Machine Learning in R

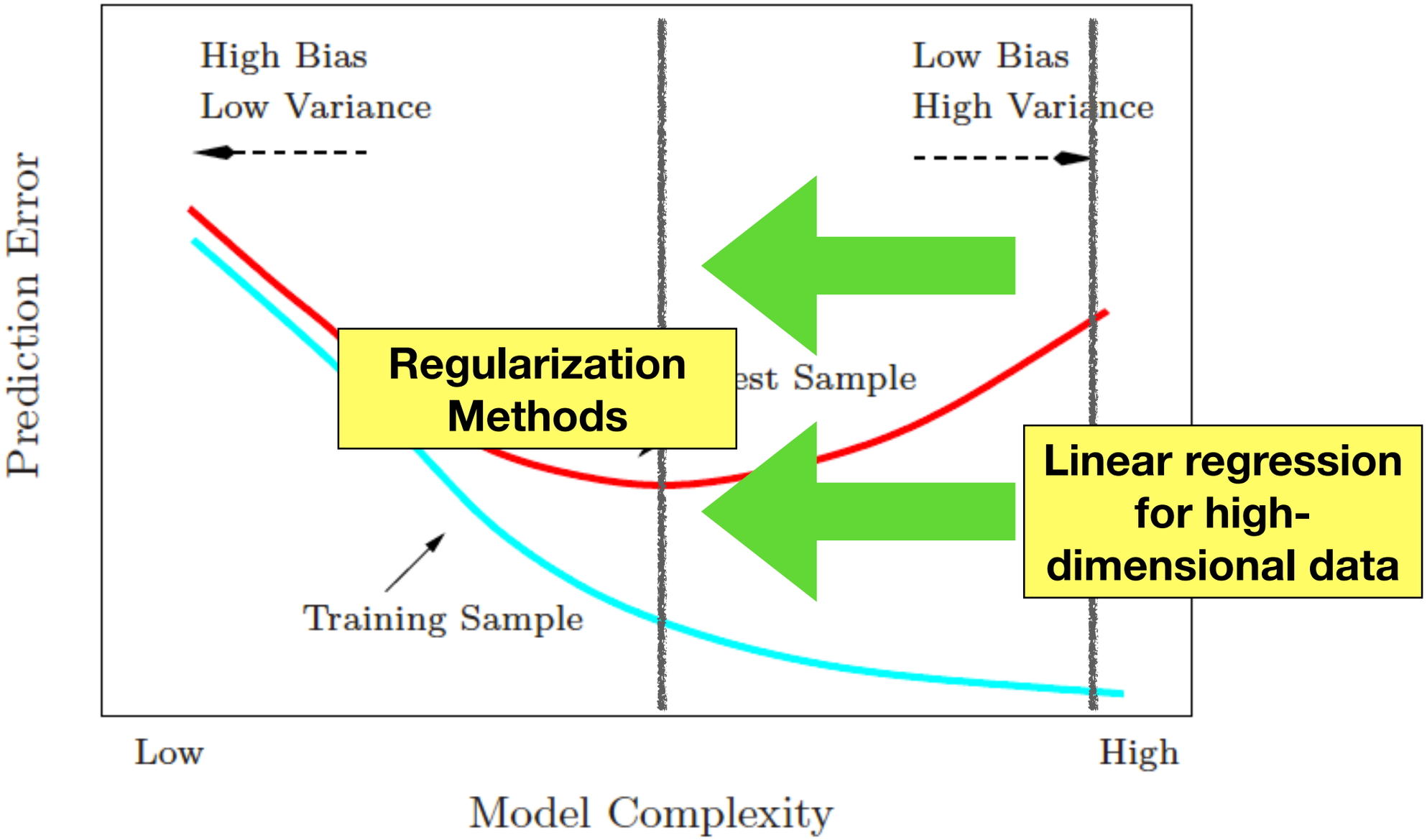
Lecture 4 – Support vector machines

Jean Feng – Epidemiology and Biostatistics

Last lecture

1. Model selection: Cross-validation
2. Low versus high-dimensional data
3. Regularization methods
    1. Ridge Regression
    2. Variable selection methods
        1. Variable subset selection
        2. Lasso
    3. Extensions

Recall: Regularization improves the bias-variance tradeoff



Recall: In penalized regression, we fit a model by minimizing the penalized empirical risk.

For example, ridge regression fits a model by solving:

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0) \right)^2}_{\text{Mean squared error}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}$$

We're going to see something like this pop up in today's lecture again!

Today's lecture

1. Maximum margin classifiers

2. Support vector classifiers

3. Support vector machines

4. SVMs as a penalization method

5. SVMs for multiple classes

Motivation

- ***Maximum margin classifier***, ***support vector classifier***, and ***support vector machine*** *(SVM)* are designed for binary classification tasks.

- Unlike logistic regression, there is no probability model for the data. Instead, SVMs are *geometrically motivated*.

- The idea is to generate ***decision boundaries*** that best separate the data.

Example dataset

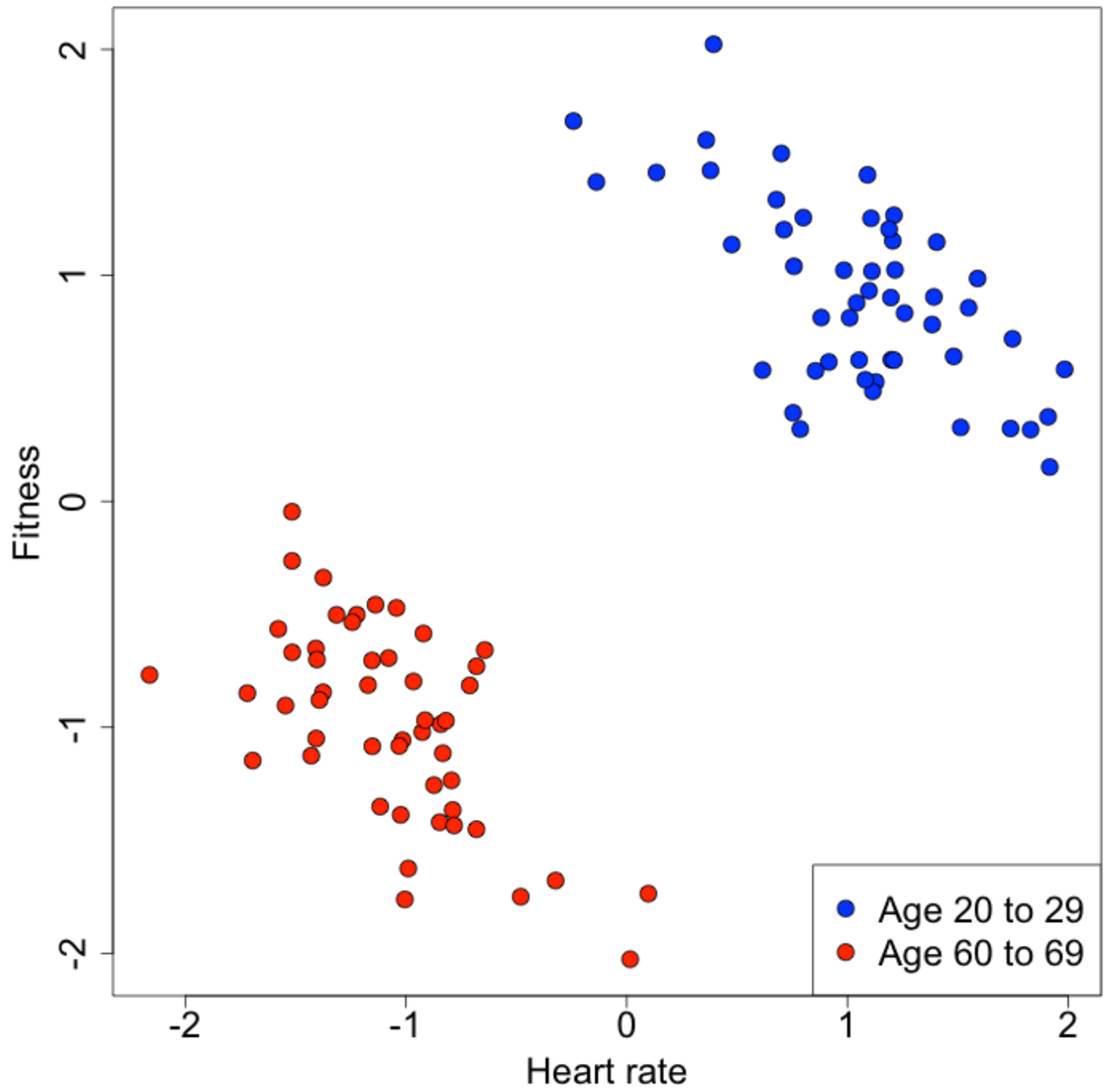
- Let's simulate data with the following measurements:
  - X_1 = normalized heart rate
  - X_2 = fitness measure
  - Y = age group
    - $Y = -1$ means the subject is 60-69 years old
    - $Y = 1$ means the subject is 20-29 years old

	HeartRate	Fitness	ageGrp
95	0.1518579	-2.51375243	-1
168	1.3867171	1.04422744	1
142	3.0515427	0.01138957	1
158	1.2044311	0.80480815	1
13	-0.3274814	-0.42026735	-1
164	1.6208782	1.64587715	1

Example dataset

- Constructing a decision rule that separates the two age groups is straightforward for this dataset.

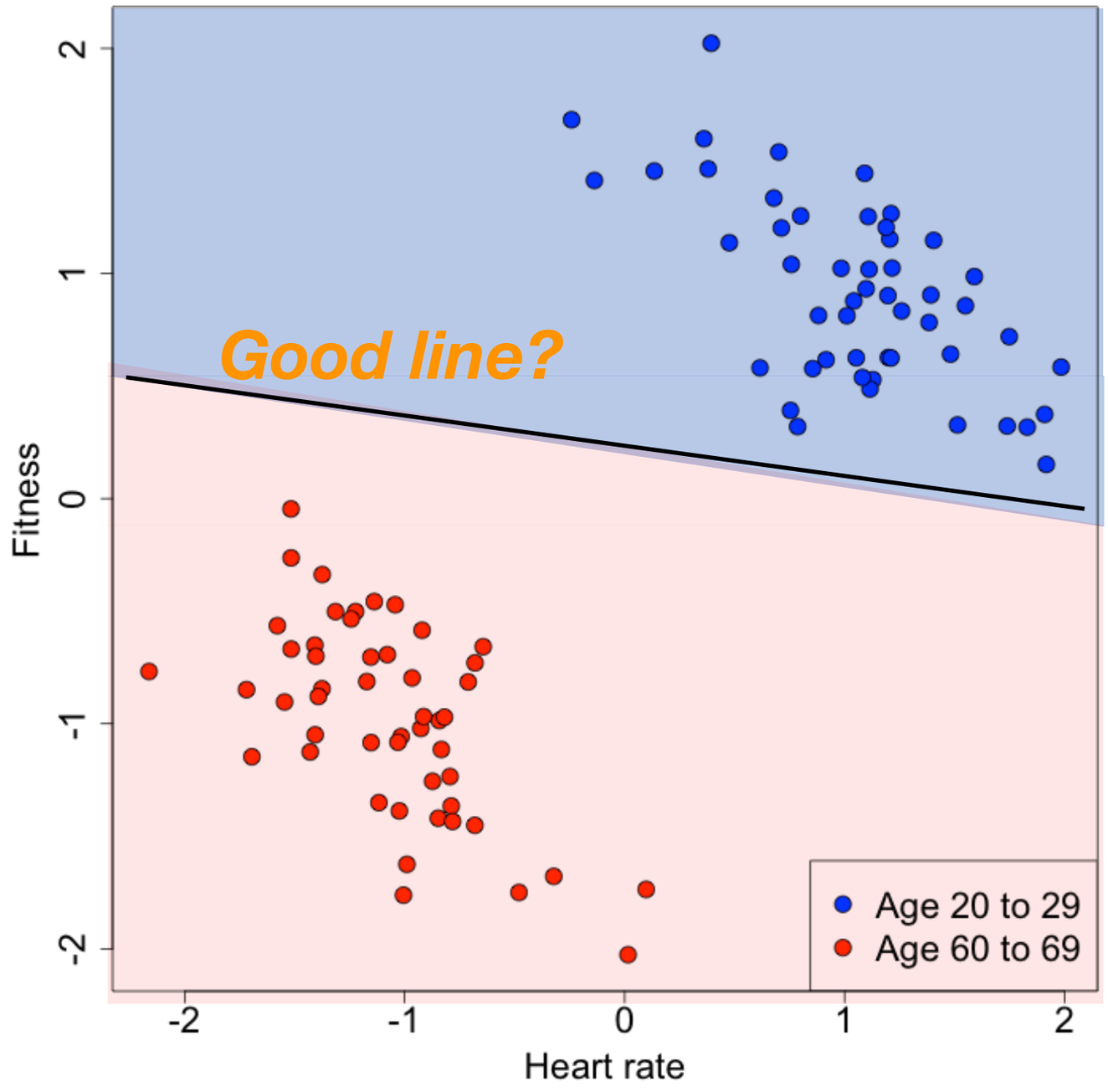
- Draw a line to separate the groups.



A separating line

- Constructing a decision rule that separates the two age groups is straightforward for this dataset.

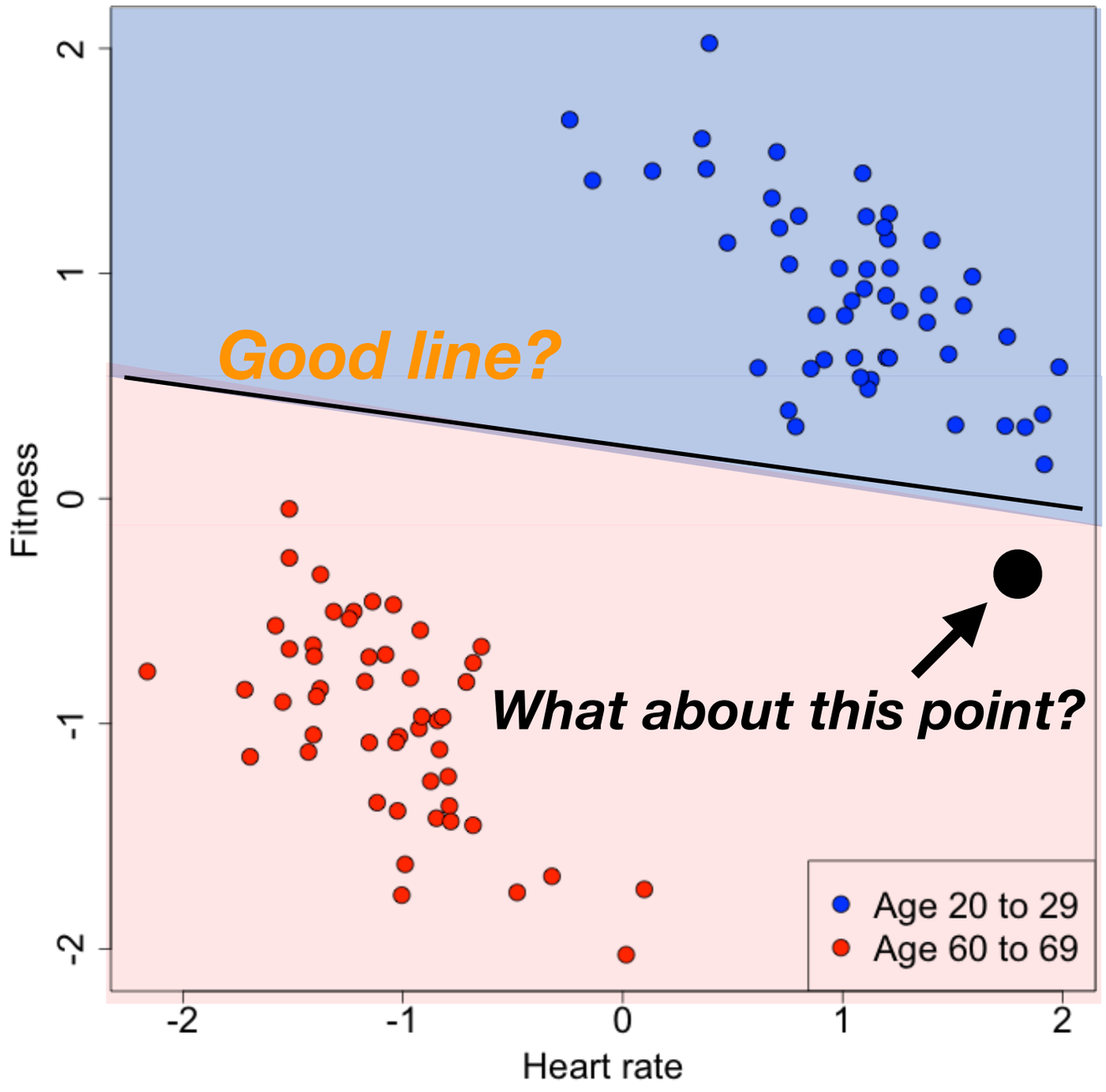
- Draw a line to separate the groups.



A separating line

- Constructing a decision rule that separates the two age groups is straightforward for this dataset.

