Probabilistic Classification

MLFA

Generative Model

- A story about how the observed data were “born”
- Story in the language of probability!

- Treat labels Y and features X as random variables
- Story outline:
- - Y created first
- - X created based on Y

Generative Model

- A story about how the observed data were “born”
- Story in the language of probability!

- Treat labels Y and features X as random variables
- Story outline:
- - Y created first p(Y): prior distribution
- - X created based on Y p(X|Y):class-conditional distribution
- Probabilistic Classification: p(Y|X): posterior distribution

Prior Distribution

- Prior: information before observing X

	Y =1	Y = 2	Y = 3	Y = 4
	80	50	40	30

- P(Y = k) = ?
- Frequentist approach: just relative frequencies!

	Y = 1	Y = 2	Y = 3	Y = 4
	0.4	0.25	0.20	0.15

Posterior Distribution

- Posterior: information after observing X
- $P(Y = k \mid X) = ?$

- Bayes Theorem:
- $P(Y = k \mid X) = (p(X \mid Y = k) * p(Y = k)) / p(X)$

$$= K * p(X \mid Y = k) * p(Y = k)$$

Class-conditional Distribution

	Y = 1	Y = 2	Y = 3	Y = 4
X < 15	40	45	10	5
X > 15	40	5	30	25

- $P(X \mid Y = k) = ??$
- $P(X<15 \mid Y=1) = 40/(40+40) = 0.5$
- $P(X>15 \mid Y=3) = 30/(30+10) = 0.75$

- $P(Y = k \mid X) = ???$

Frequentist Approach: Direct estimation

	Y = 1	Y = 2	Y = 3	Y = 4	
X < 15	40	45	10	5	100
X > 15	40	5	30	25	100

- $P(Y = 1 \mid X<15) = 40 / 100 = 0.4$

	Y = 1	Y = 2	Y = 3	Y = 4	
$p(Y \mid X < 15)$	0.4	0.45	0.10	0.05	1.0
$p(Y \mid X > 15)$	0.4	0.05	0.30	0.25	1.0

- Similar to Decision Trees

Bayesian Approach: Posterior Distribution

	Y = 1	Y = 2	Y = 3	Y = 4
X < 15	40	45	10	5
X > 15	40	5	30	25

- P(Y = 1 | X<15) = K * p(X<15 | Y=1) * p(Y=1) = K*(40/80)*(80/200)
- P(Y = 2 | X<15) = K * p(X<15 | Y=2) * p(Y=2) = K*(45/50)*(50/200)
- P(Y = 3 | X<15) = K * p(X<15 | Y=3) * p(Y=3) = K*(10/40)*(40/200)
- P(Y = 4 | X<15) = K * p(X<15 | Y=4) * p(Y=4) = K* (5/30)*(30/200)
- K = ???

Posterior Distribution

	Y = 1	Y = 2	Y = 3	Y = 4
X < 15	40	45	10	5
X > 15	40	5	30	25

	Y = 1	Y = 2	Y = 3	Y = 4
Prior p(Y)	0.40	0.25	0.20	0.15
p(Y | X < 15)	0.40	0.45	0.10	0.05
P(Y | X > 15)	0.40	0.05	0.30	0.25

Bayesian or Frequentist?

- Frequentist approach: Estimate p(Y | X) from data
- Frequentist approach: More robust if some classes are rare
- Frequentist approach: More straightforward

- Bayesian approach: Estimate p(X | Y) and p(Y) from data
- Bayesian approach: More robust if some feature values are rare
- Bayesian approach: More interpretable in many real applications
- Huge advantage for high-dimensional features!!!

Probabilistic Classifier

- Predicted label : mode of the posterior distribution!
- Y_{pred} = argmax_k p (Y = k | X)
- Confidence of the prediction = p(Y = Y_{pred}| X)

- If Bayesian approach used for p(Y | X): Bayesian Classifier!

Posterior Distribution

	Y = 1	Y = 2	Y = 3	Y = 4
X < 15	40	45	10	5
X > 15	40	5	30	25

	Y = 1	Y = 2	Y = 3	Y = 4	Ypred
Prior	0.40	0.25	0.20	0.15	1
X < 15	0.40	0.45	0.10	0.05	2
X > 15	0.40	0.05	0.30	0.25	1

Multi-dimensional Features

	Y = 1	Y = 2	Y = 3	Y = 4
X1<15	40	45	10	5
X1>15	40	5	30	25
X2 = a	40	30	15	30
X2 = b	40	20	25	0

- P(Y = k | X1=12, X2=a) = ????
- We need Joint Distribution of the features!!!

