Machine Learning and Adversaries (Part II)

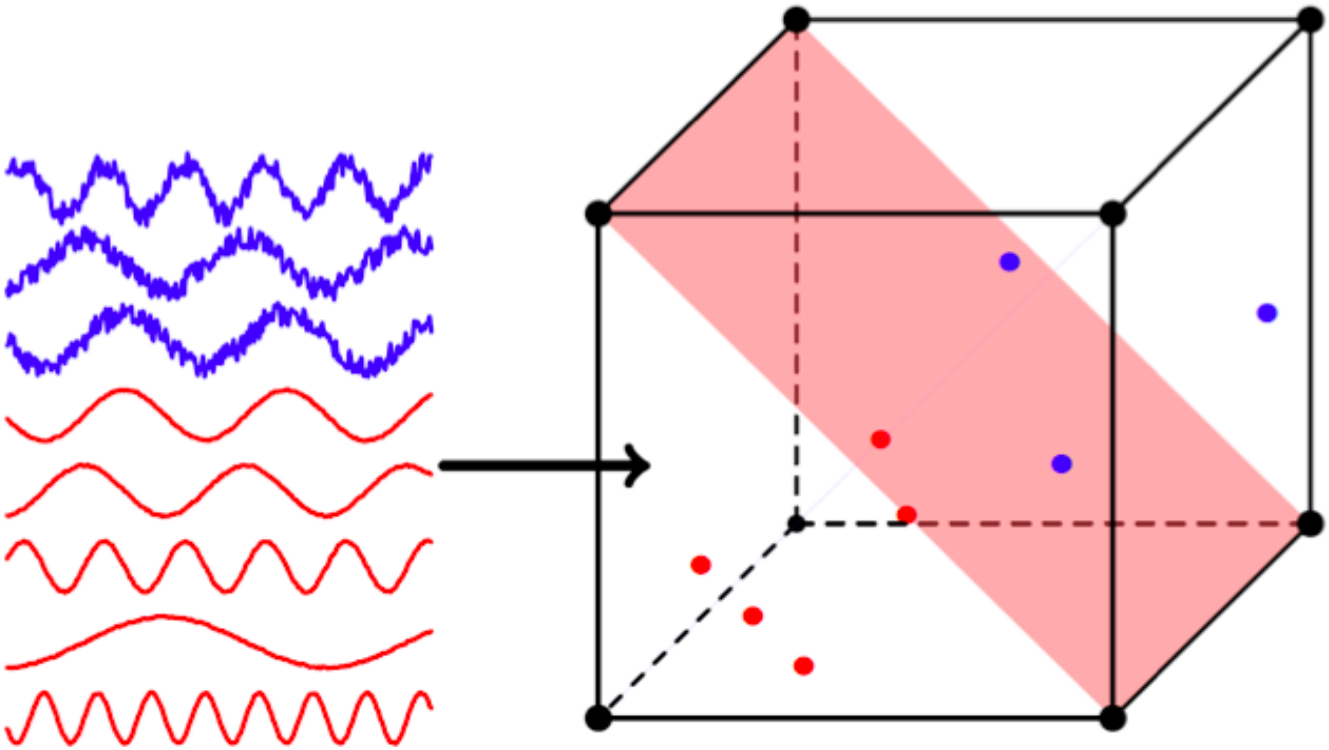
Dan Bruce, David Clark and Hector D. Menendez

Department of Computer Science
University College London

December 11, 2017

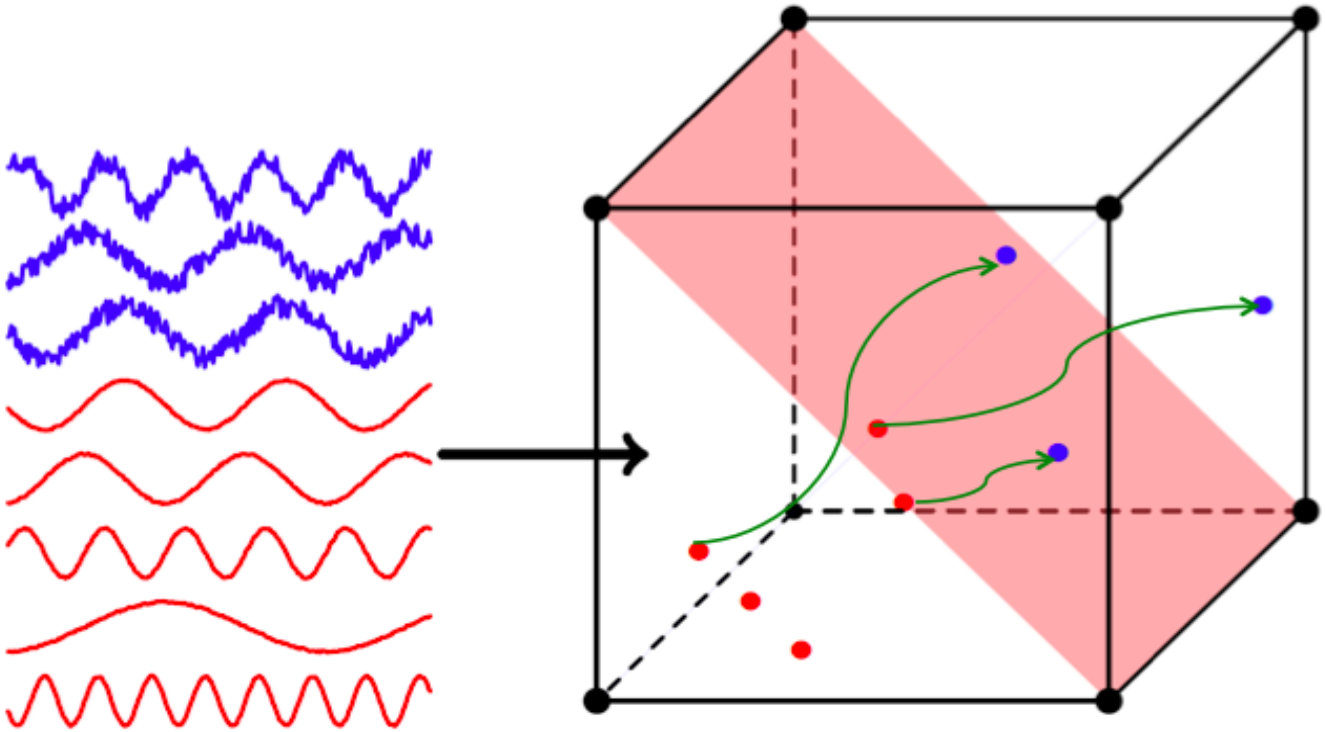
The ML solution

Machine Learning aims to solve this problem discriminating features from malicious attacks.



The Adversarial ML

Adversarial Machine Learning looks for vulnerabilities in the discrimination to cheat the algorithm.



Understanding Adversarial ML

Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

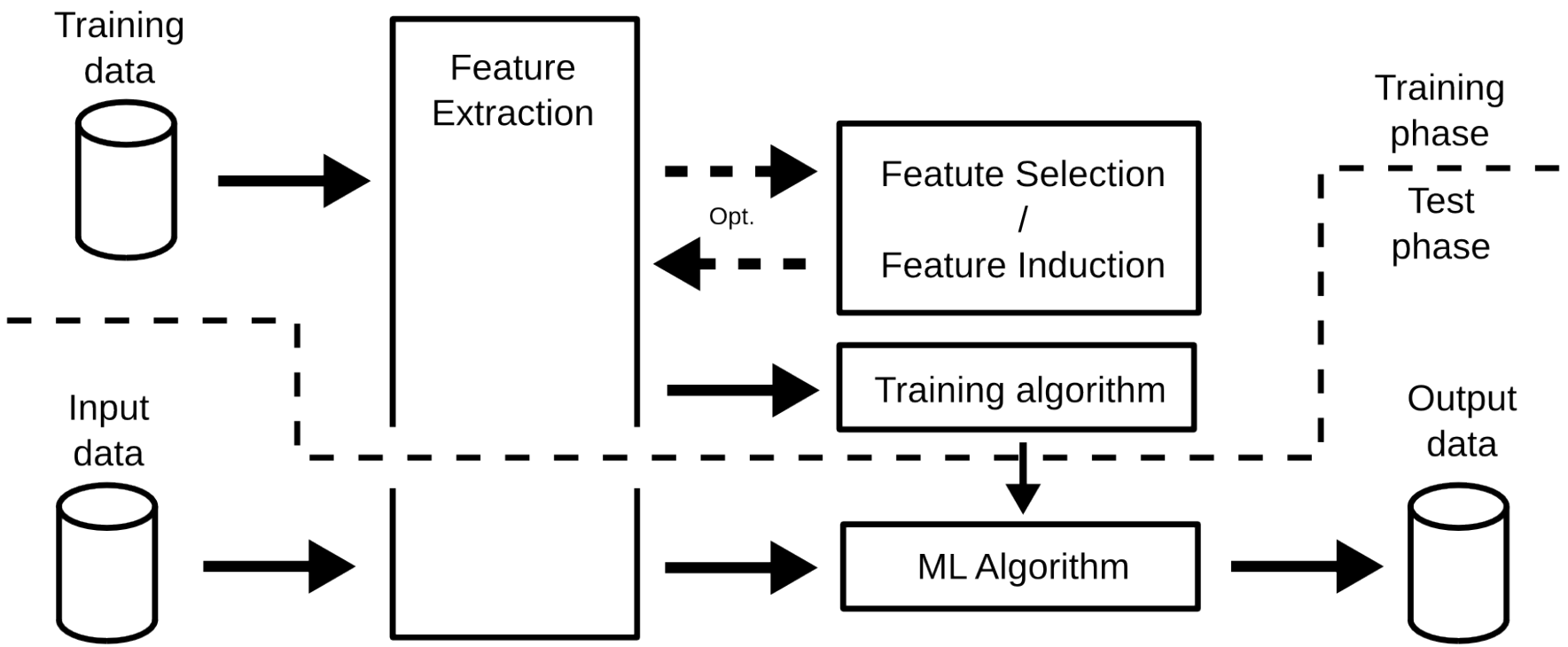
Index

Where are the machine learning vulnerabilities?

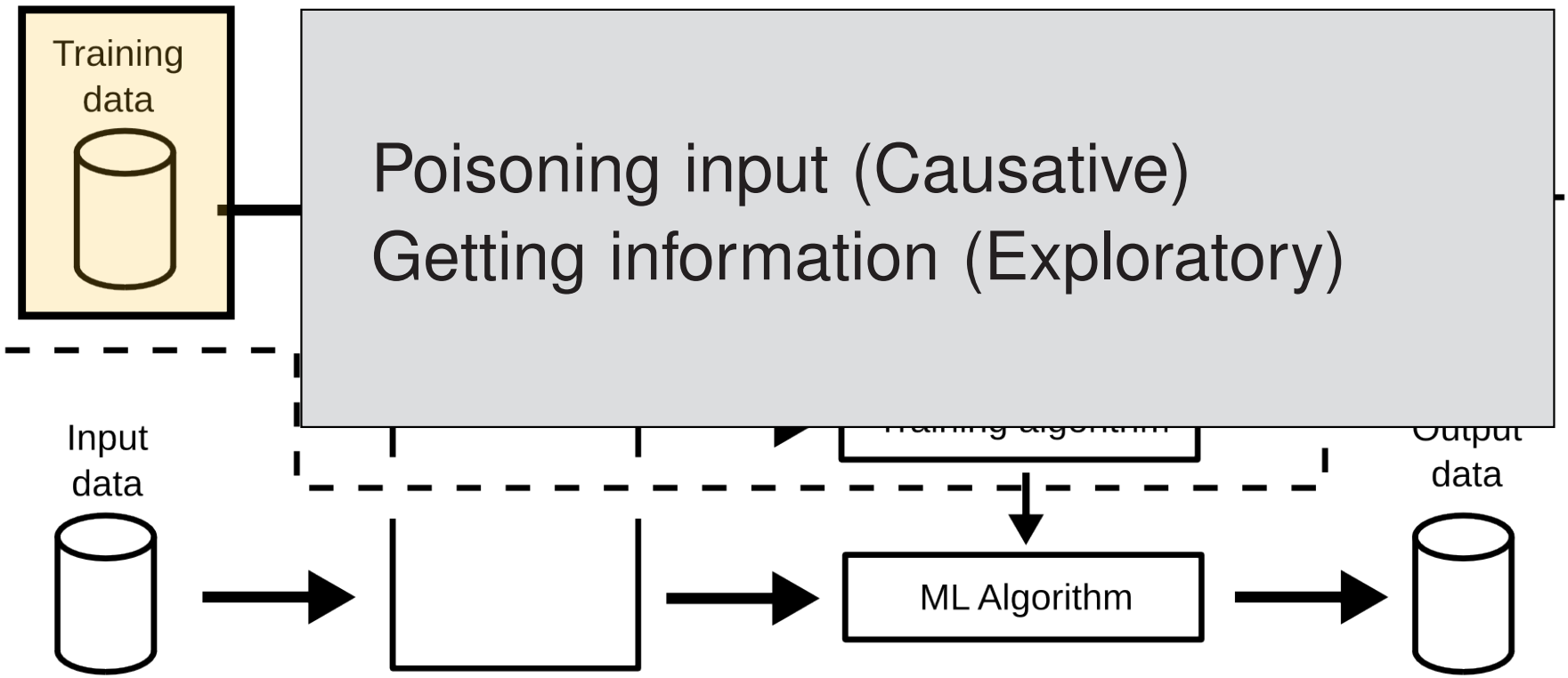
How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