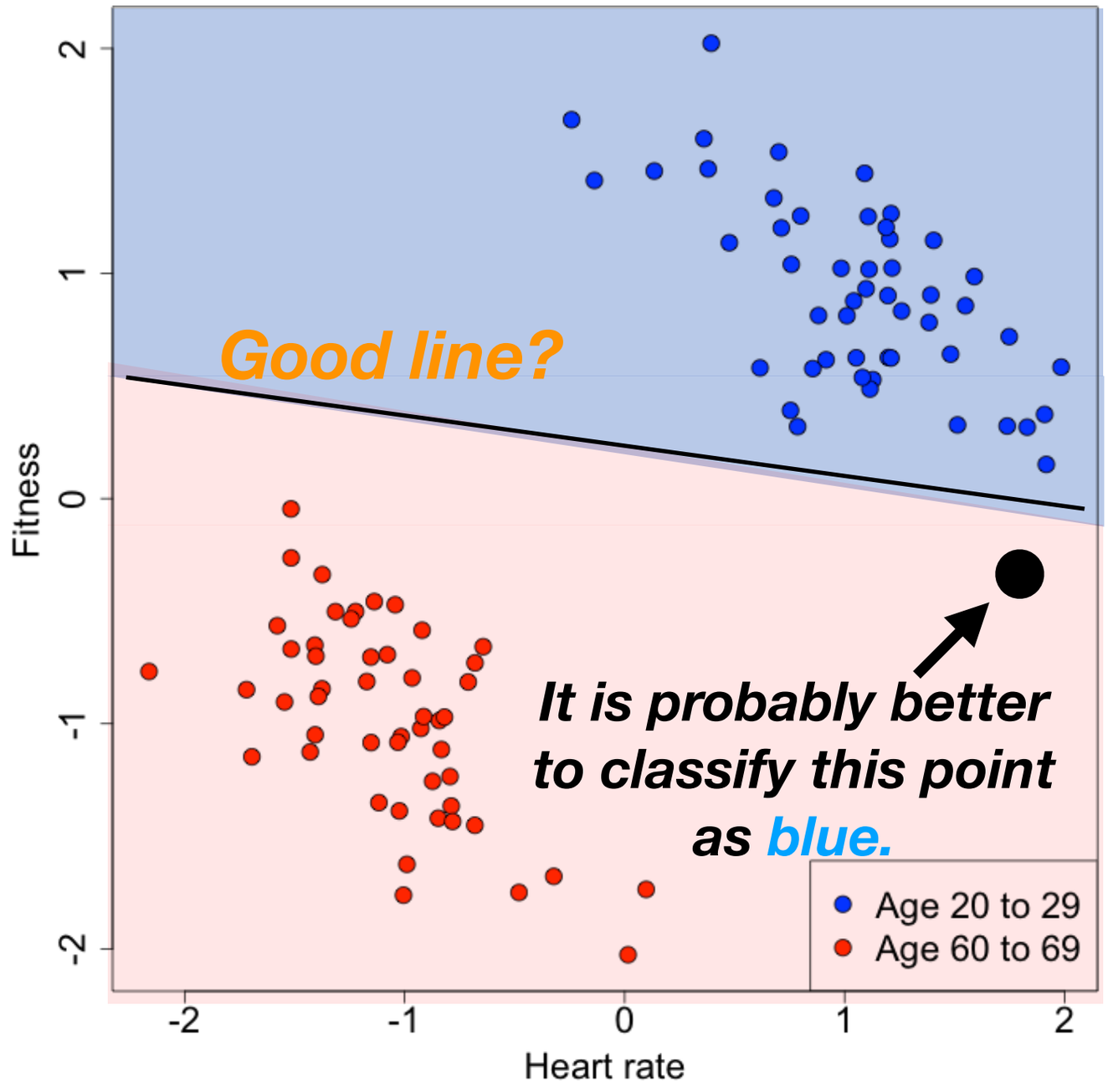
- Draw a line to separate the groups.



A separating line

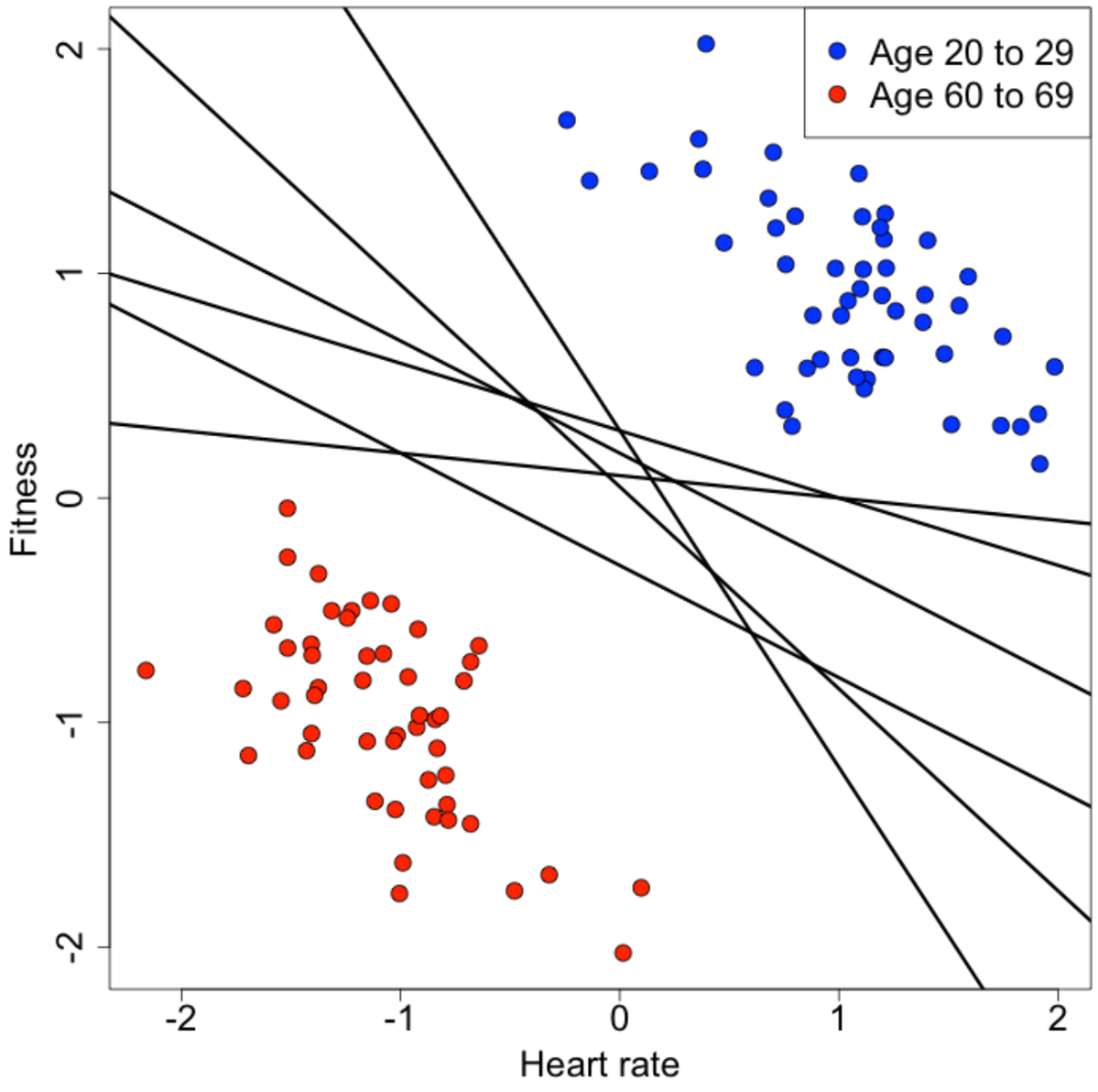
- Constructing a decision rule that separates the two age groups is straightforward for this dataset.

- Draw a line to separate the groups.



Many separating lines

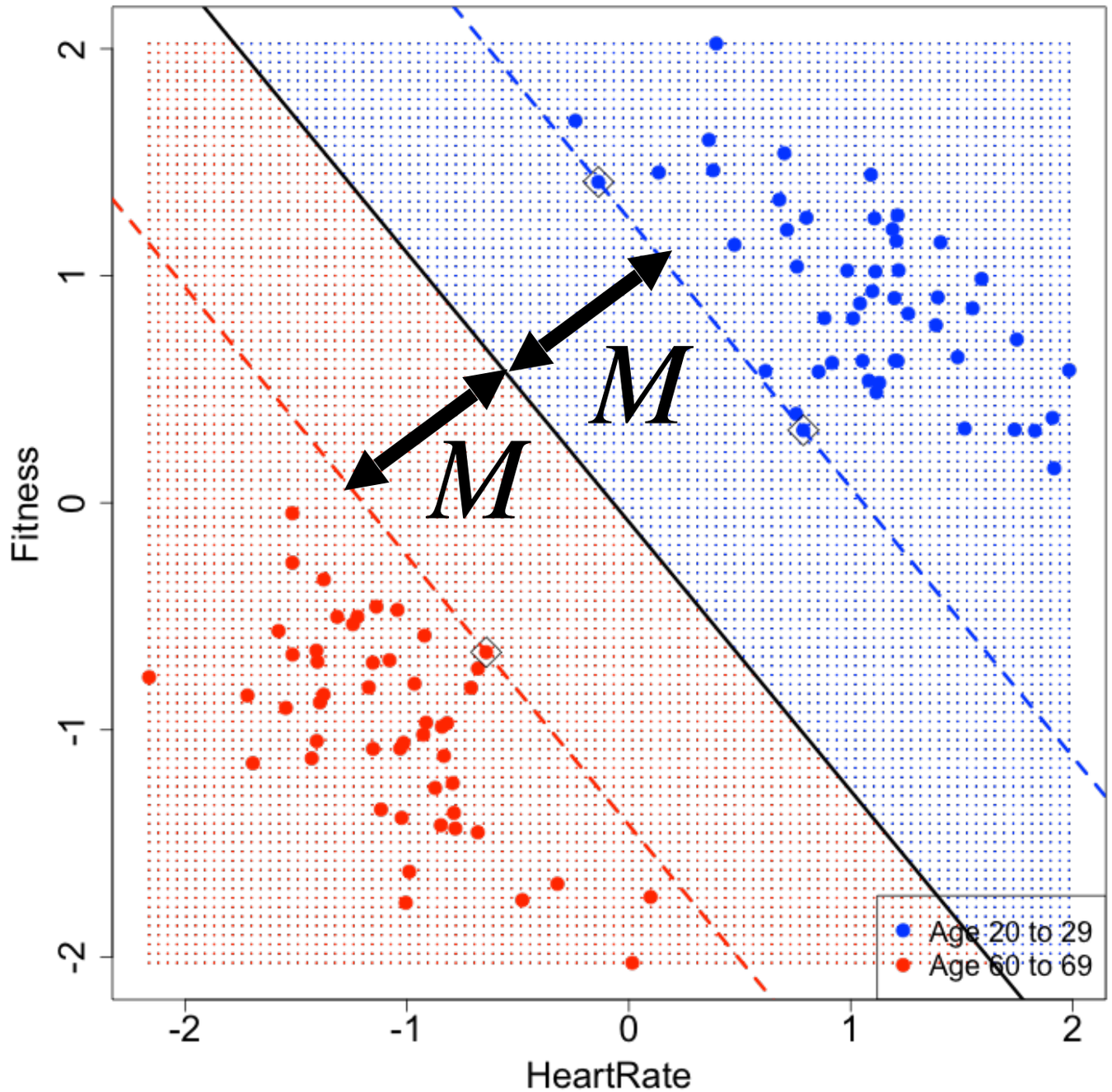
- There are actually many separating lines.
- Which one is the best one?



Maximum margin classifier

- The maximum margin classifier finds the separating line that is farthest away from any training points.

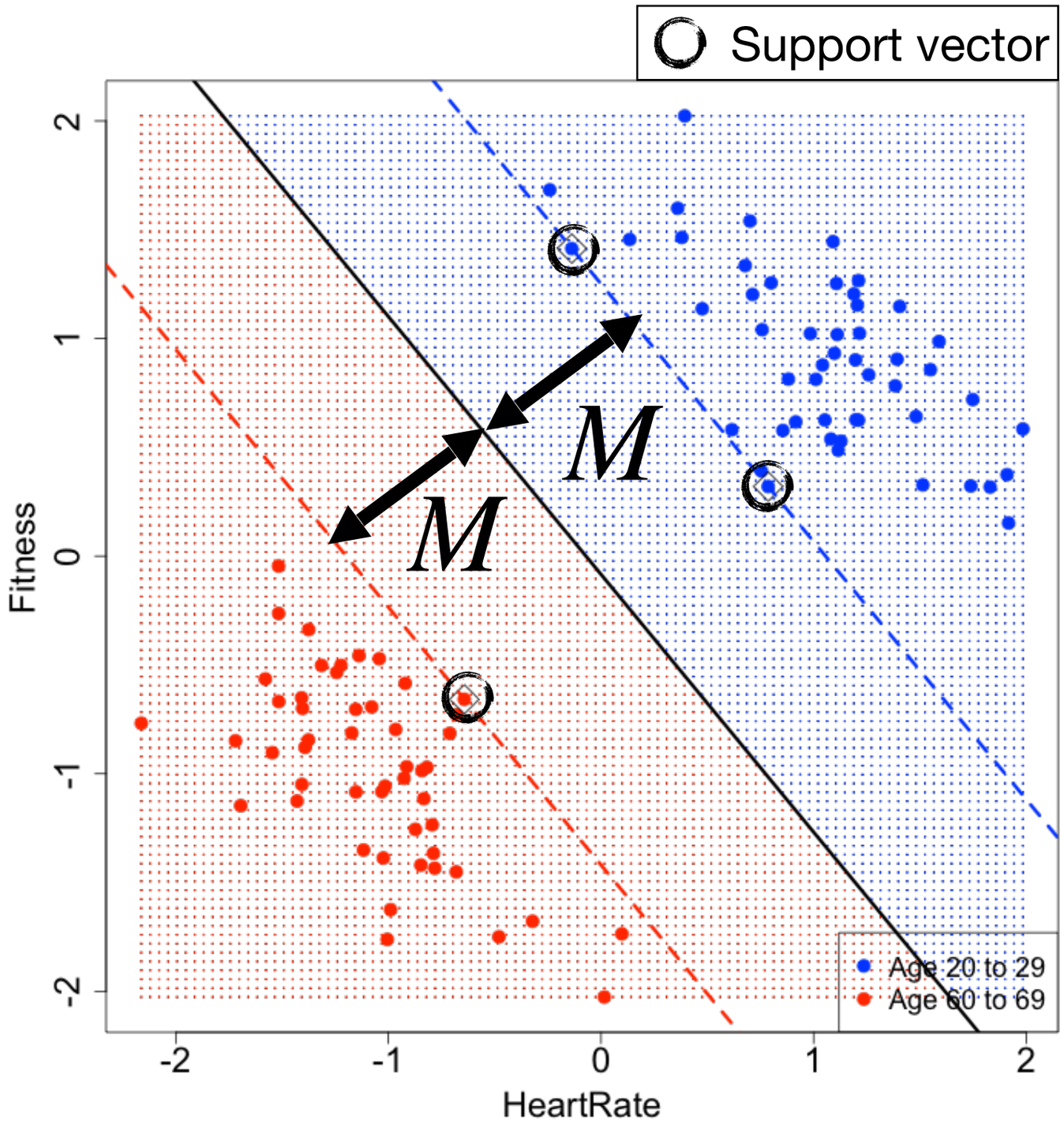
- The smallest distance between the decision boundary and training data is called the ***margin*** M.



Maximum margin classifier

- The points that are closest to the decision boundary determine the location and orientation of the decision boundary.

- These circled points are referred to as the ***support vectors***.



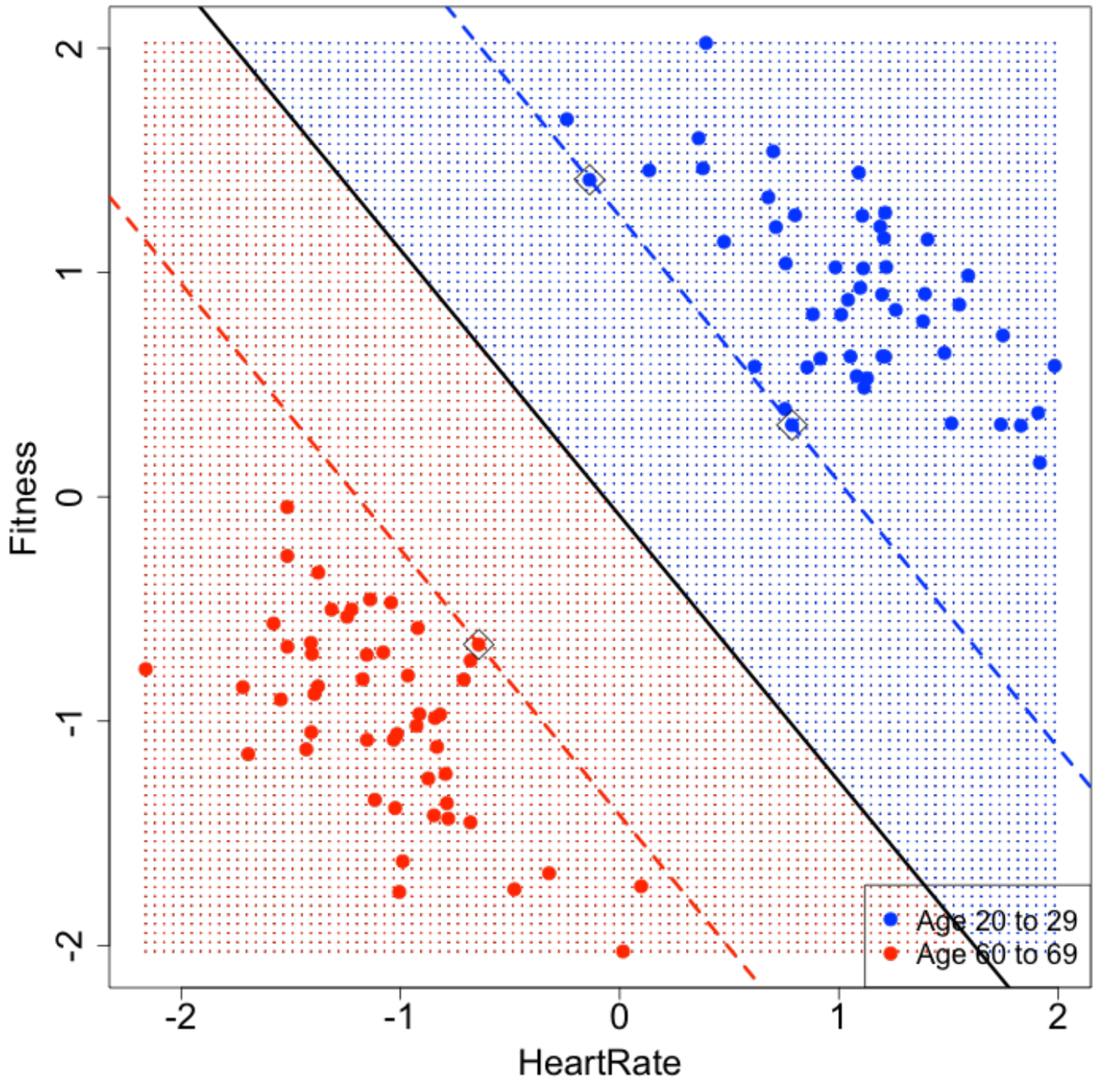
The decision function

- The maximum margin classifier fits a decision function of the form $f_{\beta_0,\beta}(x) = \beta_0 + \beta^\top x.$

- We can classify a new point x according to the sign of $f_{\beta_0,\beta}(x)$:

  - If $f_{\beta_0,\beta}(x)$ is positive, then predict class 1.