Multi-dimensional Features

	Y = 1	Y = 2	Y = 3	Y = 4
X1<15, X2=a	30	25	0	5
X1<15, X2=b	10	20	10	0
X1>15, X2=a	10	5	15	25
X1>15, X2=b	30	0	15	0

- $P(Y = 3 \mid X1=12, X2 =b) = K * P(X1<15, X2=b \mid Y = 3) * P(Y=3)$

  $= ?$

Multi-dimensional Features

	Y = 1	Y = 2	Y = 3	Y = 4
X1<15, X2=a	0.375	0.50	0	0.166
X1<15, X2=b	0.125	0.40	0.25	0
X1>15, X2=a	0.125	0.10	0.375	0.837
X1>15, X2=b	0.375	0	0.375	0

- $P(Y = 3 \mid X1=12, X2 =b) = K * P(X1<15, X2=b \mid Y = 3) *P(Y=3)$

$$= K * 0.25 * 0.20$$

Multi-dimensional Features

	Y = 1	Y = 2	Y = 3	Y = 4
X1<15, X2=a	0.375 * 0.4 * K1	0.50 * 0.25 * K1	0 * 0.2 * K1	0.166 * 0.15 * K1
X1<15, X2=b	0.125 * 0.4 * K2	0.40 * 0.25 * K2	0.25 * 0.2 * K2	0 * 0.15 * K2
X1>15, X2=a	0.125 * 0.4 * K3	0.10 * 0.25 * K3	0.375 * 0.2 * K3	0.837 * 0.15 * K3
X1>15, X2=b	0.375 * 0.4 * K4	0 * 0.25 * K4	0.375 * 0.2 * K4	0 * 0.15 * K4

- P(Y = 3 | X1=12, X2 =b) = K2 * P(X1<15, X2=b | Y = 3) *P(Y=3)

  = K2 * 0.25 * 0.20

  = ??

Naïve Bayes Classifier

- D-dimensional feature vector, M values each
- Rows of table = M**D

- Assumption: all features are independent (Naïve!)
- $P(X1<15, X2 =b) = p(X1<15) * p(X2=b)$

- D tables, rows of each table = M
- Naïve, but computationally efficient!

Naïve Bayes Classification

- $P(Y = k \mid X1<15, X2=b) = K * p(X1<15, X2=b \mid Y = k) * p(Y = k)$

  $= K * p(X1<15 \mid Y = k) * p(X2=b \mid Y = k) * p(Y = k)$

Final prediction = $\text{argmax}_k\ p(X_1 \mid Y=k)\ p(X_2 \mid Y=k)* \ldots\ldots\ p(X_D \mid Y=k)*p(Y=k)$

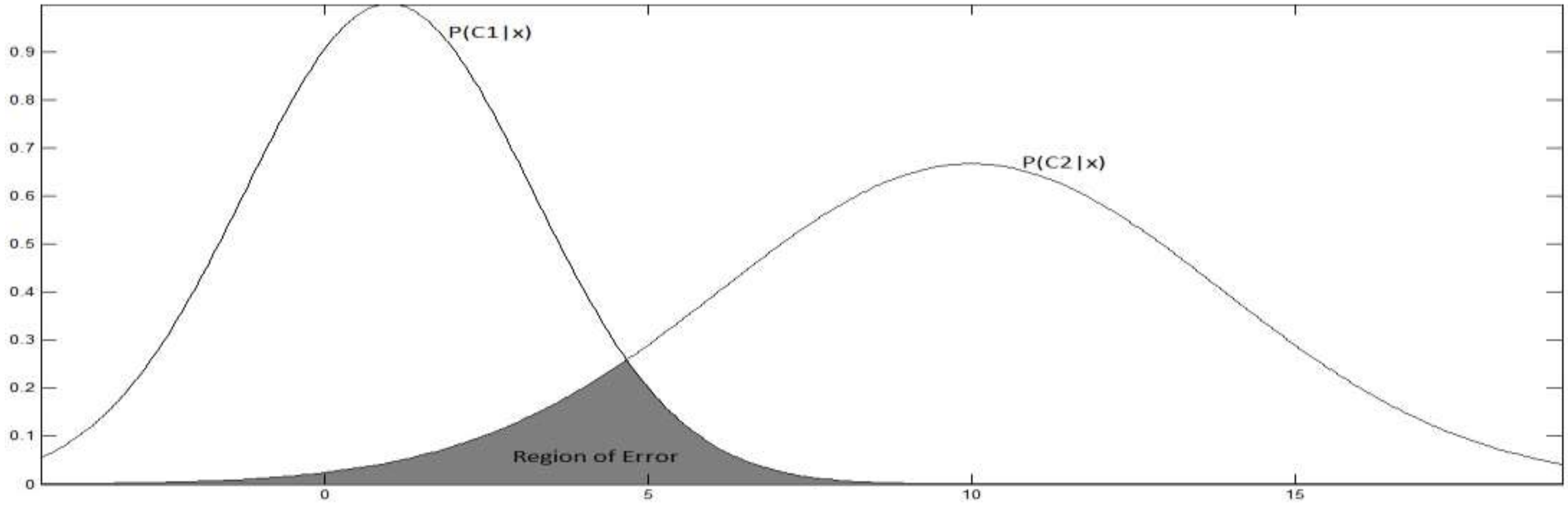
Confidence = $\max_k\ p(X_1 \mid Y=k)\ p(X_2 \mid Y=k)* \ldots\ldots\ p(X_D \mid Y=k)*p(Y=k)$

