Ensemble Methods

Ludwig Krippahl

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

Summary

- Ensemble methods
- Bagging and bragging
- Boosting and stumping

Ensemble Methods

Ensemble methods

- Combining groups of classifiers to improve classification

We'll focus on two different aproaches:

- Bootstrap aggregating : bootstrapping to train, combine predictions to reduce variance
- Boosting : training a linear combination of weak classifiers (mainly) to reduce bias

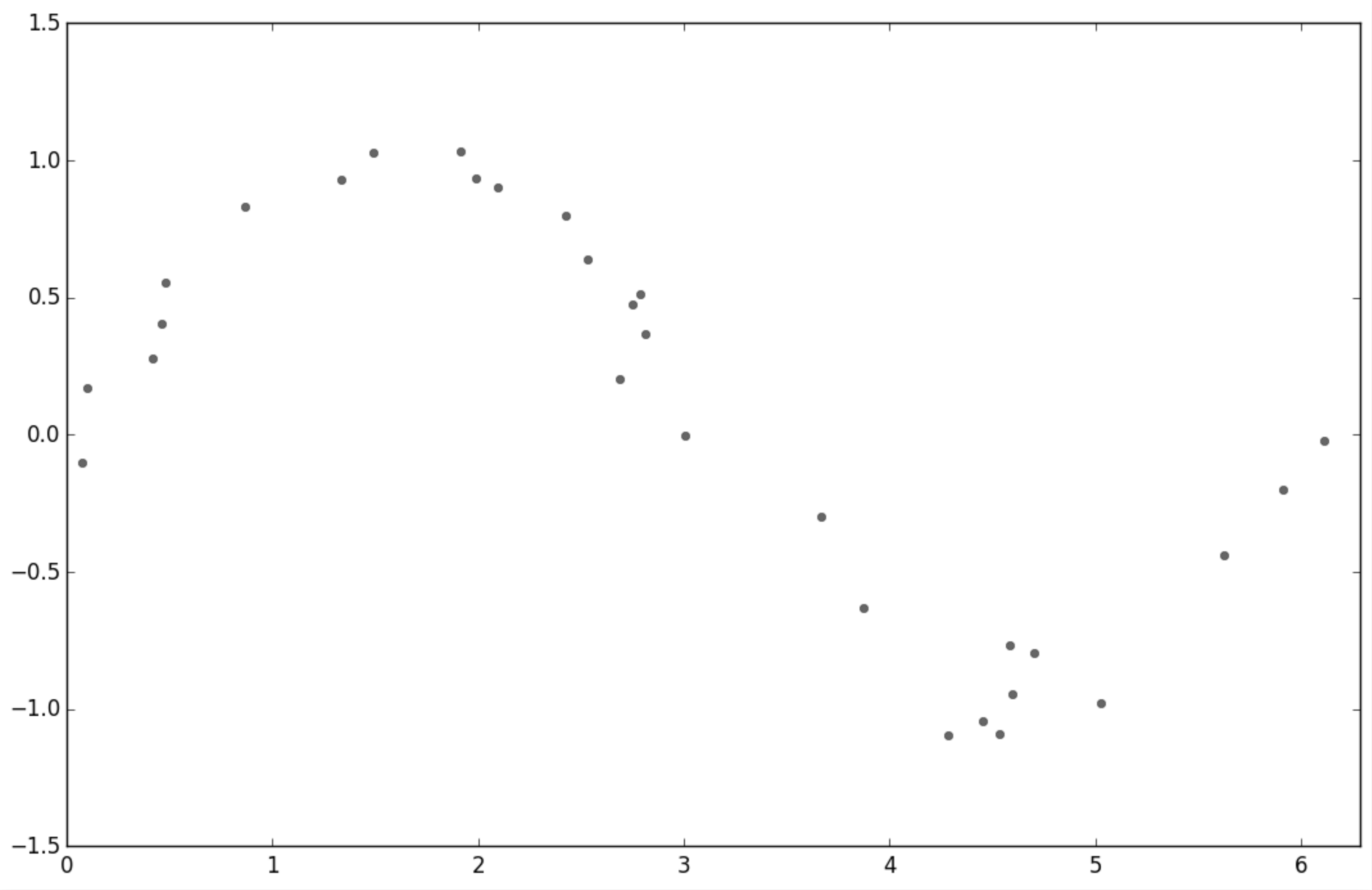
Bagging

- Bootstrap aggregating

- Use bootstrapping to generate replicas of training set
- Train model once per replica
- Aggregate the output of the hypotheses. Example: for regression, average the predictions

