CS6501: Deep Learning for Visual Recognition

Softmax Classifier + SGD

Today’s Class

Intro to Machine Learning

- What is Machine Learning?
- Supervised Learning: Classification with K-nearest neighbors
- Unsupervised Learning: Clustering with K-means clustering

Softmax Classifier

- Stochastic Gradient Descent
- Regularization

Teaching Assistants

Ziyan Yang
(tw8cb@virginia.edu)
Office Hours: Thursdays 3 to 5pm (Rice 442)

Paola Cascante-Bonilla
(pc9za@virginia.edu)
Hours: Fridays 2 to 4pm (Rice 442)

Also…

- Assignment 2 will be released between today and tomorrow.
- Subscribe and check Piazza regularly, important information about assignments will go there. Please use Piazza.

Machine Learning

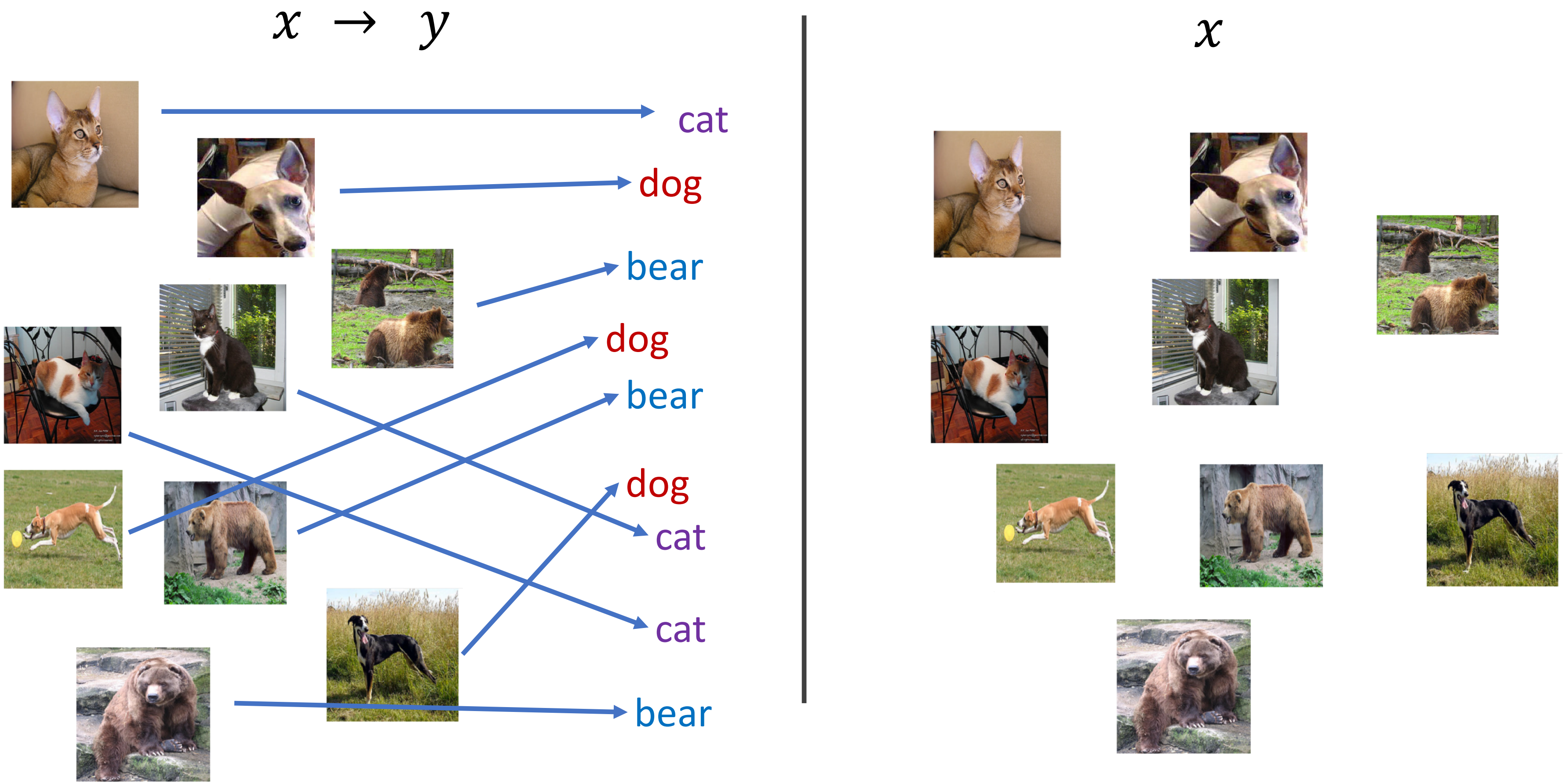
- **Machine learning** is the subfield of computer science that gives "computers the ability to learn without being explicitly programmed.”

  - term coined by Arthur Samuel 1959 while at IBM

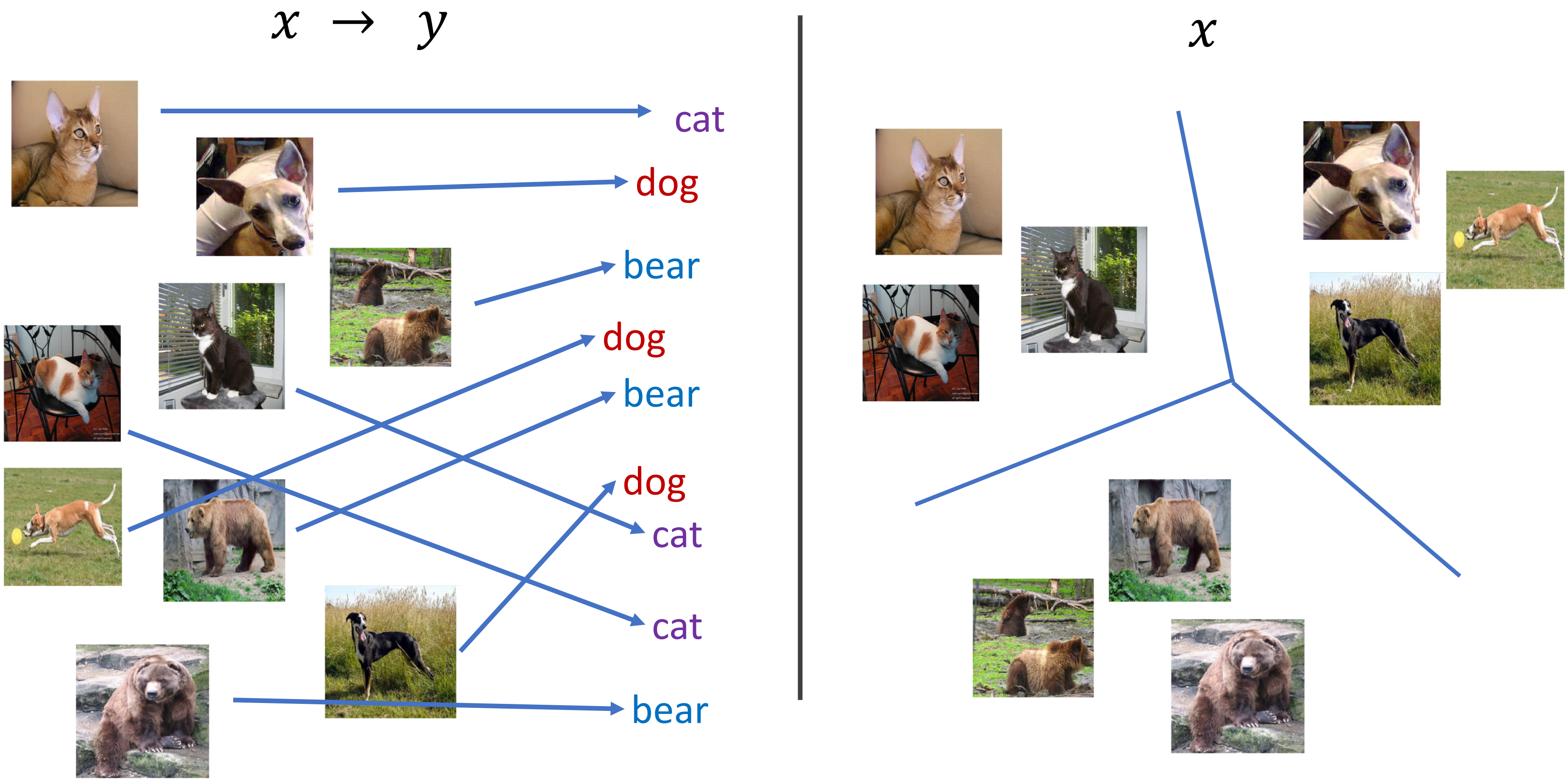
- The study of algorithms that can learn from data.