  - If $f_{\beta_0,\beta}(x)$ is negative, then predict class -1.



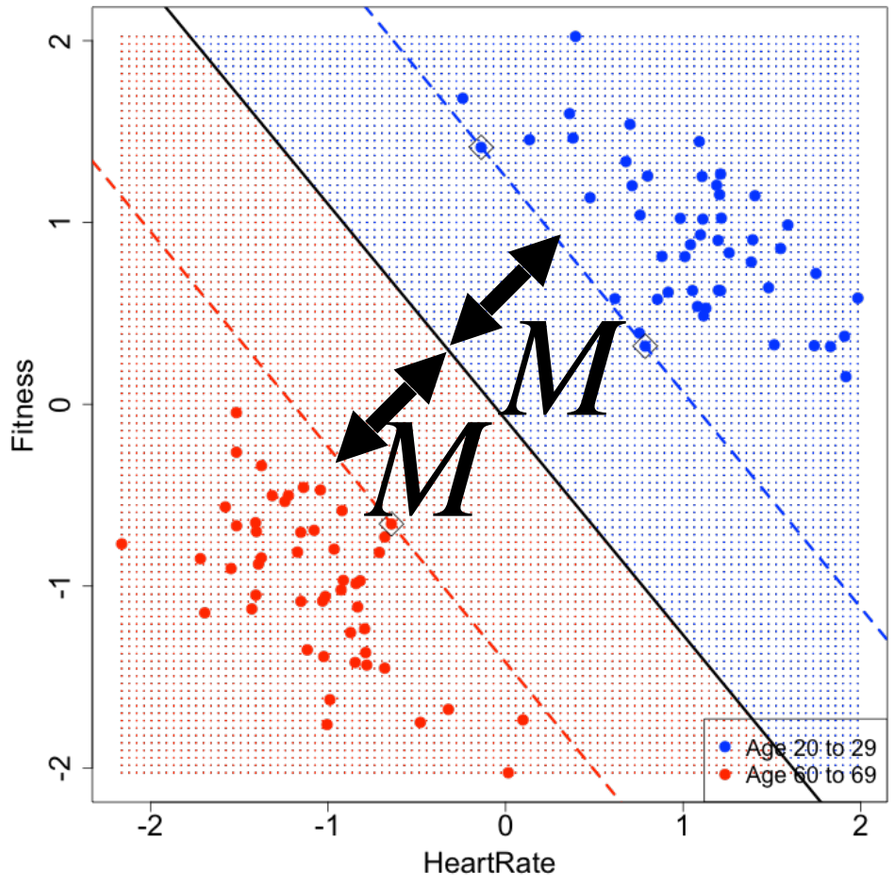
Fitting a decision function

For a training dataset $\{(x_i, y_i) : i = 1,...,n\}$, the maximum margin classifier fits the decision function by solving the following optimization problem:

$$\max_{\beta_0,\beta,M} M \quad \textit{(maximize the margin)}$$

such that $\|\beta\|_2^2 = 1$ and

$$\underbrace{y_i(\beta_0 + \beta^\top x_i)}_{\textit{Distance to the decision boundary}} \geq M \text{ for all } i = 1,...,n$$

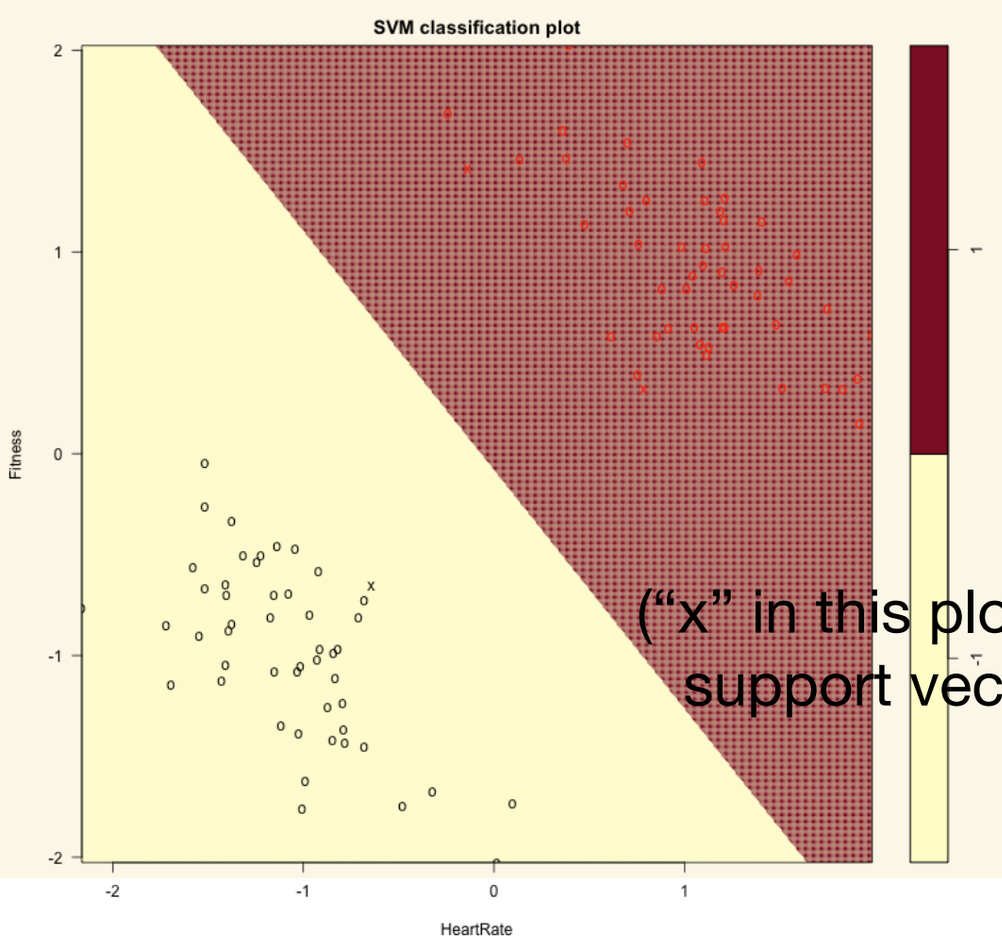


(Such that all points are correctly classified and are at least distance M away from the decision boundary)

Fitting a maximum margin classifier in R

```
> library(e1071)  ### library containing svm function
>
> svmFit <- svm(ageGrp ~ Fitness + HeartRate, data=dat, kernel="linear", cost=10, scale=FALSE)
> summary(svmFit)

Call:
svm(formula = ageGrp ~ Fitness + HeartRate, data = dat, kernel = "linear",
    cost = 10, scale = FALSE)

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10

Number of Support Vectors:  3

 ( 1 2 )


Number of Classes:  2

Levels:
 -1 1
```

```
> plot(svmFit, grid=500, data=dat)
```



("x" in this plot is a support vector)

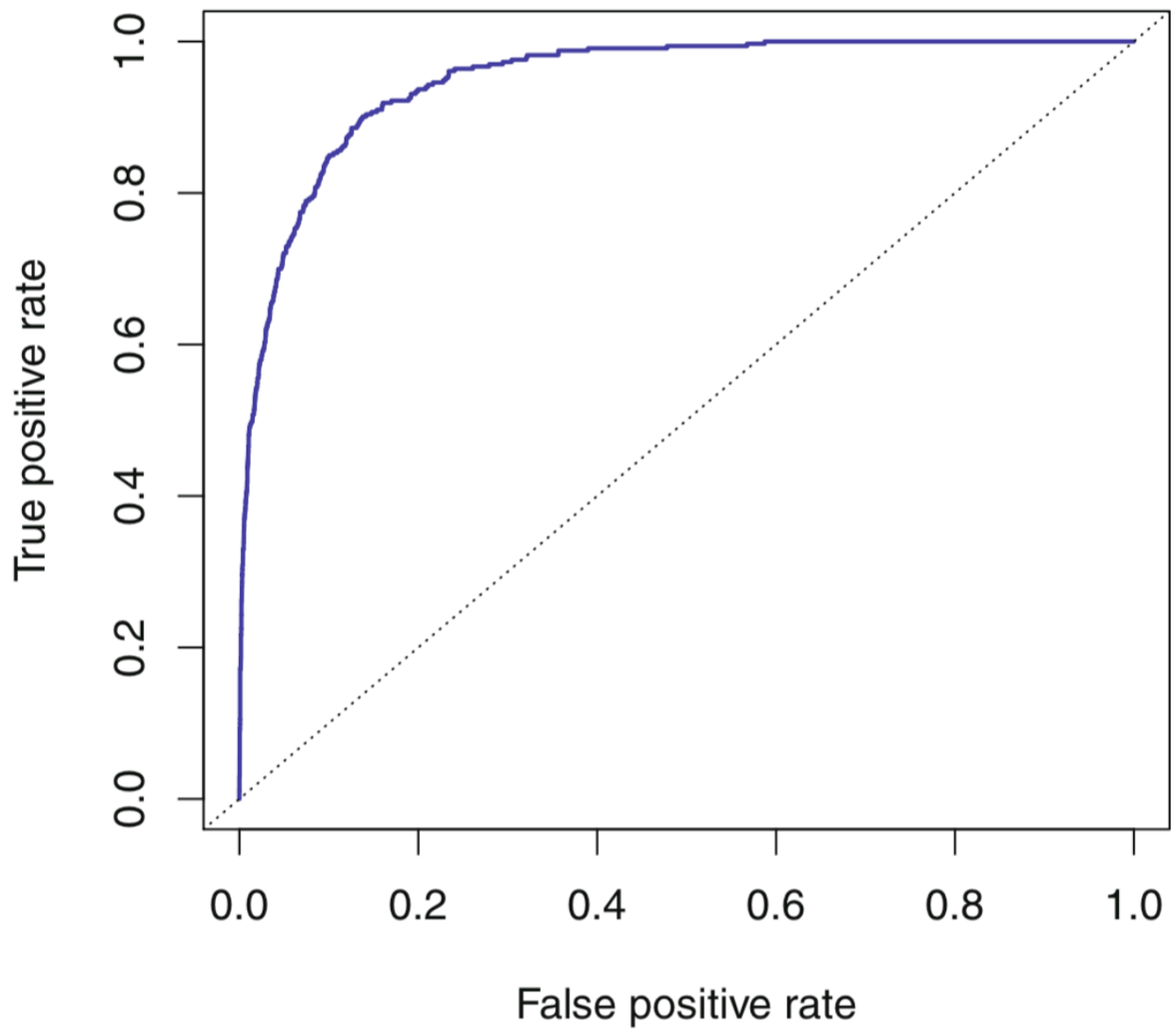
Quiz: ROC curves and SVMs

- **Q**: How would you make an ROC curve for a maximum margin classifier?

- **A**. You don't. The maximum margin classifier only outputs binary predictions.

- **B**. Threshold on the value of the decision function.

- **C**. Threshold on the distance to the decision boundary.



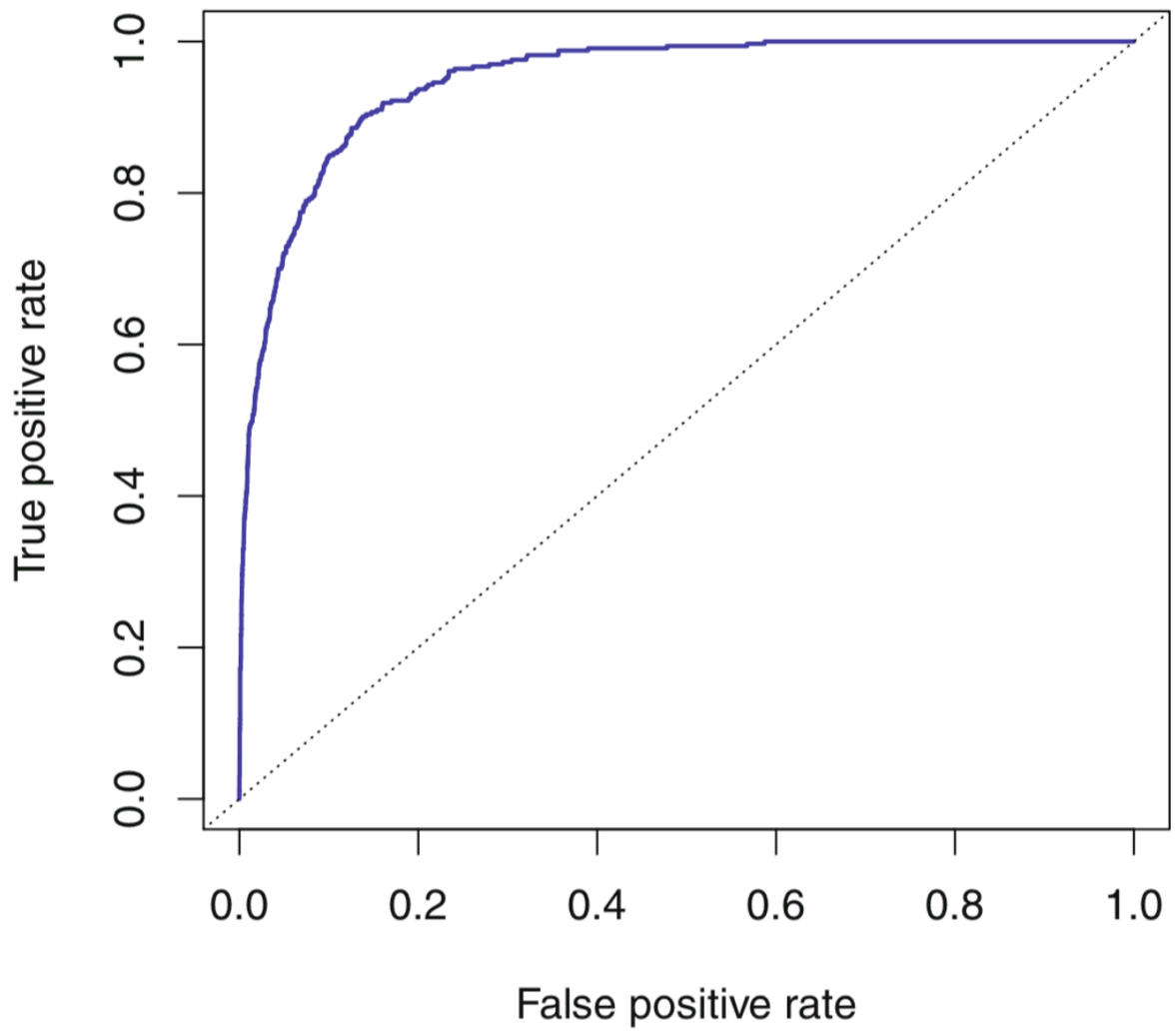
Quiz: ROC curves and SVMs

- **Q**: How would you make an ROC curve for a maximum margin classifier?

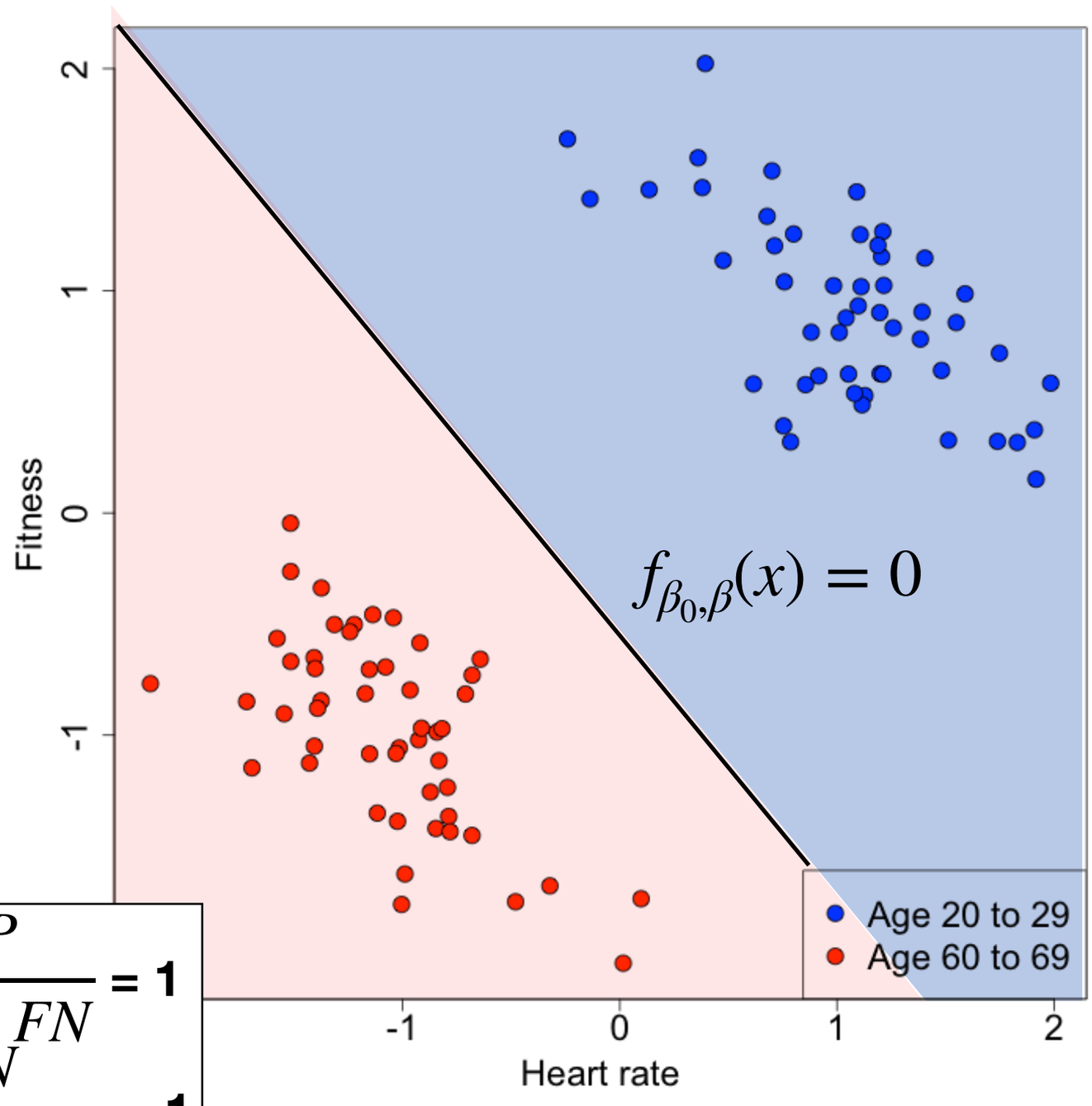
- **A**. You don't. The maximum margin classifier only outputs binary predictions.