Ensemble methods

- Example: regression



Ensemble methods

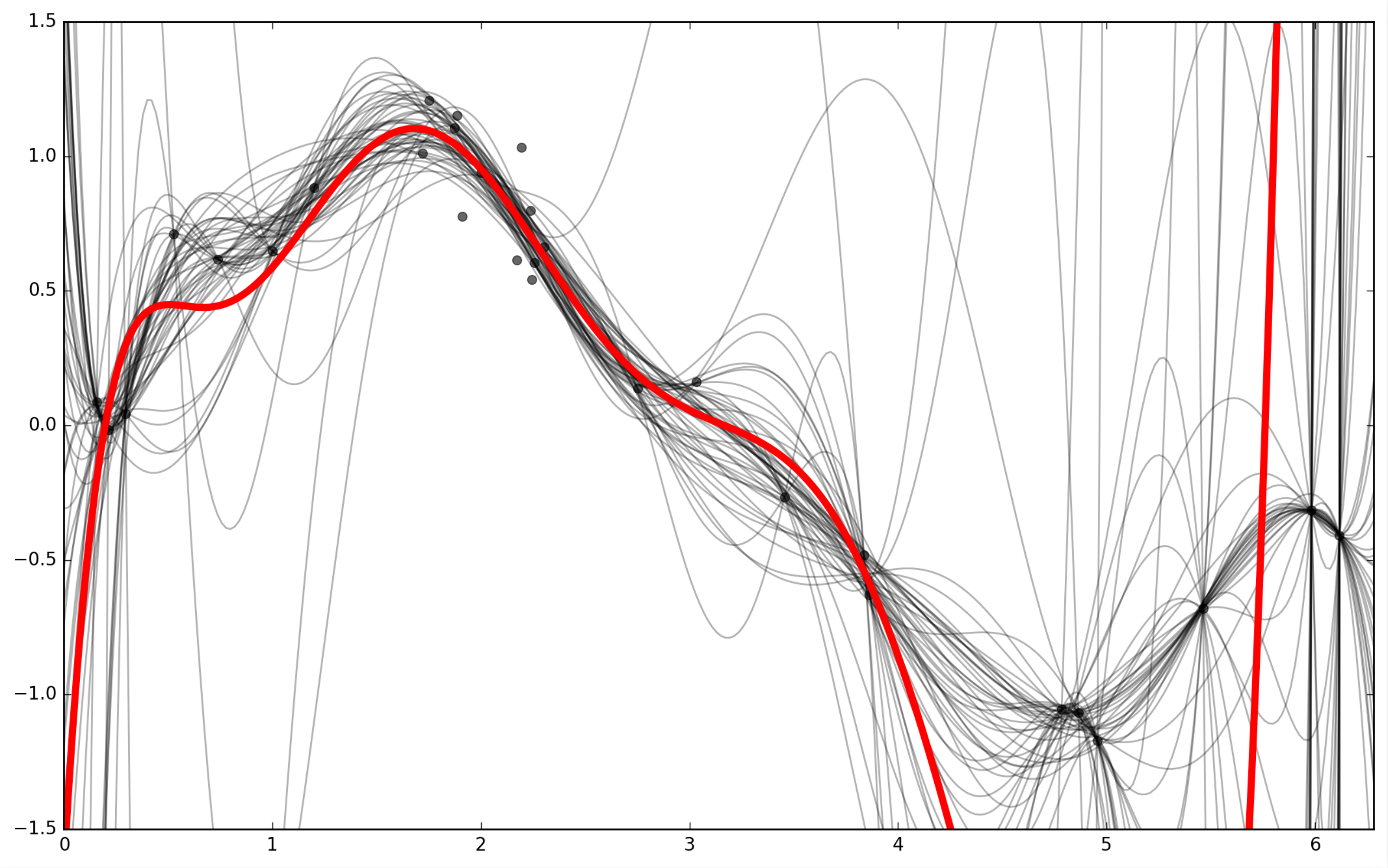
- Example: regression, mean

```
def bootstrap(samples,data):
    train_sets = np.zeros((samples,data.shape[0],data.shape[1]))
    for sample in range(samples):
        ix = np.random.randint(data.shape[0],size=data.shape[0])
        train_sets[sample,:] = data[ix,:]
    return train_sets

train_sets = bootstrap(replicas,data)
px = np.linspace(ax_lims[0],ax_lims[1],points)
preds = np.zeros((replicas,points))
for ix in range(replicas):
    coefs = np.polyfit(train_sets[ix,:,0],
                      train_sets[ix,:,1],degree)
    preds[ix,:] = np.polyval(coefs,px)
mean = np.mean(preds,axis=0)
```