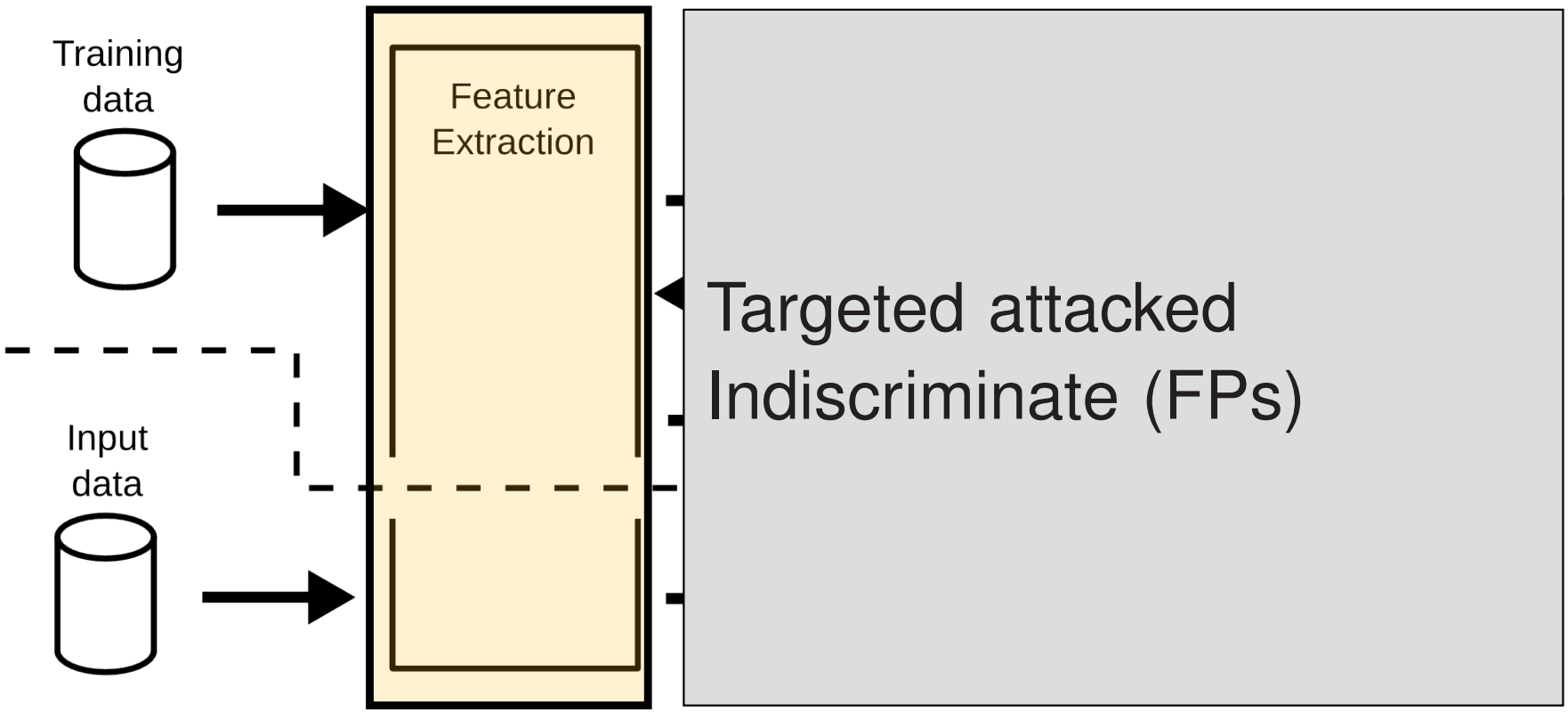
Machine Learning Vulnerabilities



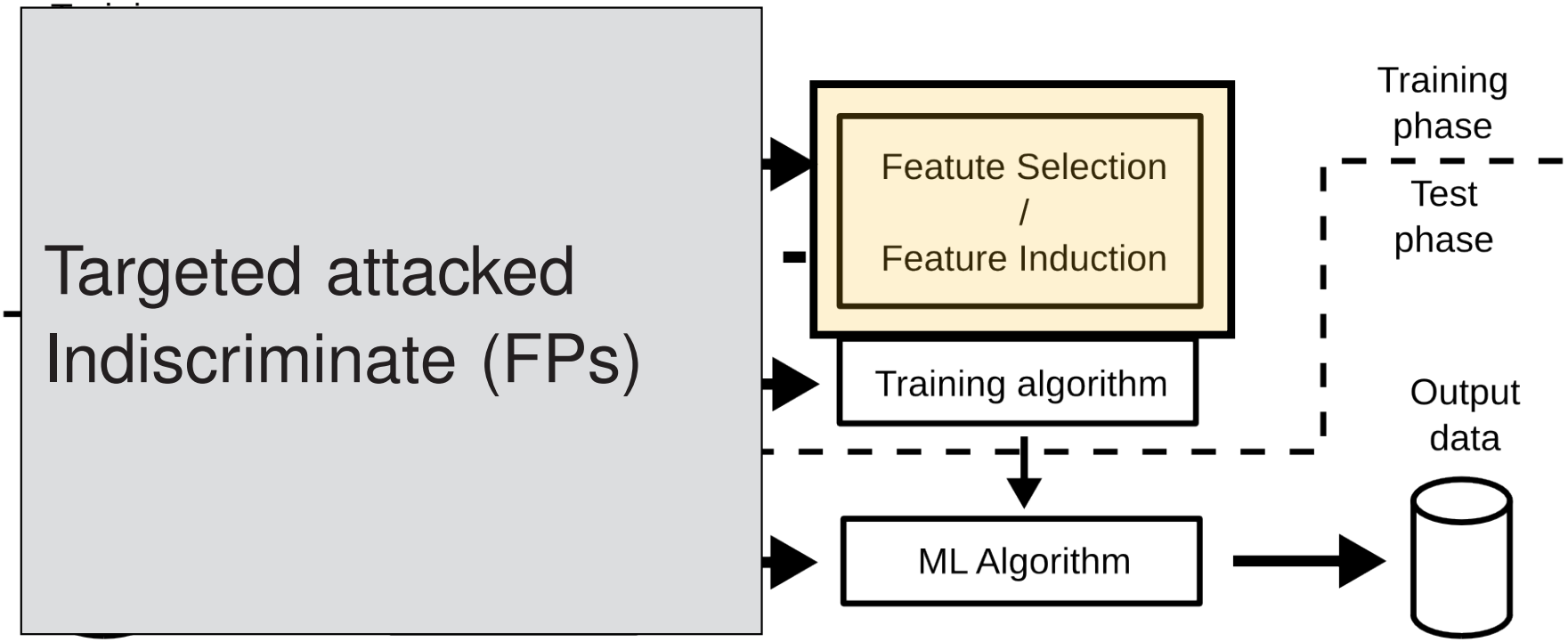
Machine Learning Vulnerabilities



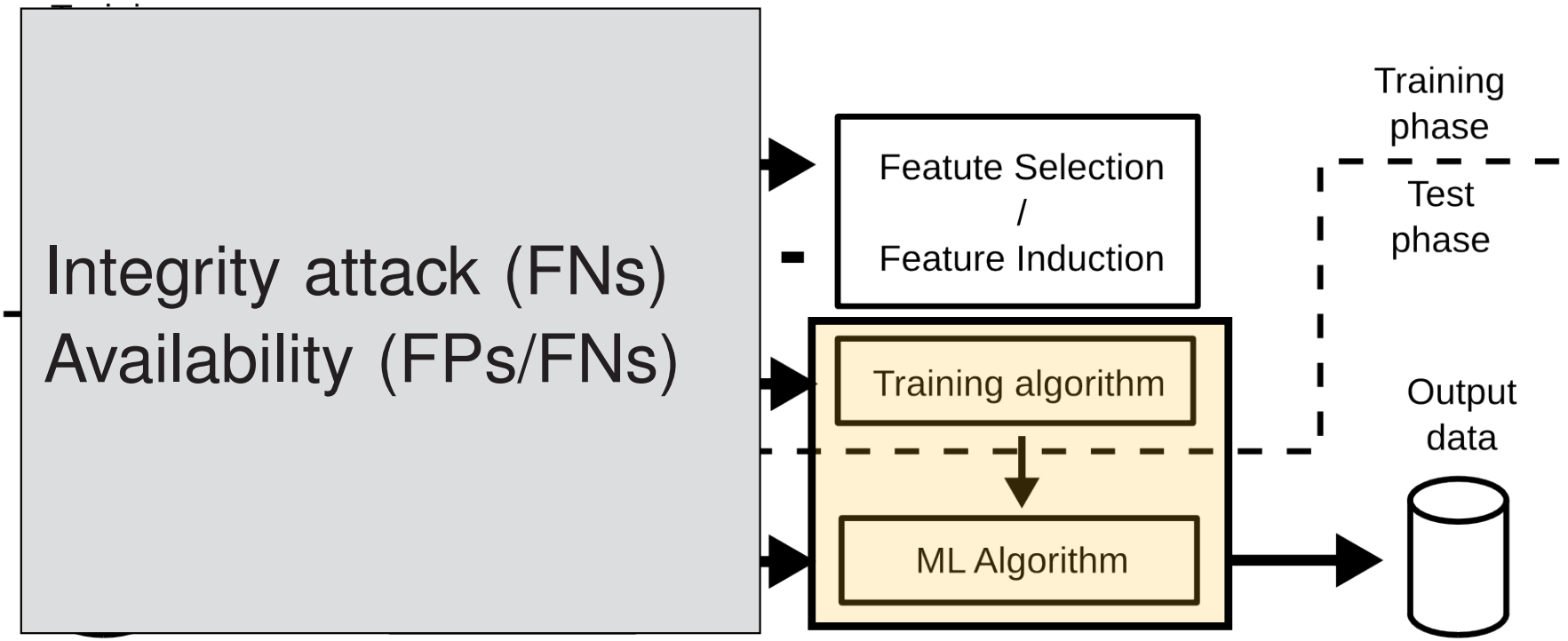
Machine Learning Vulnerabilities



Machine Learning Vulnerabilities



Machine Learning Vulnerabilities



Index

Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

What is a vulnerability?

There are three relevant agents in ML: the *oracle*, the *feature space* and the *algorithm*

The **oracle** provides the ground truth (e.g. labels)

The **feature space** represents the data features

The **algorithm** learns to discriminate using the features and the oracle information.

Train/Test Distributions

ML supposes same train and test distributions

Adversaries part from this hypothesis aiming to find mistakes on the discrimination

Where are these mistakes?

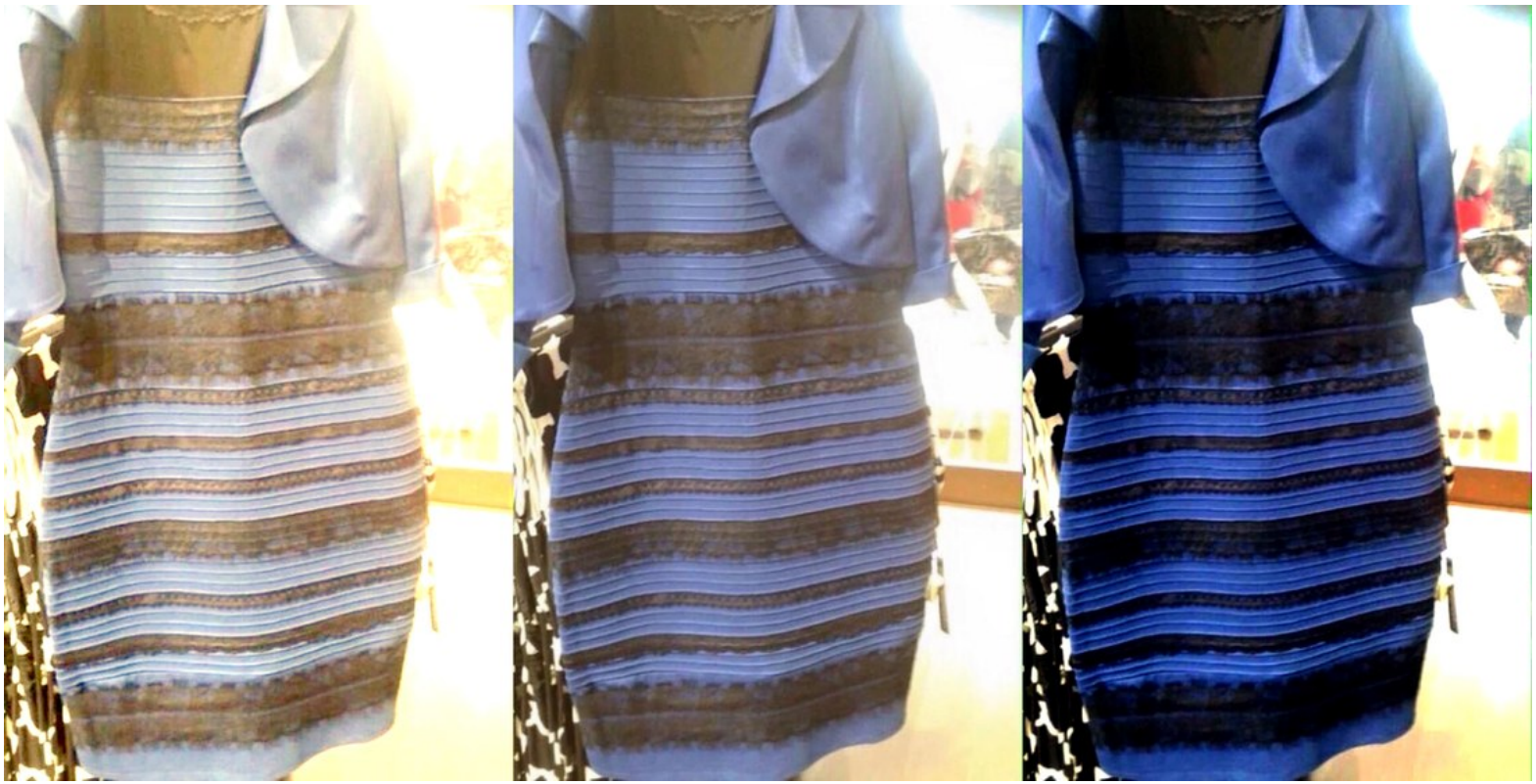
Cheating the oracle

Which color is this dress?

Cheating the oracle

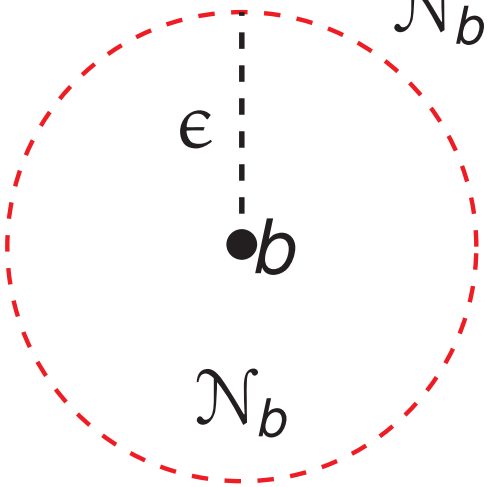
And now?

Cheating the oracle



Cheating the feature space

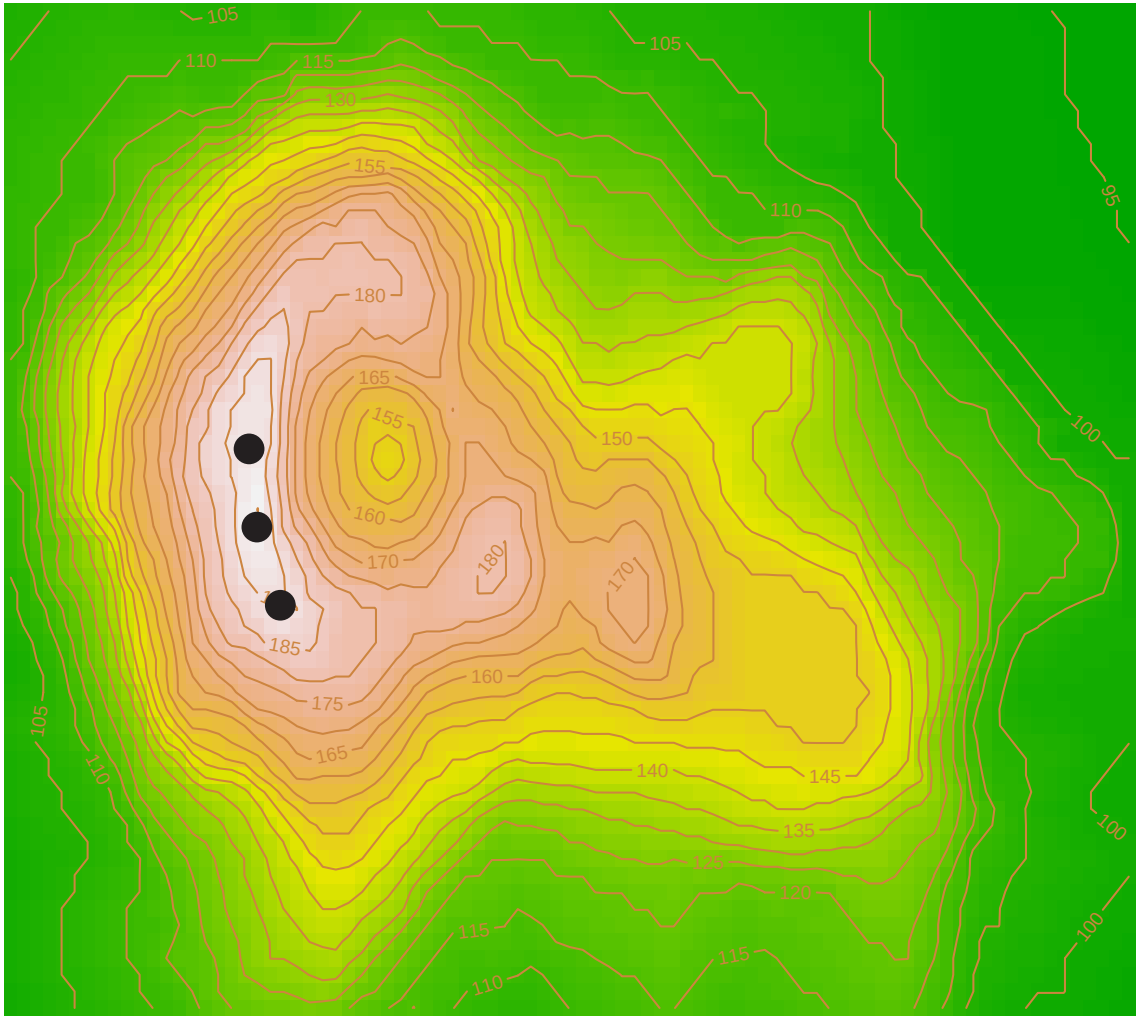
Consider b known instance.



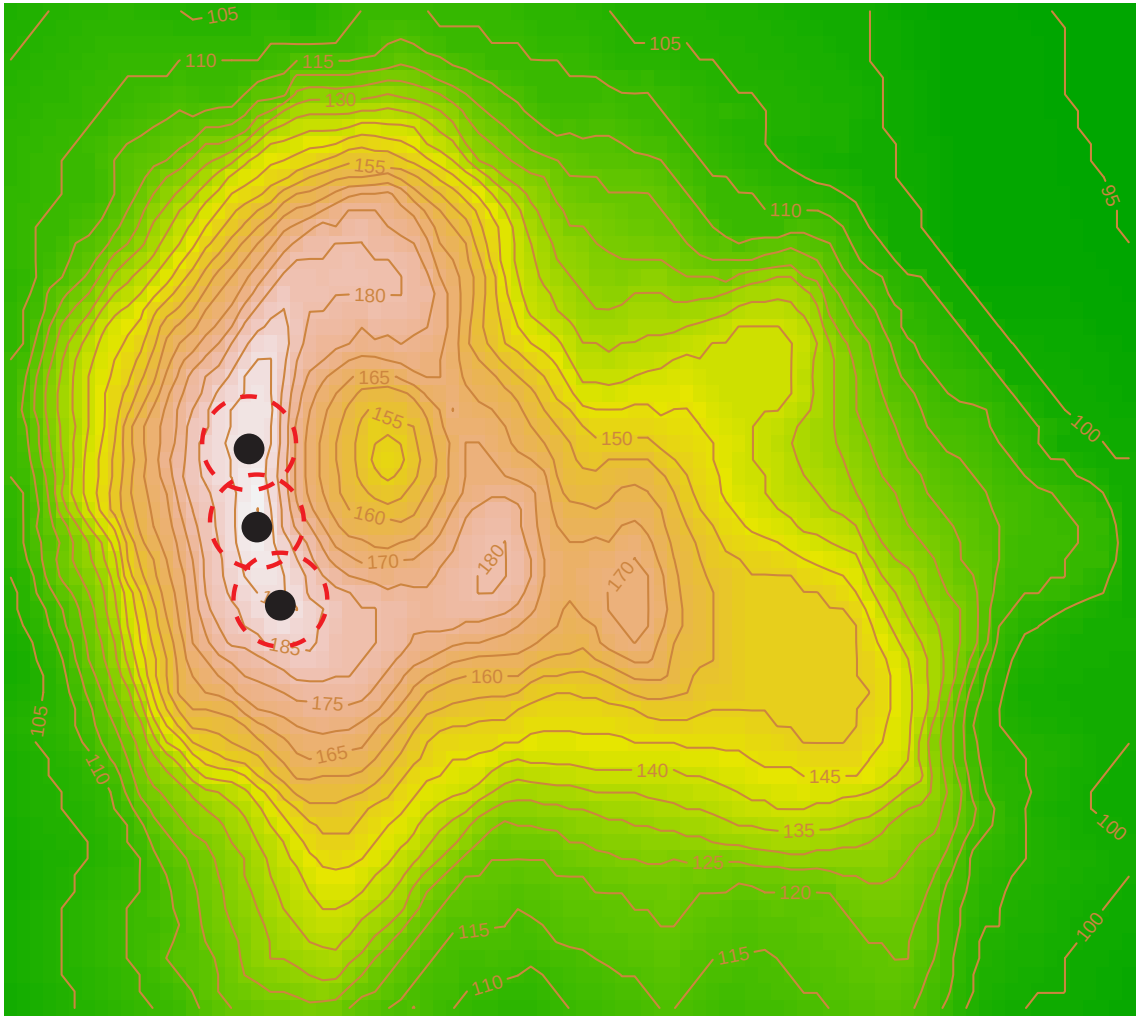
$$\mathcal{N}_b = \{x \mid d(x, b) < \epsilon \wedge C(x) = C(b)\}$$