Error in Bayes Classifier

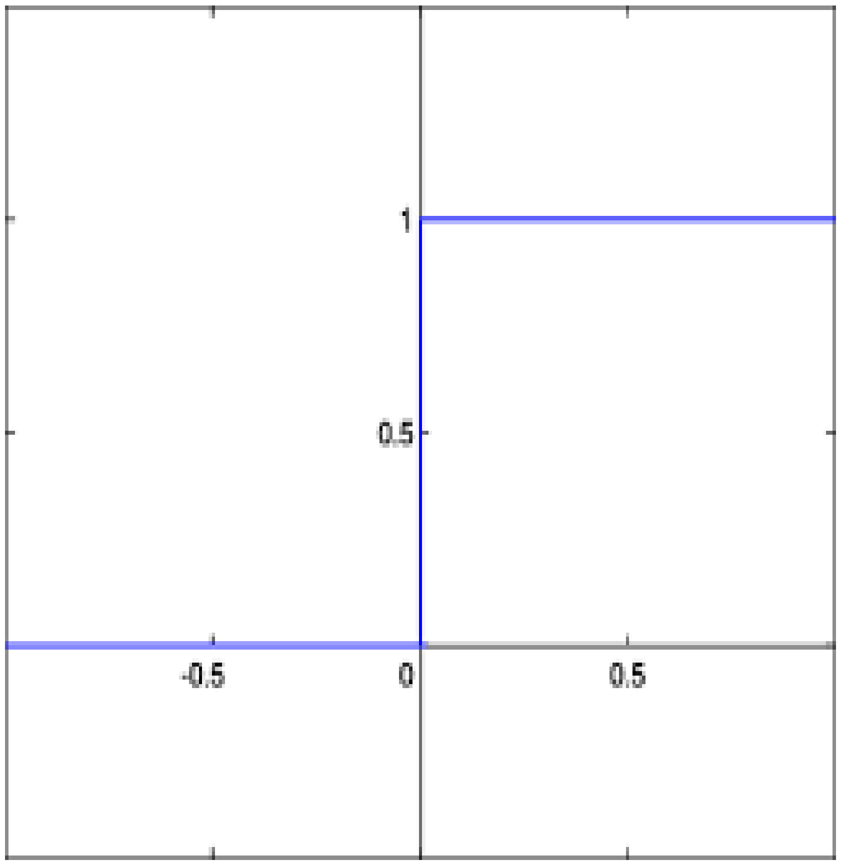
- Bayes error probability = total probability of the non-mode classes!
- Risk of prediction = 1 – confidence of prediction
- Bayes error = expected risk (expectation over all X and Y)

Error in Bayes Classifier

- Bayes error probability = total probability of the non-mode classes!
- Risk of prediction = 1 – confidence of prediction
- Bayes error = expected risk (expectation over all X and Y)