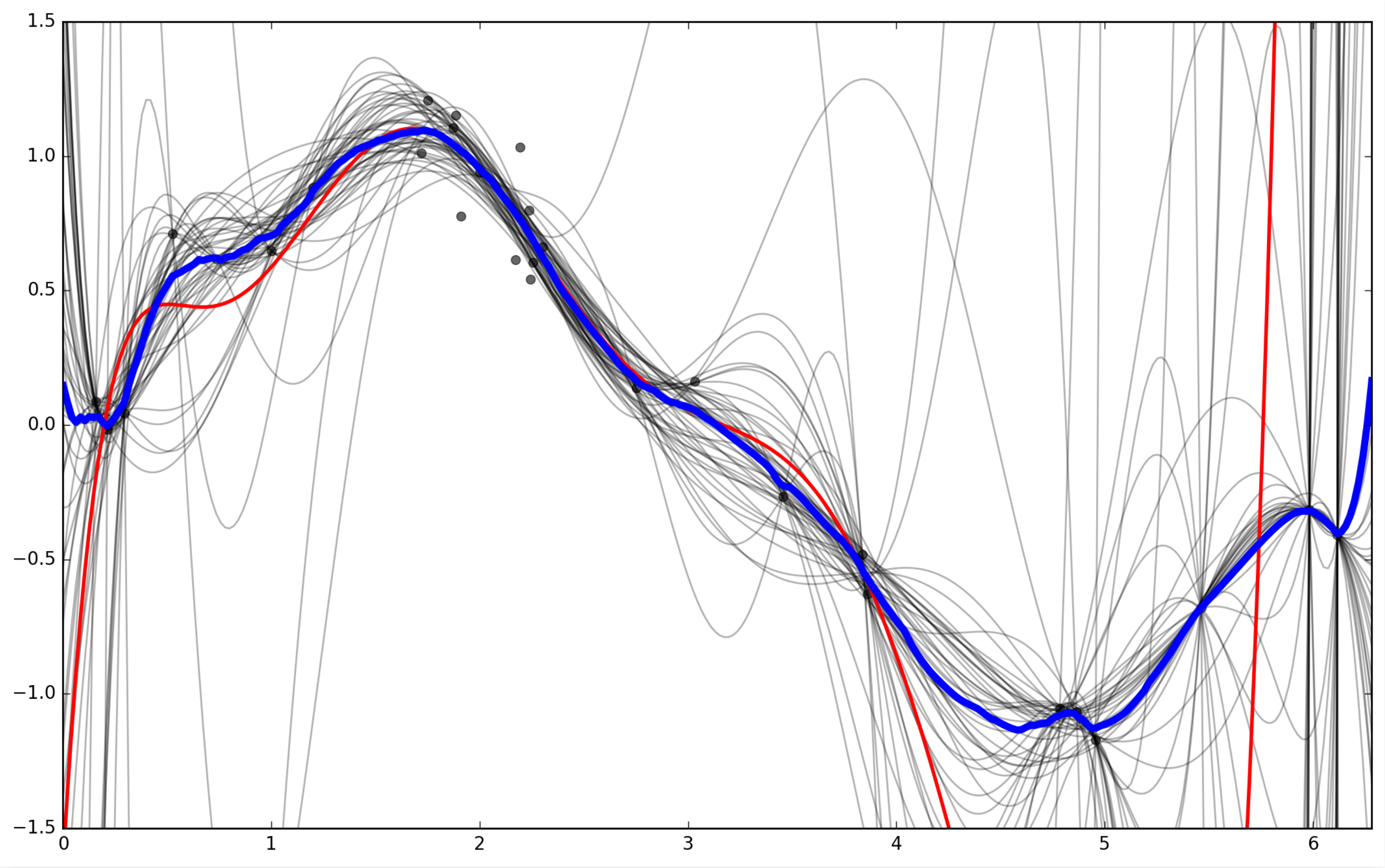
Ensemble methods

- Example: regression, mean



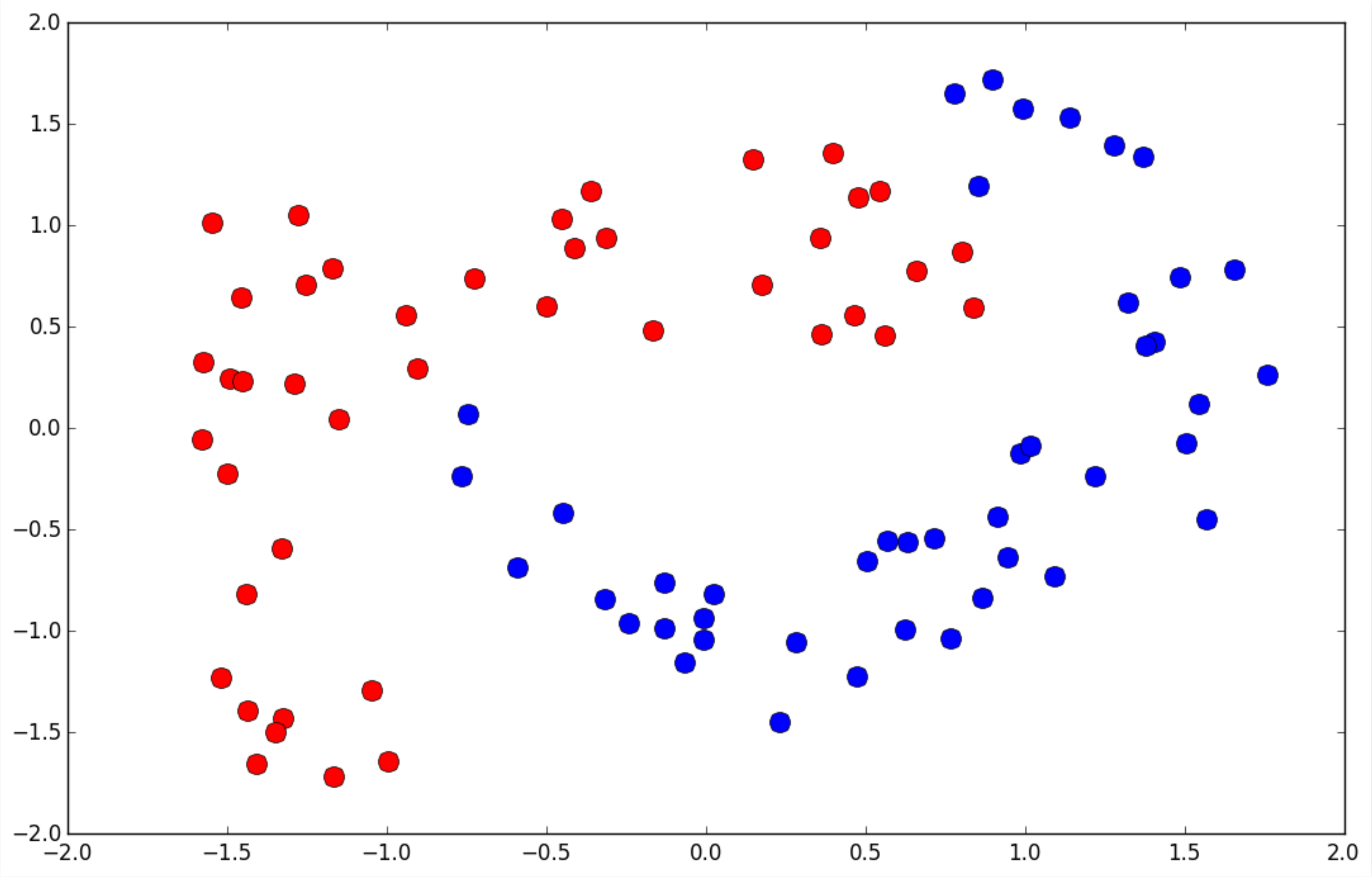
Ensemble methods

- Variation: median instead of mean



Ensemble methods

- Classification example: SVM



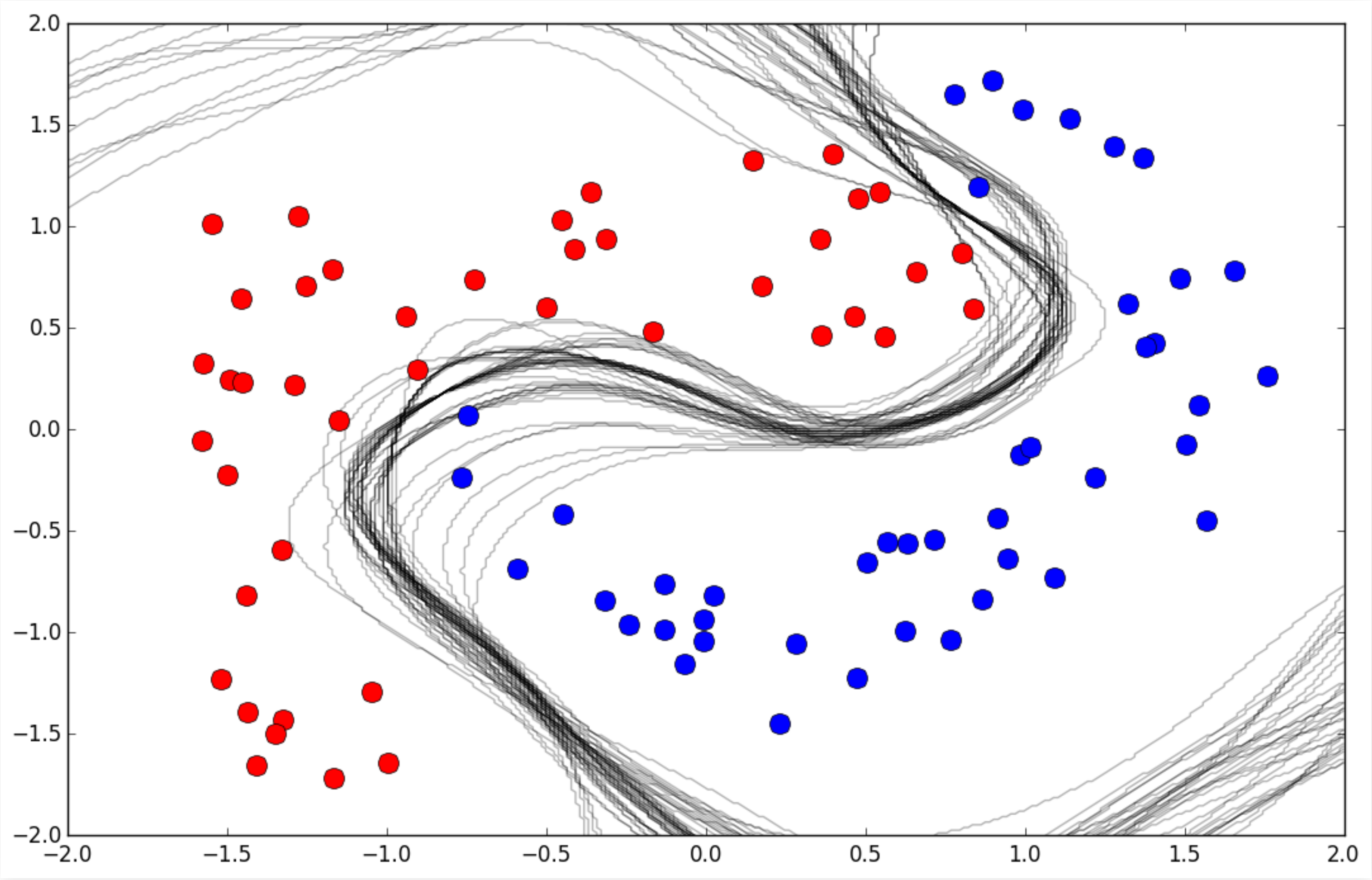
Ensemble methods

- Classification example: SVM (majority vote)

```
train_sets = bootstrap(replicas,data)
gamma = 2
C=10000
svs = []
pX,pY = np.meshgrid(pxs,pys)
pZ = np.zeros((len(pxs),len(pys)))
for ix in range(replicas):
    sv = svm.SVC(kernel='rbf', gamma=gamma,C=C)
    sv.fit(train_sets[ix,:,:-1],train_sets[ix,:,-1])
    svs.append(sv)
    preds = sv.predict(np.c_[pX.ravel(),pY.ravel()]).reshape(pZ.shape)
    pZ = pZ + preds
pZ = np.round(pZ/float(replicas))
```