- In contrast to previous Artificial Intelligence systems based on Logic, e.g. ”Expert Systems”

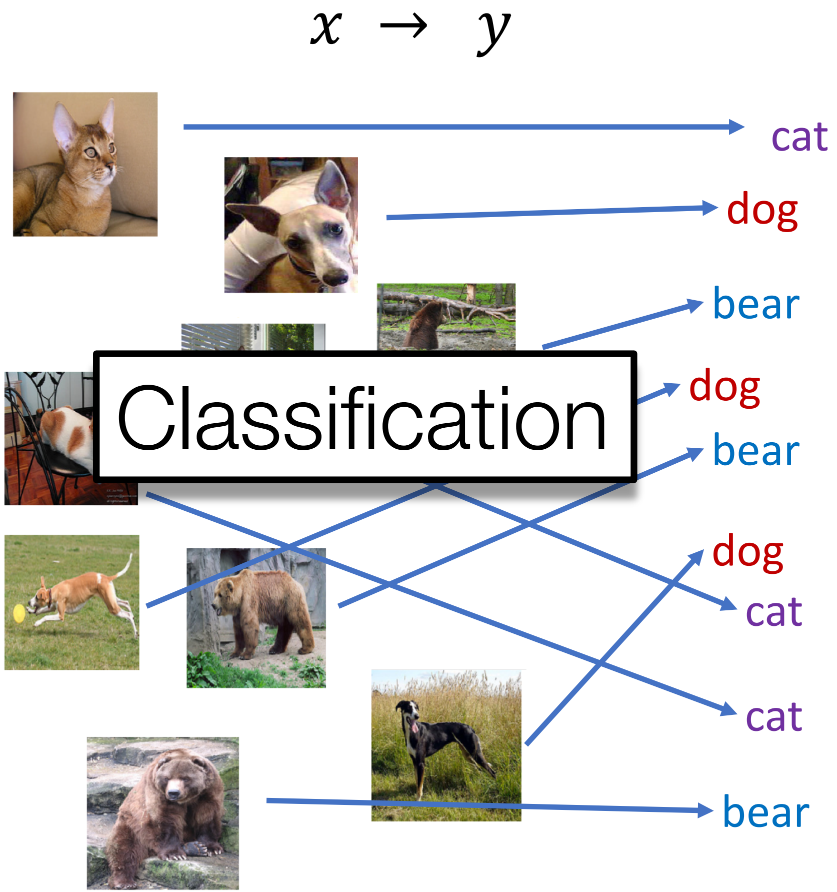
Supervised Learning vs Unsupervised Learning