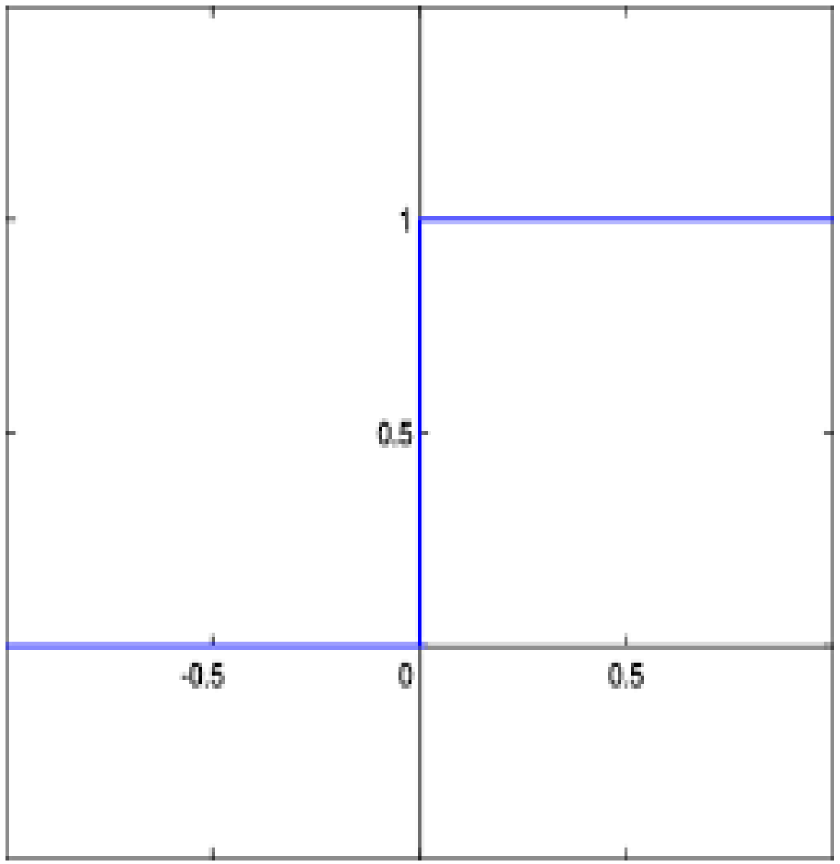


Logistic Regression for Classification

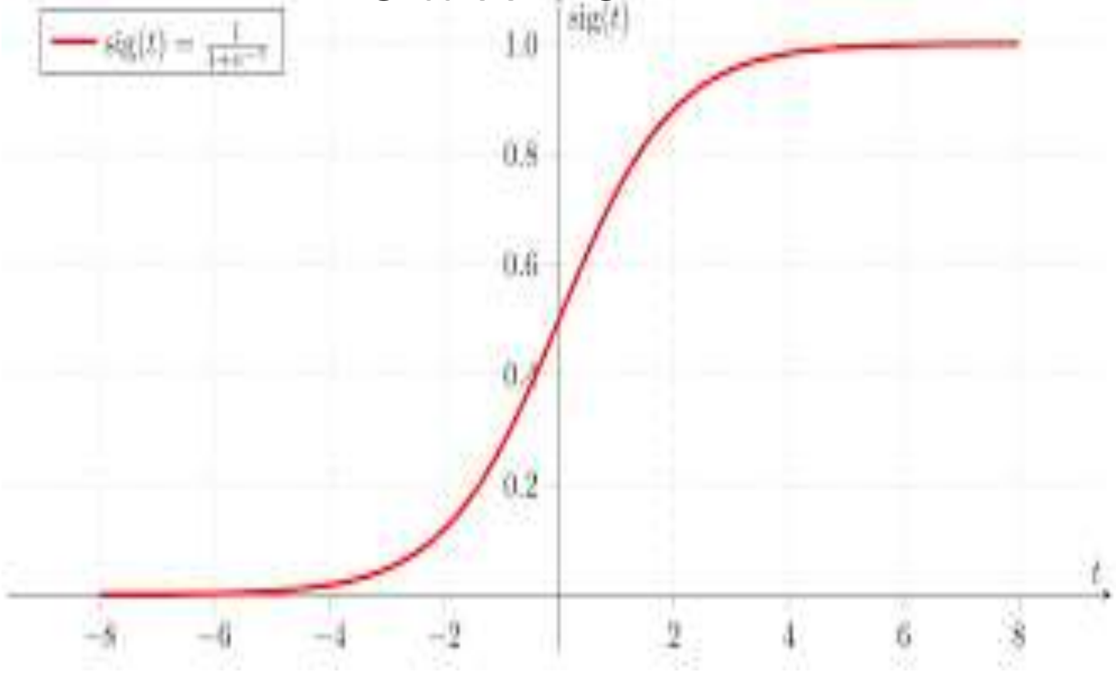


$$P(Y = 1 \mid X) = 1 \text{ if } X > 0$$
$$= 0 \text{ if } X < 0$$

Logistic Regression for Classification



$$P(Y = 1 \mid X) = 1 \text{ if } X > 0$$
$$= 0 \text{ if } X < 0$$



Logistic Regression

- $P(Y = 1 \mid X) = 1$ if $X > 0$
  $= 0$ if $X < 0$
- Approximation: $p(Y = 1 \mid X) = 1/(1 + \exp(-X))$
- Multi-dimensional features: consider weighted combination w.X
- $P(Y = 1 \mid X) = 1/(1 + \exp(-w.X))$ LOGISTIC

Logistic Regression

- $P(Y = 1 \mid X) = 1$ if $X > 0$

  $= 0$ if $X < 0$
- Approximation: $p(Y = 1 \mid X) = 1/(1 + \exp(-X))$
- Multi-dimensional features: consider weighted combination w.X
- $P(Y = 1 \mid X) = 1/(1 + \exp(-w.X))$ LOGISTIC
- But how to find w? REGRESSION!

Logistic Regression Loss Function

- Logistic regression: probabilistic binary classification
- prob(y=1 | x) = σ(w.x + b) = 1/(1 + exp(-w.x-b))   [w,b are parameters]
- prob(y=0 | x) = 1 - prob(y=1 | x)

- Loss function: -y*log(σ(w.x + b)) – (1-y)* log(1- σ(w.x + b))
- Why is this a valid loss function????