$$\exists y \in \mathcal{N}_b \mid O(y) \neq O(b)?$$

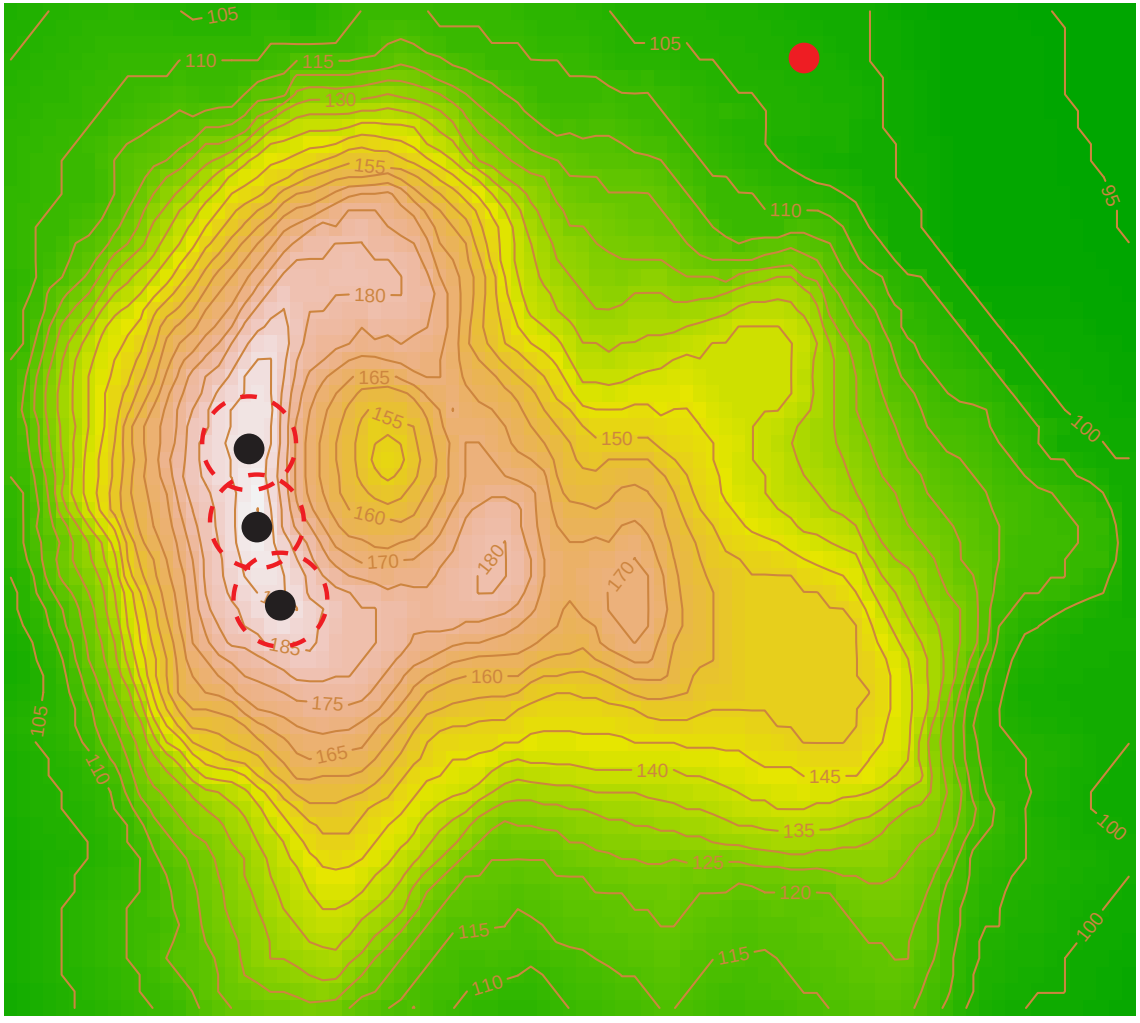
Cheating the feature space



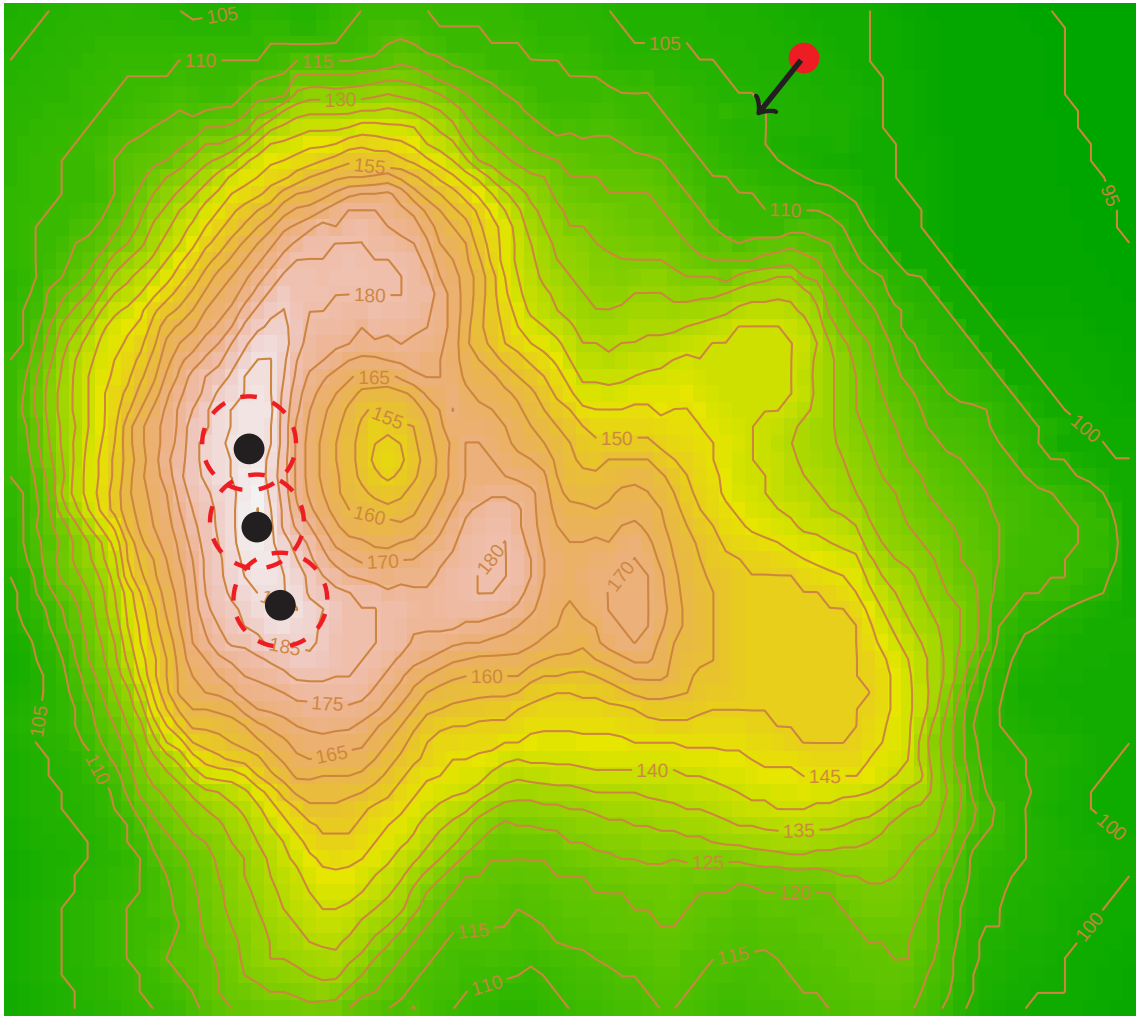
Cheating the feature space



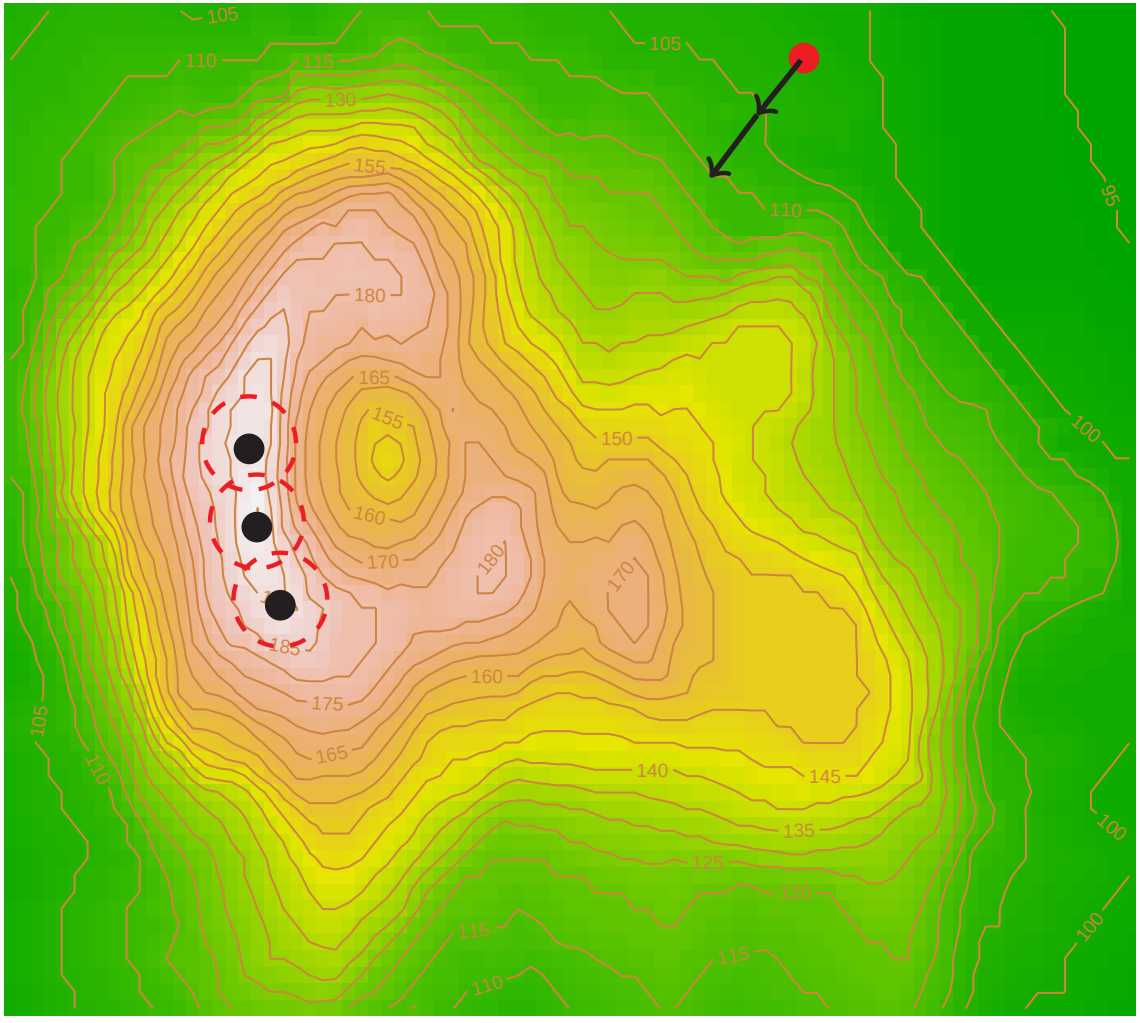
Cheating the feature space



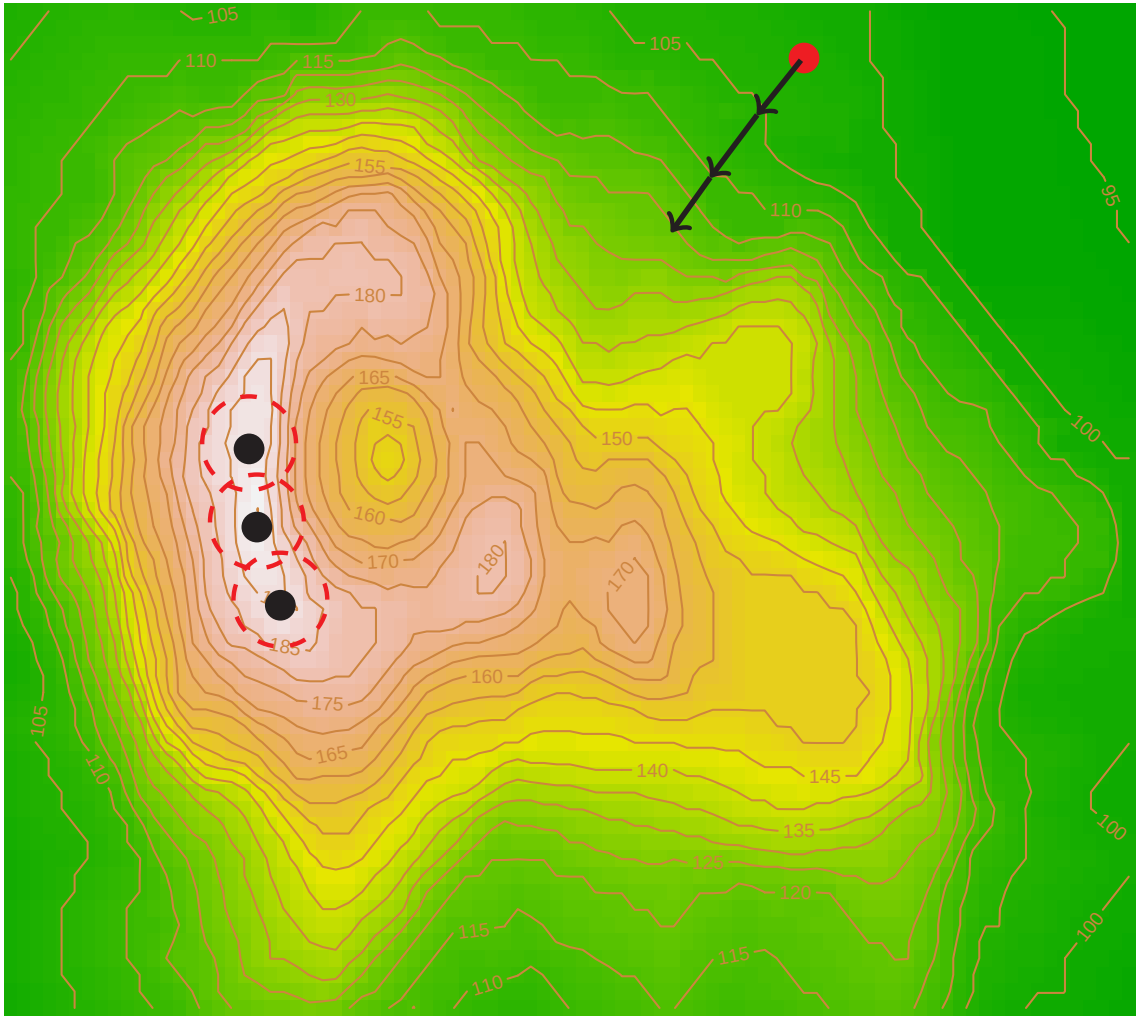
Cheating the feature space



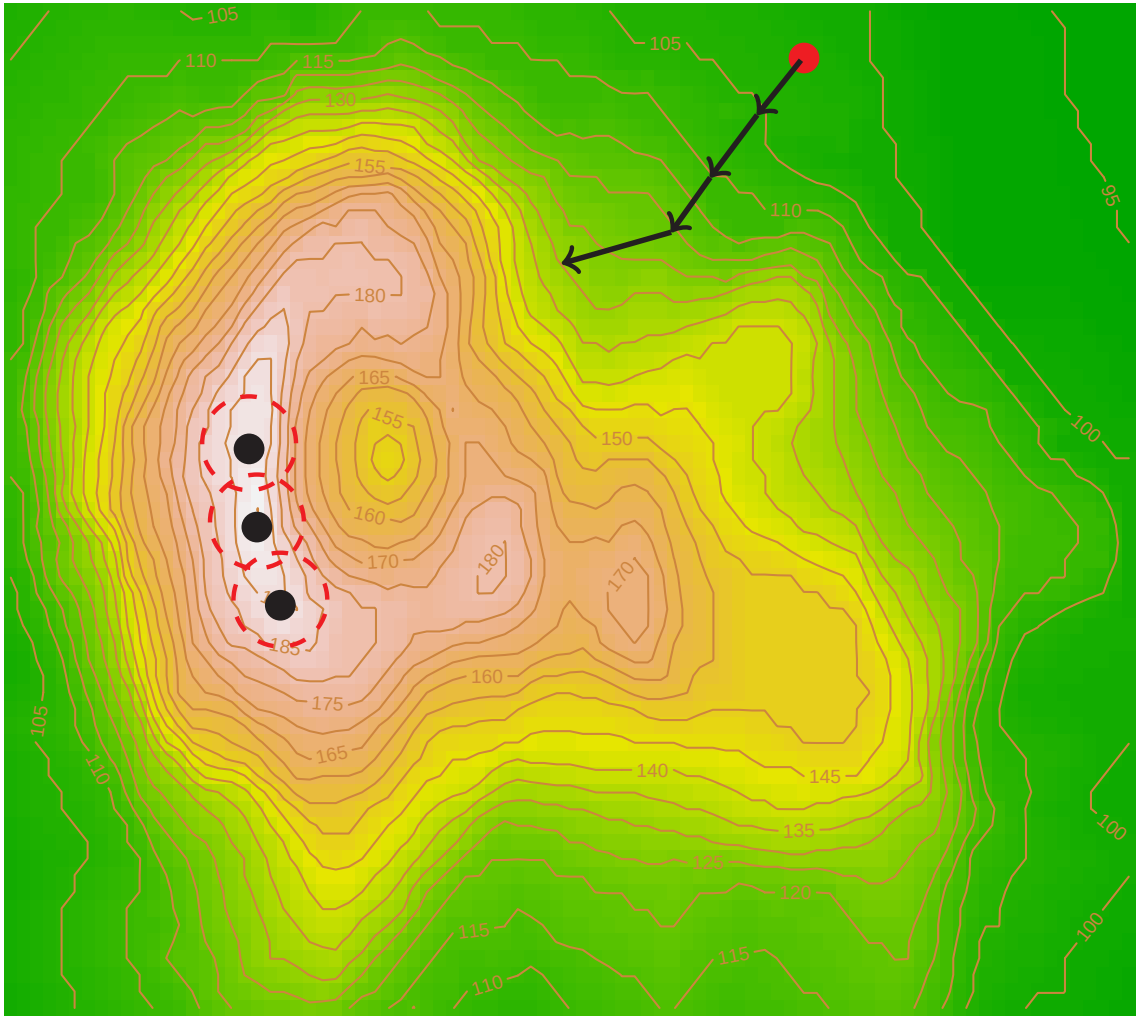
Cheating the feature space



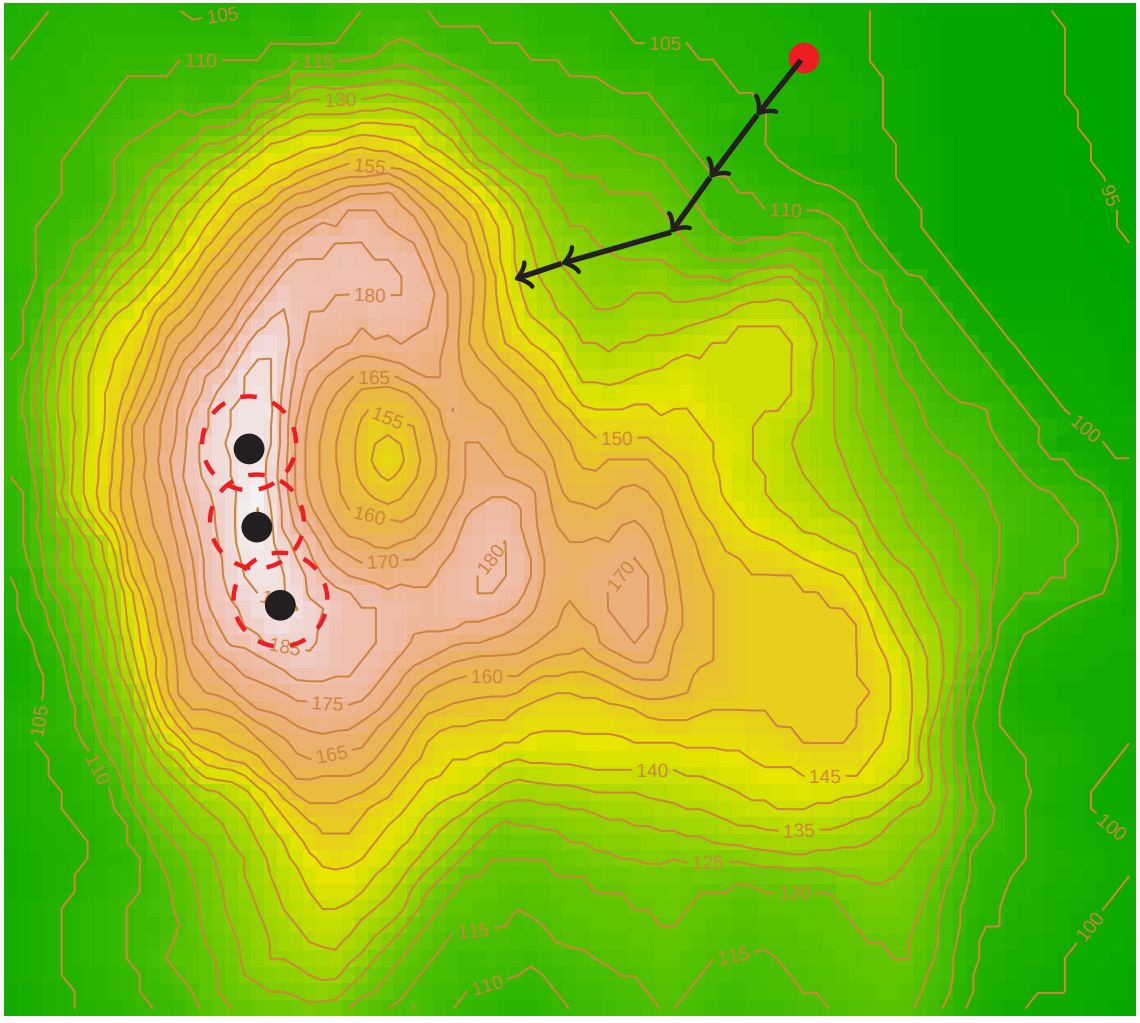
Cheating the feature space



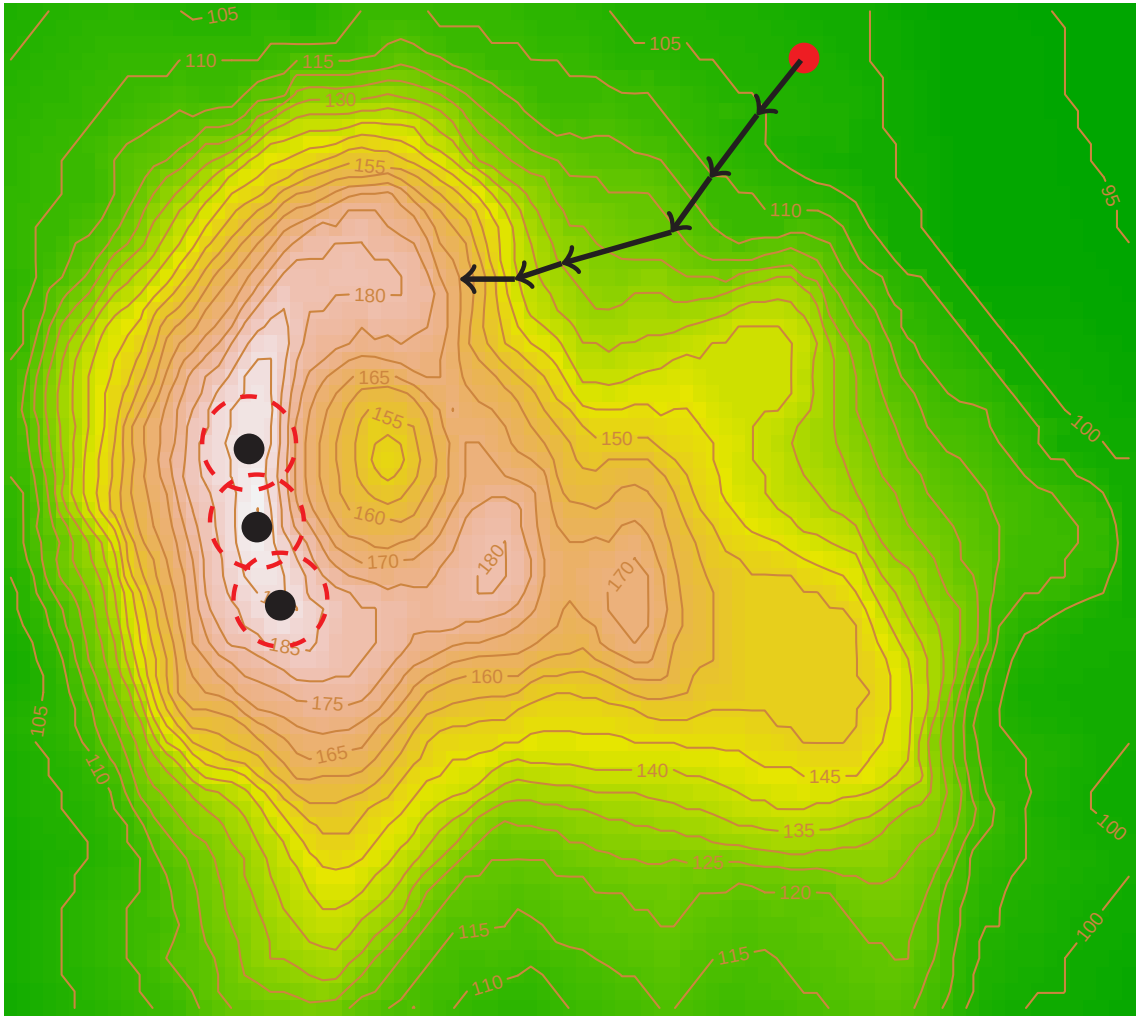
Cheating the feature space



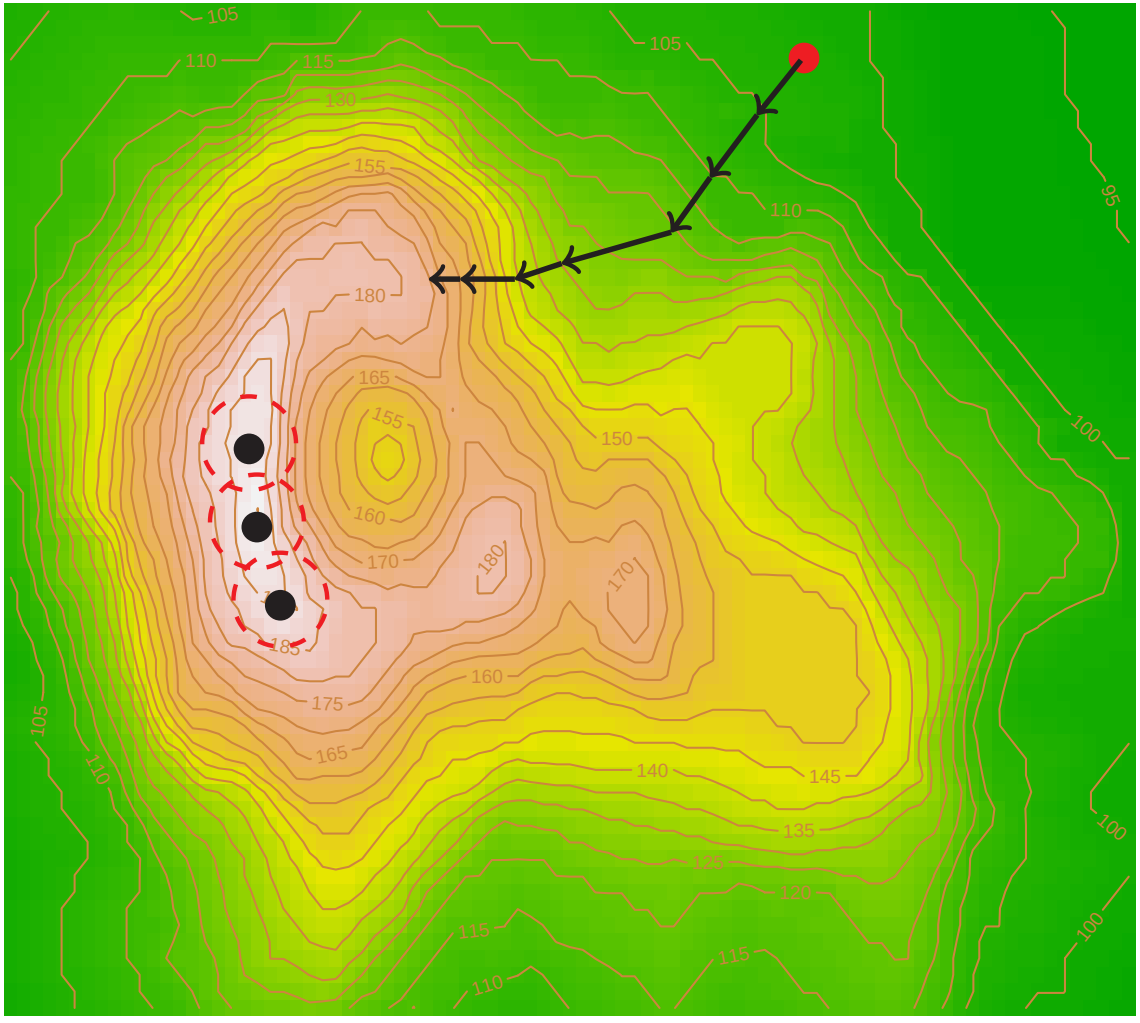
Cheating the feature space



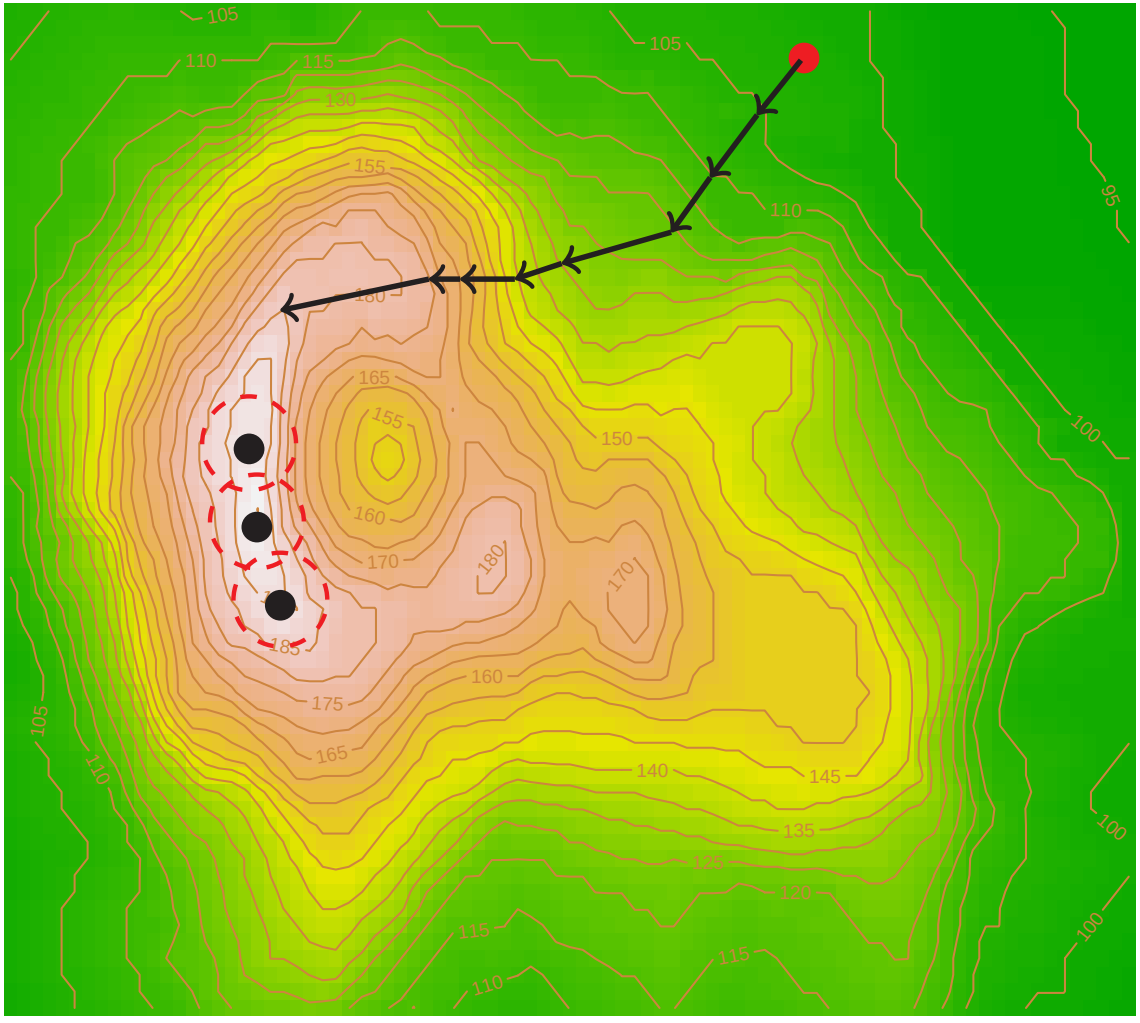
Cheating the feature space



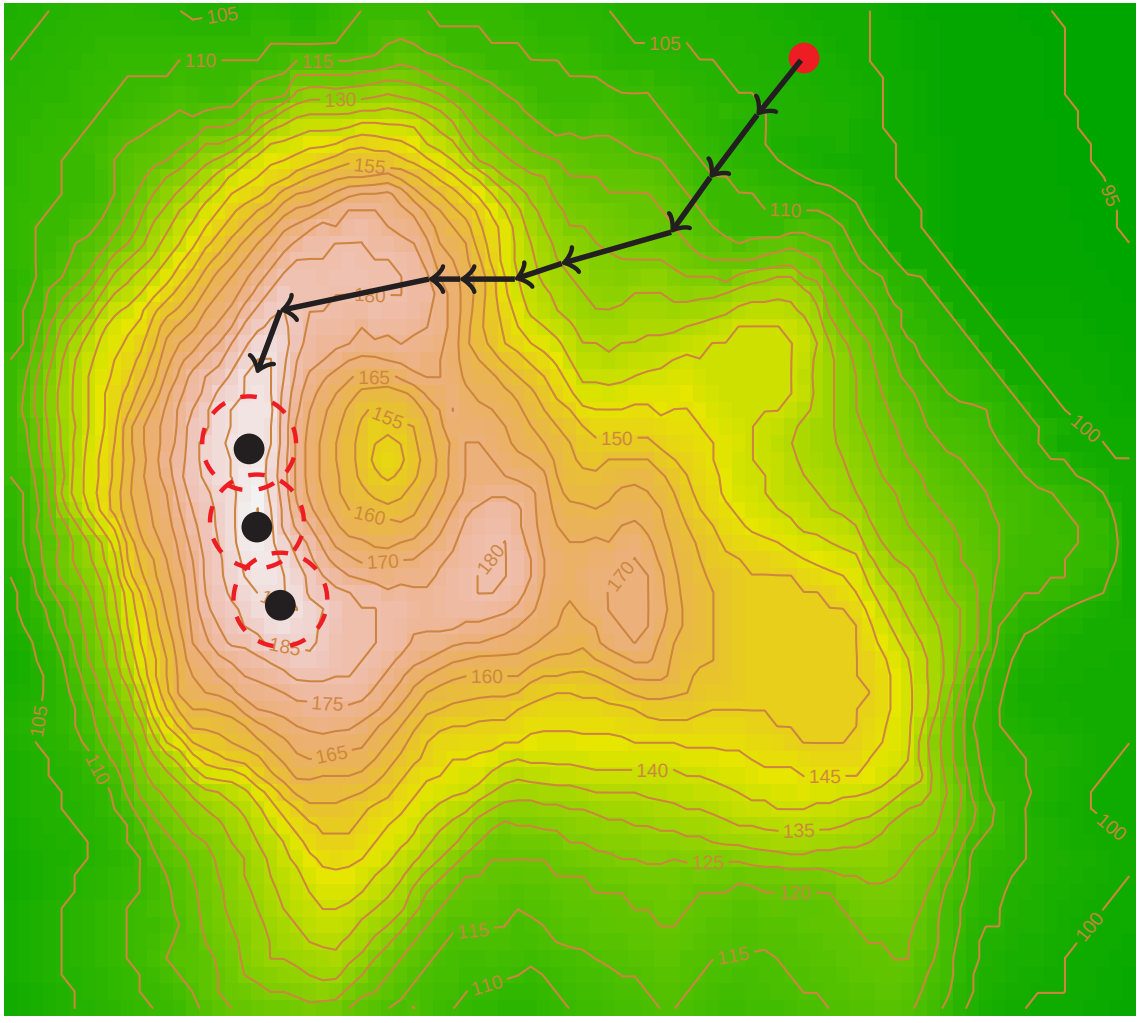
Cheating the feature space



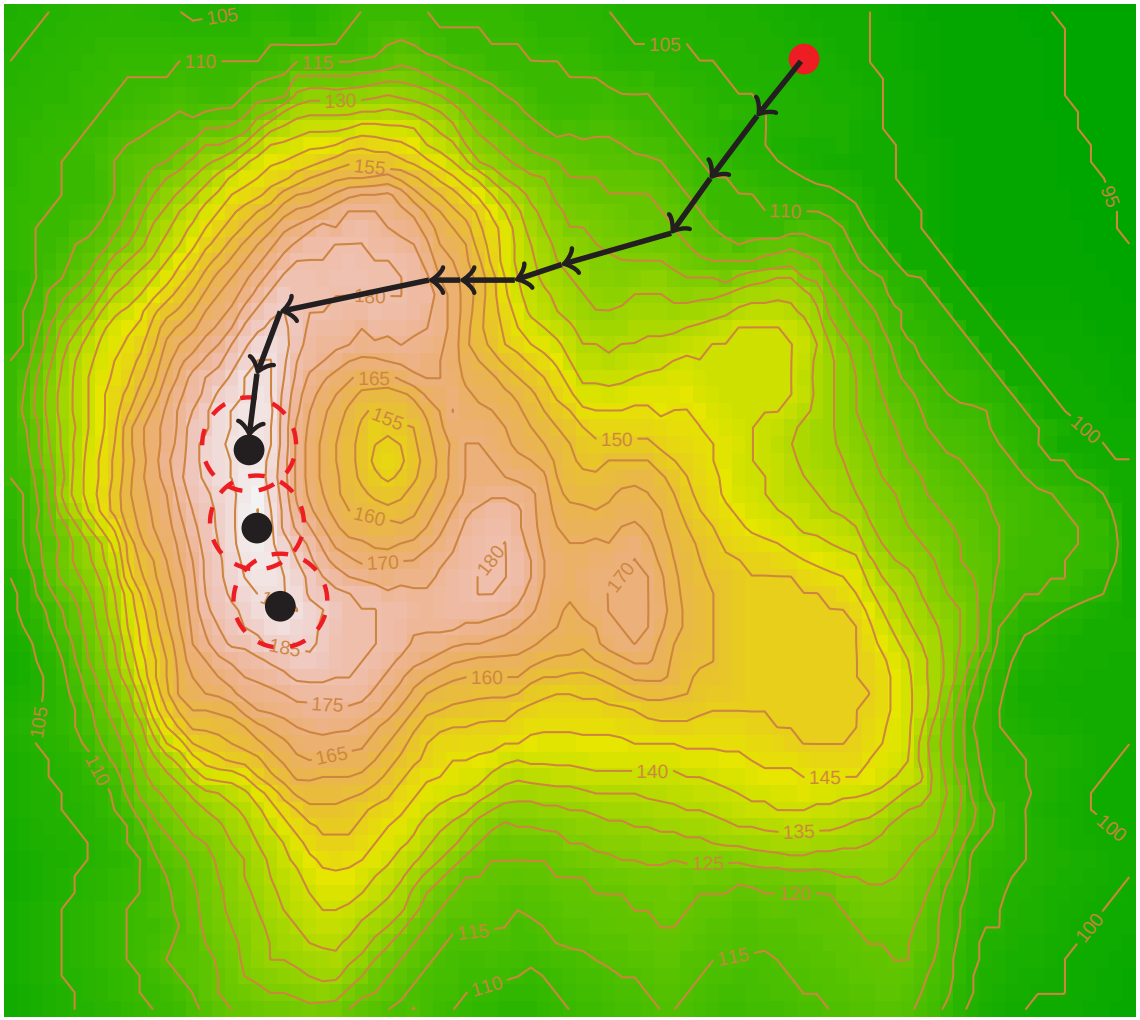
Cheating the feature space



Cheating the feature space

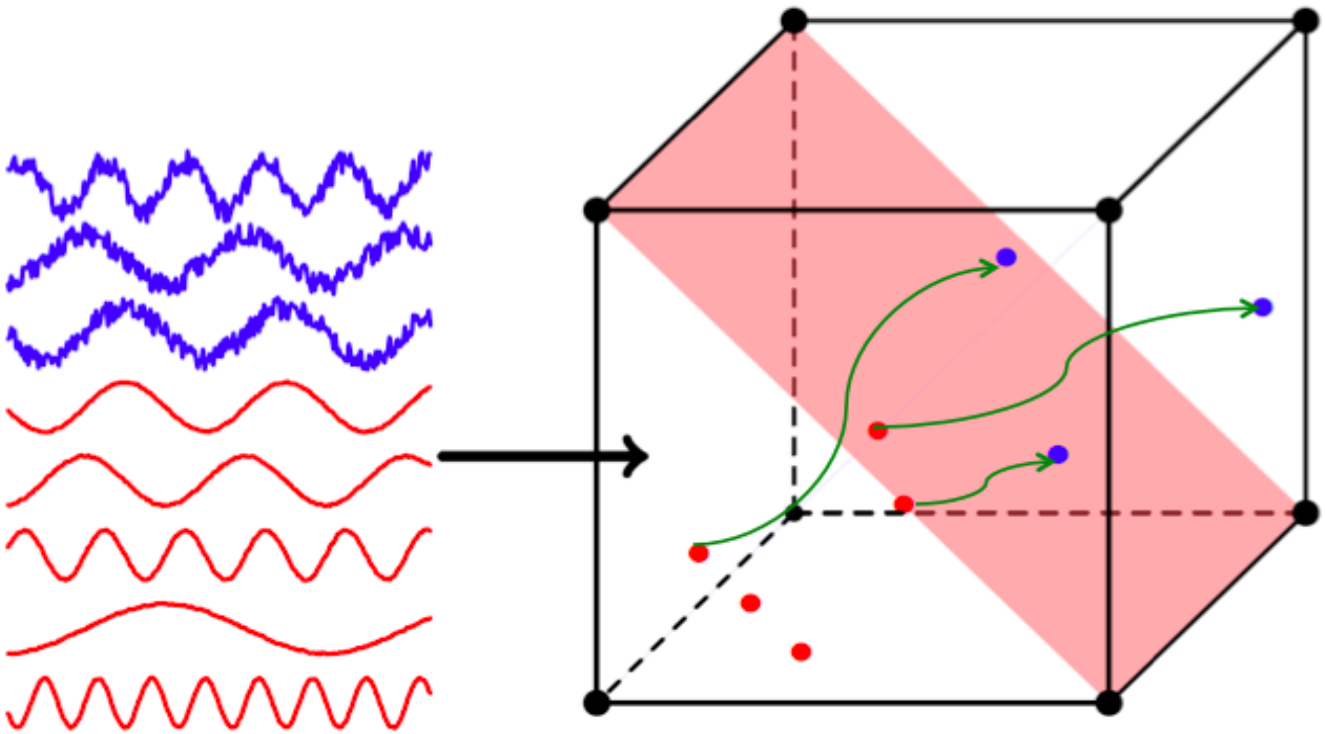


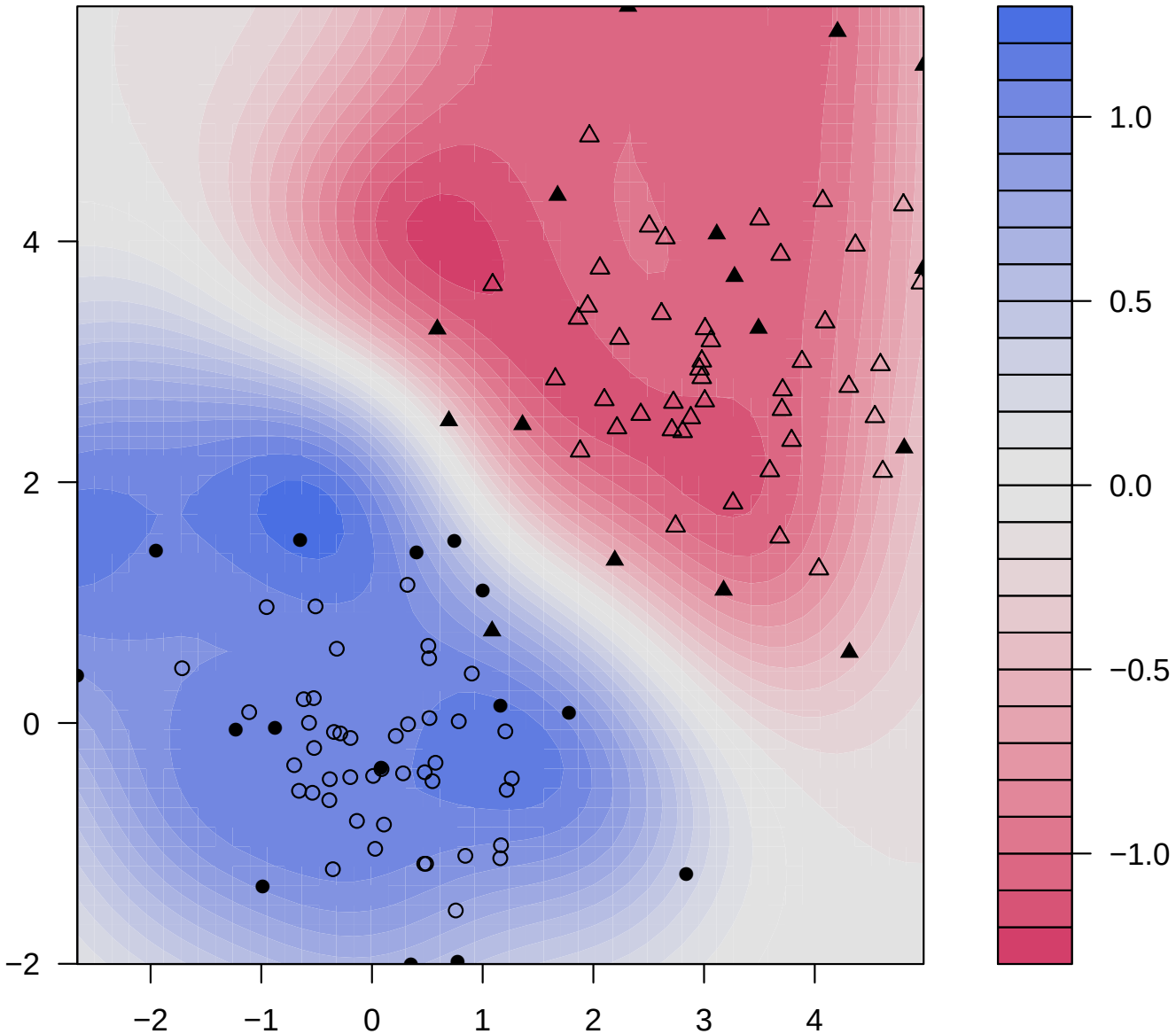
Cheating the feature space

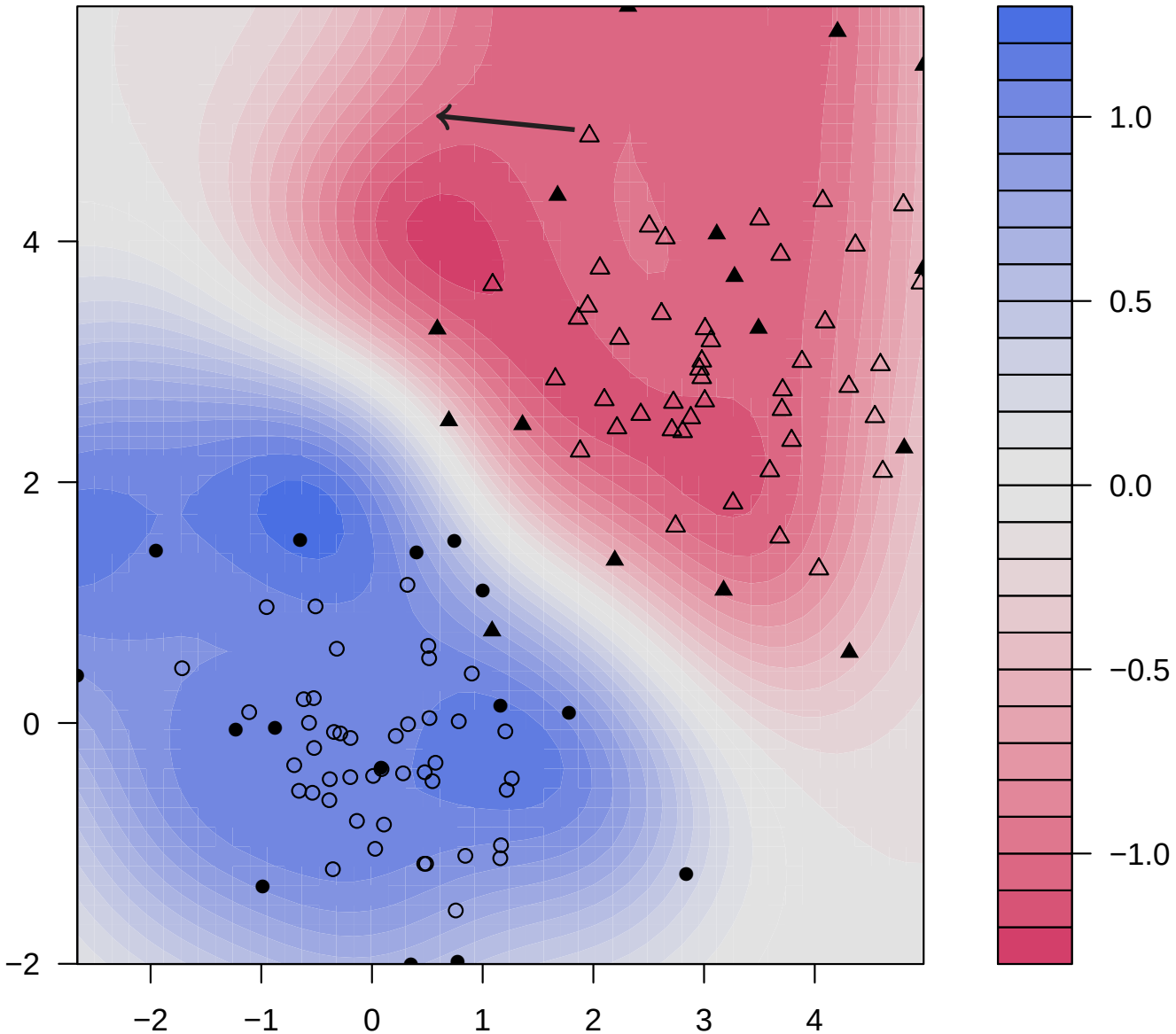


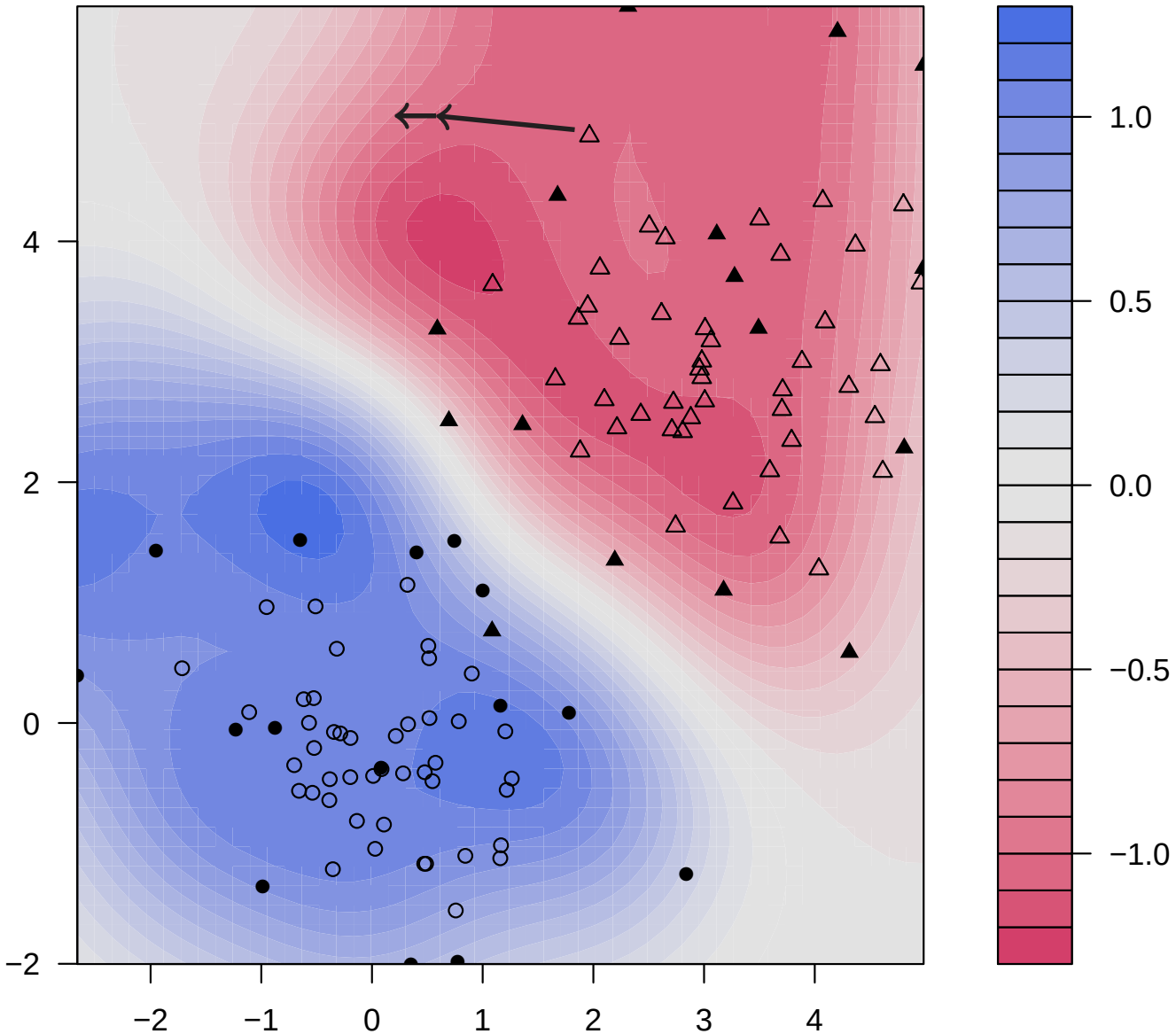