- **B**. Threshold on the value of the decision function.

- **C**. Threshold on the distance to the decision boundary.



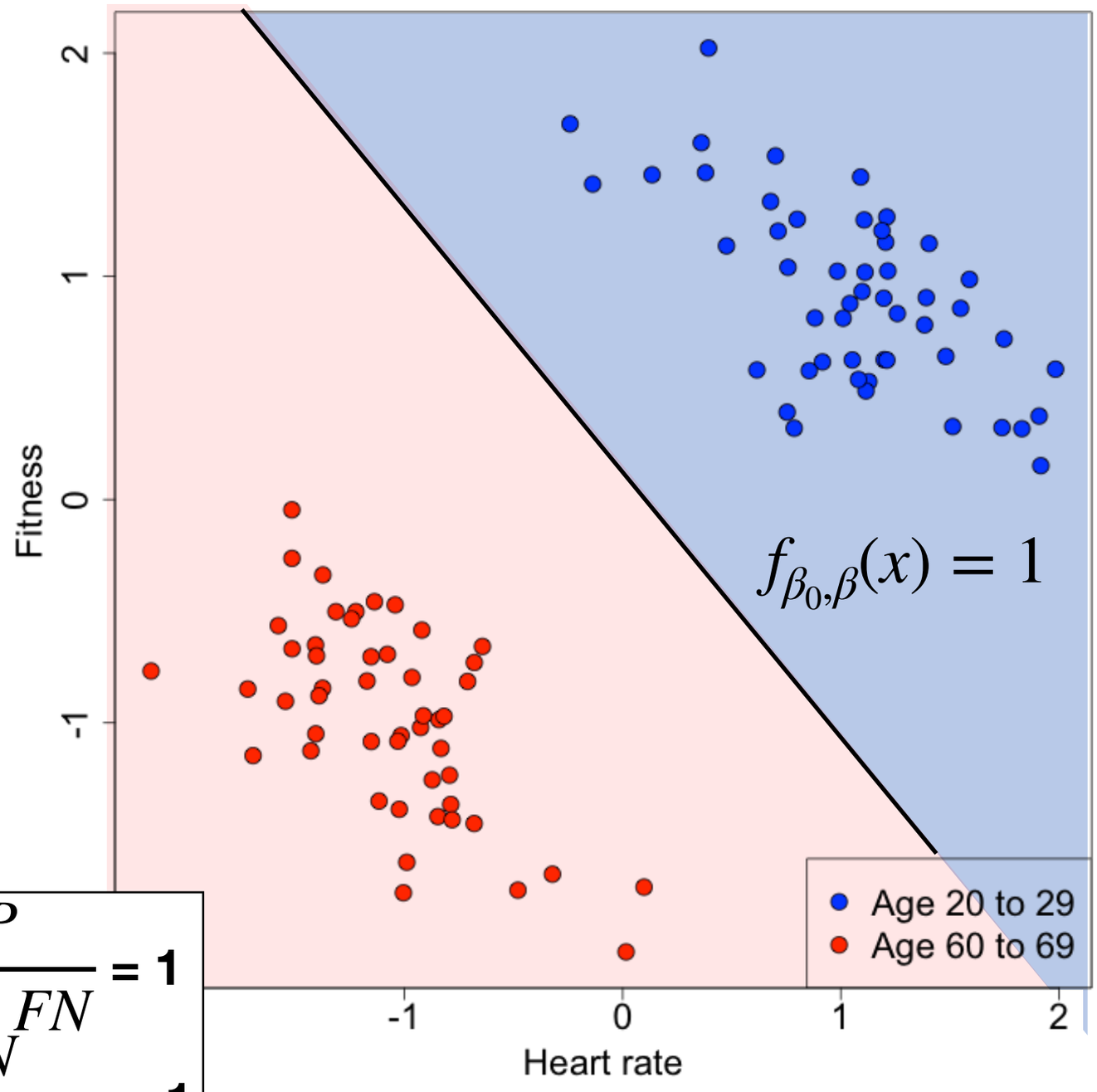
Thresholding on the decision function



$$\textbf{Sensitivity} = \frac{TP}{TP + FN} = 1$$

$$\textbf{Specificity} = \frac{TN}{TN + FP} = 1$$

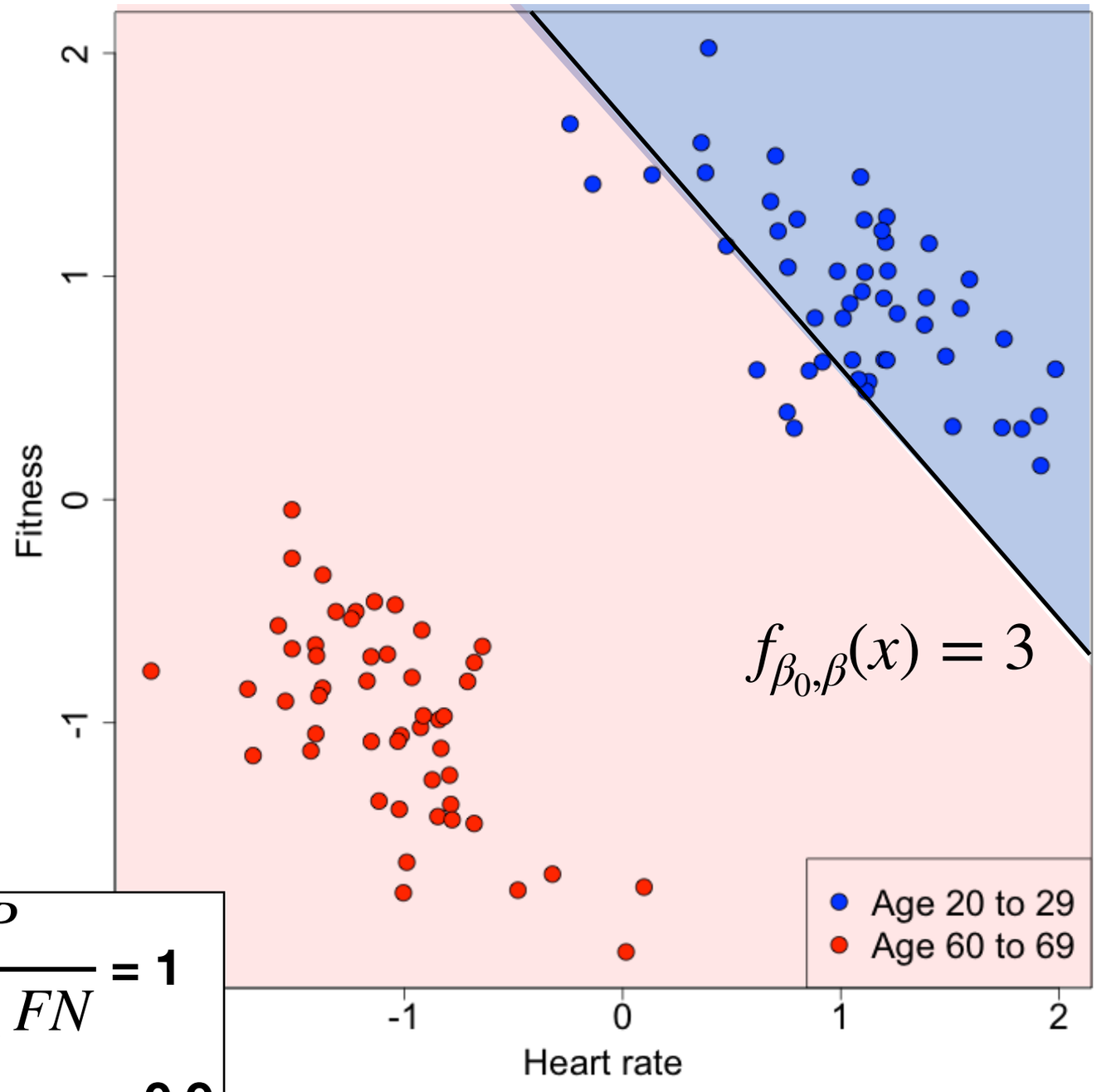
Thresholding on the decision function



$$\textbf{Sensitivity} = \frac{TP}{TP + FN} = 1$$

$$\textbf{Specificity} = \frac{TN}{TN + FP} = 1$$

Thresholding on the decision function

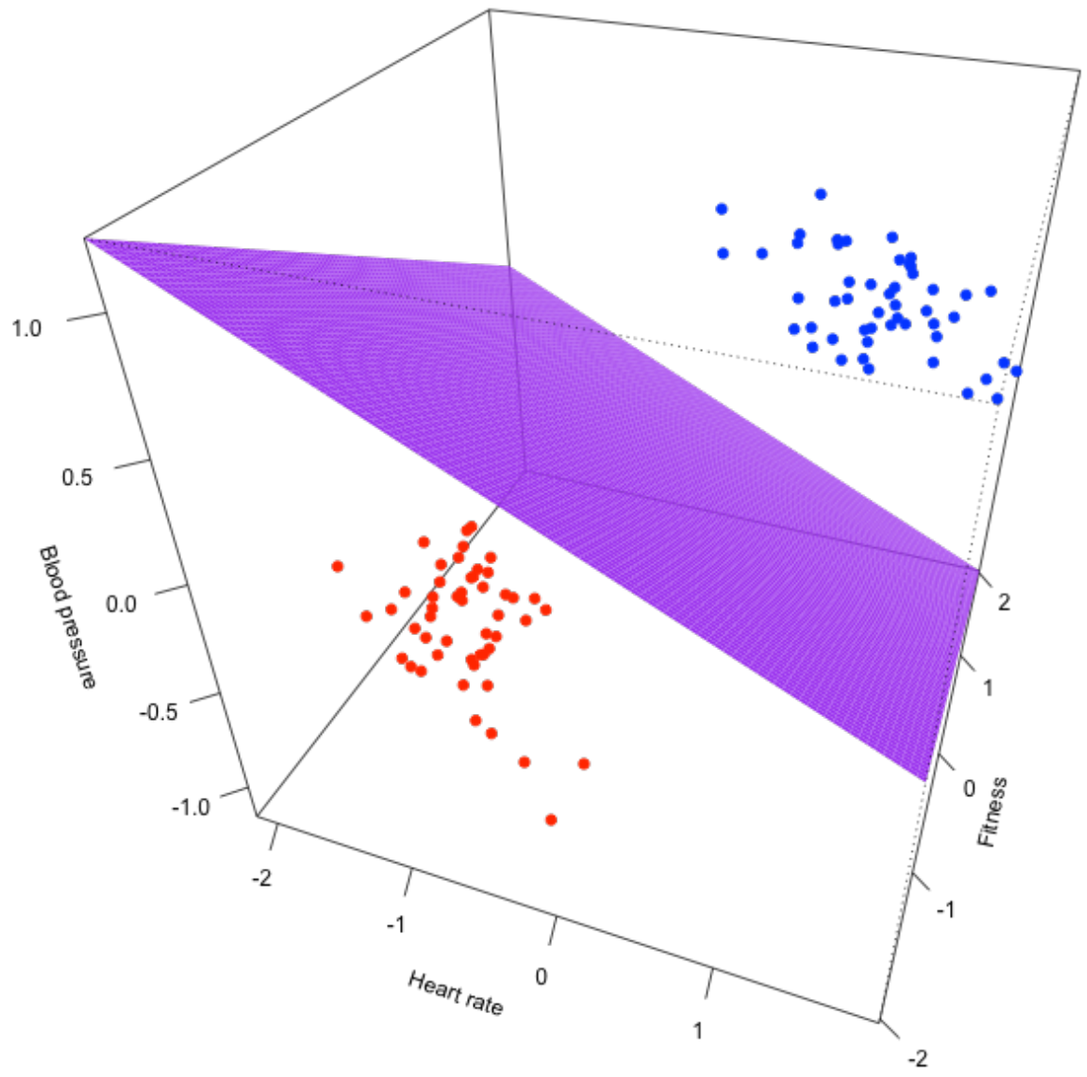


$$\textbf{Sensitivity} = \frac{TP}{TP + FN} = 1$$

$$\textbf{Specificity} = \frac{TN}{TN + FP} = 0.9$$

More than 2 predictors

- When there are p predictors where $p > 2$, we refer to the decision boundary as a ***hyperplane***. The hyperplane is a $(p - 1)$ dimensional plane.



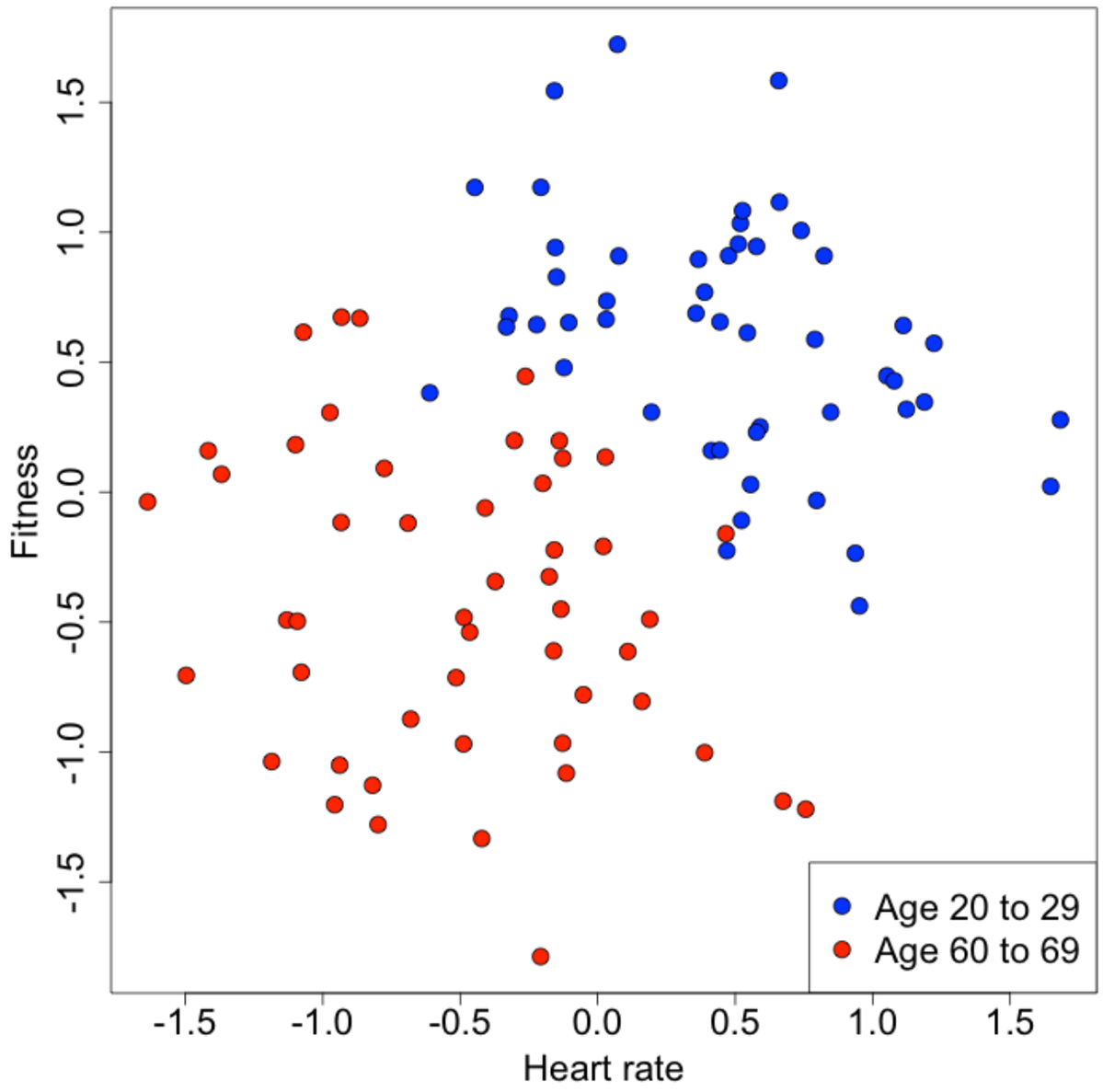
Today's lecture

1. Maximum margin classifiers
2. Support vector classifiers
3. Support vector machines
4. SVMs as a penalization method
5. SVMs for multiple classes

A more realistic dataset

- Let's simulate a more realistic dataset where the two age groups are harder to separate.