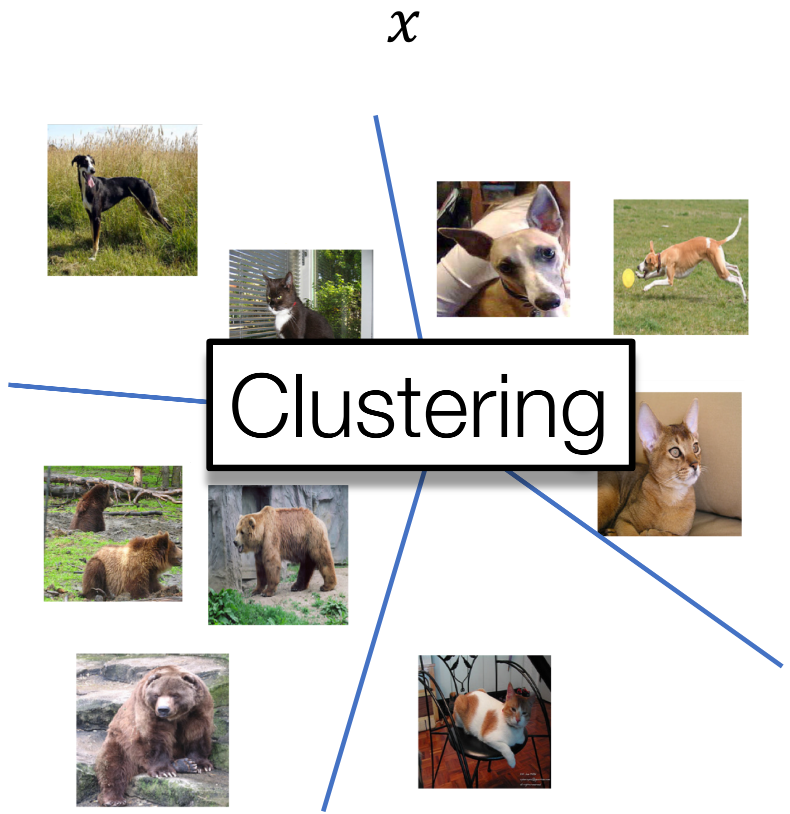


Supervised Learning vs Unsupervised Learning



Supervised Learning vs Unsupervised Learning





Supervised Learning Examples

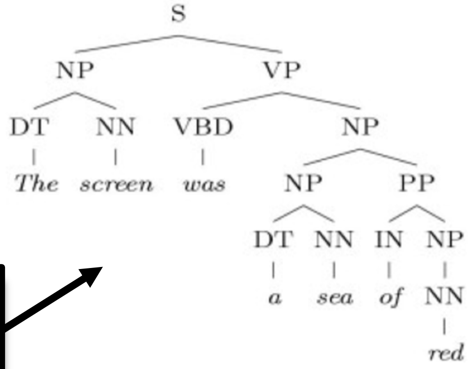
Classification

cat

Face Detection

The screen was a sea of red

Language Parsing

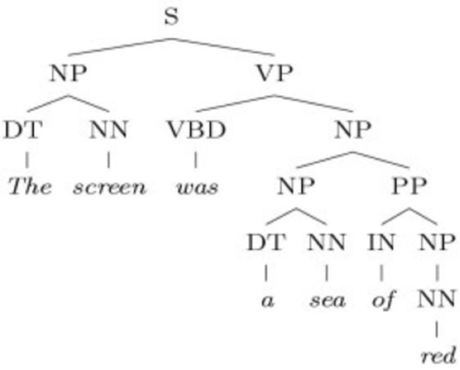


Structured Prediction

Supervised Learning Examples

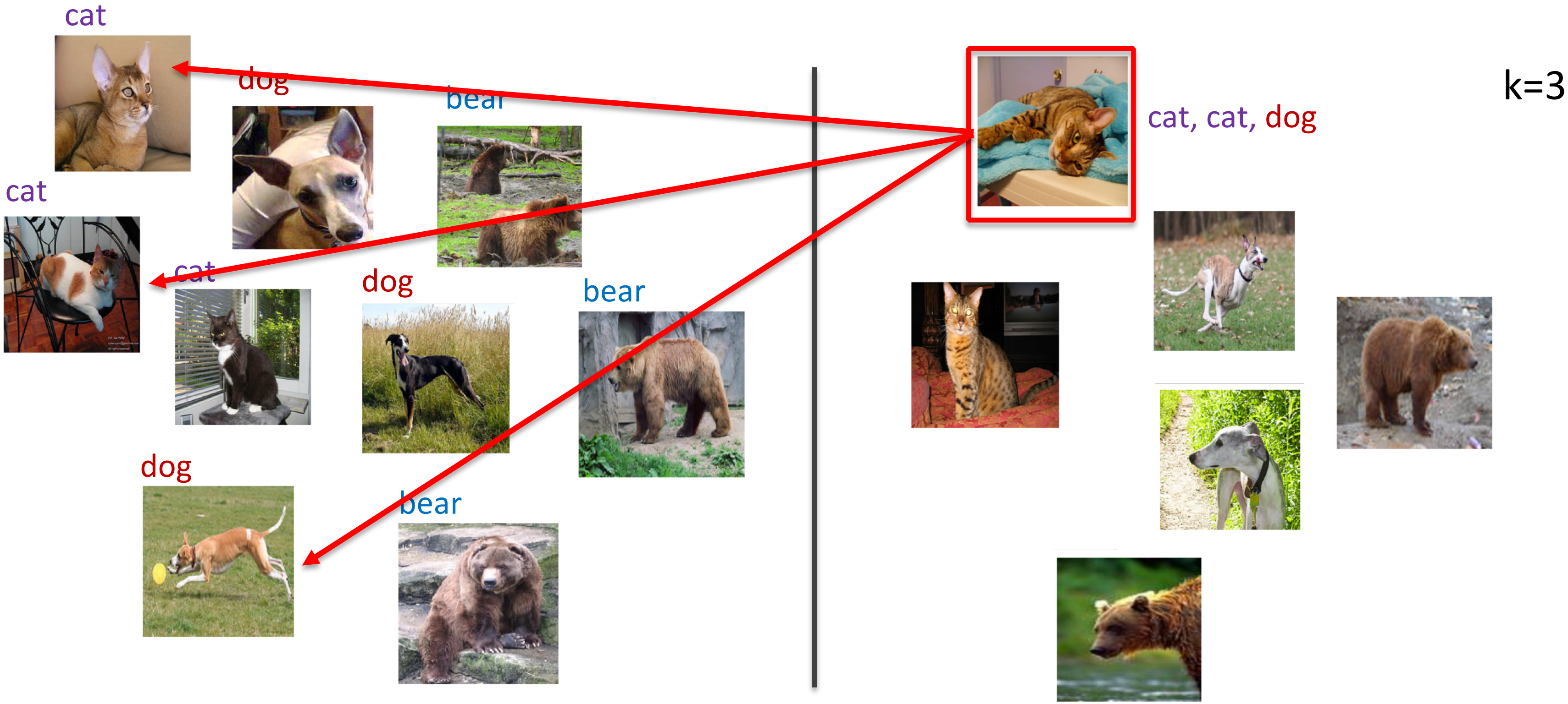
cat $= f($  $)$

 $= f($  $)$

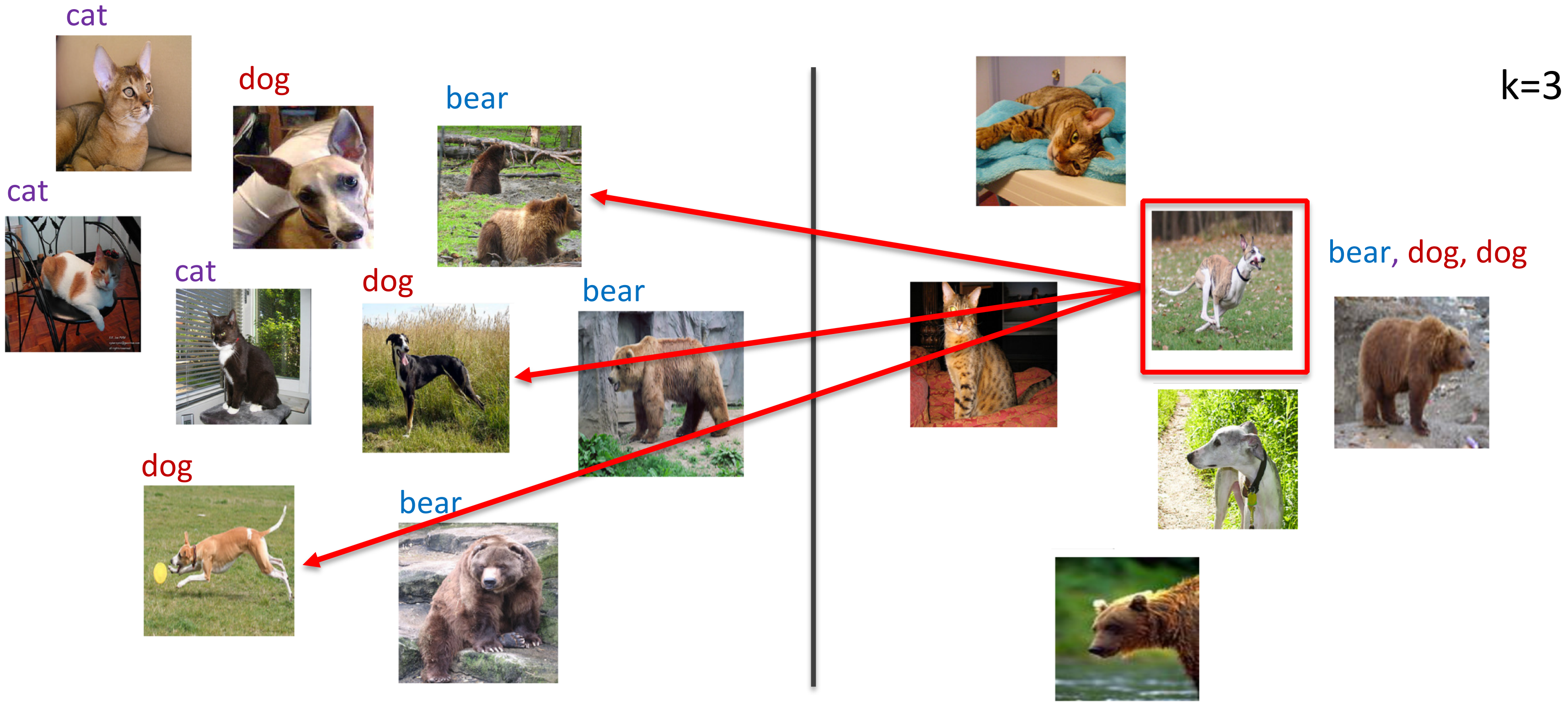


$= f($ The screen was a sea of red $)$

Supervised Learning – k-Nearest Neighbors



Supervised Learning – k-Nearest Neighbors



Supervised Learning – k-Nearest Neighbors

- How do we choose the right K?
- How do we choose the right features?
- How do we choose the right distance metric?

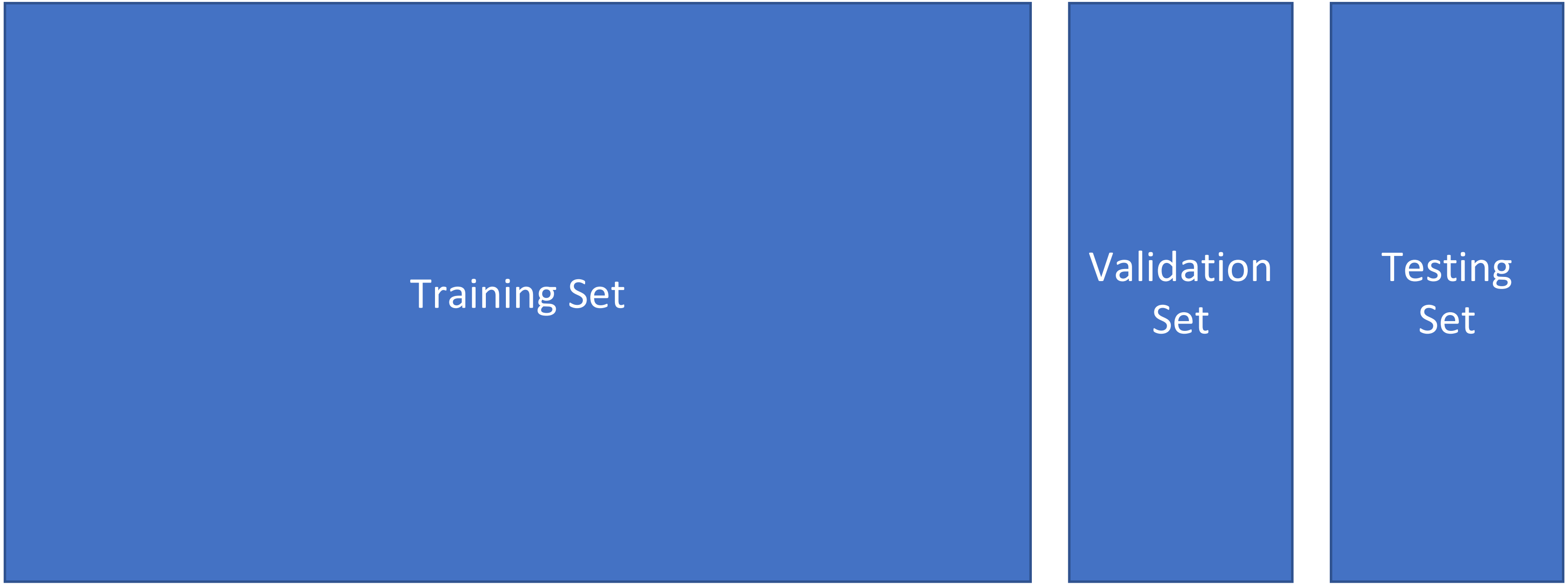
Supervised Learning – k-Nearest Neighbors

- How do we choose the right K?
- How do we choose the right features?
- How do we choose the right distance metric?

Answer: Just choose the one combination that works best!
BUT not on the test data.

Instead split the training data into a ”Training set” and a ”Validation set” (also called ”Development set”)

Training, Validation (Dev), Test Sets



Training, Validation (Dev), Test Sets

Used during development

Training, Validation (Dev), Test Sets

Only to be used for evaluating the model at the very end of development and any changes to the model after running it on the test set, could be influenced by what you saw happened on the test set, which would invalidate any future evaluation.

Unsupervised Learning – k-means clustering

k = 3

1. Initially assign all images to a random cluster

Unsupervised Learning – k-means clustering

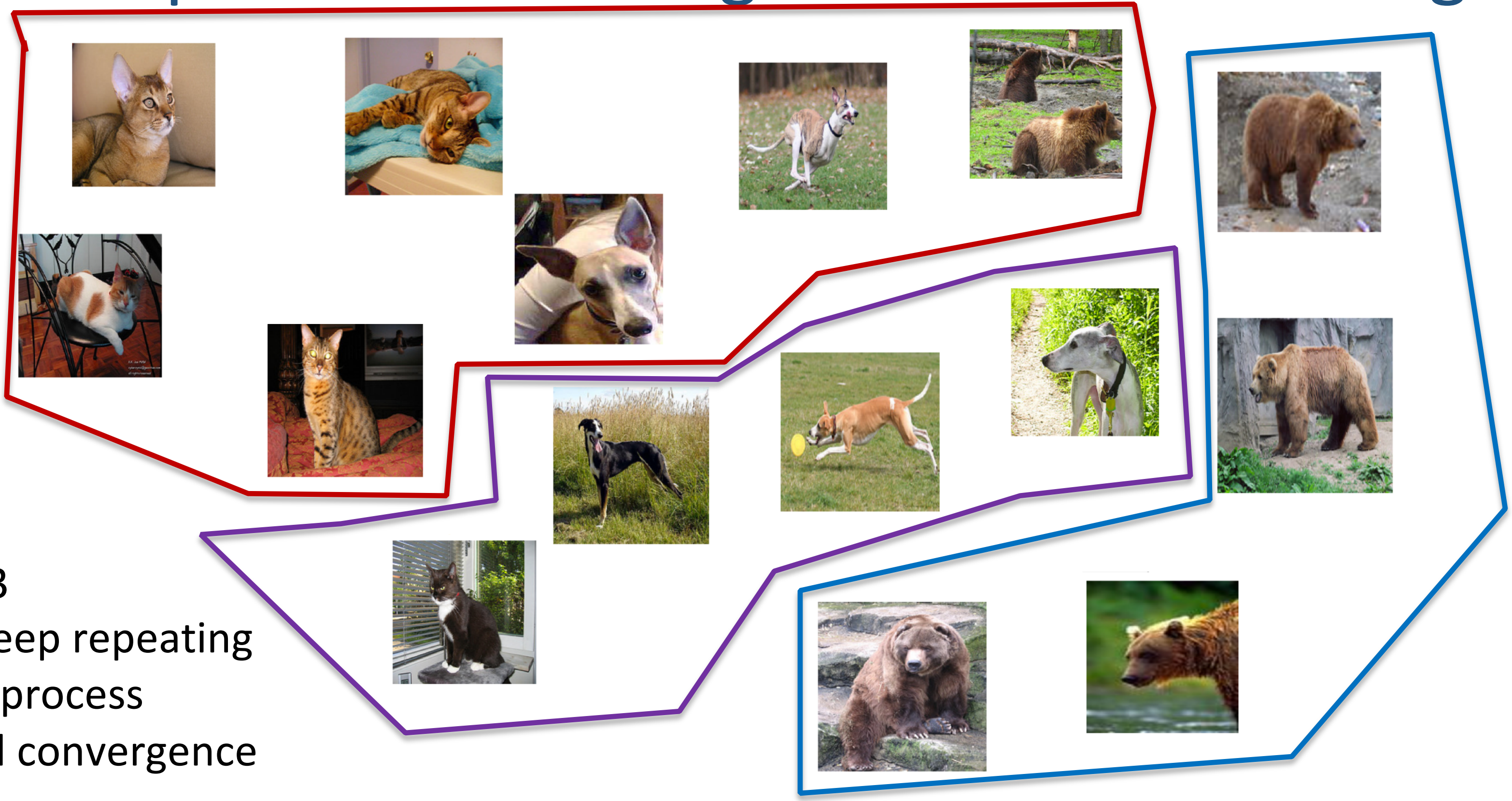
k = 3
2. Compute the mean image (in feature space) for each cluster

Unsupervised Learning – k-means clustering

k = 3

3. Reassign images to clusters based on similarity to cluster means

Unsupervised Learning – k-means clustering



k = 3

4. Keep repeating this process until convergence

Unsupervised Learning – k-means clustering

k = 3

4. Keep repeating this process until convergence

Unsupervised Learning – k-means clustering

k = 3
4. Keep repeating this process until convergence

Unsupervised Learning – k-means clustering

- How do we choose the right K?
- How do we choose the right features?
- How do we choose the right distance metric?
- How sensitive is this method with respect to the random assignment of clusters?