Logistic Regression Loss Function

- Logistic regression: probabilistic binary classification
- prob(y=1 | x) = σ(w.x + b) = 1/(1 + exp(-w.x-b))   [w,b are parameters]
- prob(y=0 | x) = 1 - prob(y=1 | x)

- Loss function: -y*log(σ(w.x + b)) – (1-y)* log(1- σ(w.x + b))
- Why is this a valid loss function????

	y=1	y=0
	-log(σ(w.x + b))	-log(1- σ(w.x + b))

Logistic Regression Loss Function

- Logistic regression: probabilistic binary classification
- prob(y=1 | x) = σ(w.x + b) = 1/(1 + exp(-w.x-b))   [w,b are parameters]
- prob(y=0 | x) = 1 - prob(y=1 | x)

- Loss function: -y*log(σ(w.x + b)) – (1-y)* log(1- σ(w.x + b))
- Why is this a valid loss function????

	y=1	**y=0**
σ(w.x + b) = 0.99	-log(0.99) = 0	-log(0.01) = 4.6
σ(w.x + b) = 0.01	-log(0.01) = 4.6	-log(0.99) = 0

Minimization of Loss functions

- We want to find classifier/regressor function "h" which minimizes loss function on full training set

- Training loss = $\sum_i L(h(x_i), y_i)$
- Optimal classifier/regressor $h_{OPT} = \text{argmin}_h \sum_i L(h(x_i), y_i)$

- How to calculate this argmin?

Analytical Approach

- Linear Regression: squared error loss function $(h(x) - y)^2$
- $h(x) = W.X + b$
- $(W,b)_{OPT} = \text{argmin} \sum_i (W.X_i + b - y_i)^2$
- Optimal values can be calculated by equating the derivatives to 0

Numerical Approach

- In some cases, analytical approach does not work
- Reasons:

  1) loss function not differentiable

     e.g. 0-1 loss function

  2) derivative equations cannot be solved

     e.g. Loss function for logistic regression

- In such cases, we need numerical approach!

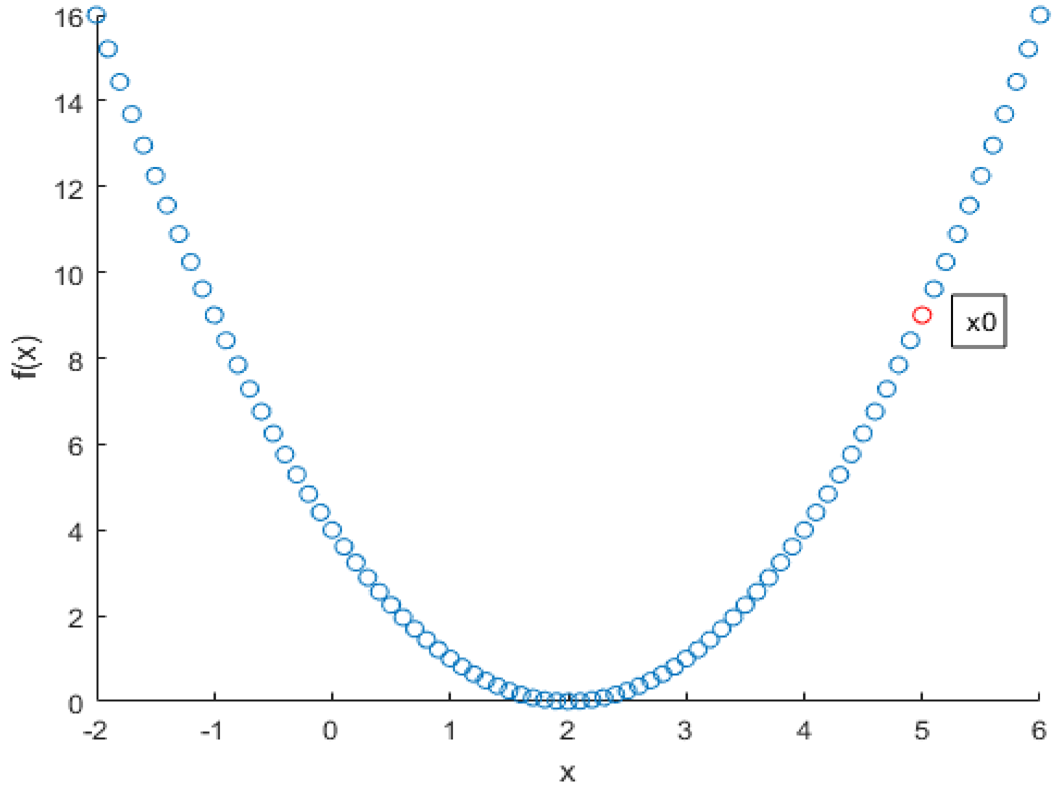
Numerical Approach

- Numerical approach to minimizing any function f:

  1) Start with any initial point x0
  2) Move to point x1 = x0 – a.f'(x0) /// f'(x0): derivative of f(x) at x=x0
     /// a: constant learning rate
  3) If x1 = x0, STOP /// x0 is a minima of function f
     else set x0 = x1, GOTO 2