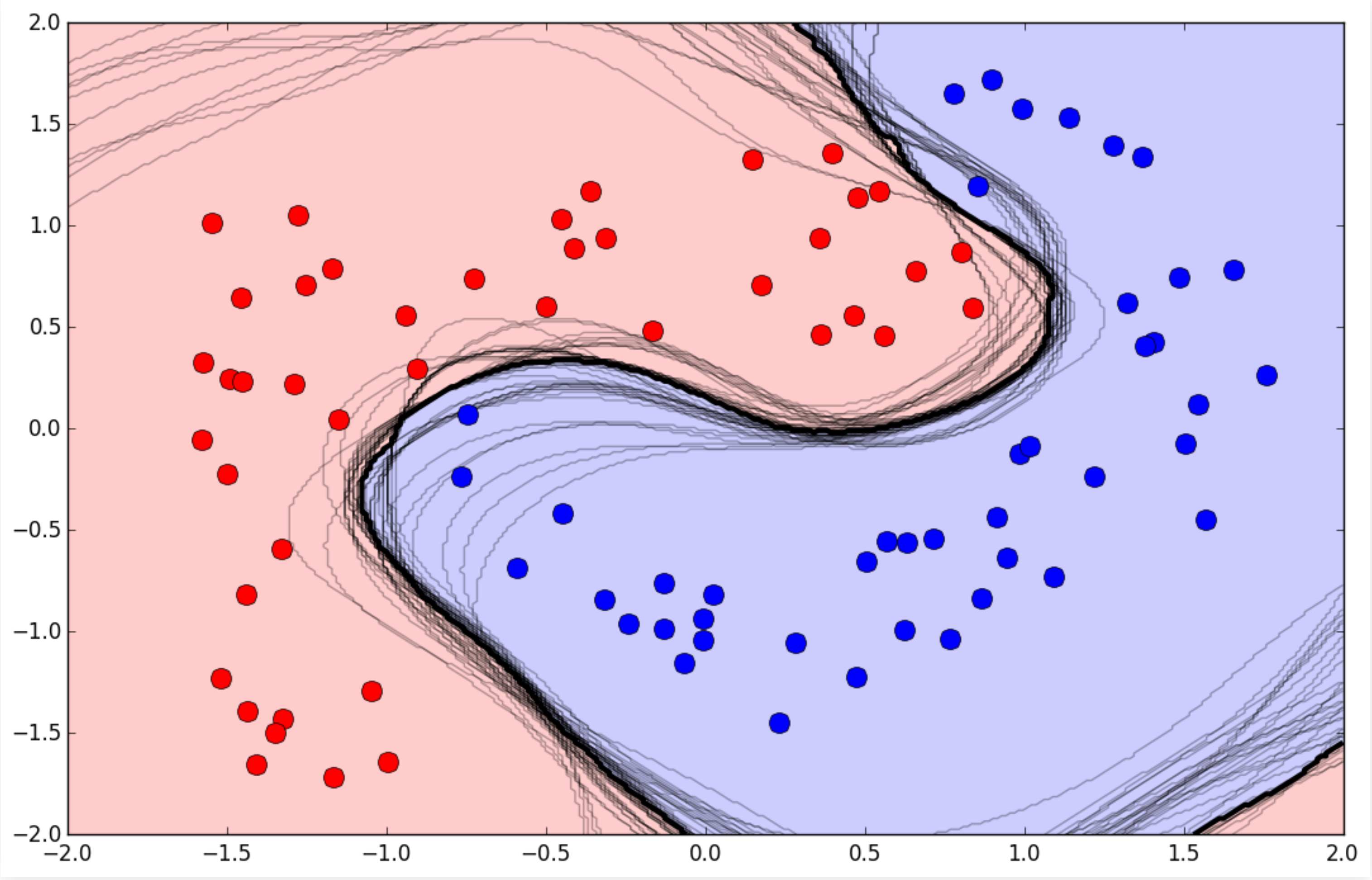
Ensemble methods

- 50 SVM, trained with bootstrapping



Ensemble methods

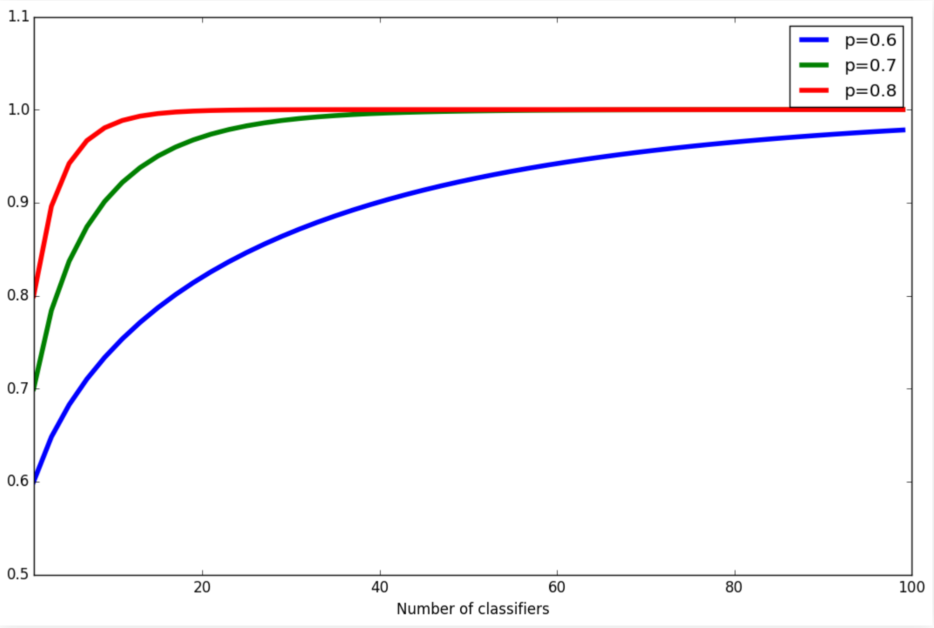
- Majority class of 50 SVM



Bagging (Bootstrap aggregating)

- Averaging reduces variance and overfitting, increasing probability of correct classification as number of classifiers increases

$$\sum_{k=T/2+1}^{T} \binom{T}{k} p^k (1-p)^{T-k}$$



Bagging (Bootstrap aggregating)

- Bootstrap aggregating classifiers

$$\sum_{k=T/2+1}^{T} \binom{T}{k} p^k (1-p)^{T-k}$$

- This assumes classifiers are independent
- If classifiers are correlated, this does not work so well
- Bagging is best for unstable algorithms
  - (susceptible to input variations)

Boosting

Boosting

- Learn a linear combination of weak classifiers
- Individual classifiers must have error rate below 0.5
- Combination of classifiers has a lower bias and better classification power

AdaBoost

- Initialize sample weights: $w_n = 1/N$
- Fit classifier $y_m(x)$ by minimizing weighted error

$$J_m = \sum_{n=1}^{N} w_m^n I(y_m(x^n) \neq t^n)$$

- Compute weighted error on training set:

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_m^n I(y_m(x^n) \neq t^n)}{\sum_{n=1}^{N} w_m^n}$$

Ensemble methods

- Compute classifier weight:

$$\alpha_m = \ln \frac{1 - \epsilon_m}{\epsilon_m}$$

  - Original (Freund and Schapire,2003): $\alpha_m = \frac{1}{2} \ln \frac{1-\epsilon_m}{\epsilon_m}$

- Compute new sample weights (and normalize):

$$w_{m+1}^n = w_m^n \exp\left(\alpha_m I(y_m(x^n) \neq t^n)\right)$$

- Increases weight of misclassified points
- Stop when ϵ_m is zero or greater than 0.5
- Output of the boosted classifier is weighted sum of classifiers:

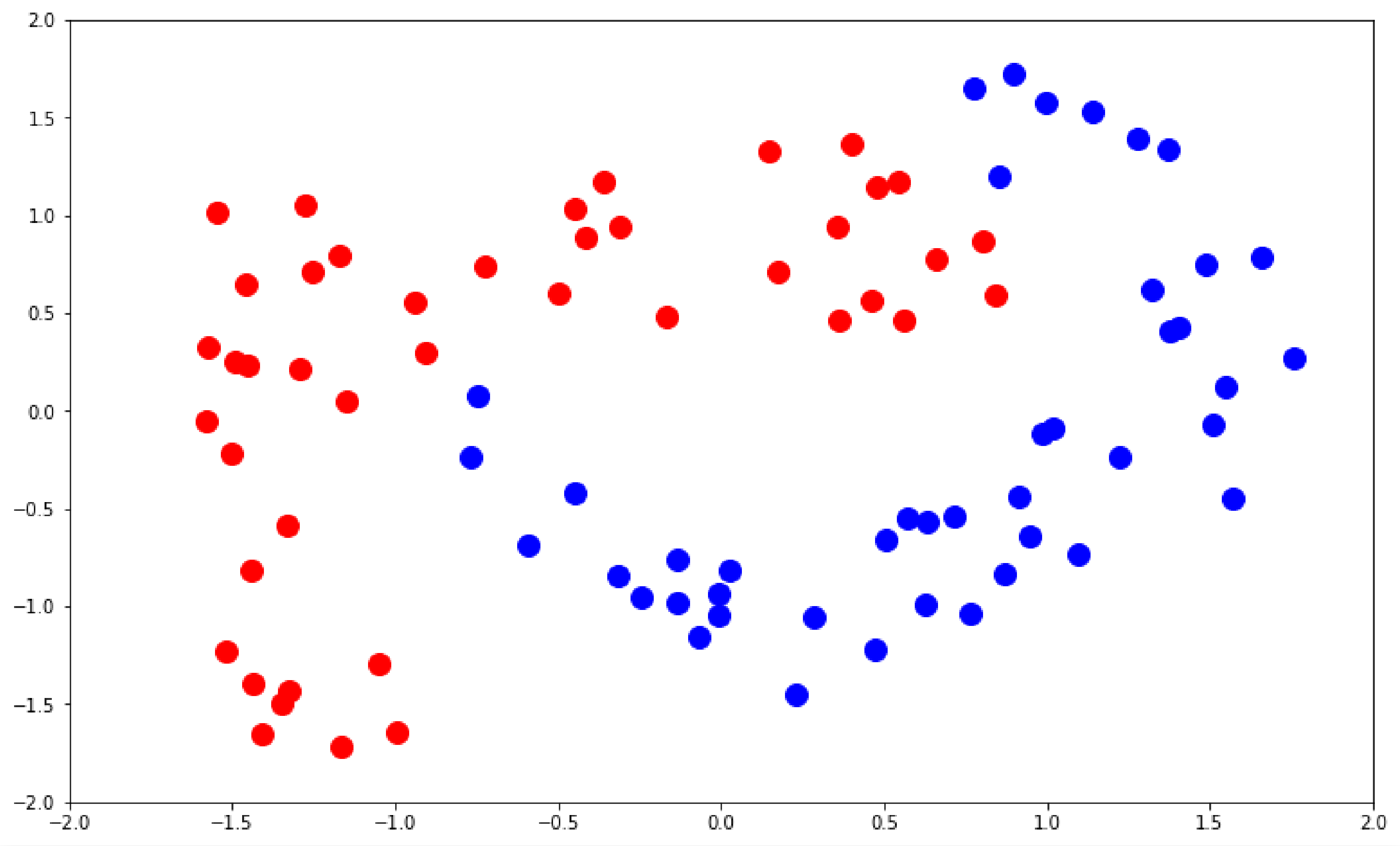
$$f(x) = sign \sum_{m=1}^{M} \alpha_m y_m(x)$$

Decision tree algorithm

- Split data into 2 subsets according to some feature and rule
  - e.g. $x_1 \leq 1$
- Use some measure of information gain to evaluate the split
  - Classification error: assuming most common class in each subset
  - Gini Index: $G = 1 - \sum_c p_c^2$
  - Information Entropy: $Entropy = \sum_c p_c \log p_c$
- Choose one feature and rule that optimizes information gain
- Repeat for each subset with mixed classes