Cheating the classifier

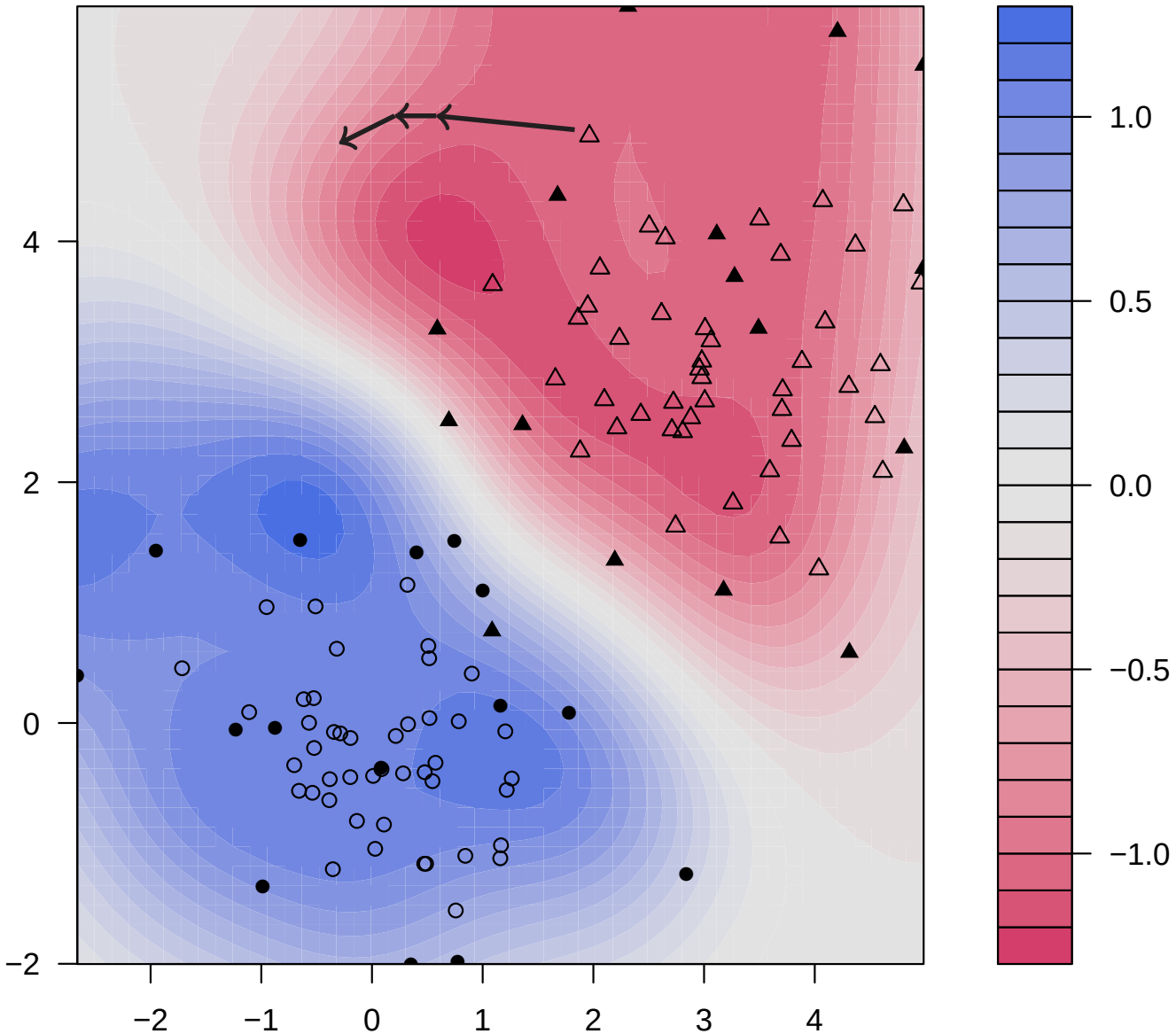
Jump the wall strategy

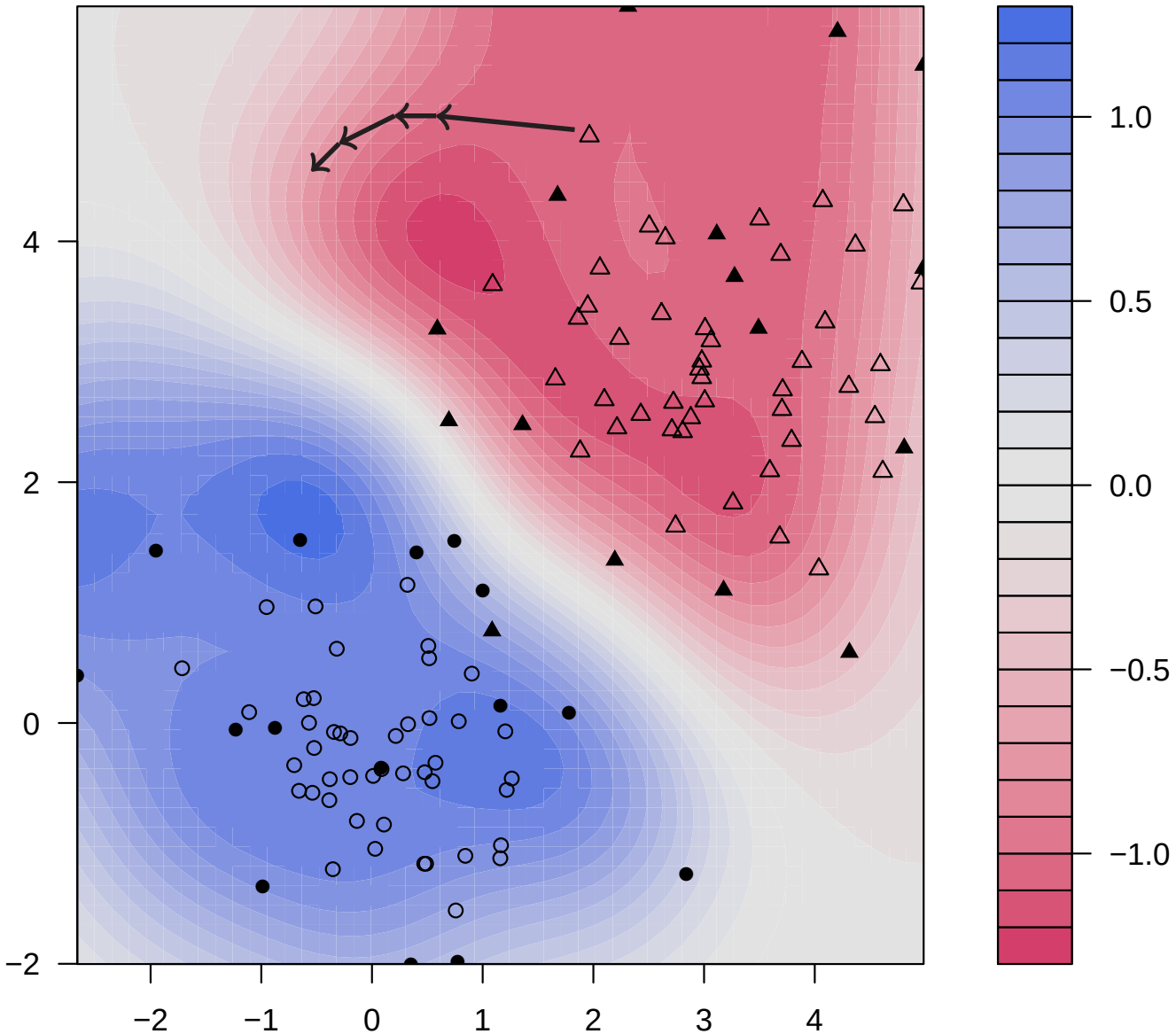


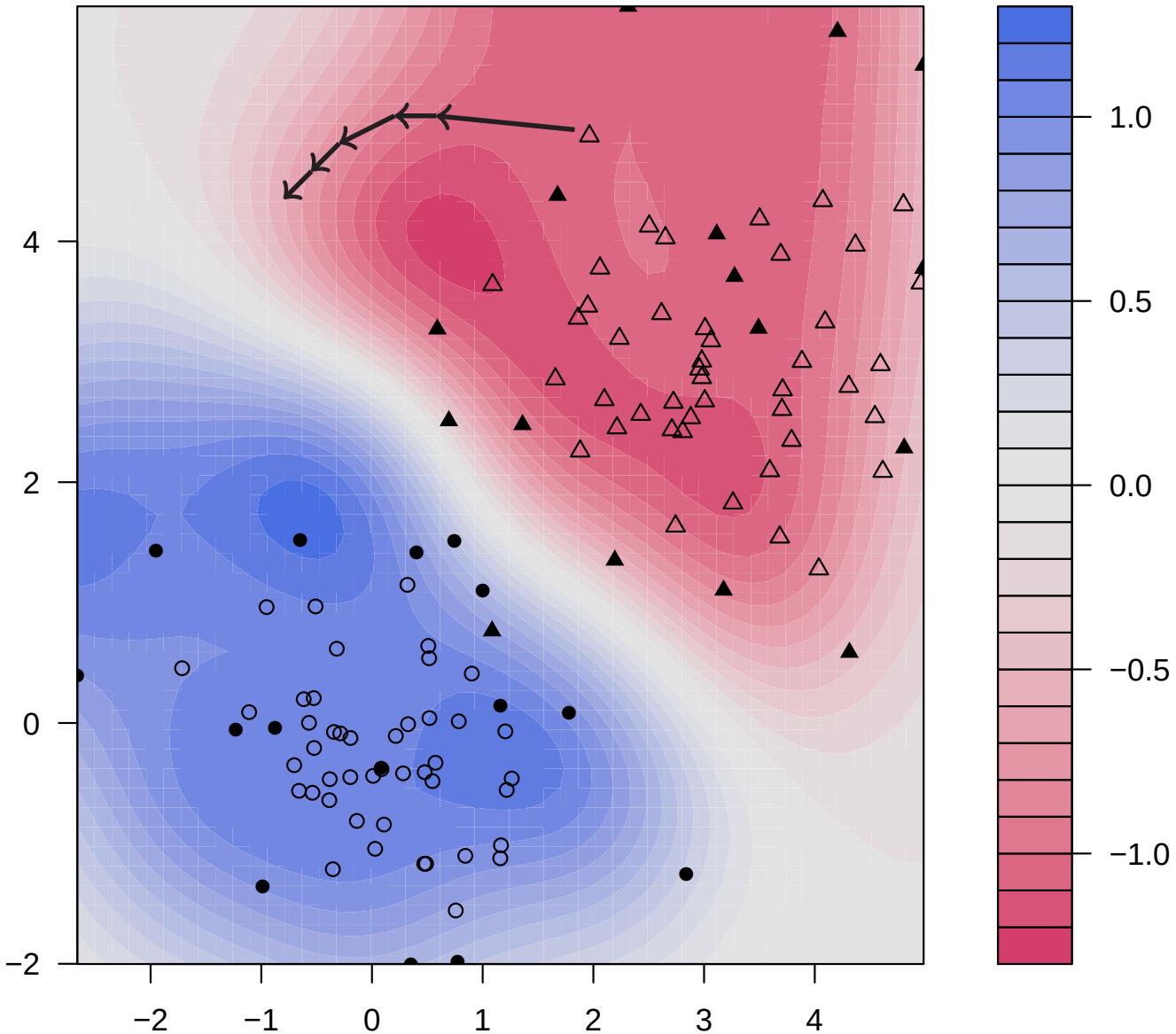

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

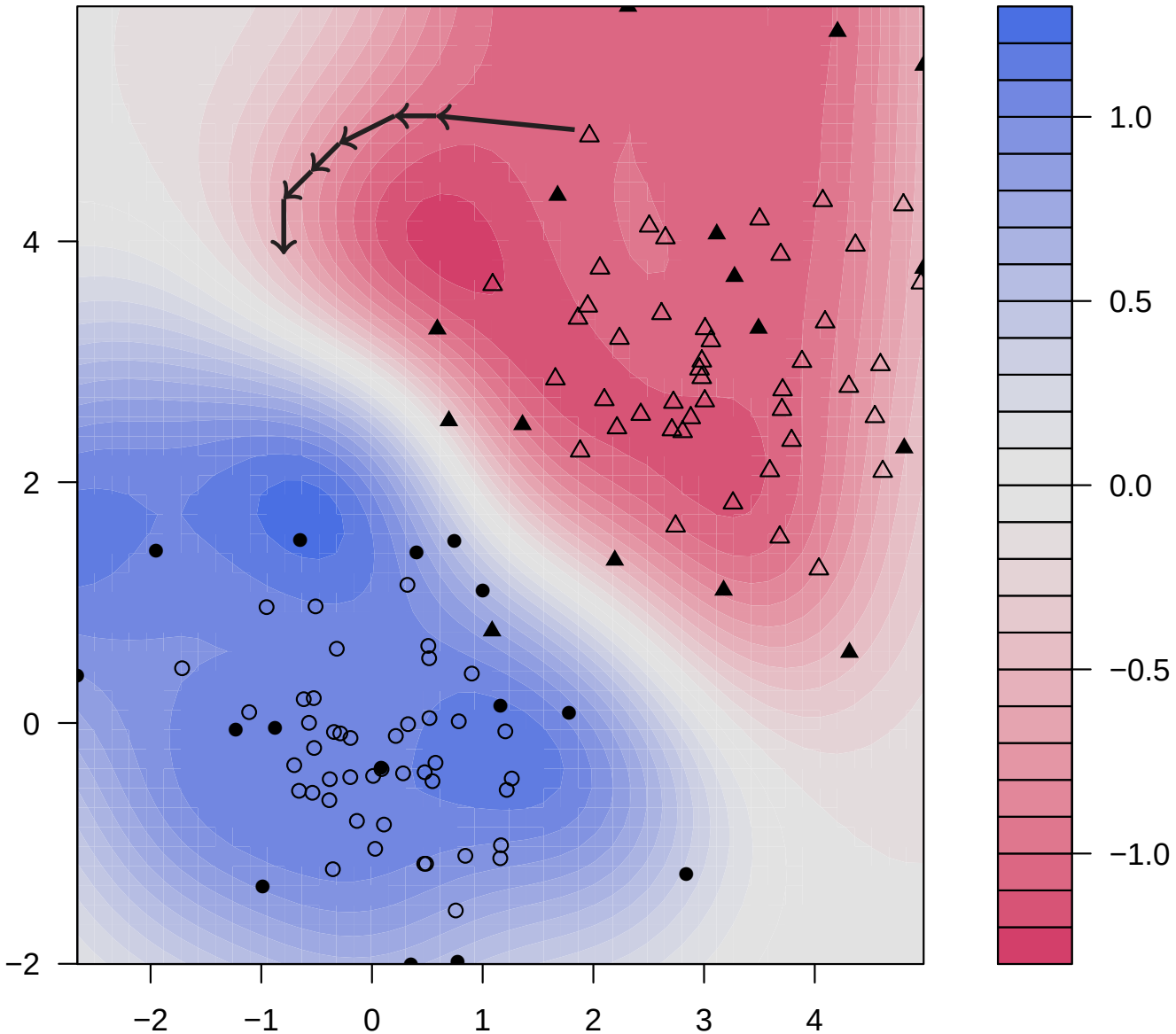

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

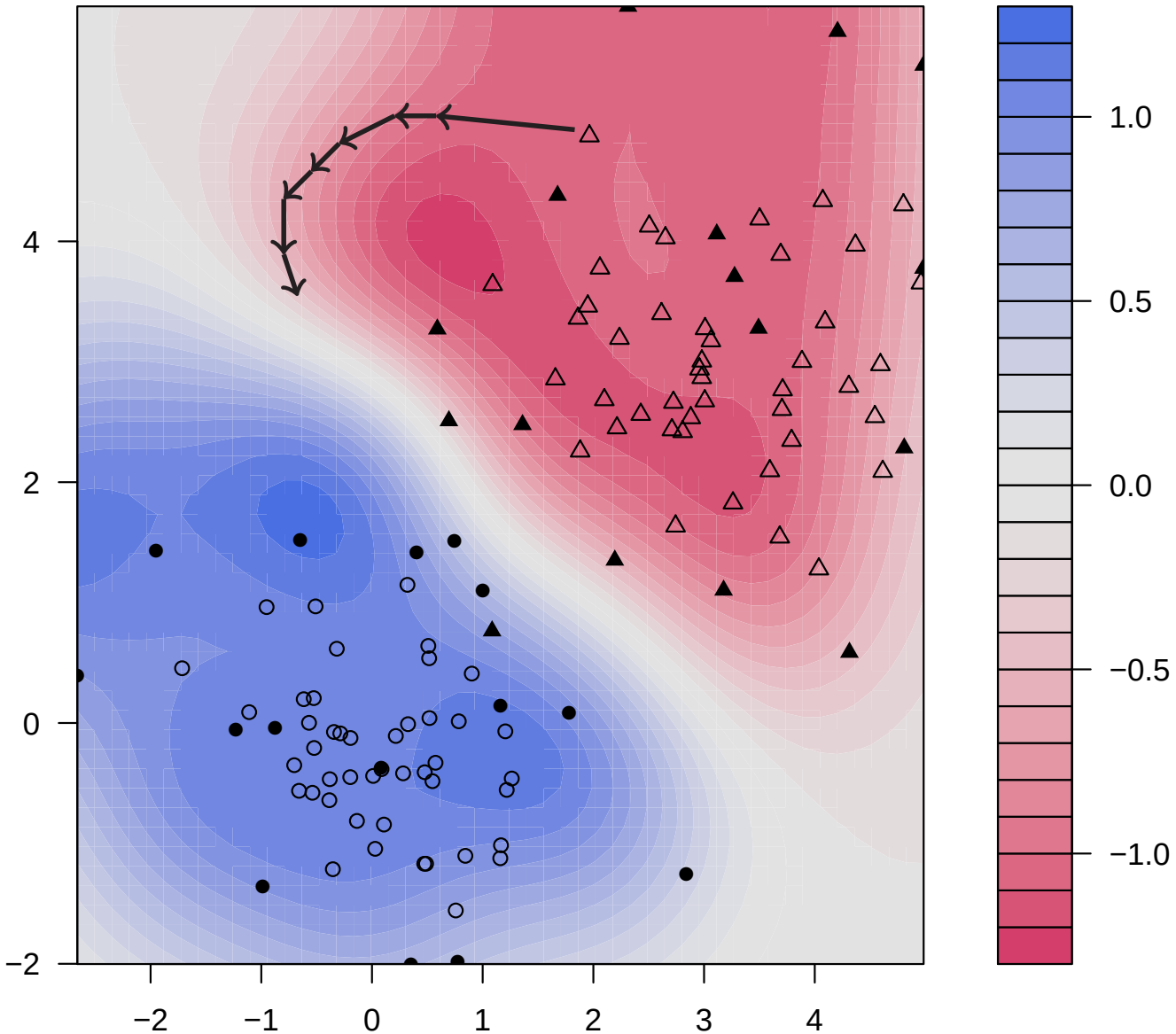

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

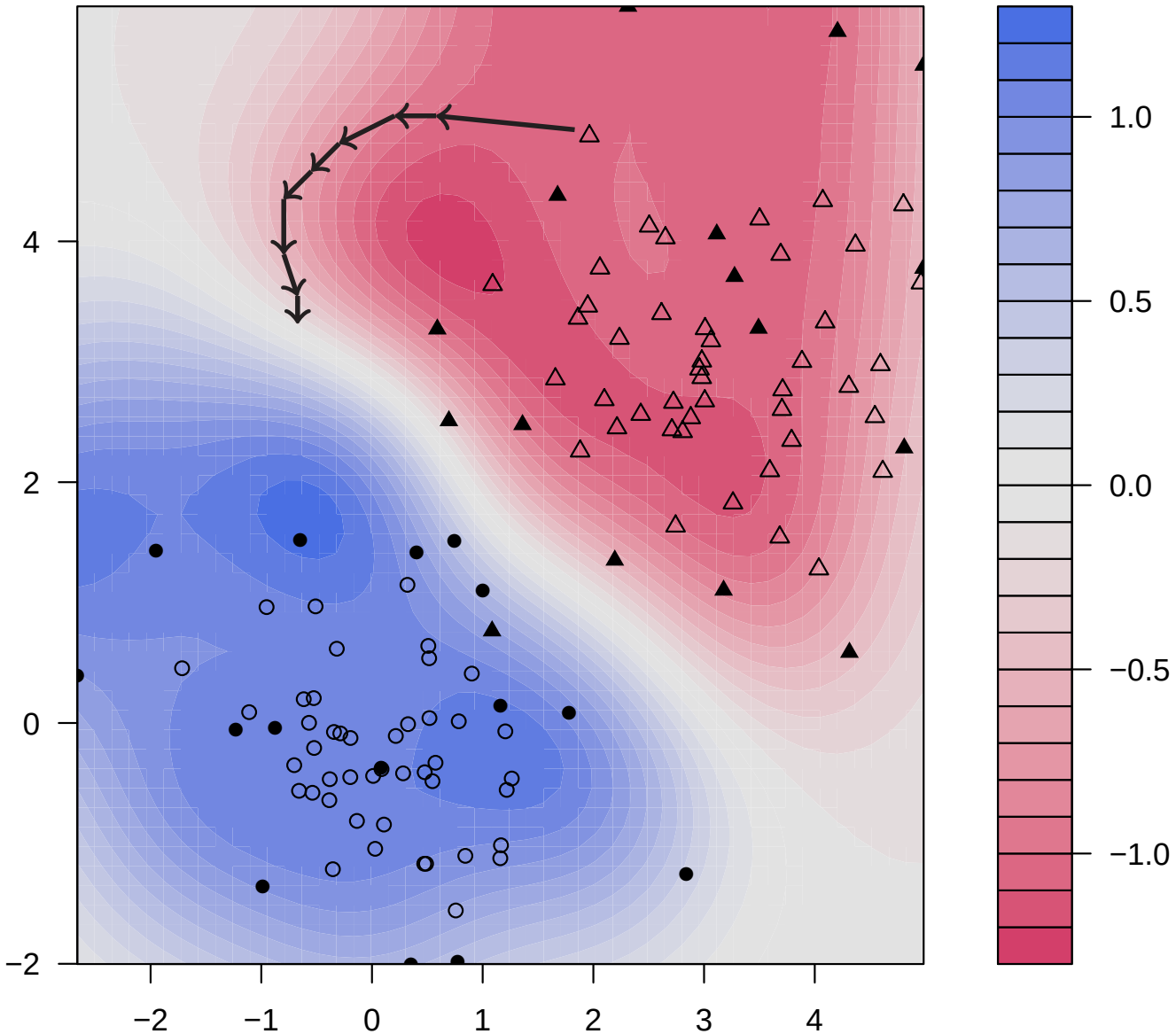

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

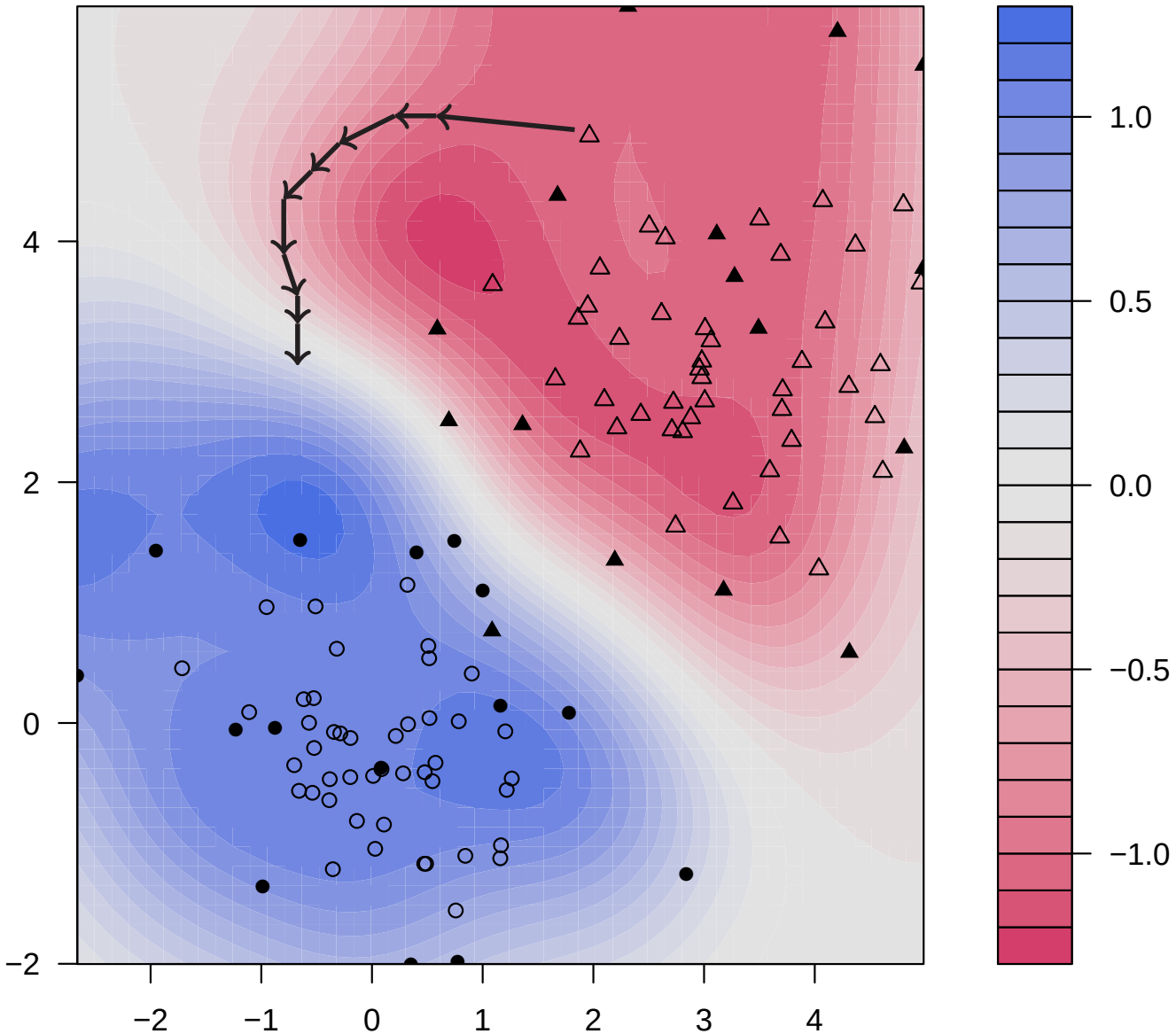

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

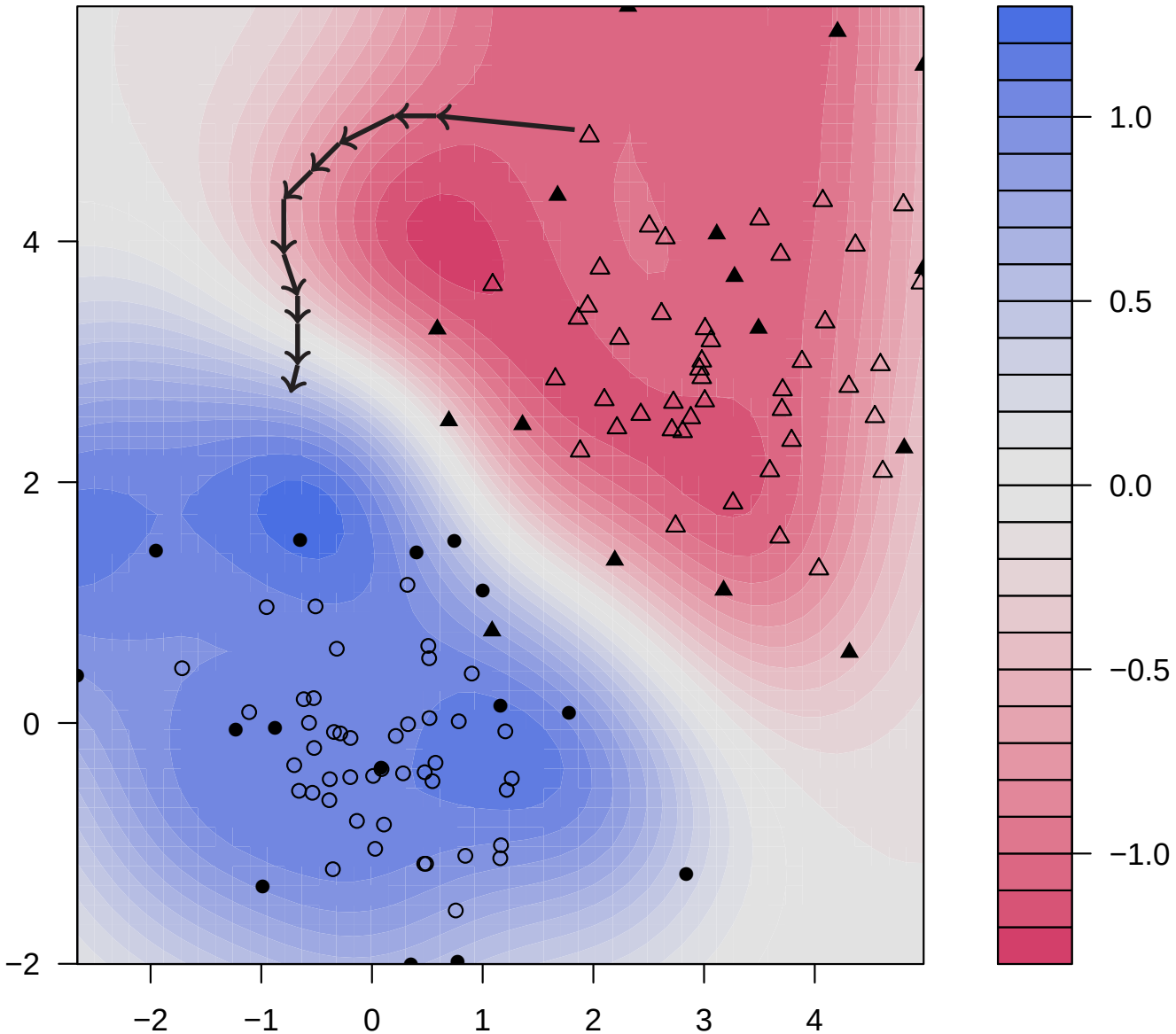

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

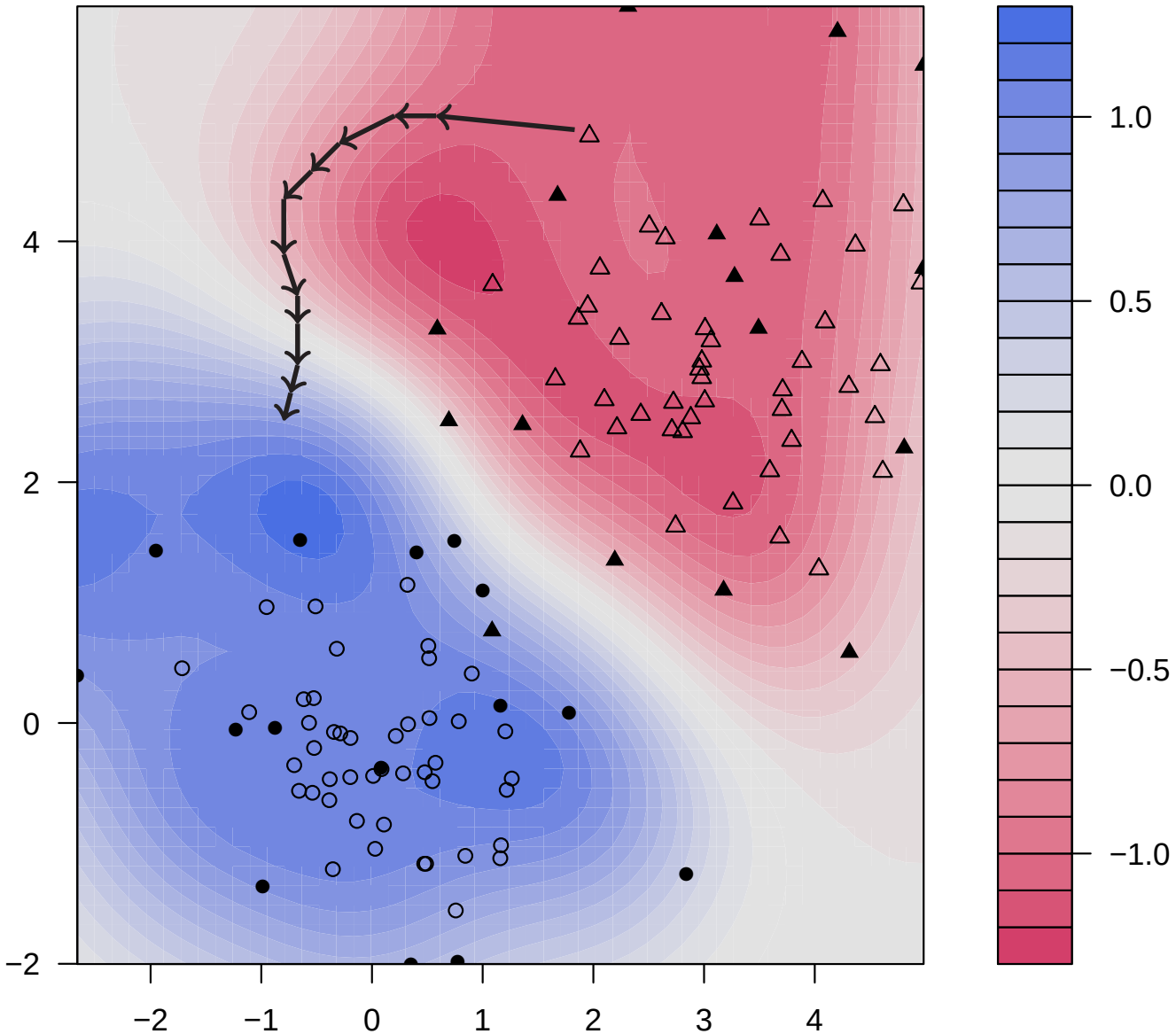

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

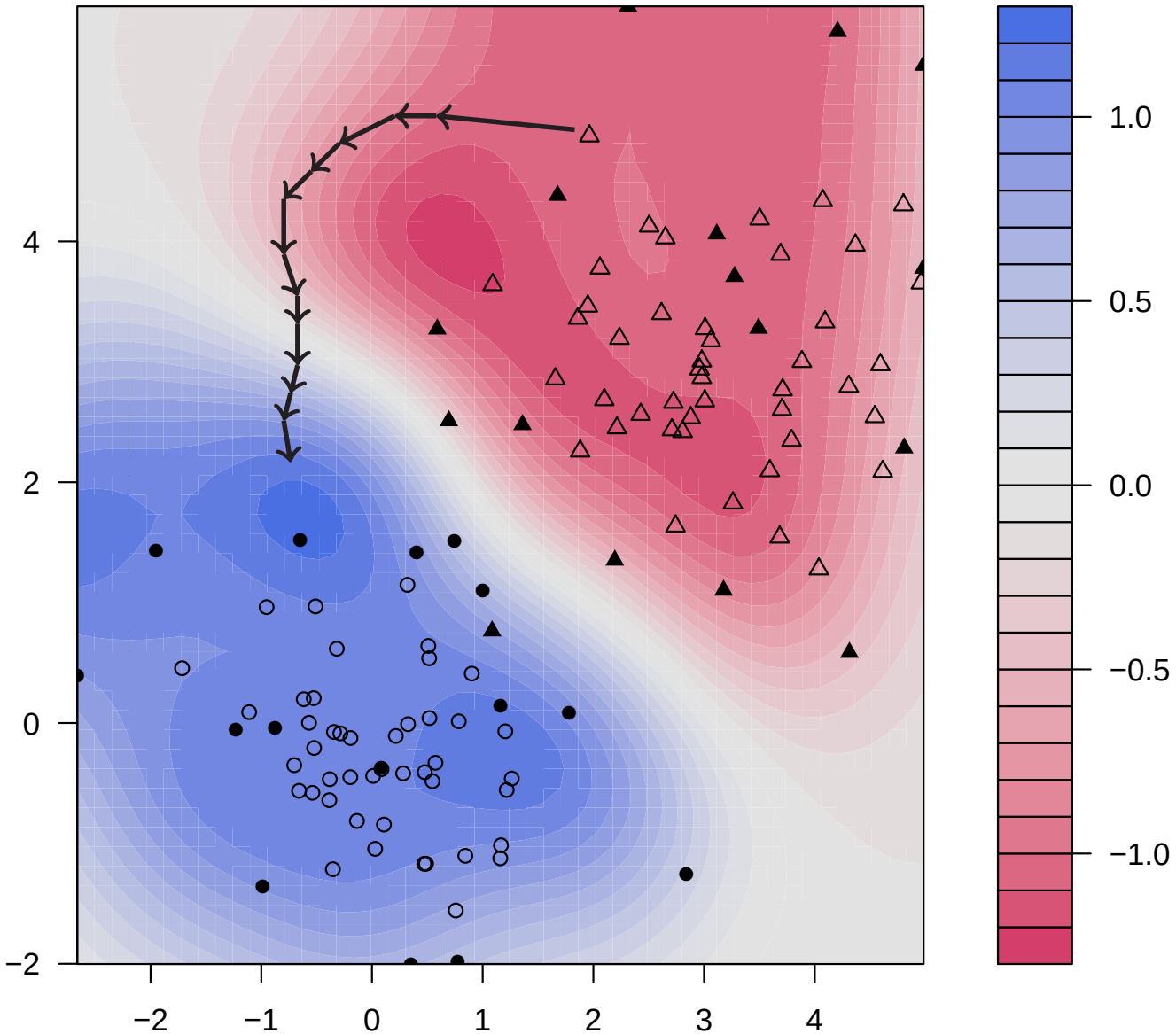

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

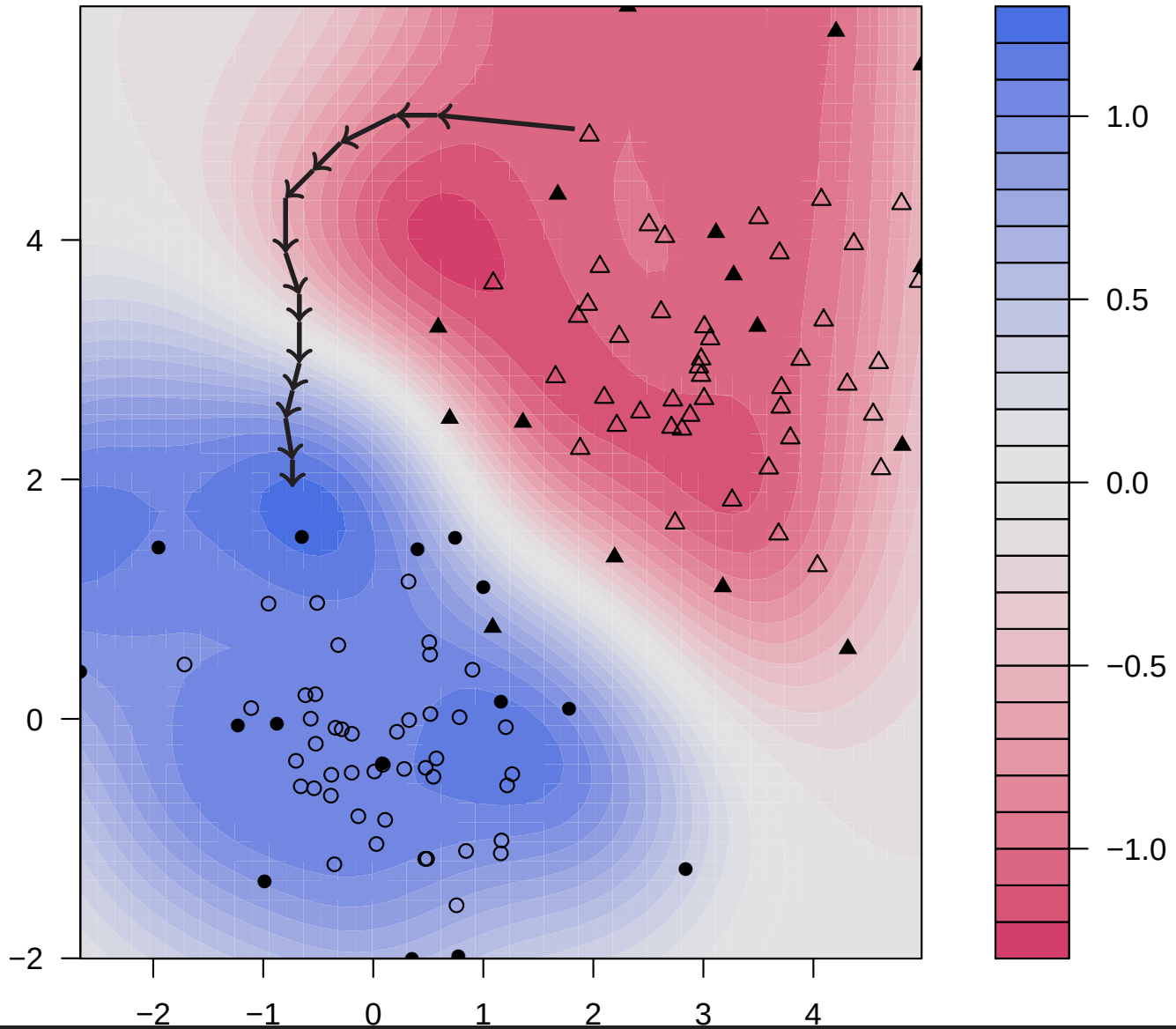

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0


4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