Numerical Approach

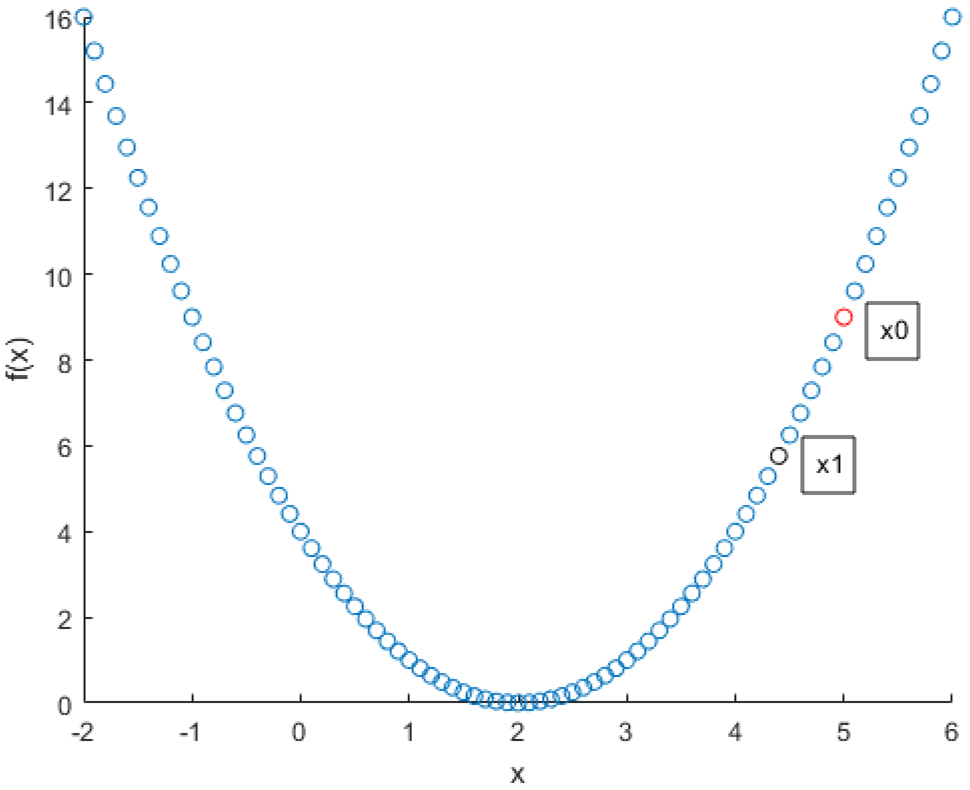
- Consider $f(x) = x^2 - 4x + 4$; $f'(x) = 2x - 4$
- Set initial point x=5, learning rate a = 0.1



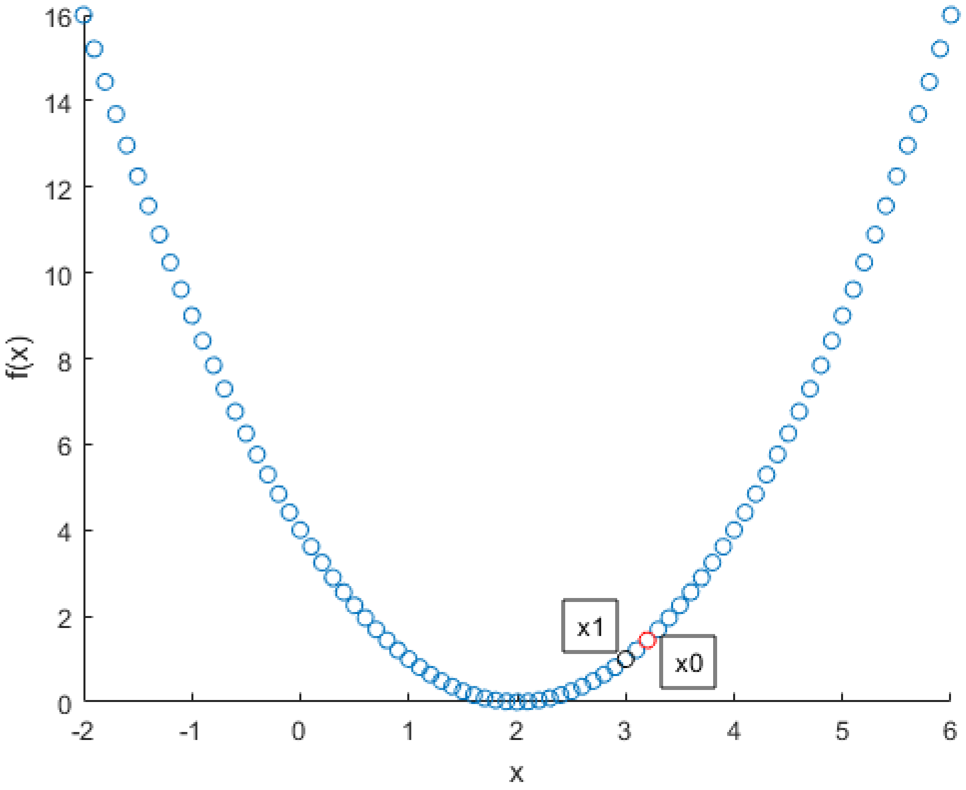
INITIALIZATION

Numerical Approach

- Consider $f(x) = x^2 - 4x + 4$; $f'(x) = 2x - 4$
- Set initial point x=5, learning rate a = 0.1