Answer: Just choose the one combination that works best!
BUT not on the test data.

Instead split the training data into a ”Training set” and a ”Validation set” (also called ”Development set”)

Supervised Learning - Classification

Training Data


cat

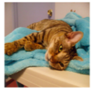

dog


cat

.
.
.


bear

Test Data



.
.
.

Supervised Learning - Classification

Training Data

$x_1 = [$  $]$ $y_1 = [$cat $]$

$x_2 = [$  $]$ $y_2 = [$dog $]$

$x_3 = [$  $]$ $y_3 = [$cat $]$

.
.
.

$x_n = [$  $]$ $y_n = [$bear $]$

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

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

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

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

.
.
.

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

Supervised Learning – Linear Softmax

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \quad x_{12} \quad x_{13} \quad x_{14}]$	$y_1 = [1 \quad 0 \quad 0]$	$\hat{y}_1 = [0.85 \quad 0.10 \quad 0.05]$
$x_2 = [x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]$	$y_2 = [0 \quad 1 \quad 0]$	$\hat{y}_2 = [0.20 \quad 0.70 \quad 0.10]$
$x_3 = [x_{31} \quad x_{32} \quad x_{33} \quad x_{34}]$	$y_3 = [1 \quad 0 \quad 0]$	$\hat{y}_3 = [0.40 \quad 0.45 \quad 0.15]$
$\vdots$		
$x_n = [x_{n1} \quad x_{n2} \quad x_{n3} \quad x_{n4}]$	$y_n = [0 \quad 0 \quad 1]$	$\hat{y}_n = [0.40 \quad 0.25 \quad 0.35]$

Supervised Learning – Linear Softmax

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

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

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

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

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

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

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

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)

$\lambda = 0.01$

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

Initialize w and b randomly

for e = 0, num_epochs **do**

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

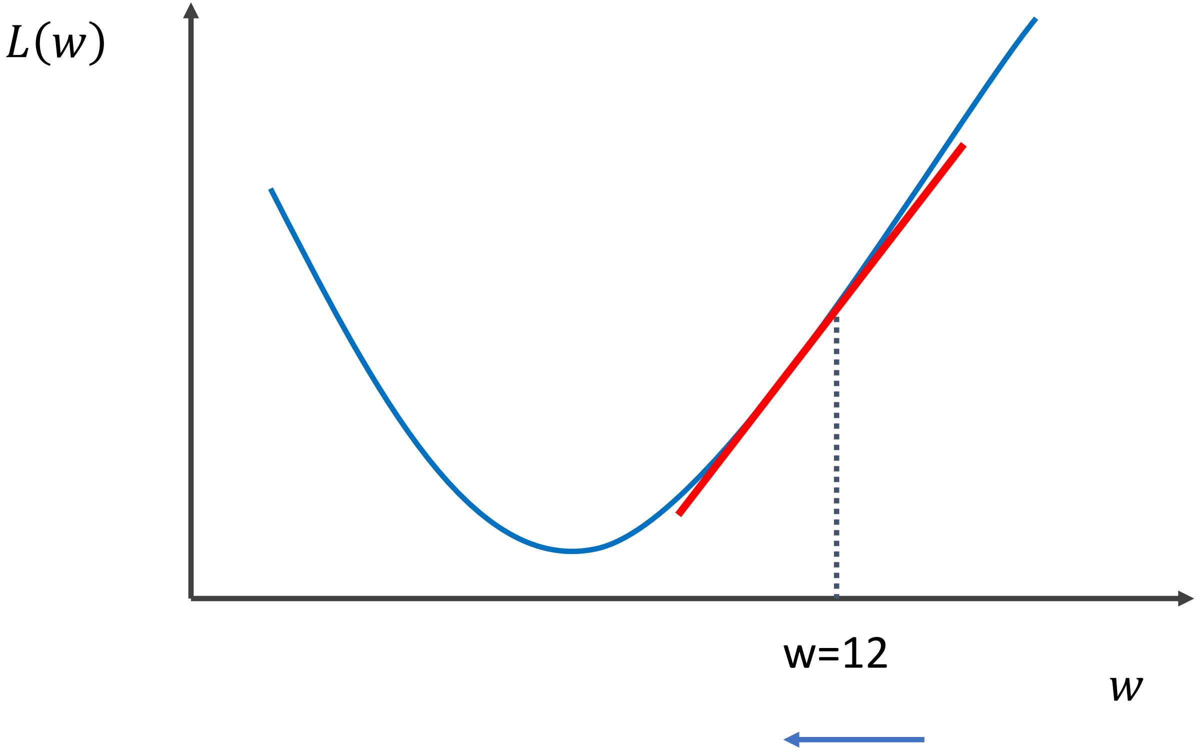
Update w: $w = w - \lambda\, dL(w,b)/dw$

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) (idea)



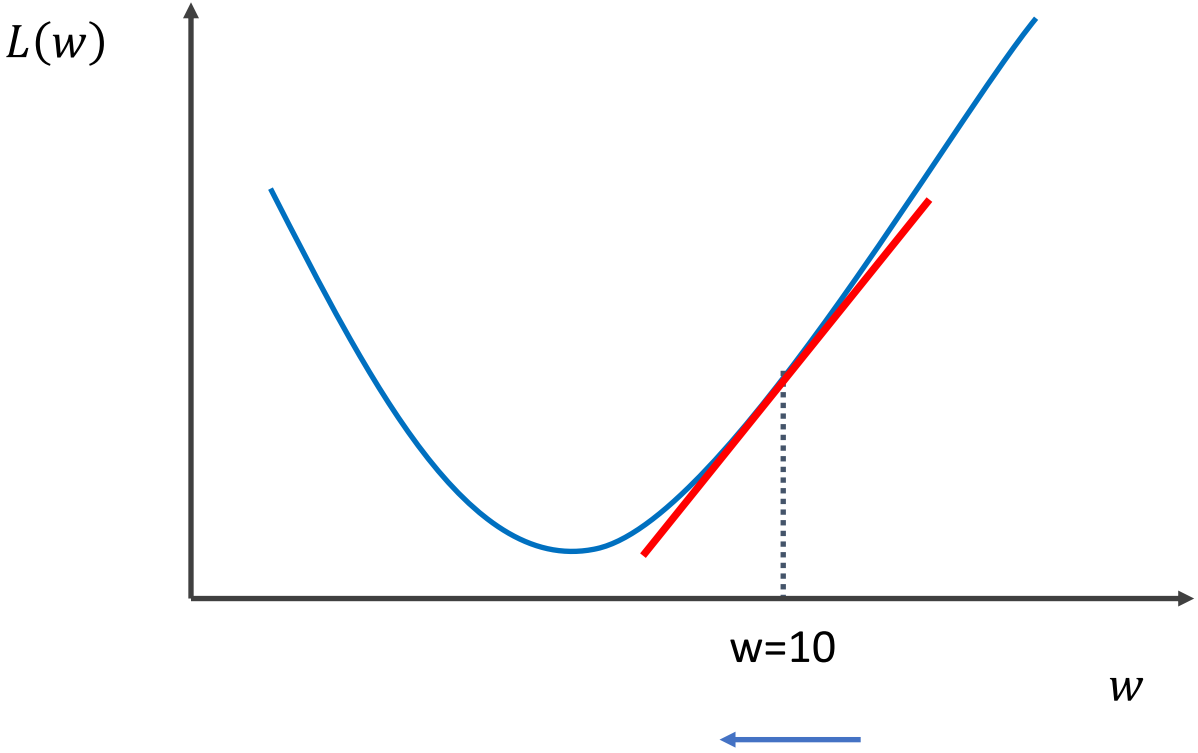
1. Start with a random value of w (e.g. w = 12)

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)

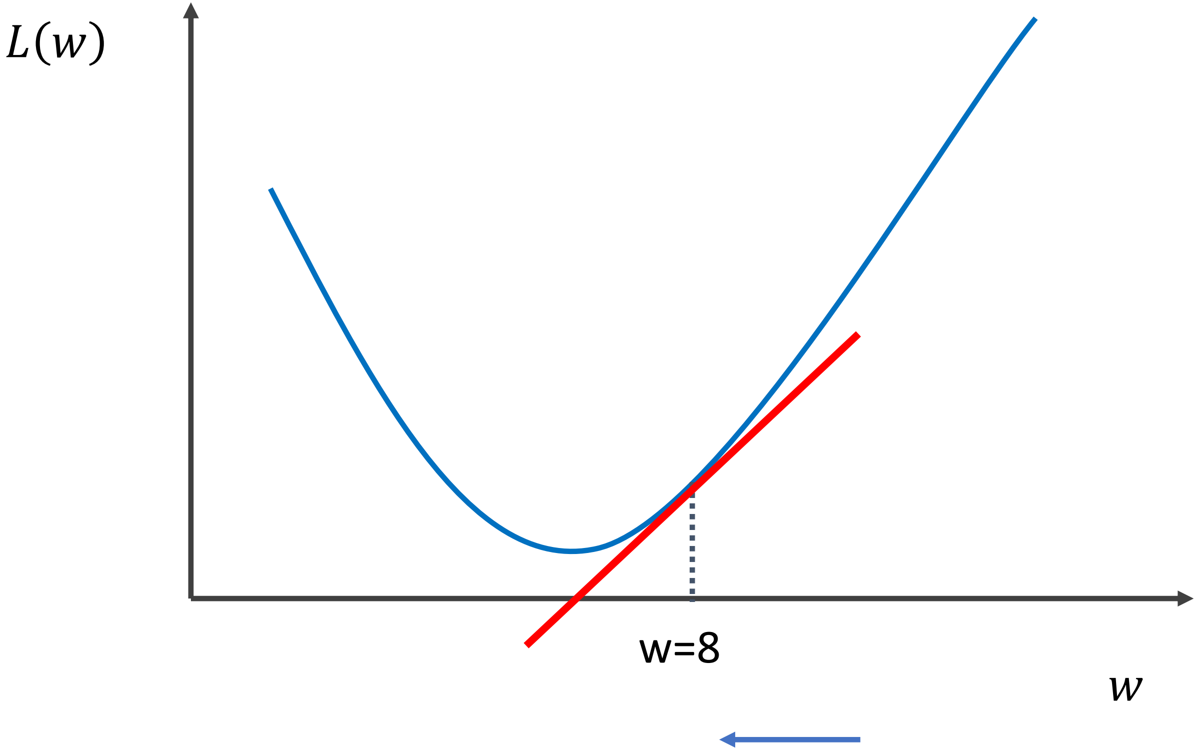


2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

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)