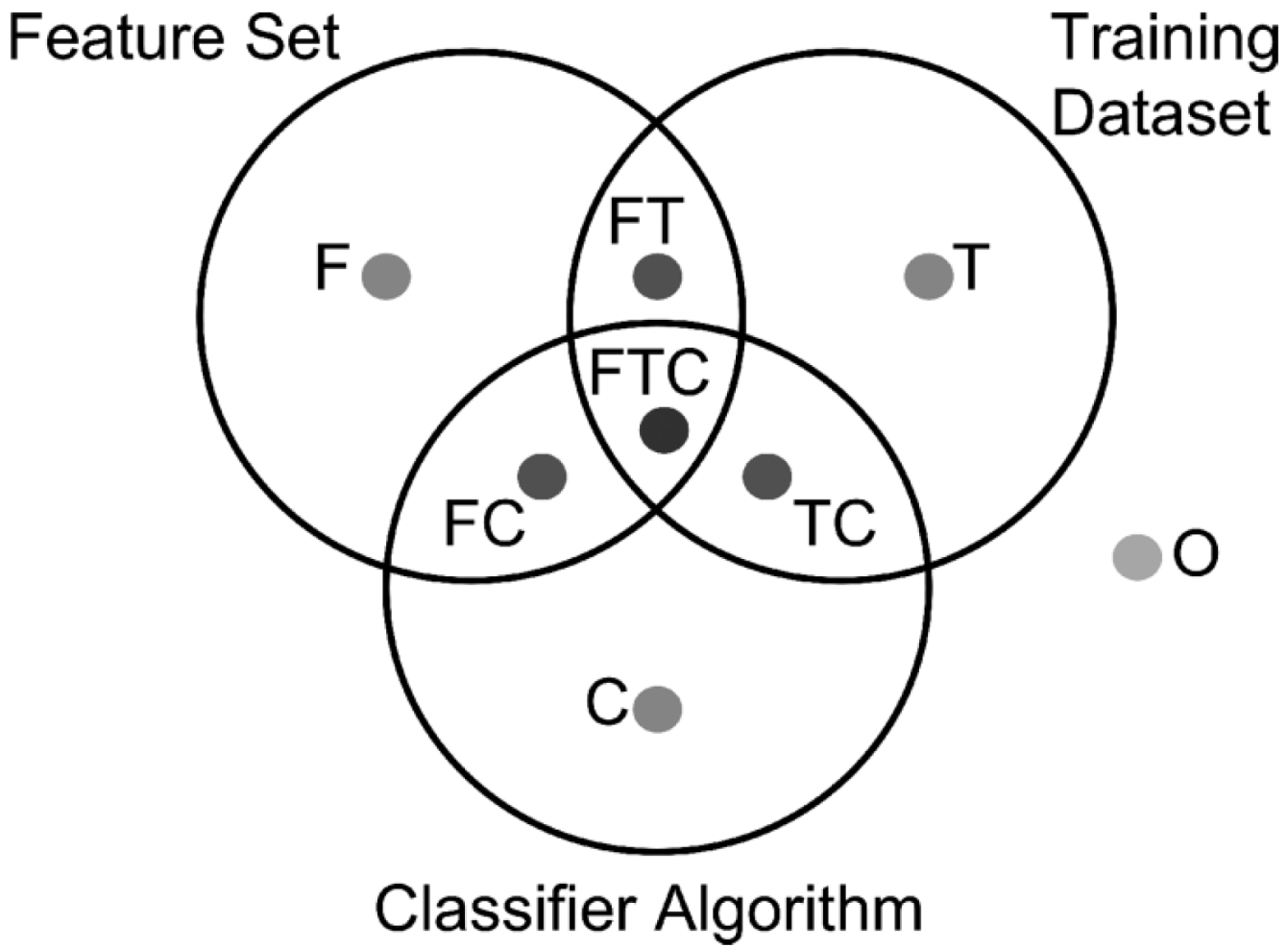

4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0


4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0


4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0


4
2
0
−2
−2
−1
0
1
2
3
4
1.0
0.5
0.0
−0.5
−1.0

What the adversary knows?



What the adversary knows?

Level 0 (basic): Knows the oracle decision and has access to the detector $\Longrightarrow$ Blind feedback

Level 1: Knows the classifier $\Longrightarrow$ Construction vulnerabilities

Level 2: Knows the feature space $\Longrightarrow$ Knows the relevant features

Level 3: Knows the training data $\Longrightarrow$ Replication

Index

Where are the machine learning vulnerabilities?

How does Adversarial ML exploit vulnerabilities?

How does Adversarial ML work in practice?

The Attack

First, we are going to create the model, in this case we suppose that we have complete information about everything

We are going to suppose that the adversary can increment or reduce any feature up to 4

In this scenario we are going to look for a malicious instance that will be classified as benign