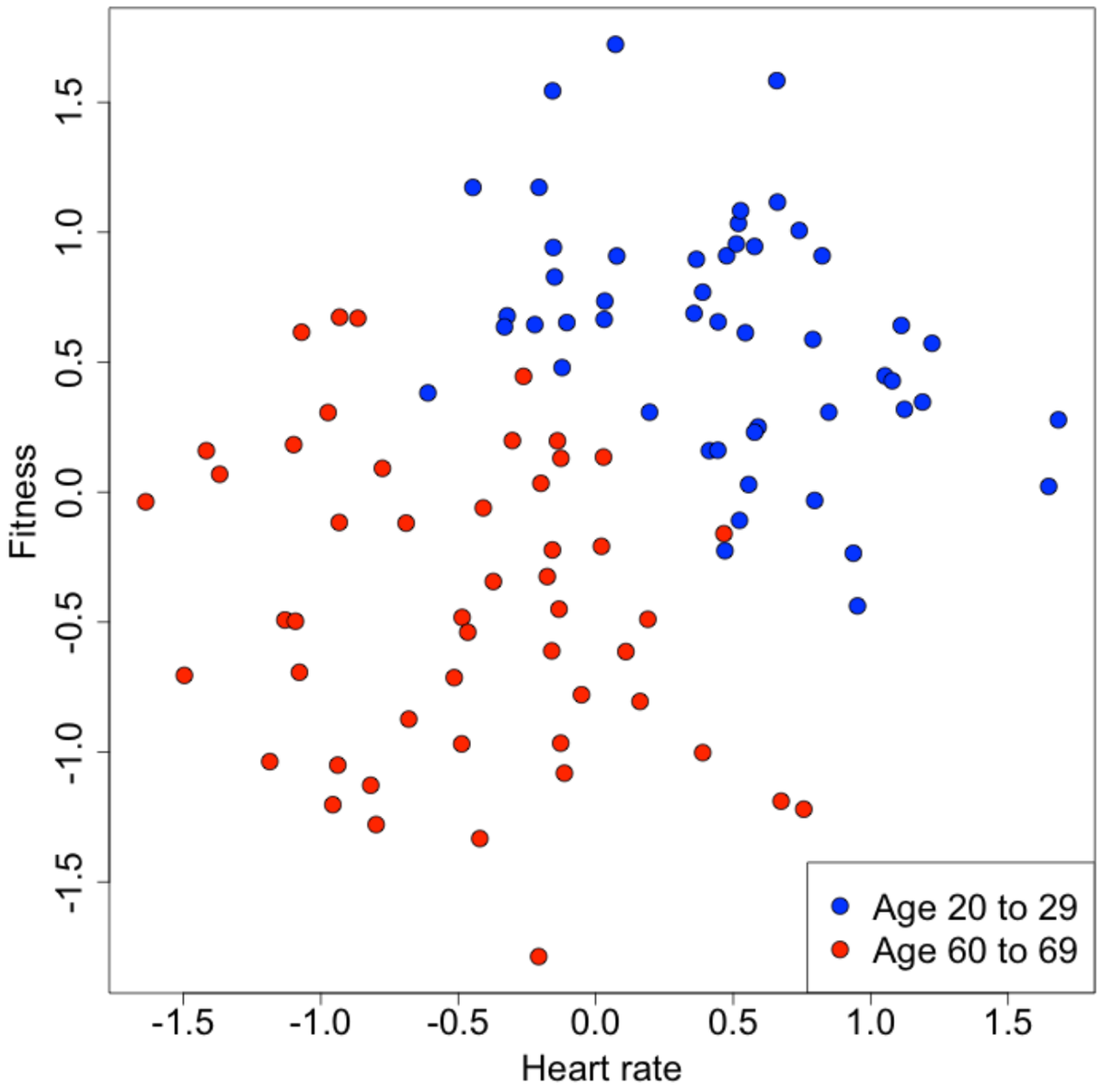
- What if there is no separating hyperplane?



What if there is no separating hyperplane?

- **Option 1**: Allow for misclassifications.
  - ***Support vector classifiers***

- **Option 2**: Draw a squiggly decision boundary between the two classes.
  - ***Support vector machines***



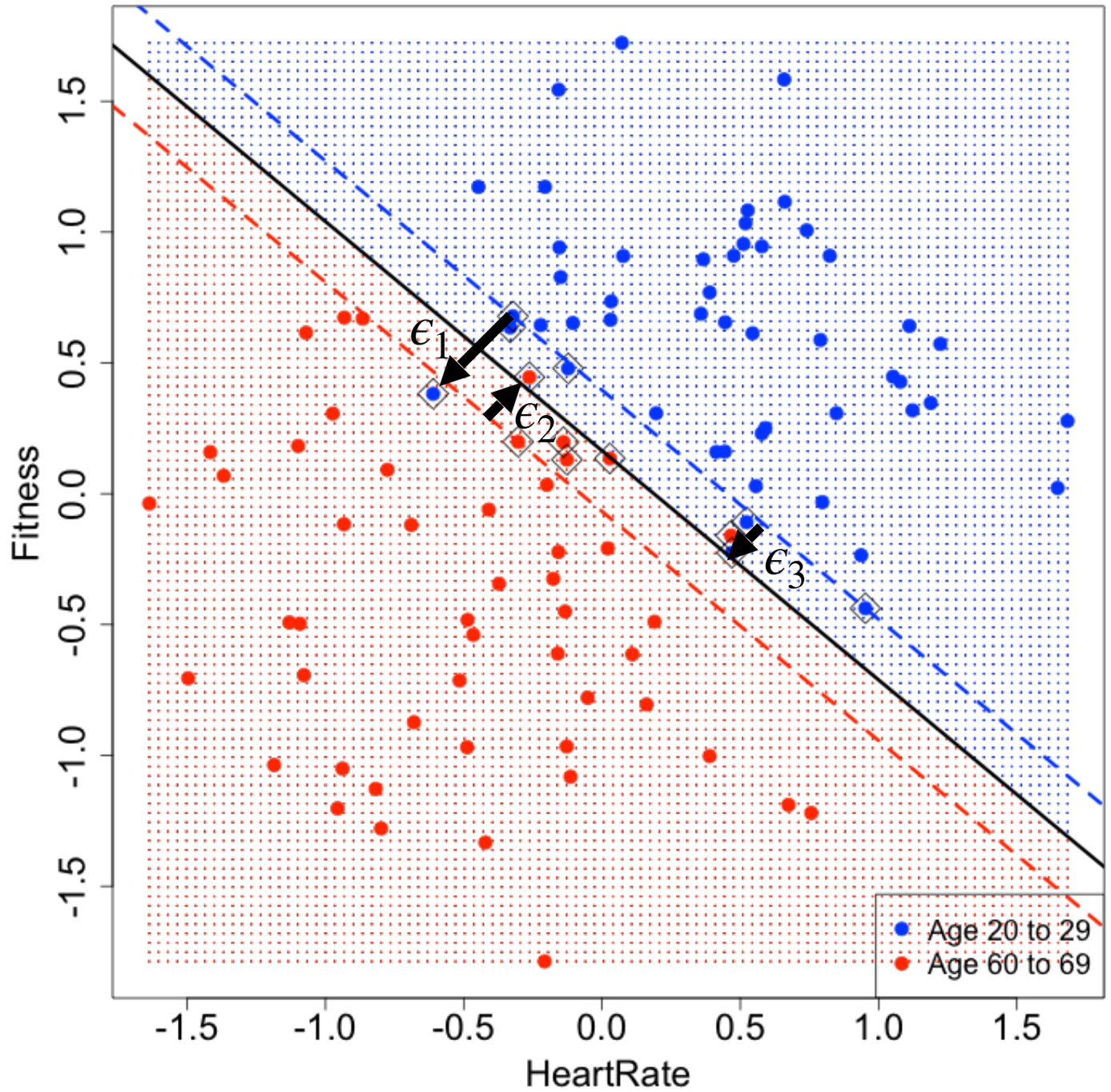
Support vector classifiers

(Also known as support vector machines with a linear kernel)

- Support vector classifiers allow for violations.

- The violation $\epsilon_i \geq 0$ for the i-th point measures how far the point is from the correct side of the margin boundary.

- ϵ_i is more commonly referred to as a ***slack variable***.



Support vector classifiers

For a training dataset $\{(x_i, y_i) : i = 1,...,n\}$, the support vector classifier finds a linear decision boundary that maximizes the margin M such that the **total allowed violation** is no more than some tuning parameter $C \geq 0$:

$$\max_{\beta_0,\beta,\epsilon,M} M$$ *(maximize the margin)*

such that $\|\beta\|_2^2 = 1$ and

$$\epsilon_i \geq 0 \text{ and } \sum_{i=1}^{n} \epsilon_i \leq C$$ *(constraint on the total violation)*

$$\underbrace{y_i(\beta_0 + \beta^\top x_i)}_{\text{Distance to the decision boundary}} \geq \underbrace{M(1 - \epsilon_i)}_{\text{A shrunken margin}} \text{ for all } i = 1,...,n$$

Maximum margin classifier

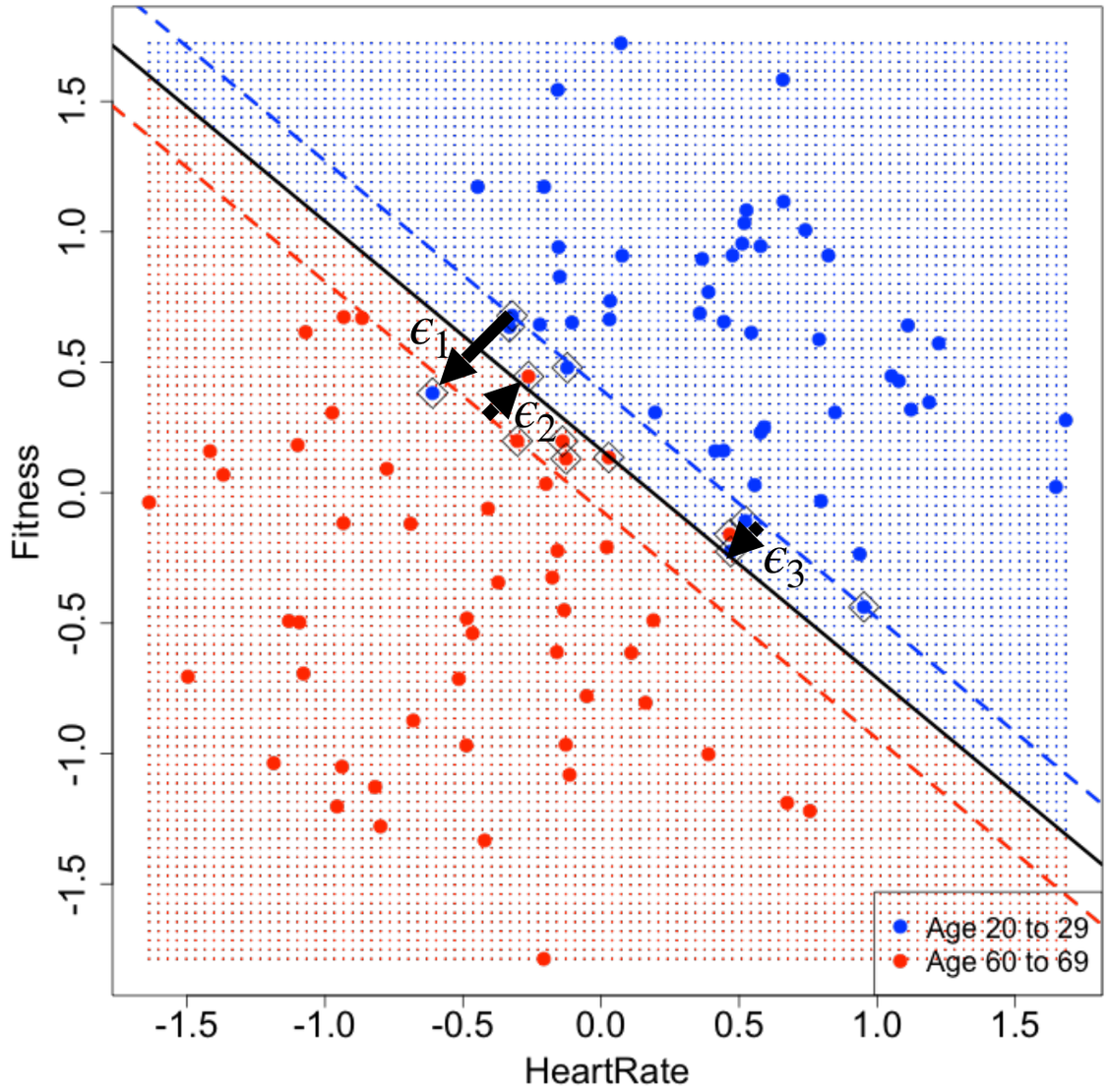
$$\max_{\beta_0,\beta,M} M$$

such that $\|\beta\|_2^2 = 1$ and

$$y_i(\beta_0 + \beta^\top x_i) \geq M \text{ for all } i$$

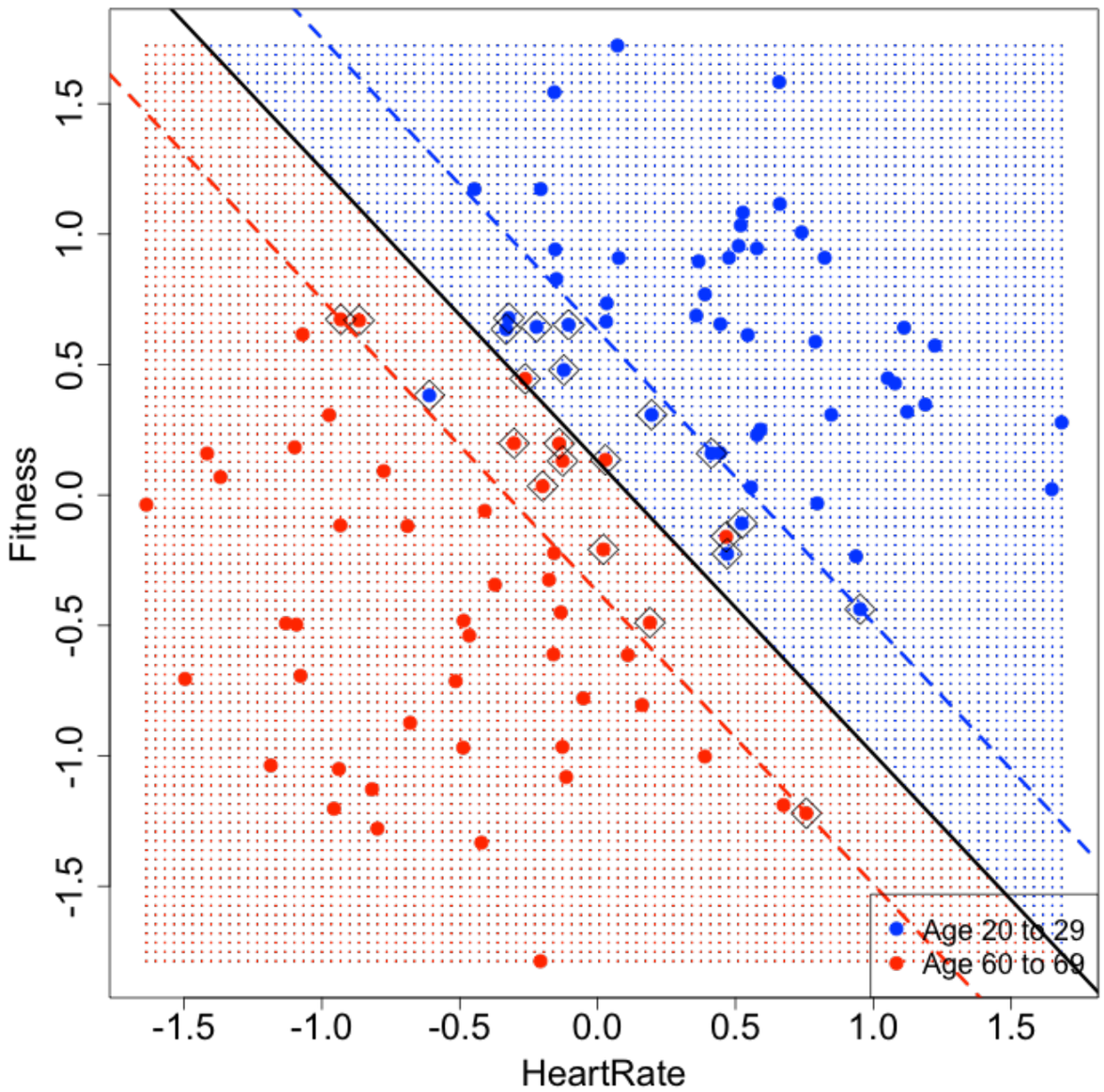
Support vector classifiers

- The value of slack variable ϵ_i indicates where the i-th observation is located:
  - $\epsilon_i = 0$ means the point is on the correct side of the margin boundary.
  - $0 < \epsilon_i \leq 1$ means the point is on the wrong side of the margin boundary but is on the correct side of the decision boundary.
  - $\epsilon_i > 1$ means the point is on the wrong side of the decision boundary.
- Support vectors = observations that determine the location/orientation of the decision boundaries, which includes points on the margin and points with $\epsilon_i > 0$

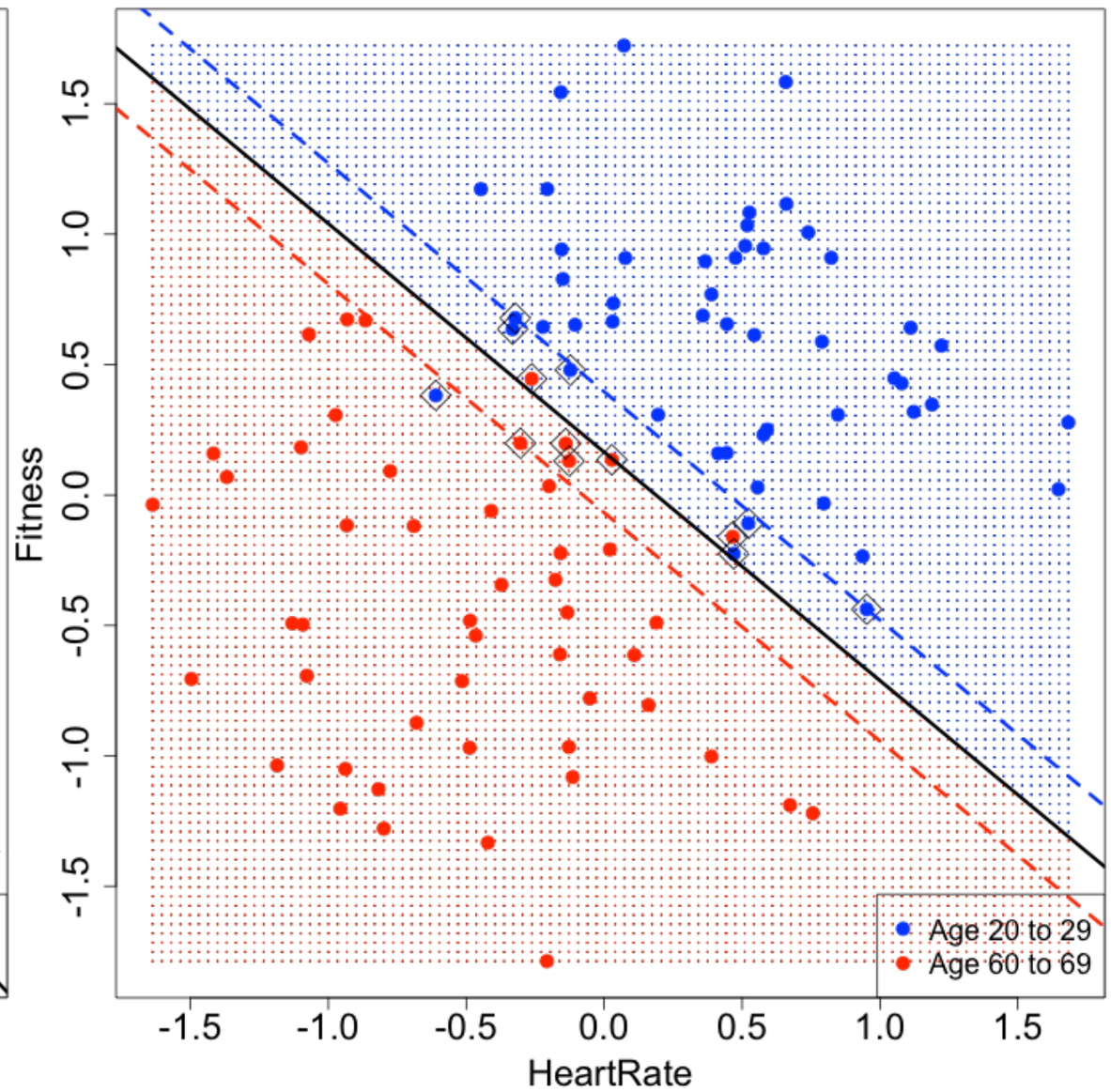


Different values of C

Allow for more violations (bigger C)