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

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

Gradient Descent (GD)

$\lambda = 0.01$

Initialize w and b randomly

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

expensive

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

(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 $dl(w,b)/db$

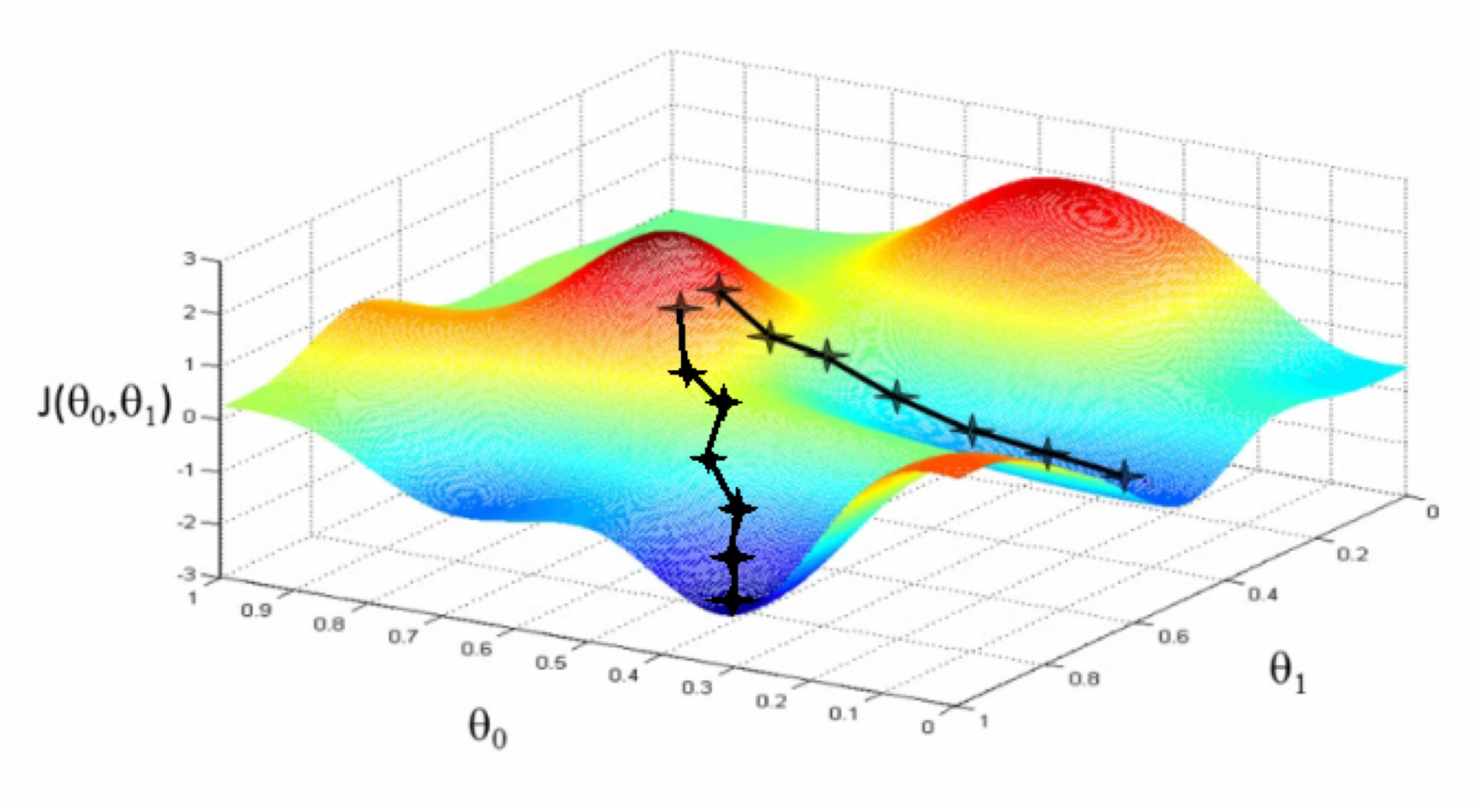
Update w: $w = w - \lambda\, dl(w,b)/dw$

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



Source: Andrew Ng

(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 $dl(w, b)/db$ 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: $a_i = (w_{i,1}x_1 + w_{i,2} + w_{i,3} + w_{i,4}) + b_i$

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

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

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

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

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

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

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

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

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

Computing Analytic Gradients

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

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

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

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

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

when $i \neq label$:

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

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

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

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

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

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

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

when $i \neq label$:

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

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

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

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

Computing Analytic Gradients

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

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

when $i = label$:

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

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

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

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

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

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

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

More …

- Regularization
- Momentum updates
- Hinge Loss, Least Squares Loss, Logistic Regression Loss

Assignment 2 – Linear 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]

Supervised Learning – Linear Softmax

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

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

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

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

How do we find a good w and b?

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

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?

Regression vs Classification

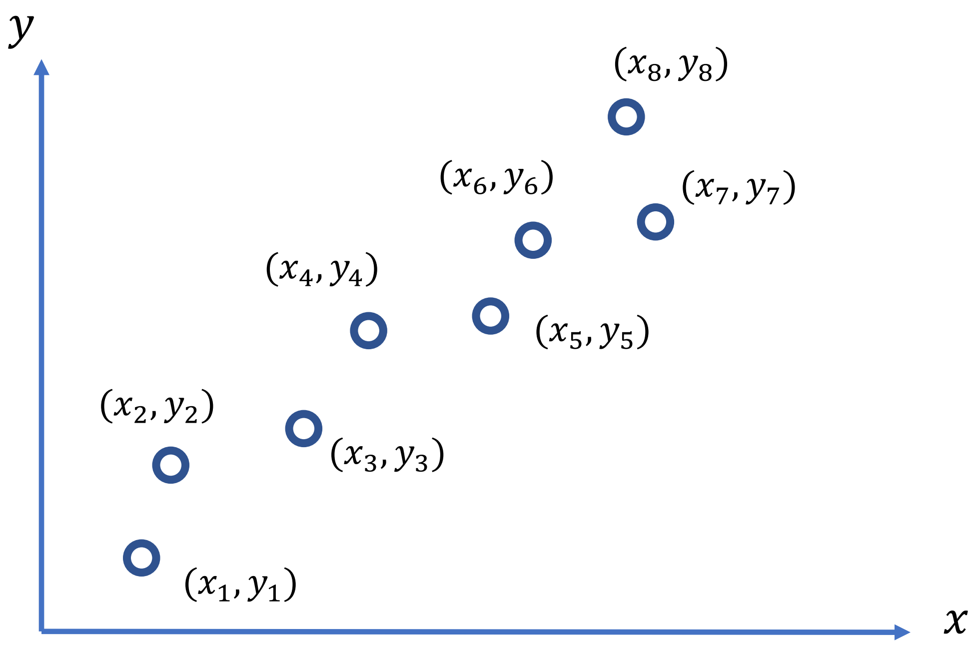
Regression

- 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.

