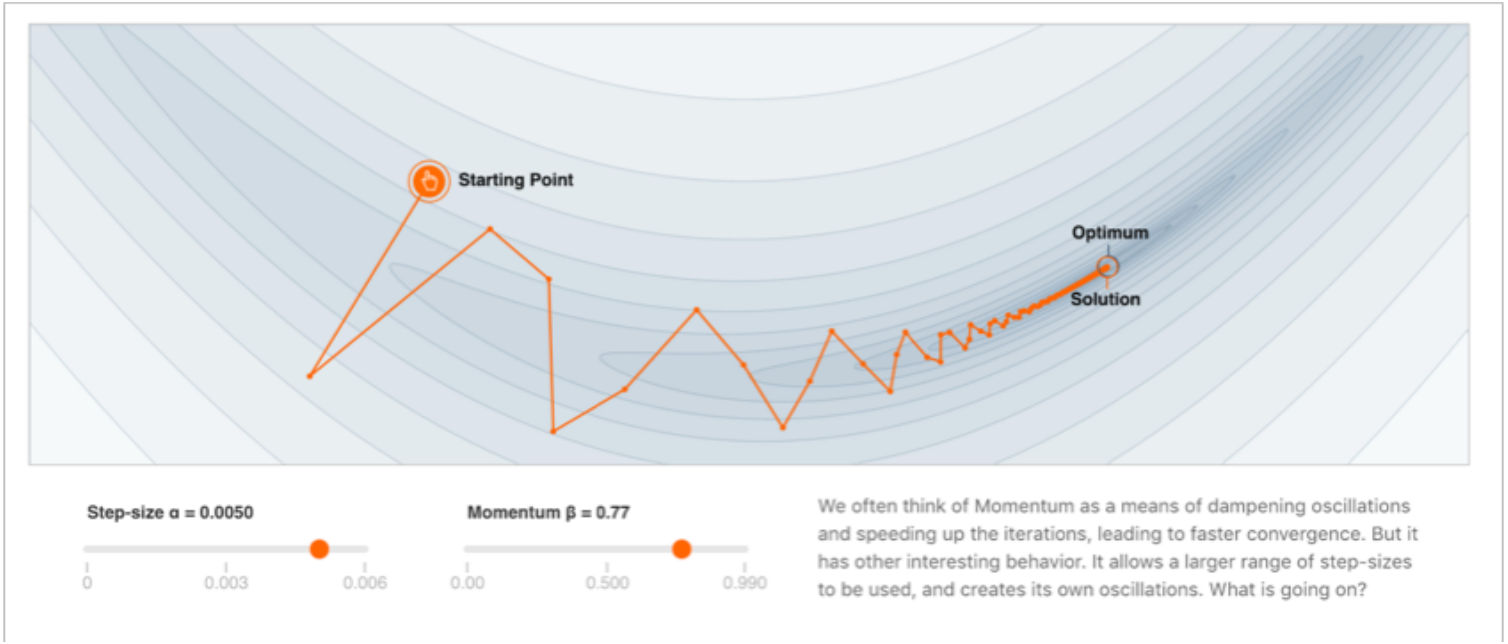
Print: $l(w,b)$ // Useful to see if this is becoming smaller or not.

end

end

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

More on Momentum



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

Questions?