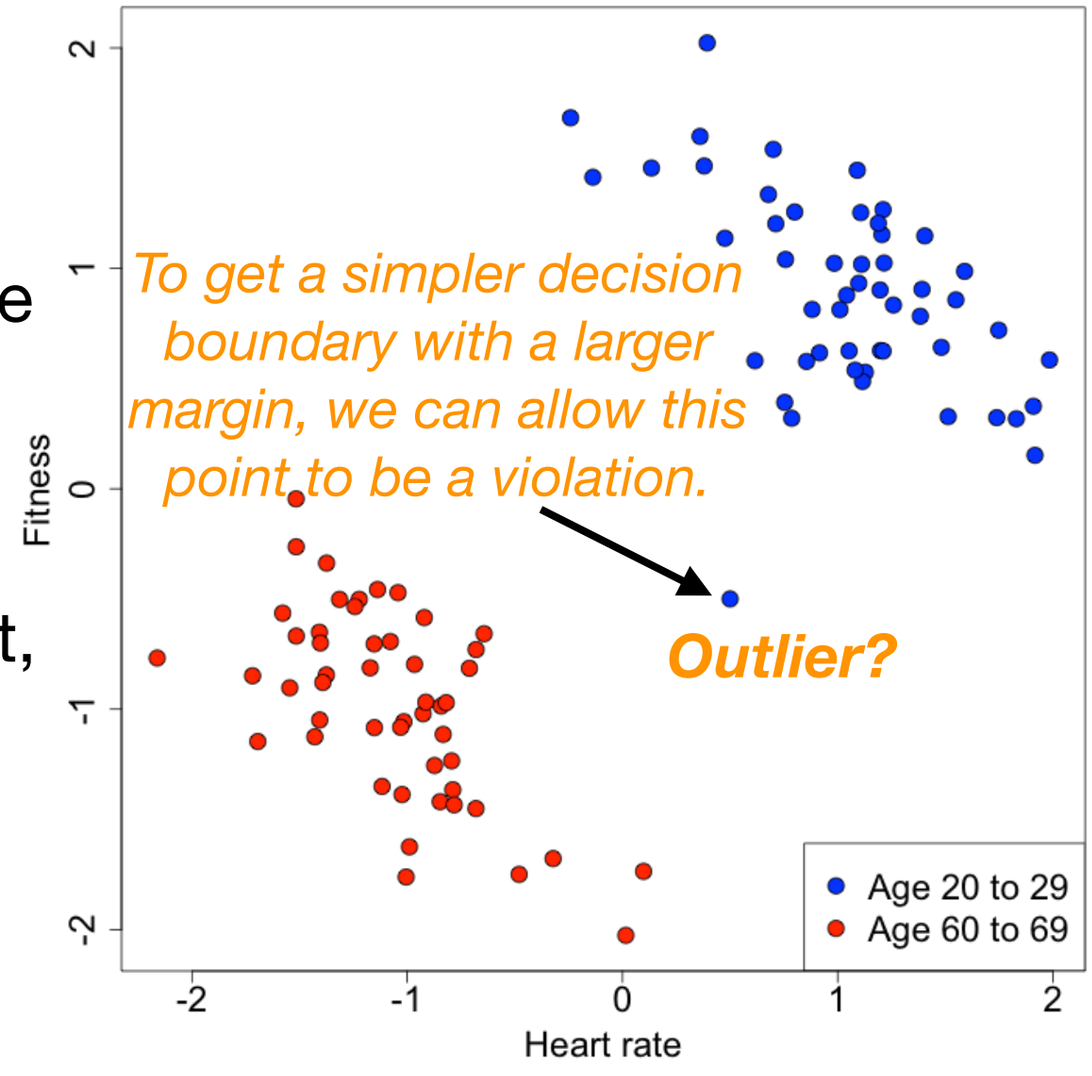


Allow for fewer violations (smaller C)



Support vector classifiers as a regularization method

- You can also view hyperparameter C as how much we want to regularize the decision boundary.
    - Bigger C means more regularization. As we allow for more violations, we can find "simpler" decision boundaries.
- Even for a perfectly separable dataset, you might still want to allow for violations (i.e. $C > 0$) because regularization can improve generalizability.



Fitting a support vector classifier in R

Same command, but now let's pay attention to the "cost" argument

```
> svmFit <- svm(ageGrp ~ Fitness + HeartRate, data=dat, kernel="linear", cost=10, scale=FALSE)
> summary(svmFit)

Call:
svm(formula = ageGrp ~ Fitness + HeartRate, data = dat, kernel = "linear",
    cost = 10, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10

Number of Support Vectors:  3

 ( 1 2 )


Number of Classes:  2

Levels:
 -1 1
```

- `cost` in `svm` is the cost of violating, rather than the total violation hyperparameter C that we've been discussing.

- So higher `cost` in `svm` corresponds to smaller total allowed violation.

Cross-validation for support vector classifiers in R

- You can tune the value of the `cost` parameter using the `caret` package.

Support Vector Machines with Linear Kernel

```
method = 'svmLinear2'
```

Type: Regression, Classification

Tuning parameters:

- `cost` (Cost)

Required packages: `e1071`

http://topepo.github.io/caret/train-models-by-tag.html#support-vector-machines

```
train_control <- trainControl(method="cv", number=5)
cv_train <- train(
  ageGrp ~ Fitness + HeartRate,
  data = dat,
  method = "svmLinear2",
  trControl = train_control,
  tuneGrid = expand.grid(cost = exp(seq(-5, 5, length = 10)))
)
Support Vector Machines with Linear Kernel

100 samples
  2 predictor
  2 classes: '-1', '1'


No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 80, 80, 80, 80, 80
Resampling results across tuning parameters:

  cost          Accuracy  Kappa
  6.737947e-03  0.93      0.86
  2.046808e-02  0.95      0.90
  6.217652e-02  0.95      0.90
  1.888756e-01  0.95      0.90
  5.737534e-01  0.96      0.92
  1.742909e+00  0.96      0.92
  5.294490e+00  0.96      0.92
  1.608324e+01  0.95      0.90
  4.885657e+01  0.93      0.86
  1.484132e+02  0.93      0.86

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cost = 0.5737534.
```

Today's lecture

1. Maximum margin classifiers

2. Support vector classifiers

3. Support vector machines

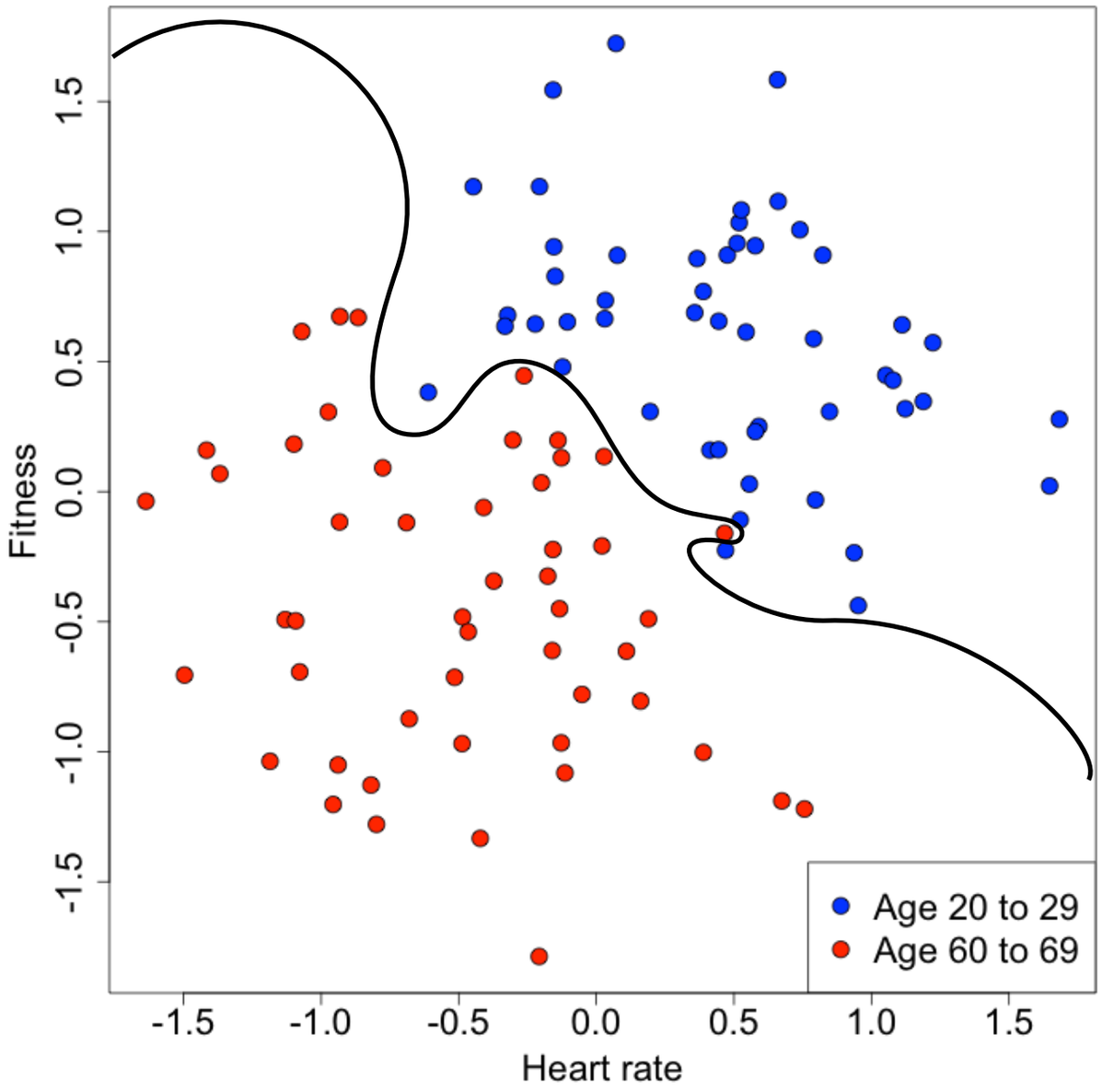
4. SVMs as a penalization method

5. SVMs for multiple classes

What if there is no separating hyperplane?

- **Option 1**: Allow for misclassifications.
  - ***Support vector classifiers***

- **Option 2**: Draw a squiggly decision boundary between the two classes.
  - ***Support vector machines***

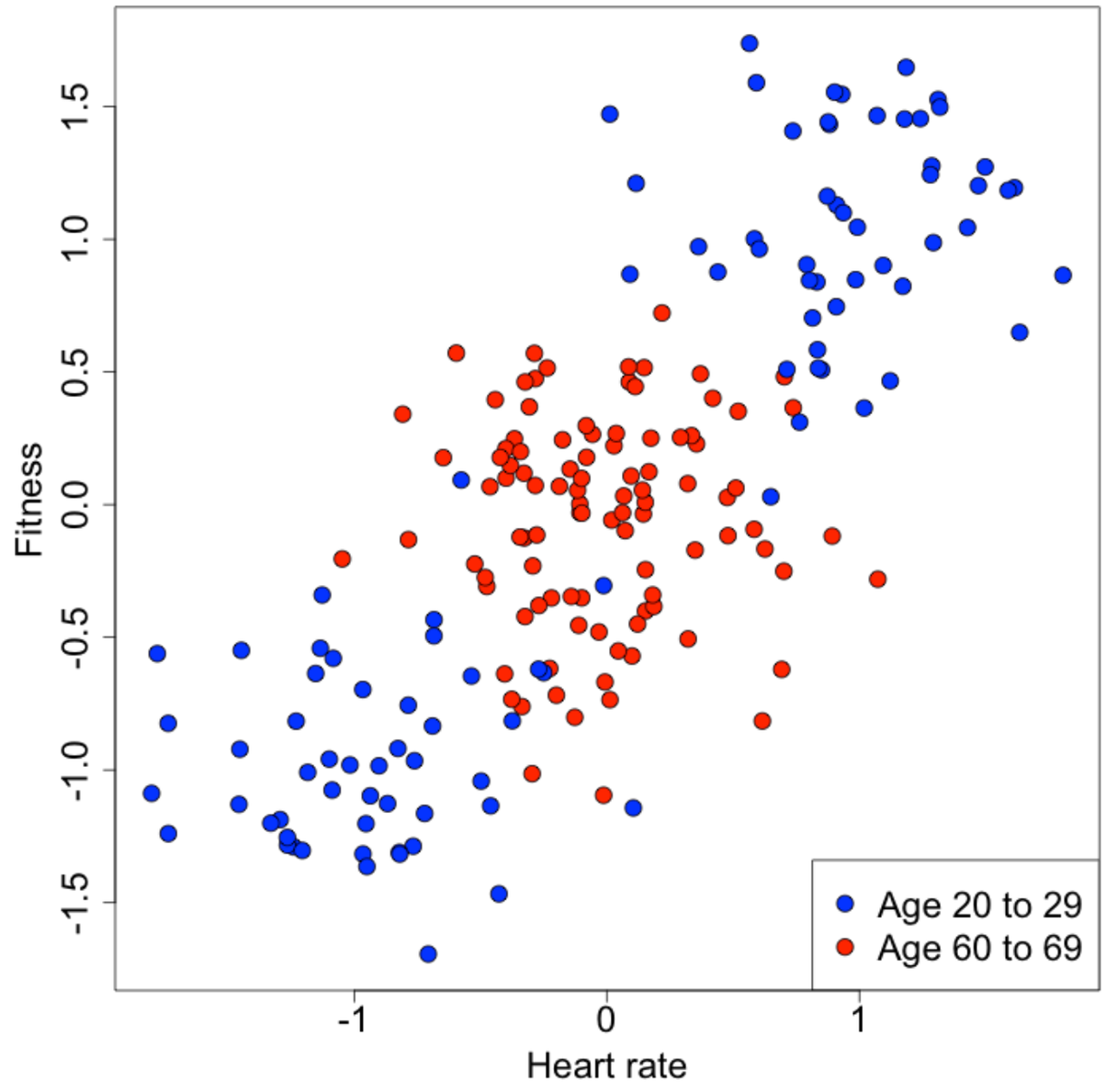


Warning: This decision boundary is probably not appropriate for this dataset

Example where a nonlinear decision boundary can help

- **Option 1**: Allow for misclassifications.
  - ***Support vector classifiers***

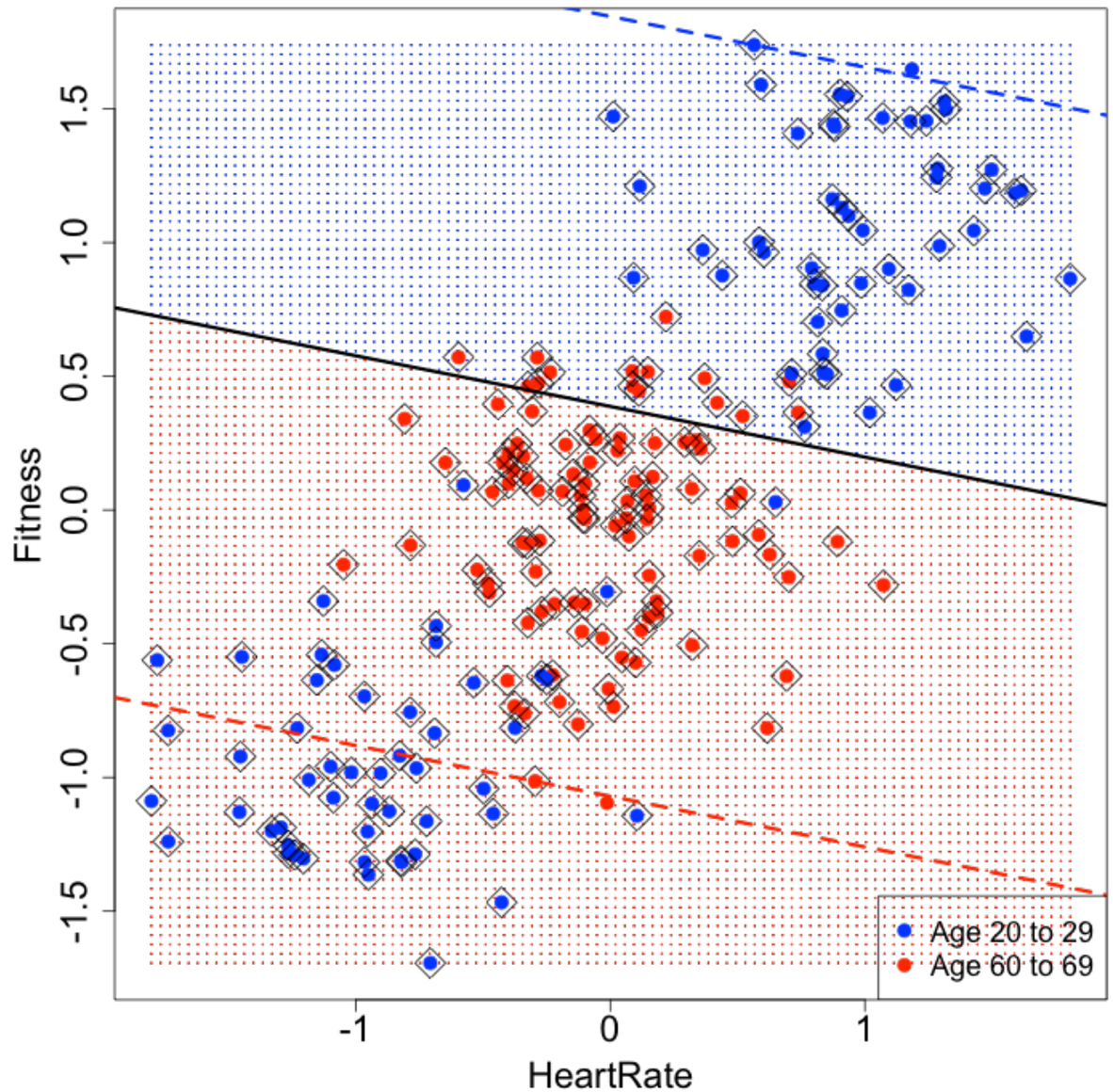
- **Option 2**: Draw a squiggly decision boundary between the two classes.
  - ***Support vector machines***