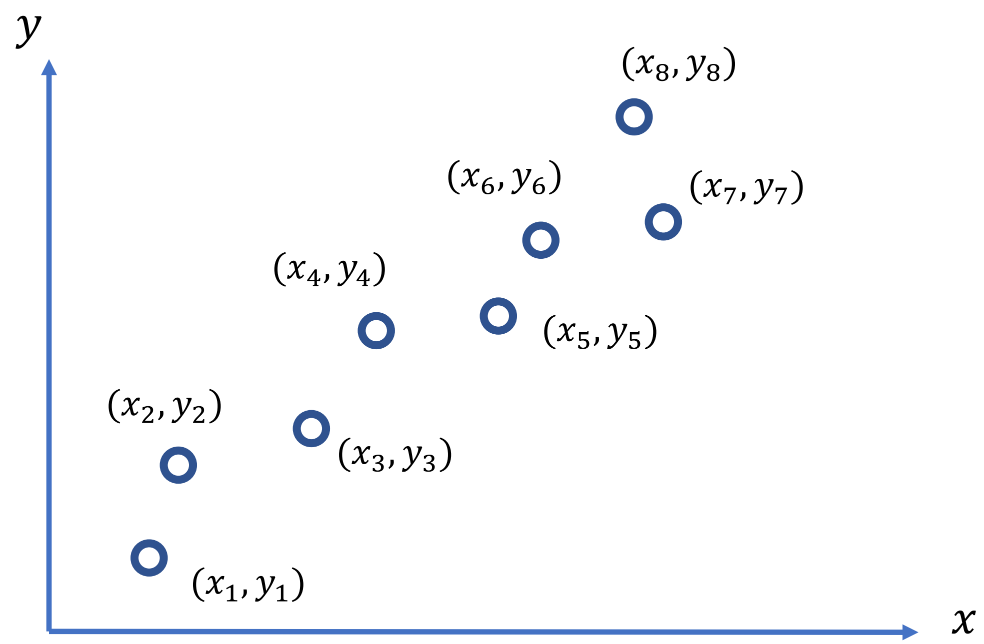
Classification

- 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

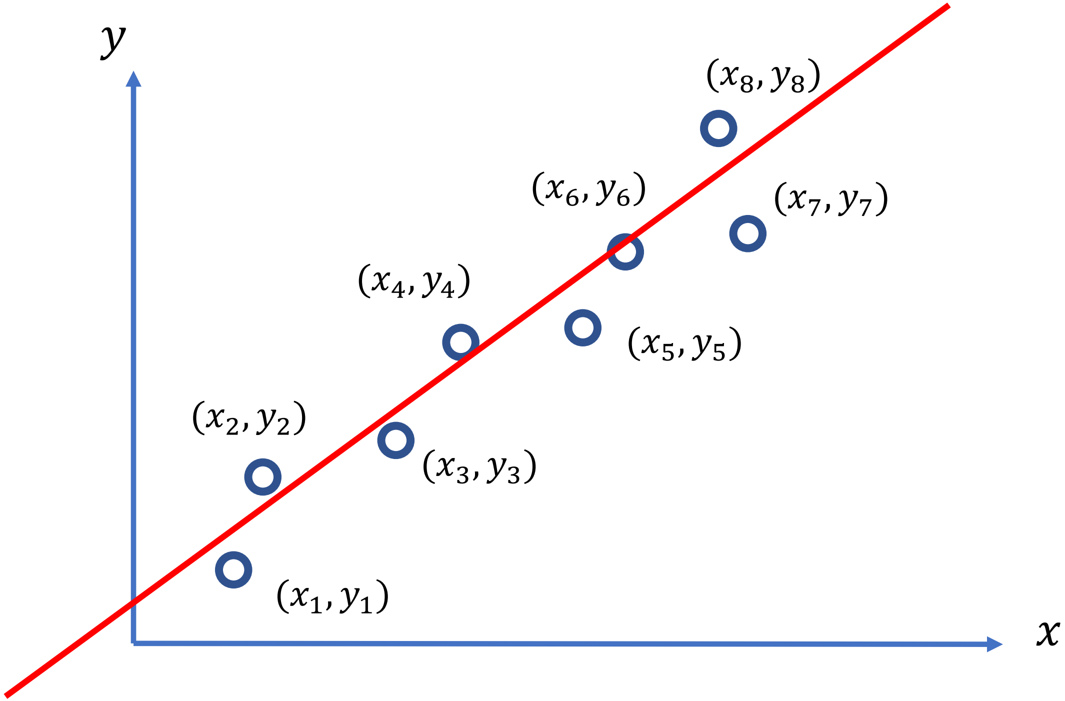


Linear Regression – 1 output, 1 input



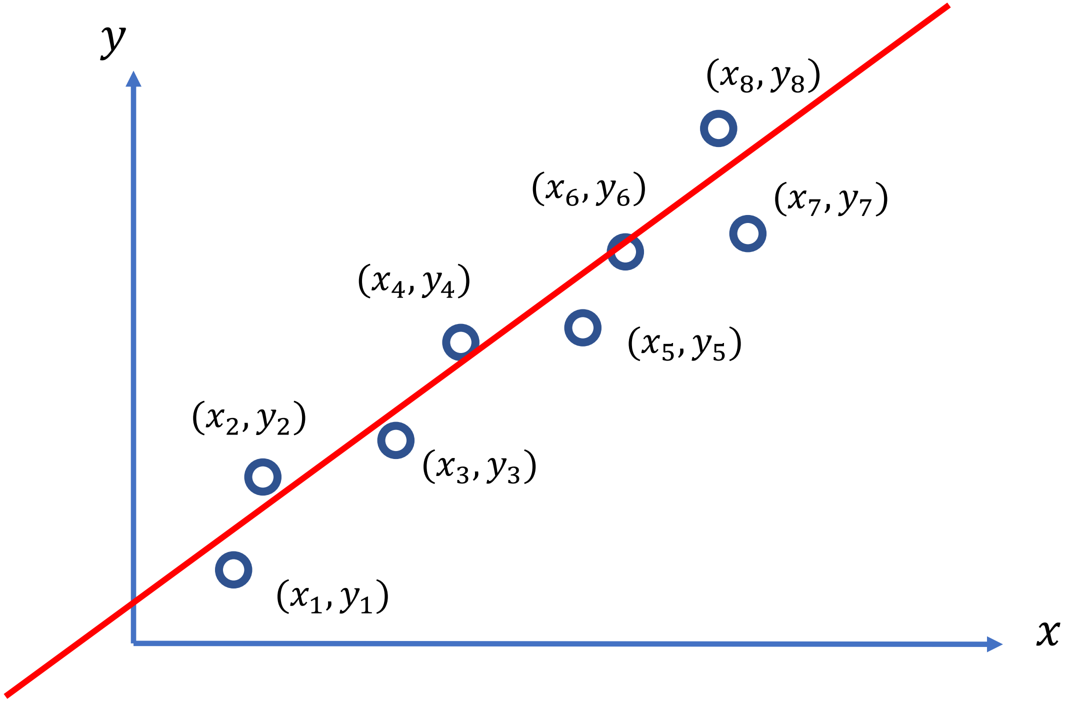
Model: $\hat{y} = wx + b$

Linear Regression – 1 output, 1 input



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

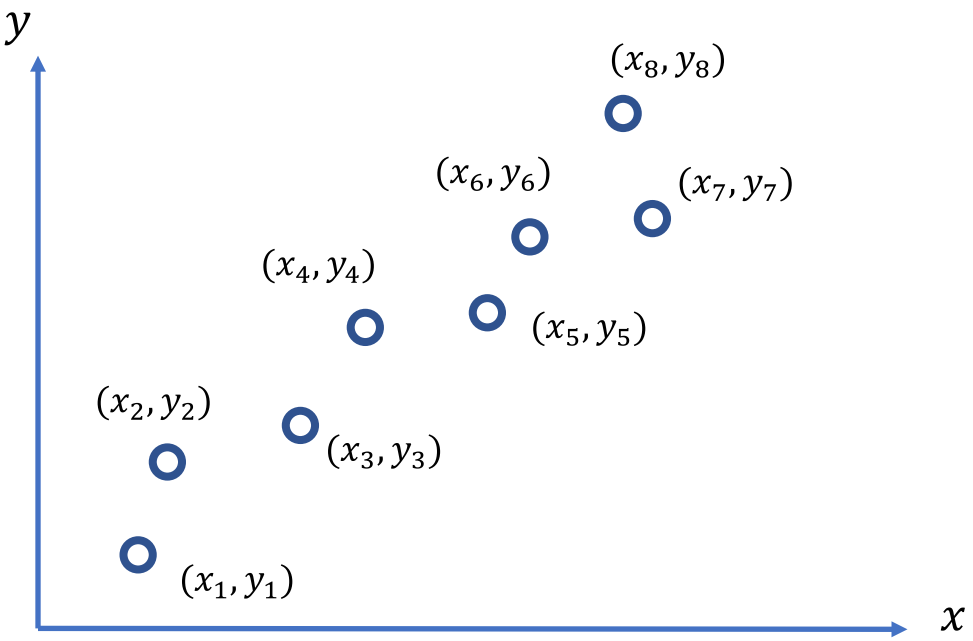
Linear Regression – 1 output, 1 input



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

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

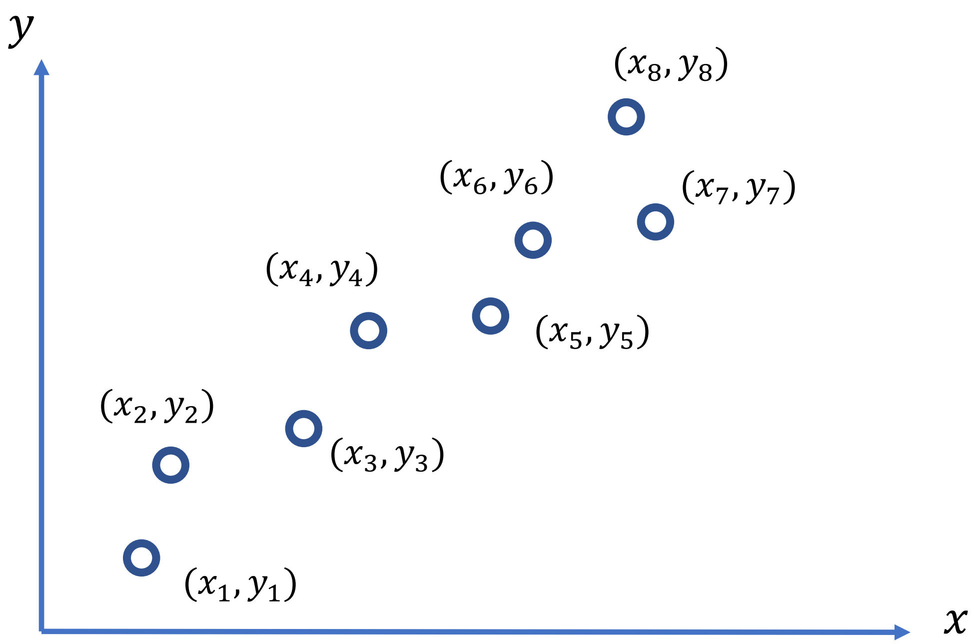
Quadratic Regression



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

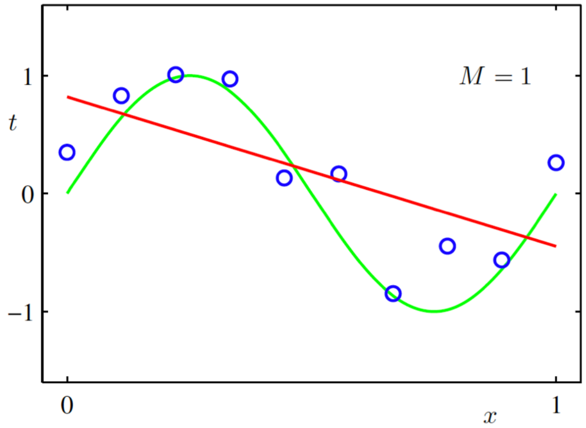


Model: $\hat{y} = w_n x^n + \cdots + w_1 x + b$

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

Overfitting

f is linear

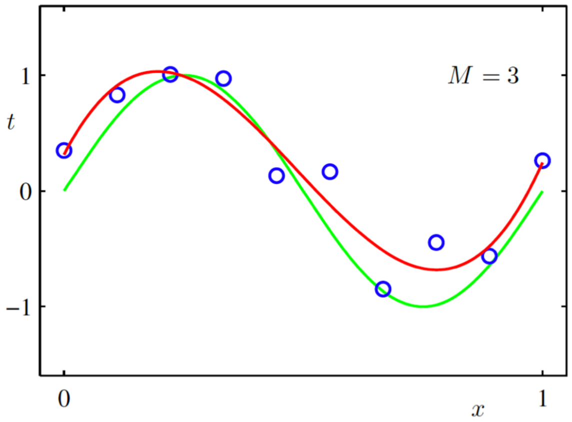


$Loss(w)$ is high

Underfitting

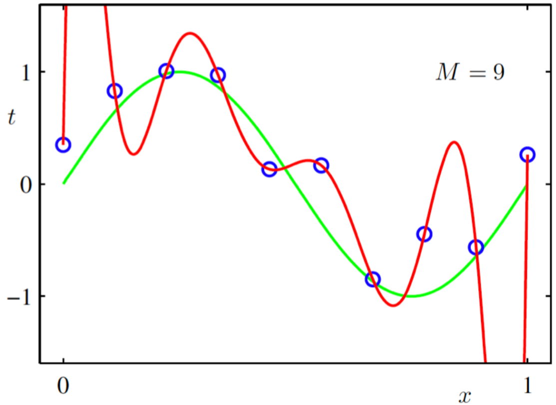
High Bias

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: $dl(w, b)/dw$ 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

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