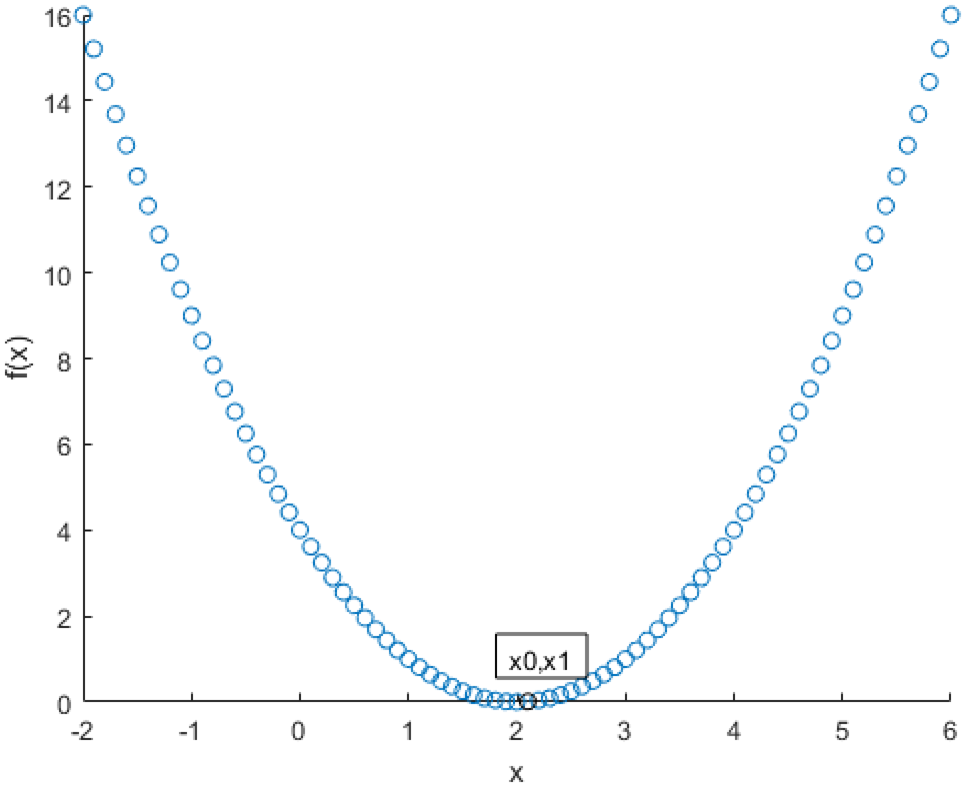
ITERATION 1



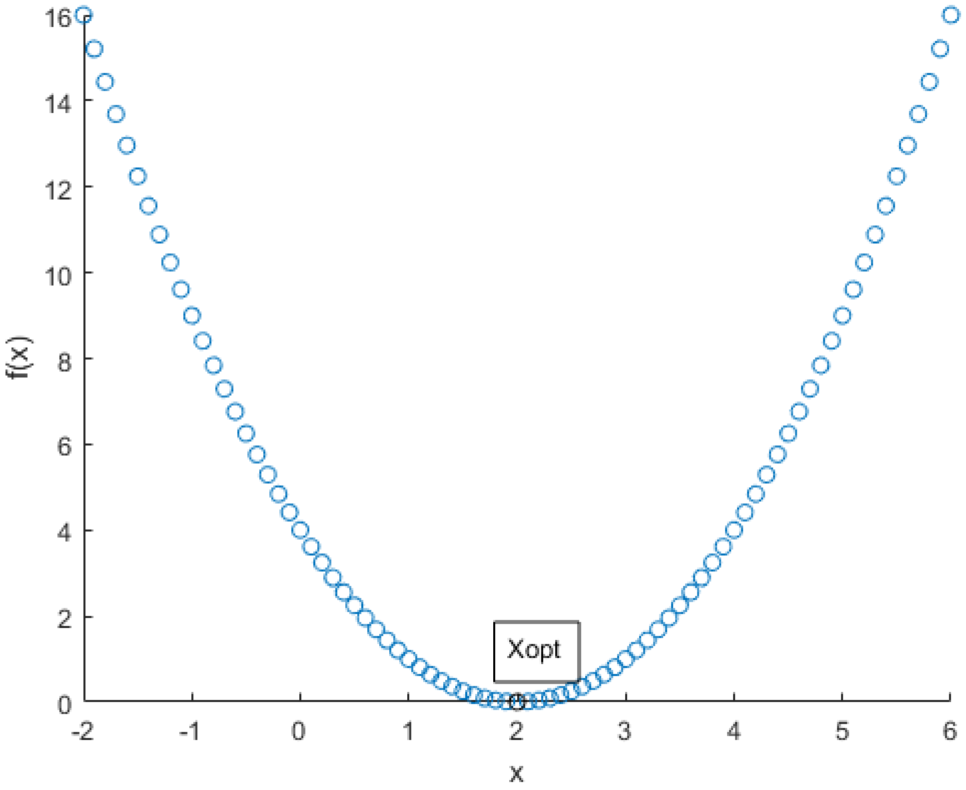
ITERATION 5

Numerical Approach

- Consider $f(x) = x^2 - 4x + 4$; $f'(x) = 2x - 4$
- Set initial point x=5, learning rate a = 0.1