Example where a nonlinear decision boundary can help

- **Option 1**: Allow for misclassifications.
  - ***Support vector classifiers***

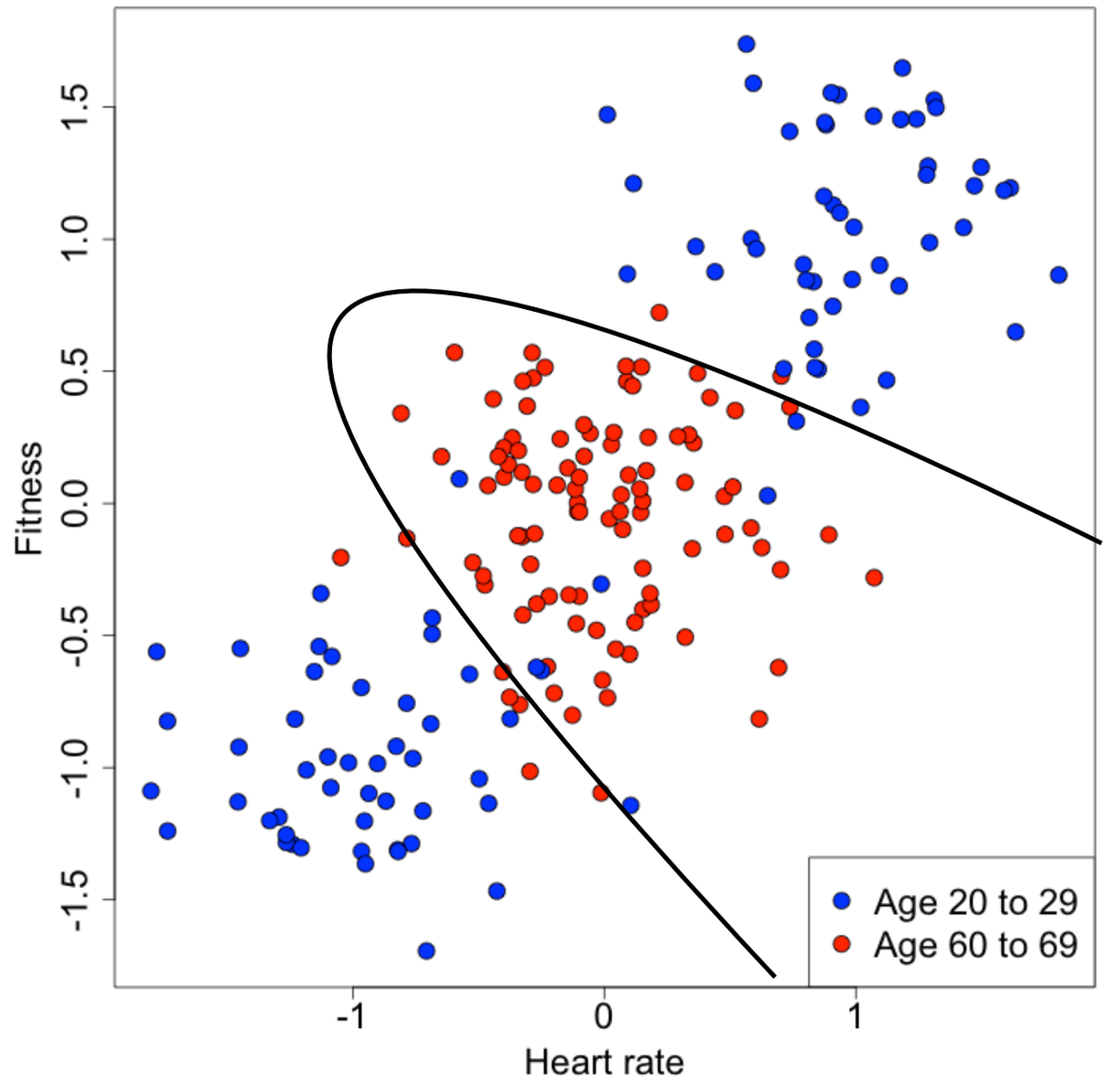
- **Option 2**: Draw a squiggly decision boundary between the two classes.
  - ***Support vector machines***



Example where a nonlinear decision boundary can help

- **Option 1**: Allow for misclassifications.
  - ***Support vector classifiers***

- **Option 2**: Draw a squiggly decision boundary between the two classes.
  - ***Support vector machines***



Suppose we had two variables (X_1, X_2). Which of these methods fits a quadratic decision boundary?

- **A**: Fit a support vector classifier on an augmented set of variables: $(X_1, X_2, X_1X_2, X_1^2, X_2^2)$

  This fits a quadratic decision boundary and accounts for interactions

- **B**: Fit a support vector classifier on an augmented set of variables: (X_1, X_2, X_1^2, X_2^2)

  This fits a quadratic decision boundary, but doesn't allow for interactions

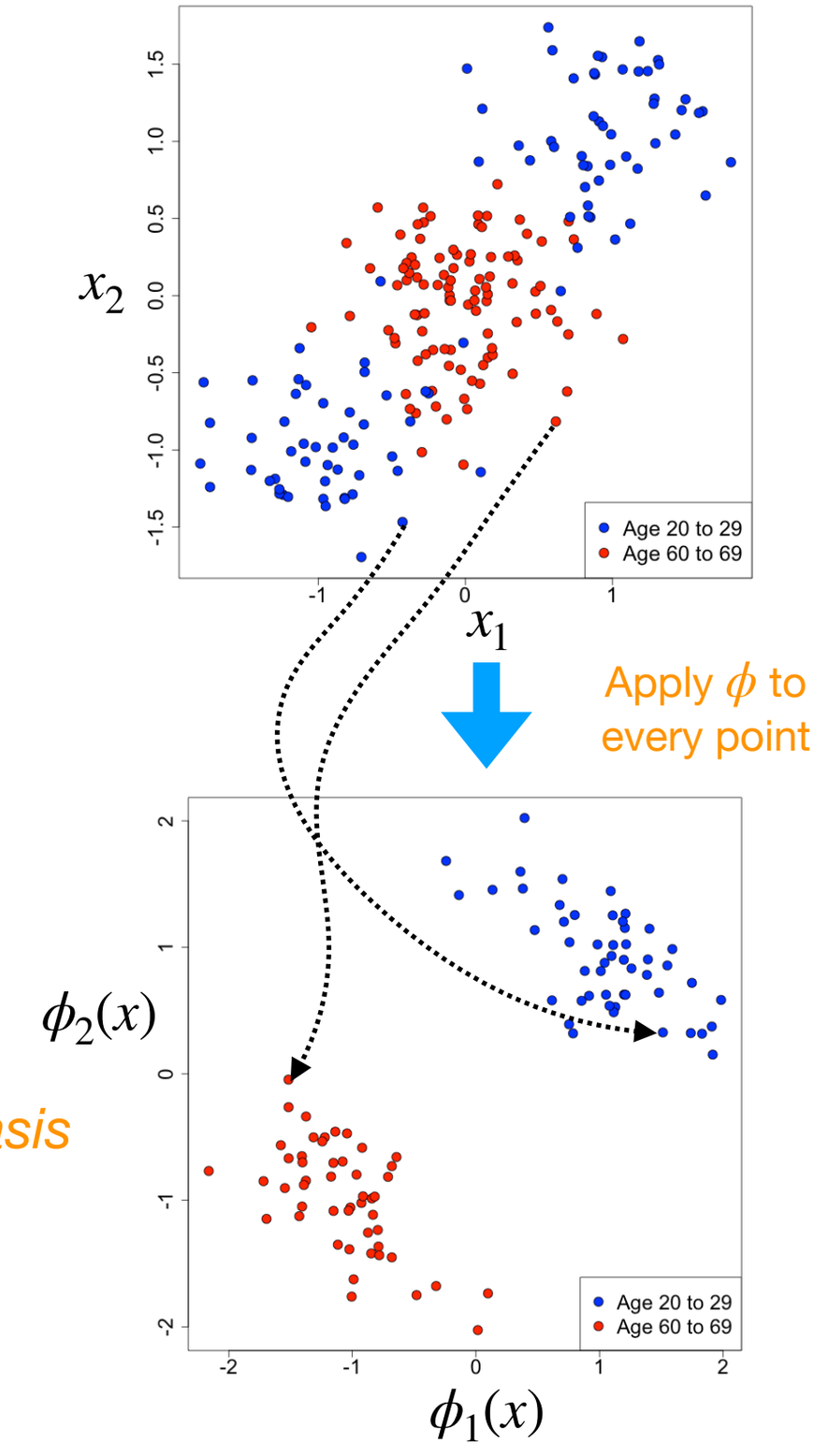
- **C**: Fit a support vector classifier for the variables: (X_1, X_2)

  This fits a linear decision boundary.

Recall: Polynomial regression

- Remember that we can fit non-linear models using linear regression by expanding terms. For example:
  - $(x_1) \rightarrow (x_1, x_1^2, x_1^3)$
  - $(x_1, x_2) \rightarrow (x_1, x_1^2, x_2, x_2^2, x_1 x_2)$
- More generally, we can perform a **basis expansion**. That is, we create a new feature set using a sequence of basis functions $\{\phi_1, \cdots, \phi_k\}$ as follows:
  - $x \rightarrow (\phi_1(x), \phi_2(x), \cdots, \phi_k(x))$

Polynomial basis
Spline basis
Radial basis
…



Fitting a non-linear decision boundary

- We can fit a non-linear decision boundary by applying a basis expansion and before fitting a support vector classifier.

- The decision function will have the form

$$f_{\beta_0,\beta}(x) = \beta_0 + \sum_{j=1}^{k} \beta_j \phi_j(x).$$

- However, this procedure can be very computationally expensive if k is very big and impossible to run if we want to consider $k = \infty$.

Support vector machines

- Support vector machines extend support vector classifiers to fit non-linear decision boundaries.

- SVMs use the "***Kernel Trick***" to efficiently estimate a nonlinear decision boundary when k is very large.

The Kernel Trick

It turns out that…

<u>Support vector classifier</u>

$$\max_{\beta_0,\beta,M,\epsilon} M$$

such that $\|\beta\|_2^2 = 1, \epsilon_i \geq 0\,, \sum_{i=1}^{n} \epsilon_i \leq C$

$y_i(\beta_0 + \beta^\top x_i) \geq M(1-\epsilon_i)$ for all i

must have a solution of the form:

$$f_{\beta_0,\alpha}(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

where $\langle x, z \rangle = \sum_{j=1}^{p} x_j z_j$.

Inner product with the i-th training observation

<u>Support vector machine</u>

$$\max_{\beta_0,\beta,M,\epsilon} M$$

such that $\|\beta\|_2^2 = 1, \epsilon_i \geq 0\,, \sum_{i=1}^{n} \epsilon_i \leq C$

$y_i(\beta_0 + \beta^\top \phi(x_i)) \geq M(1-\epsilon_i)$ for all i

must have a solution of the form:

$$f_{\beta_0,\alpha}(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle \phi(x), \phi(x_i) \rangle.$$

Inner product with the i-th (transformed) training observation

The Kernel Trick

Support vector machine

$$\max_{\beta_0,\beta,M,\epsilon} M$$

such that $\|\beta\|_2^2 = 1, \epsilon_i \geq 0\ , \sum_{i=1}^{n} \epsilon_i \leq C$

$y_i(\beta_0 + \beta^\top \phi(x_i)) \geq M(1 - \epsilon_i)$ for all i

must have a solution of the form:

$$f_{\beta_0,\alpha}(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle \phi(x), \phi(x_i) \rangle.$$

Inner product with the i-th (transformed) training observation

- To solve this optimization problem, all we need to compute are the inner products. We don't need to actually compute $\phi(x)$.
- For special sequences of basis functions ϕ, the ***inner product*** can be computed very efficiently using a so-called "***Kernel function***": $K(x, x') = \langle \phi(x), \phi(x') \rangle$
- d-order Polynomial Kernel:

$$K(x, x') = \left(r + \sum_{j=1}^{p} x_i x'_j\right)^d$$

- Radial Basis Kernel:

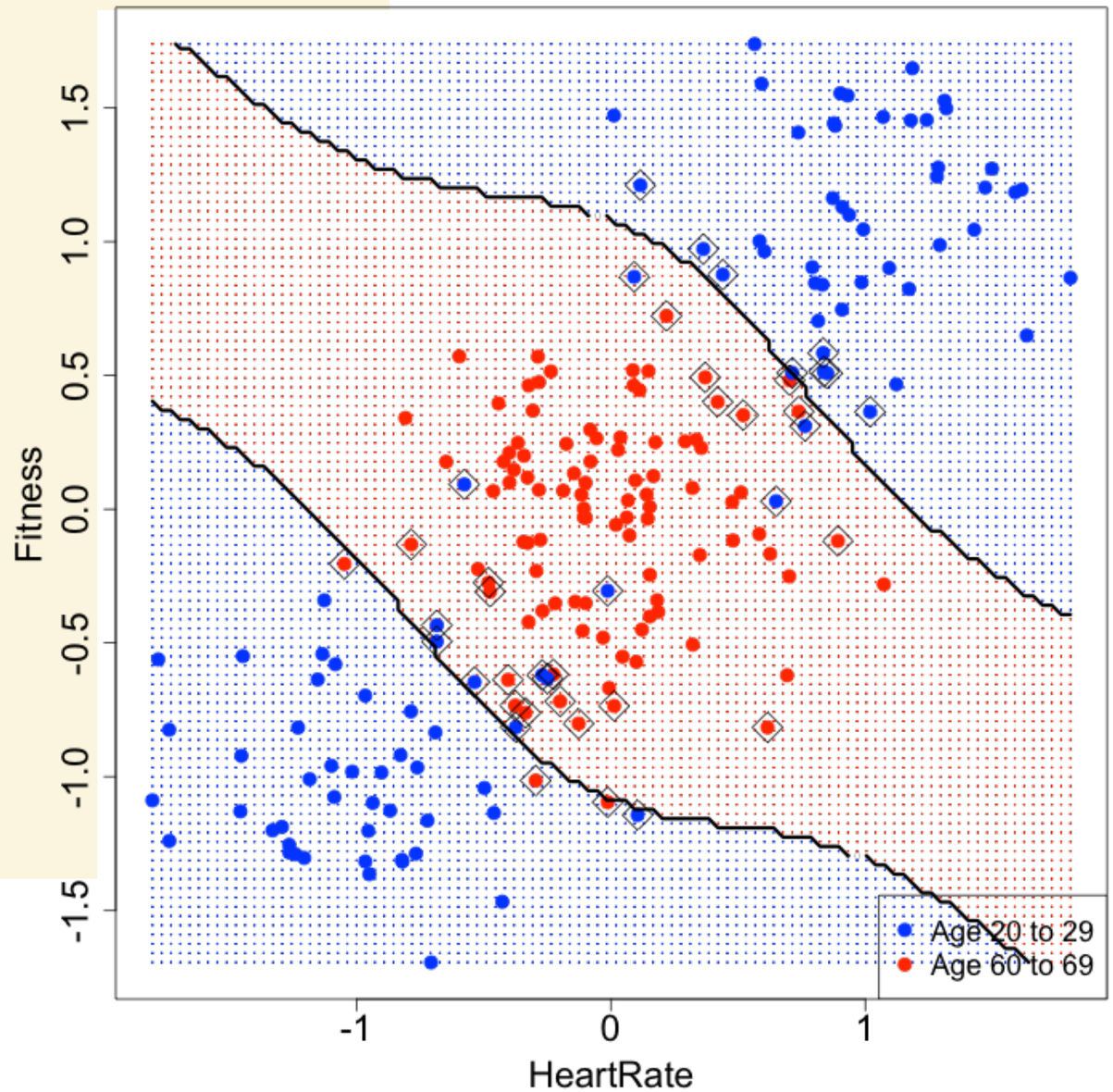
$$K(x, x') = \exp\left(-\gamma \left\| x - x' \right\|_2^2\right)$$

(In fact, $k = \infty$ for the radial basis kernel!)

Example: polynomial kernel

- Fit an SVM using a polynomial kernel with $d = 4$ and cost = 10.