The classifier: SVM

```
1  library(e1071)
2  library(caret)
3  library(mlbench)
4  data(BreastCancer)
5  dataset <- BreastCancer[complete.cases(BreastCancer),]
6  dataset<-as.matrix(sapply(dataset[-1],as.numeric))
7  correlations <- cor(dataset[,-10])
8  hcorrelations <- findCorrelation(correlations, cutoff=0.85)
9  dataset <- dataset[,-hcorrelations]
10 trainIndex<-sample(1:length(dataset[,1]),length(dataset[,1])*2/3)
11 trainData<-dataset[trainIndex,]
12 testData<-dataset[-trainIndex,]
13 svm_model <- svm(Class ~ ., data=trainData, probability=TRUE)
14 pred <- predict(svm_model,testData[,-9])
15 table(round(pred),testData[,9])
```

Collecting training information

We can check the range of values and the mean

```
malicious <- trainData[trainData[,"Class"]==2,]
benign <- trainData[trainData[,"Class"]==1,]
apply(benign,2,max)
apply(benign,2,min)
apply(malicious,2,max)
apply(malicious,2,min)
```

Check the efficiency of moving

Our oracle: We can move any feature +4 to -4:

```
predict(svm_model, malicious[,-9])[1]
malicious[1,]<-malicious[1,]-1
predict(svm_model, malicious[,-9])[1]
malicious[1,]<-malicious[1,]+1
predict(svm_model, malicious[,-9])[1]
```

Create variations and variants

We create a variation matrix: from +4 to -4

We create variants for the first elements using these variations

```
variations<-matrix(sample(-4:4,800,replace=TRUE),100,8)
variants <- t(variations) + malicious[1,-9]
variants[variants <=0]<-1
variants[variants >=11]<-10
prediction<-predict(svm_model, t(variants))
round(prediction)
sum(round(prediction)==1)#Benign
sum(round(prediction)==2)#Malicious
```

The Lowest variations

Finding the lowest number of variations gives us the most sensitive features

It is the shortest path between malicious and benign

```
1 succVar<-variants[,which(round(prediction)==1)]
2 apply(abs(succVar-malicious[1,-9]),2,sum)
```

Manufacturing variants

```
1  createVariants <- function(ind,model)
2  {
3     variations<-matrix(sample(-4:4,800,replace=TRUE),100,8)
4     variants <- t(variations) + ind
5     variants[variants <=0]<-1
6     variants[variants >=11]<-10
7     prediction<-predict(model, t(variants))
8     succVar<-variants[,which(round(prediction)==1)]
9     return(succVar)
10 }
```

Manufacturing variants per unit

```
1 totalVariants<-data.frame(row.names=colnames(malicious[,-9]))
2 for( i in 1:100){
3   vars<-createVariants(malicious[1,-9],svm_model)
4   if(length(vars)>0){
5     predict(svm_model,t(vars))
6     totalVariants <- cbind(totalVariants, vars)
7     totalVariants <- totalVariants[,duplicated(t(totalVariants))
          ==FALSE]
8   }
9 }
```

Manufacturing variants in general

```
1  totalVariants<-data.frame(row.names=colnames(malicious[,-9]))
2  for( j in 1:length(malicious[,1])){
3   for( i in 1:10){
4    vars<-createVariants(malicious[j,-9],svm_model)
5    if(length(vars)>0){
6      predict(svm_model,t(vars))
7      totalVariants <- cbind(totalVariants, vars)
8      totalVariants <- totalVariants[,duplicated(t(totalVariants))
           ==FALSE]
9    }
10  }
11 }
```

Exercise

Train three different classifiers and attack them using the same rules

Which one is more sensitive?