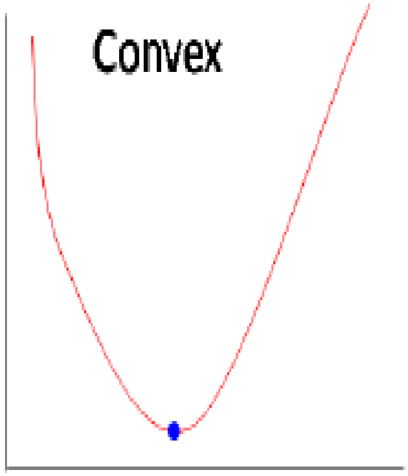
ITERATION 15

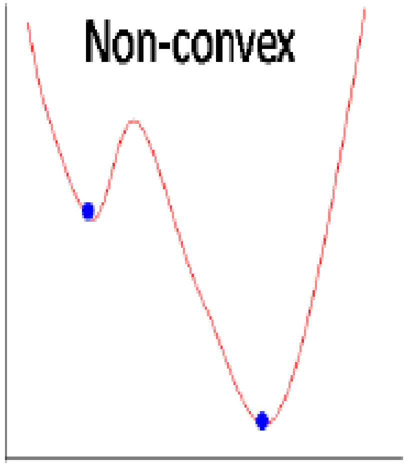


ITERATION 20: CONVERGENCE!

Gradient Descent Issues

- Does it always converge?
  - Depends on the "learning rate"
  - low learning rate: slow convergence
  - high learning rate: may oscillate around the minima!
- Does it always give the optimal solution?
  - Yes, if the function is **Convex** (has *unique minima*)
  - Otherwise, it converges at *any minima*





Logistic Regression Loss Function

Loss function $$L(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))$$

First task: find the derivative L'(w)!

$$\begin{aligned}
\mathrm{g} = \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \frac{\partial}{\partial \boldsymbol{w}}[-\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))] \\
&= -\sum_{n=1}^{N}\left(y_n \boldsymbol{x}_n - \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n))}\boldsymbol{x}_n\right) \\
&= -\sum_{n=1}^{N}(y_n - \mu_n)\boldsymbol{x}_n = \mathbf{X}^\top(\boldsymbol{\mu} - \boldsymbol{y})
\end{aligned}$$

Gradient Descent for Logistic Regression

- Initialize $\boldsymbol{w}^{(1)} \in \mathbb{R}^D$ randomly.
- Iterate the following until convergence

$$\underbrace{\boldsymbol{w}^{(t+1)}}_{\text{new value}} = \underbrace{\boldsymbol{w}^{(t)}}_{\text{previous value}} - \eta \underbrace{\sum_{n=1}^{N}(\mu_n^{(t)} - y_n)\boldsymbol{x}_n}_{\text{gradient at previous value}}$$

where η is the learning rate and $\mu^{(t)} = \sigma(\boldsymbol{w}^{(t)^\top}\boldsymbol{x}_n)$ is the predicted label probability for $\boldsymbol{x}_n$ using $\boldsymbol{w} = \boldsymbol{w}^{(t)}$ from the previous iteration

Multi-class Classification

Suppose Y can take K values instead of (0,1)

We consider (w_k,b_k) for each of the K classes

$$\text{prob}(y=k \mid x) = C.\sigma(w_k.x + b_k) \quad \text{where C is the normalizing constant}$$

$$= \exp(w_k.x + b_k)/(\textstyle\sum_j \exp(w_j.x + b_j))$$

Loss function: $-\sum_j I(y=j)*\log(\sigma(w_j.x + b_j))$

Apply Gradient Descent to compute (w_k,b_k) for each of the K classes

Stochastic Gradient Descent for Perceptron

Instead of all data-points, compute loss function for any one data-point

Update the weights

$L(w) = (y_i – w.x_i)^2$

$\Delta L(w) = (y_i – w.x_i)x_i$

$= \text{error} * \text{input}$

```
In [5]: # Estimate Perceptron weights using stochastic gradient descent
def train_weights(train, l_rate, n_epoch):
	weights = [0.0 for i in range(len(train[0]))]
	for epoch in range(n_epoch):
		sum_error = 0.0
		for row in train:
			prediction = predict(row, weights)
			error = row[-1] - prediction
			sum_error += error**2
			weights[0] = weights[0] + l_rate * error
			for i in range(len(row)-1):
				weights[i + 1] = weights[i + 1] + l_rate * error * row[i]
		print('>epoch=%d, lrate=%.3f, error=%.3f' % (epoch, l_rate, sum_error))
	return weights
```