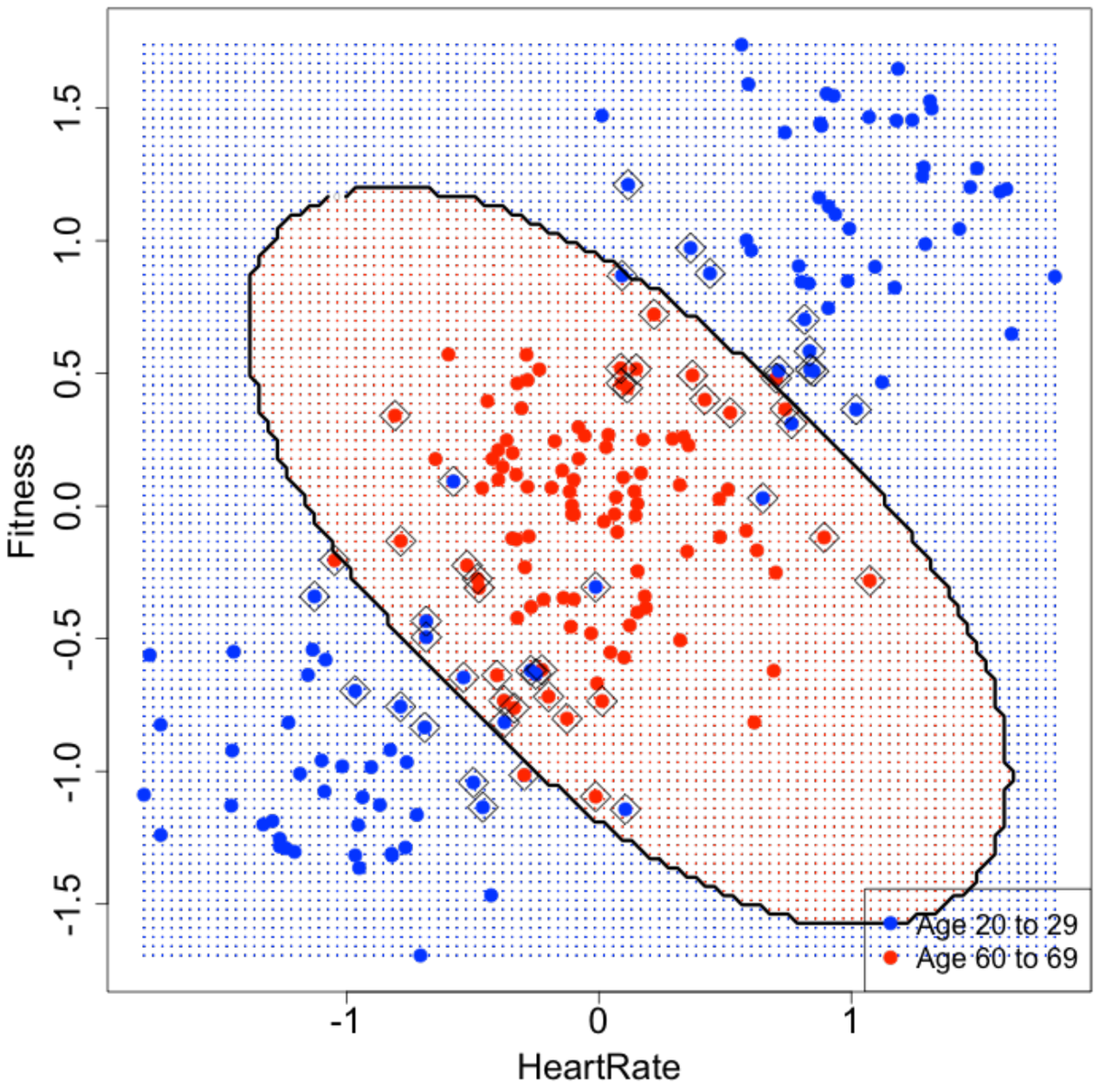
```
> svmFitSVM <- svm(ageGrp ~ Fitness + HeartRate, data=dat, kernel="polynomial",
 degree=4, cost=10, scale=FALSE)
> summary(svmFitSVM)

Call:
svm(formula = ageGrp ~ Fitness + HeartRate, data = dat, kernel = "polynomial",
    degree = 4, cost = 10, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  10
     degree:  4
     coef.0:  0

Number of Support Vectors:  41

 ( 21 20 )


Number of Classes:  2

Levels:
 -1 1
```



Example: Radial basis kernel

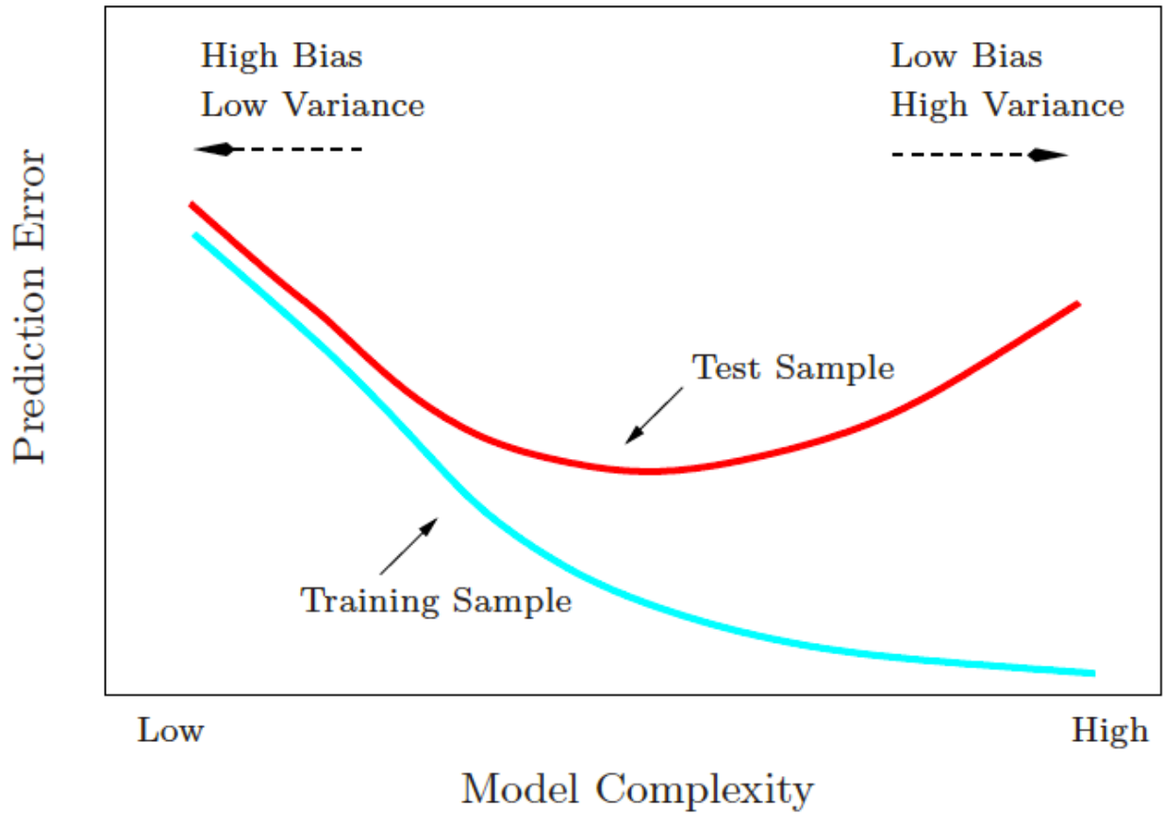
- Fit an SVM using a radial basis kernel with $\gamma = 0.1$ and cost=10.

```
> svmFit_radial <- svm(ageGrp ~ Fitness + HeartRate, data=dat, kernel="radial",
  gamma=0.15, cost=10, scale=FALSE)
```



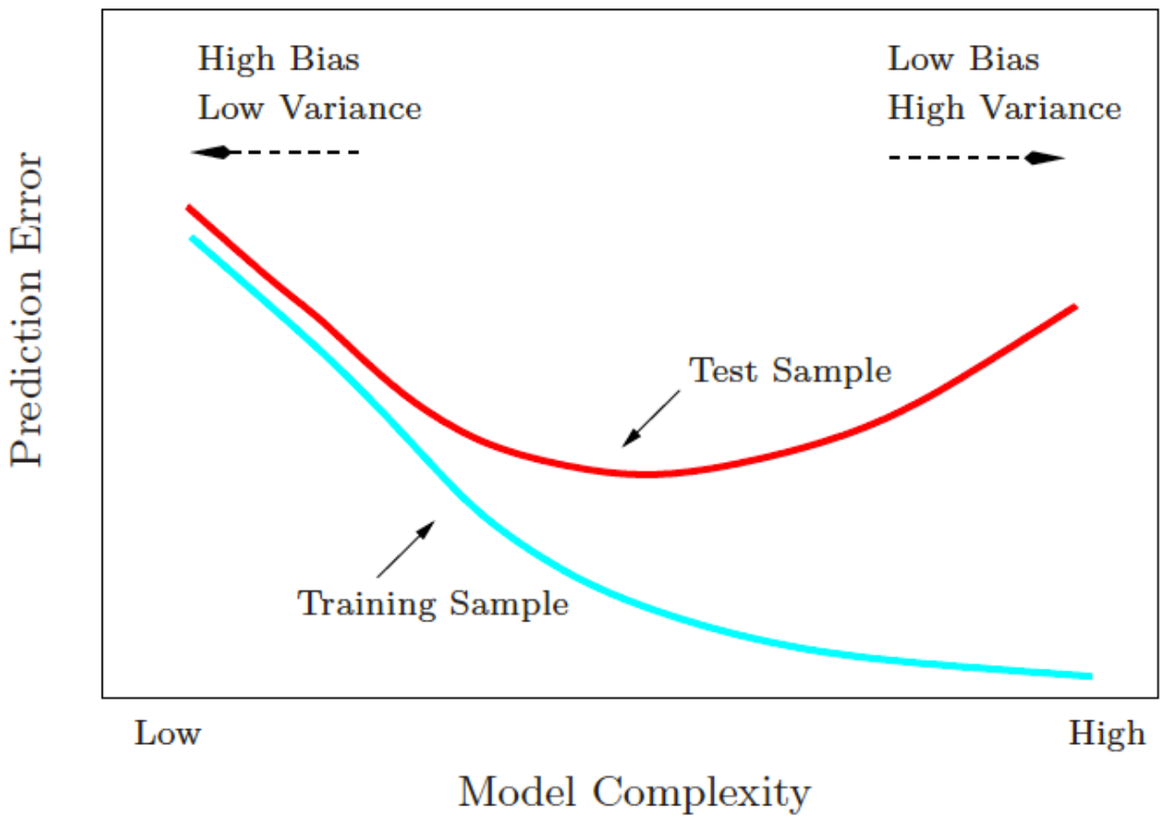
$$K(x, x') = \exp\left(-\gamma \parallel x - x' \parallel_2^2\right)$$

Q: In terms of the bias-variance tradeoff, how would you rank an SVM with a linear kernel versus an SVM with a non-linear kernel?



- **A**: SVM with a linear kernel has lower model complexity.
- **B**: SVM with a non-linear kernel has lower model complexity.

Q: In terms of the bias-variance tradeoff, how would you rank an SVM with a linear kernel versus an SVM with a non-linear kernel?



- **A**: SVM with a linear kernel has lower model complexity.

- **B**: SVM with a non-linear kernel has lower model complexity.

Estimating a non-linear decision boundary accurately generally requires more observations and is particularly difficult in high-dimensional settings where $p \gg n$.

How should I choose between using a linear versus non-linear kernel?

- **A**: Hard to tell. Use cross-validation.
    - Good choice!
- **B**: If the true decision boundary is non-linear, use a non-linear kernel.
    - A non-linear kernel is most effective when 1) *you have lots of observations relative to the number of features* and 2) the boundary is truly non-linear.
- **C**: If the number of observations is small, it will be impossible to accurately estimate a non-linear decision boundary.
    - Estimating a non-linear decision boundary accurately generally requires more observations and is particularly difficult in high-dimensional settings where $p \gg n$.

Today's lecture

1. Maximum margin classifiers

2. Support vector classifiers

3. Support vector machines

4. SVMs as a penalization method

5. SVMs for multiple classes

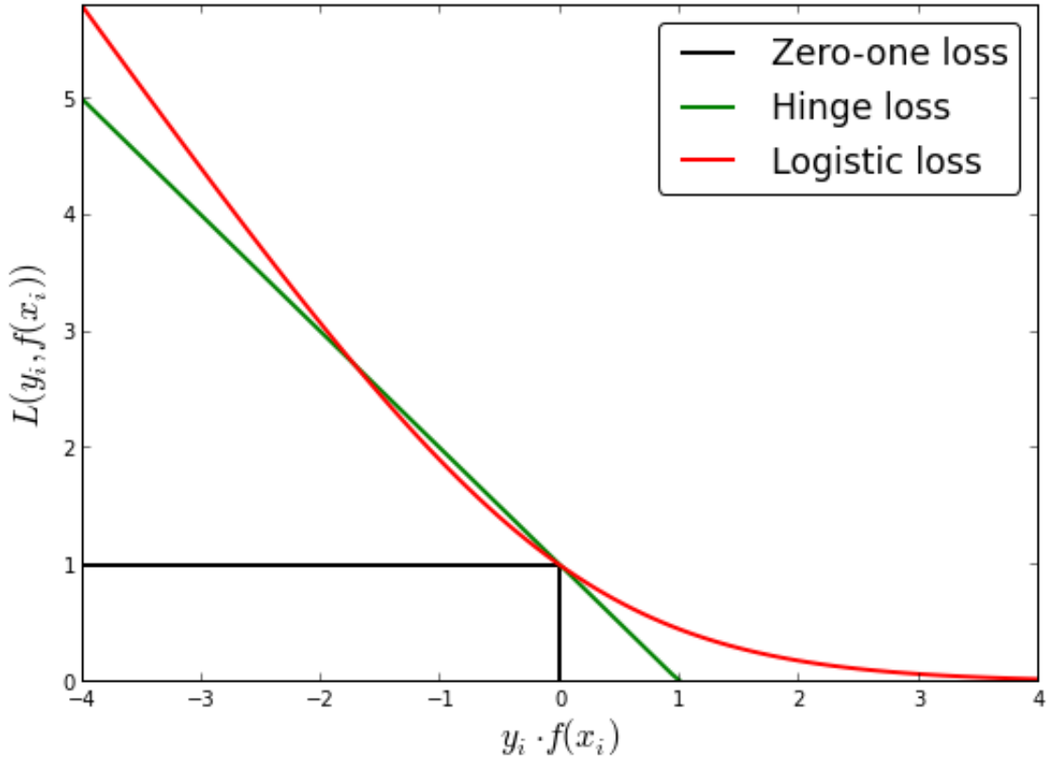
SVMs as a penalized regression method

Support vector classifier (SVC)

$$\max_{\beta_0,\beta,M,\epsilon} M$$

such that $\|\beta\|_2^2 = 1, \epsilon_i \geq 0\,, \sum_{i=1}^{n} \epsilon_i \leq C$

$y_i(\beta_0 + \beta^\top x_i) \geq M(1-\epsilon_i)$ for all i



The same thing (SVC)

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f_\beta(x_i)\right) + \lambda\|\beta\|^2$$

with hinge loss:

$\ell(y, f(x)) = \max(0, 1 - f(x)y)$

Logistic regression with a ridge penalty

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f_\beta(x_i)\right) + \lambda\|\beta\|^2$$

with logistic loss

$\ell(y, f(x)) = \log(1 + \exp(-f(x)y))$

Solving the more general penalized regression problem…

Support vector classifier

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f_\beta(x_i)\right) + \lambda \|\beta\|^2$$

with hinge loss:

$$\ell(y, f(x)) = \max(0, 1 - f(x)y)$$

Logistic regression with a ridge penalty

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f_\beta(x_i)\right) + \lambda \|\beta\|^2$$

with logistic loss

$$\ell(y, f(x)) = \log(1 + \exp(-f(x)y))$$

Introduce non-linearities using basis functions $\{\phi_1, \ldots, \phi_k\}$

Support vector ***machine***

$$\min_{\beta_0, \alpha} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, \beta_0 + \sum_{j=1}^{k} \alpha_j \phi_j(x_i)\right) + \lambda \|\alpha\|^2$$

with hinge loss:

$$\ell(y, f(x)) = \max(0, 1 - f(x)y)$$

Logistic regression with a ridge penalty

$$\min_{\beta_0, \alpha} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, \beta_0 + \sum_{j=1}^{k} \alpha_j \phi_j(x_i)\right) + \lambda \|\alpha\|^2$$

with logistic loss

$$\ell(y, f(x)) = \log(1 + \exp(-f(x)y))$$

We can use the ***kernel trick*** *with* $K(x, x') = \langle \phi(x), \phi(x') \rangle$ *to solve* ***both*** *of these optimization problems efficiently*

Is the Kernel Trick unique to SVMs?

- **No**. Both SVMs and logistic regression can be written as minimizers of a penalized empirical loss. Both can be combined with the Kernel Trick to fit non-linear decision boundaries.

- For historical reasons, non-linear kernels are much more common in the context of SVMs.

In high dimensions…

- In SVMs, the tuning parameter C controls the amount of flexibility of the classifier.

- Because SVMs can be written as ridge regression but with a hinge loss, tuning C is similar to tuning the penalty parameter in ridge regression. The SVM decision rule involves all p variables.

- You can fit a sparse SVM by replacing the ridge penalty with a lasso penalty; this yields a decision rule involving only a subset of the features.

- People used to claim that SVMs overcome the "curse of dimensionality". This is not true!

Today's lecture

1. Maximum margin classifiers

2. Support vector classifiers

3. Support vector machines

4. SVMs as a penalization method

5. SVMs for multiple classes

Multiple classes

- The separating hyperplane idea in SVMs does not lend itself naturally to more than 2 classes. Still, there are some extensions that are quite popular:

  - **One-versus-One** (OVO): Construct an SVM for every pair of classes. For a test observation, predict the class that won the most pairwise competitions.

  - **One-versus-All** (OVA): Fit K SVMs, where each SVM predicts a class vs. the rest. For a test observation, predict the class with the largest decision function.

Summary

- Pros:
    - SVMs are very good off-the-shelf binary classifiers. They have won many machine learning competitions.
    - SVMs are computationally efficient.
- Cons:
    - When non-linear decision boundaries are used, they are not easily interpretable.
    - It doesn't output probabilities for each class.

Today's lecture

1. Maximum margin classifiers

2. Support vector classifiers

3. Support vector machines

4. SVMs as a penalization method

5. SVMs for multiple classes