global v

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

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

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

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

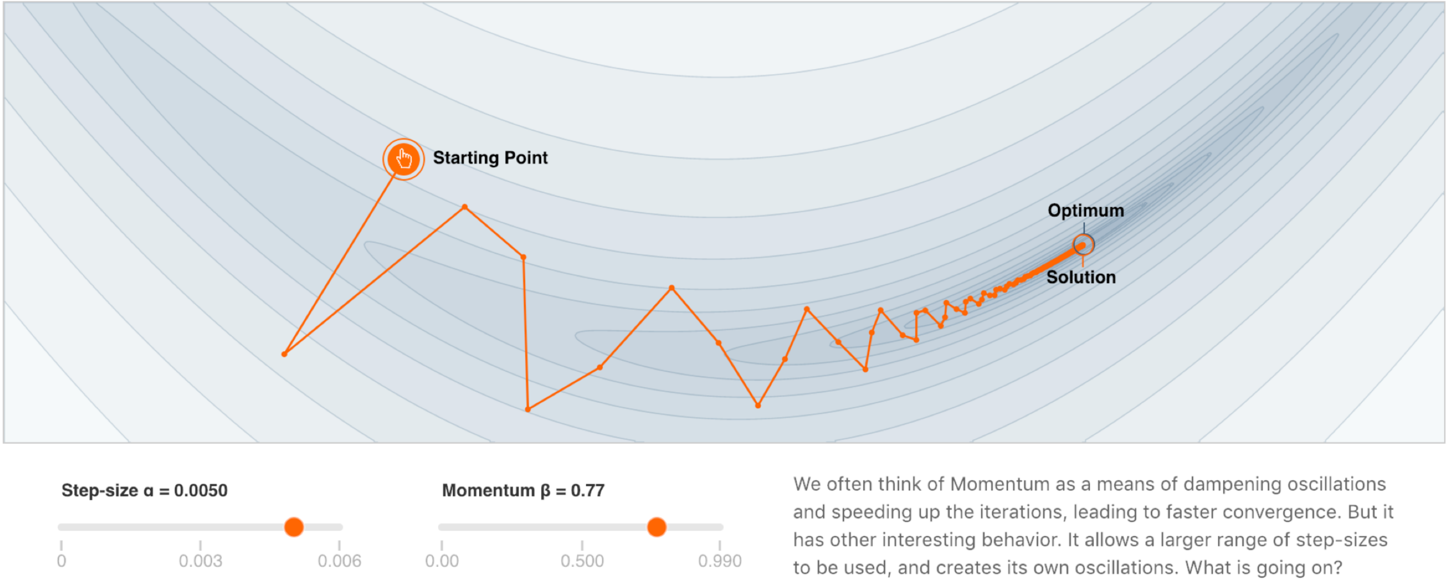
Print: $l(w,b)$ // Useful to see if this is becoming smaller or not.

end

end

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

More on Momentum

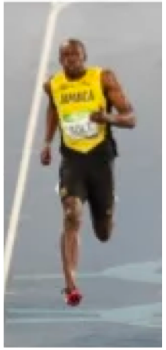


We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

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

Image Features: HoG

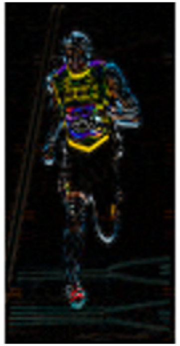
Compute gradients



$*$ $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ →

I_x

I_y



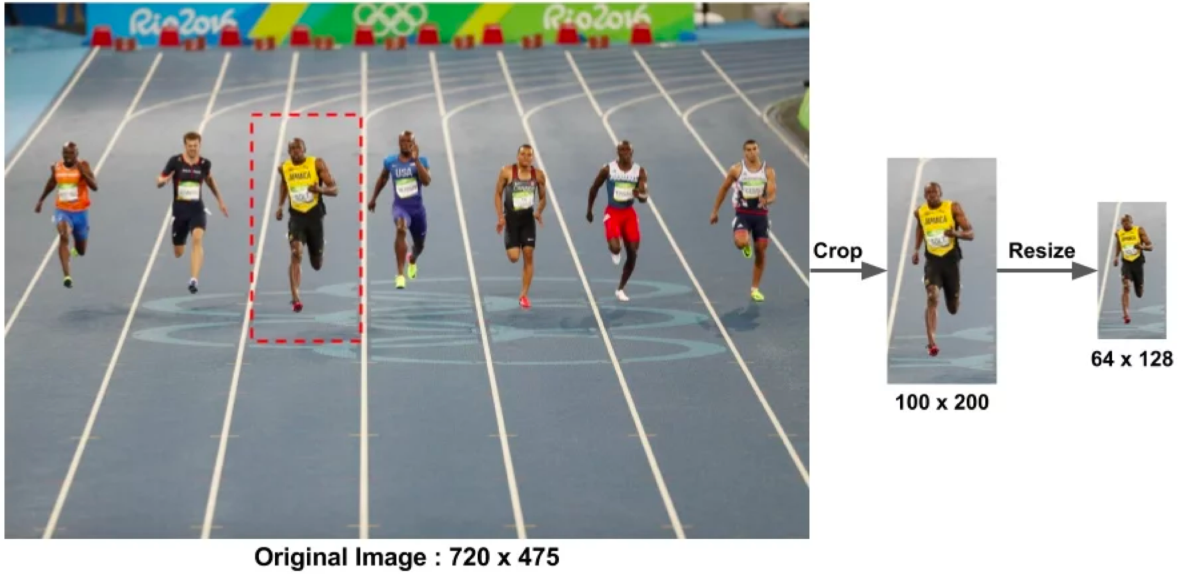
$\sqrt{I_x^2 + I_y^2}$

Paper by Navneet Dalal & Bill Triggs presented at CVPR 2005 for detecting people.

Scikit-image implementation

Images by Satya Mallick https://www.learnopencv.com/histogram-of-oriented-gradients/

Image Features: HoG



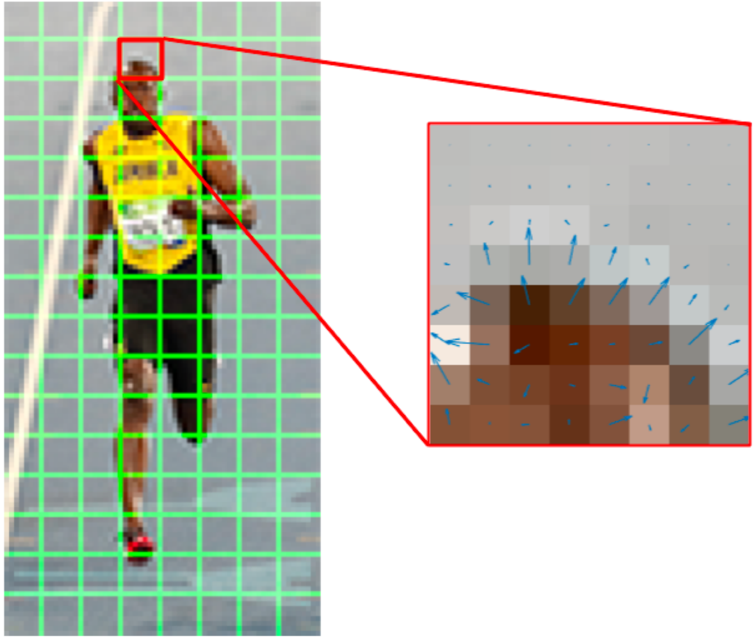
Paper by Navneet Dalal & Bill Triggs presented at CVPR 2005 for detecting people.

Scikit-image implementation

Images by Satya Mallick https://www.learnopencv.com/histogram-of-oriented-gradients/

Image Features: HoG

We will aggregate gradient magnitude and directions in 8x8 pixel regions



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

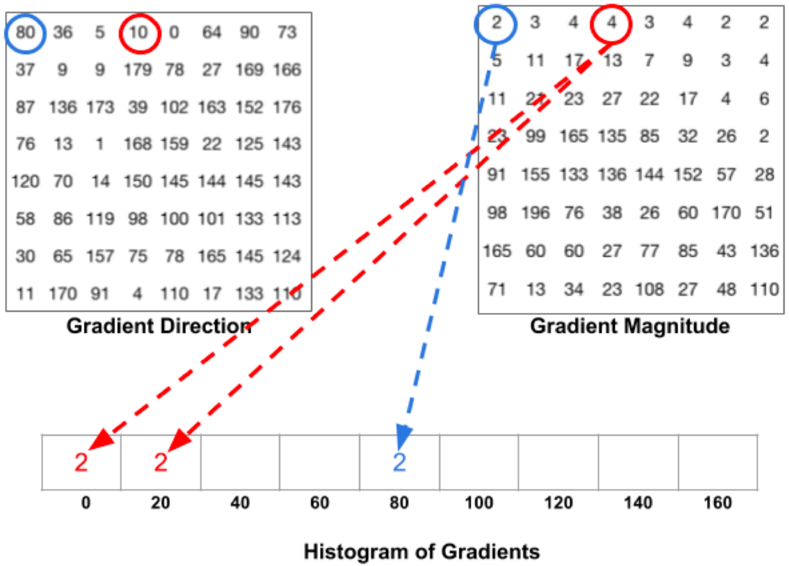
Paper by Navneet Dalal & Bill Triggs presented at CVPR 2005 for detecting people.

Scikit-image implementation

Images by Satya Mallick https://www.learnopencv.com/histogram-of-oriented-gradients/

Image Features: HoG

Compute a histogram with 9 bins for angles from 0 to 180



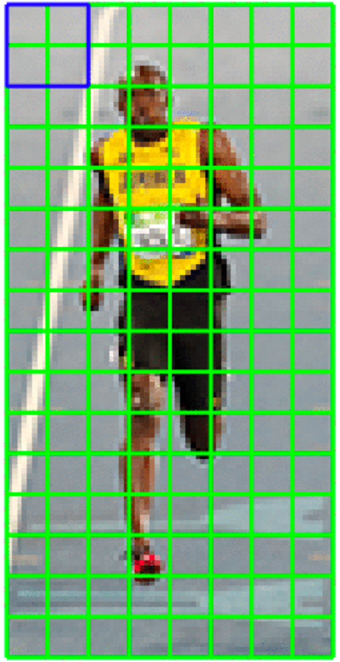
Paper by Navneet Dalal & Bill Triggs presented at CVPR 2005 for detecting people.

Scikit-image implementation

Images by Satya Mallick https://www.learnopencv.com/histogram-of-oriented-gradients/

Image Features: HoG

Normalize histograms with respect to histograms of adjacent neighbors.



Paper by Navneet Dalal & Bill Triggs presented at CVPR 2005 for detecting people.

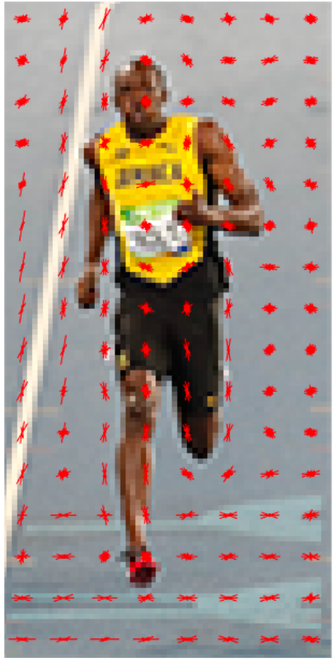
Scikit-image implementation

Images by Satya Mallick https://www.learnopencv.com/histogram-of-oriented-gradients/

Image Features: HoG

Image (or image region) represented by a vector containing all the histograms.

In this case how long is that vector?

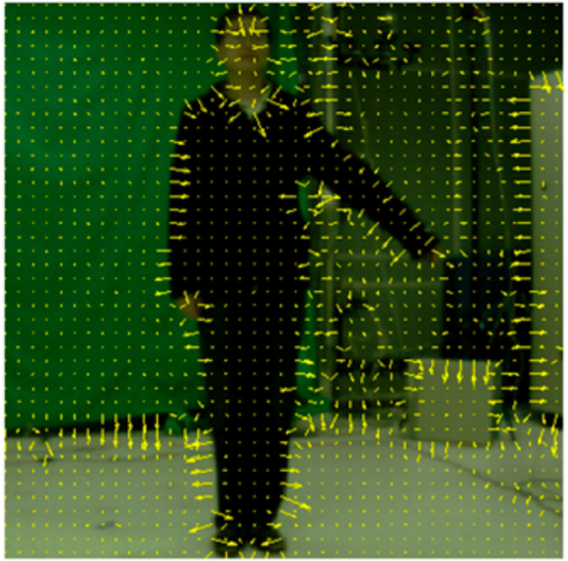


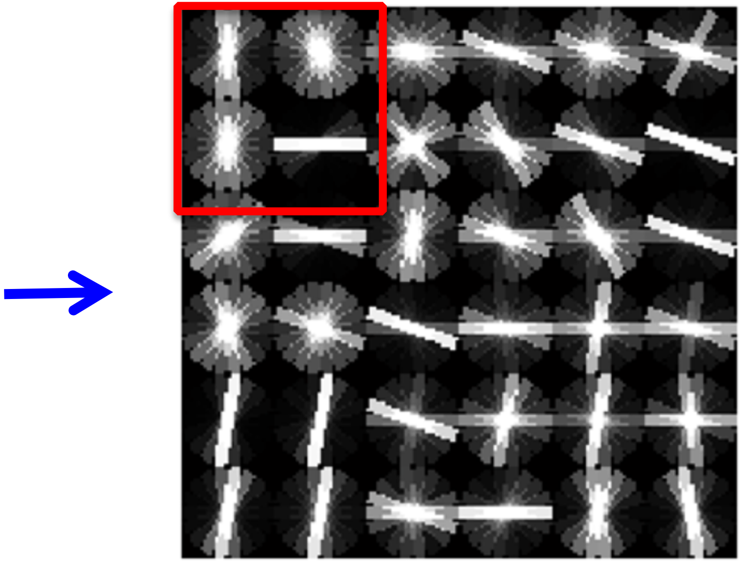
Paper by Navneet Dalal & Bill Triggs presented at CVPR 2005 for detecting people.

Scikit-image implementation

Images by Satya Mallick https://www.learnopencv.com/histogram-of-oriented-gradients/

Image Features: HoG



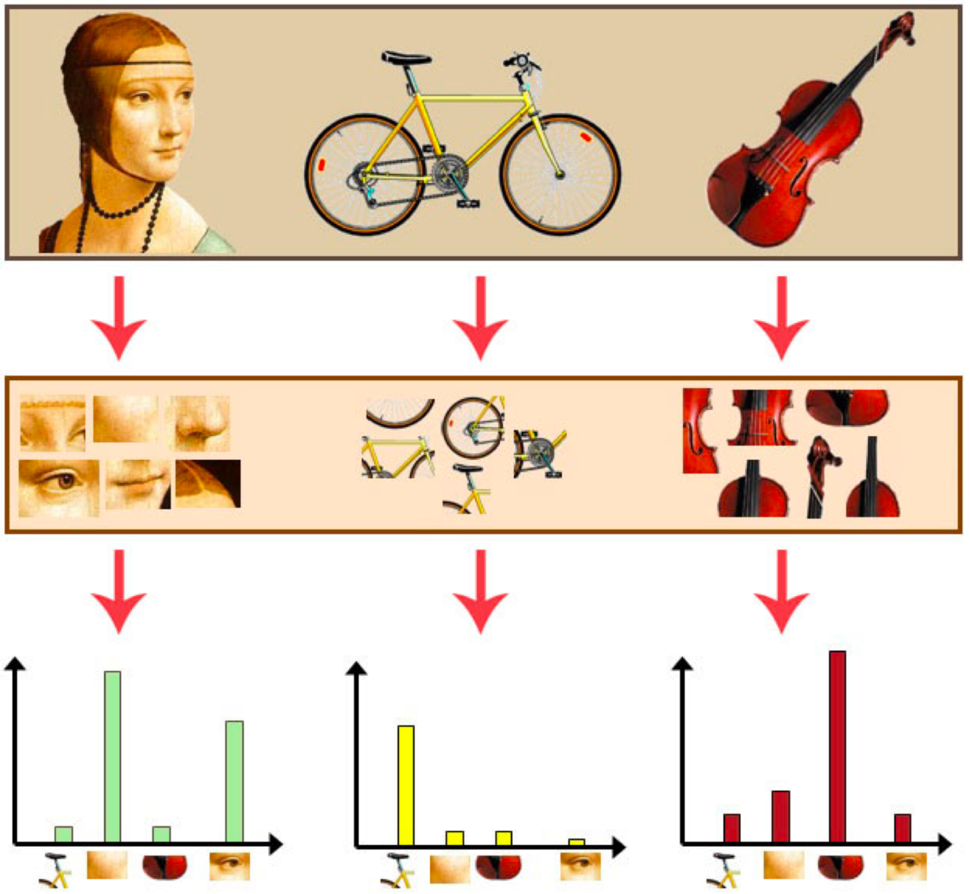


+ Block Normalization

Paper by Navneet Dalal & Bill Triggs presented at CVPR 2005 for detecting people.
Figure from Zhuolin Jiang, Zhe Lin, Larry S. Davis, ICCV 2009 for human action recognition.

Extract SIFT Feature Descriptors

Compute Histograms of Features



slide by Fei-fei Li

Summary: Image Features

- Largely replaced by Neural networks
- Still useful to study for inspiration in designing neural networks that compute features.

- Many other features proposed
  - LBP: Local Binary Patterns: Useful for recognizing faces.
  - Dense SIFT: SIFT features computed on a grid similar to the HOG features.
  - etc.

Questions?