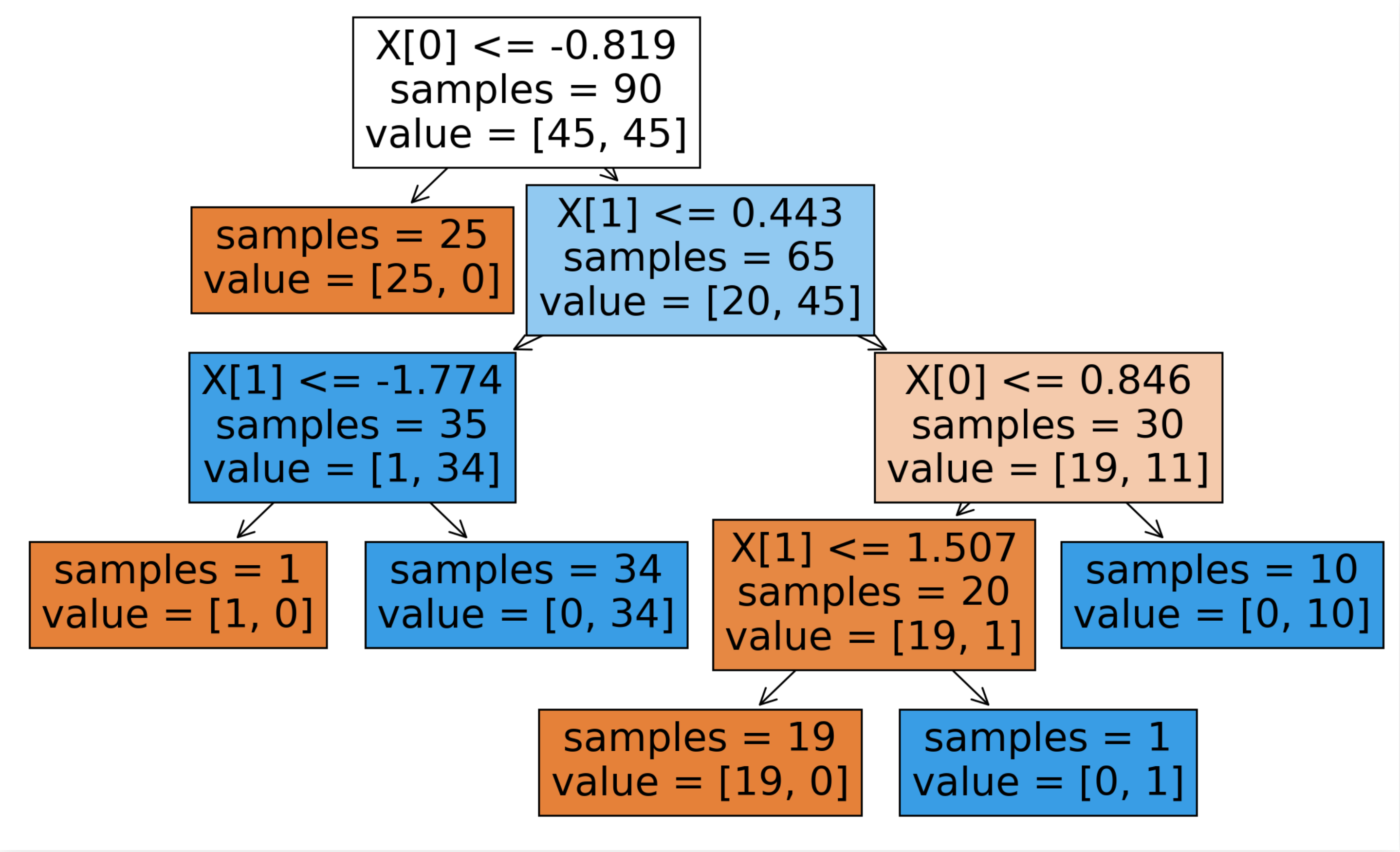
Decision tree algorithm

- Example:



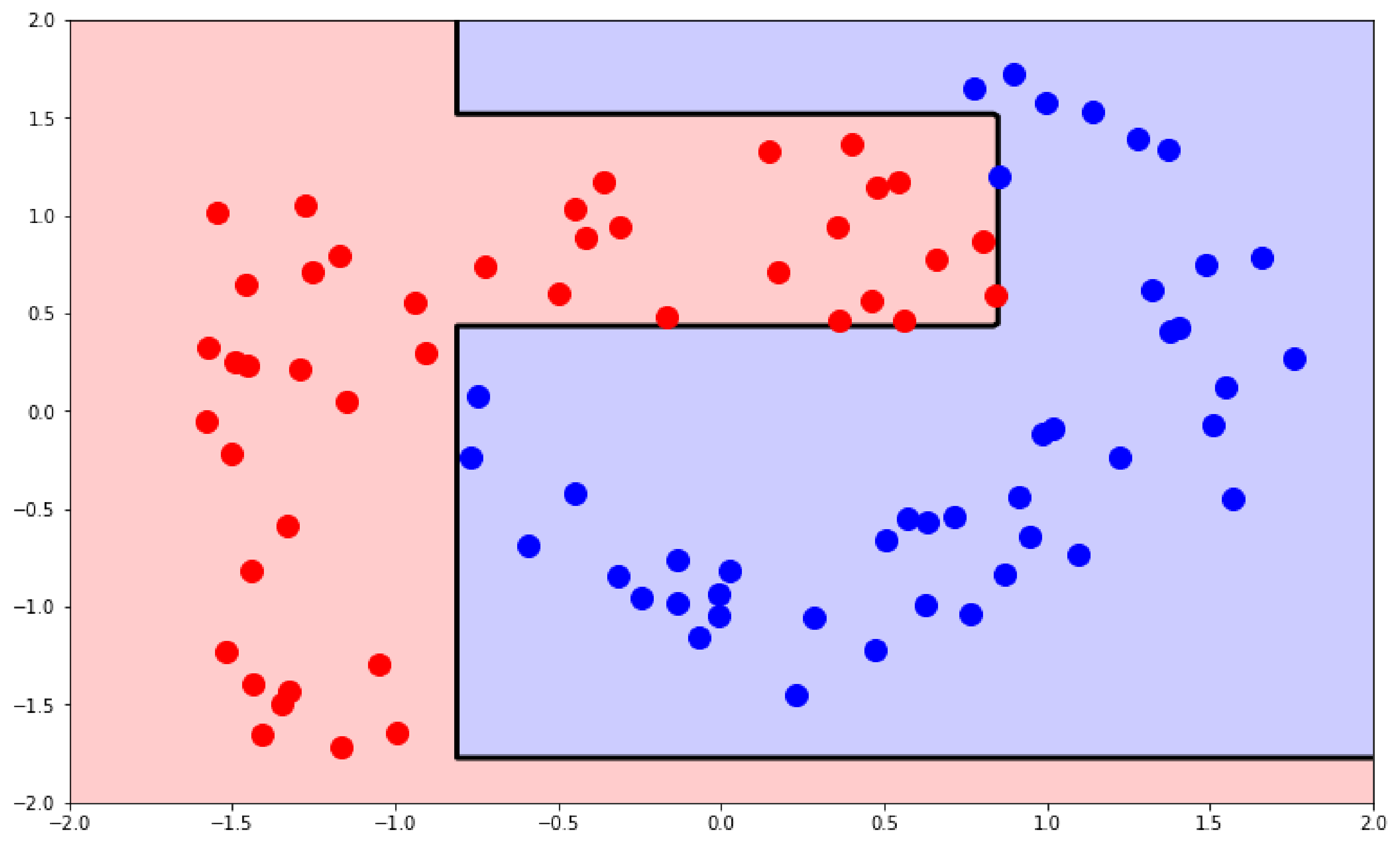
Decision tree algorithm

- Example:



Decision tree algorithm

- Example:



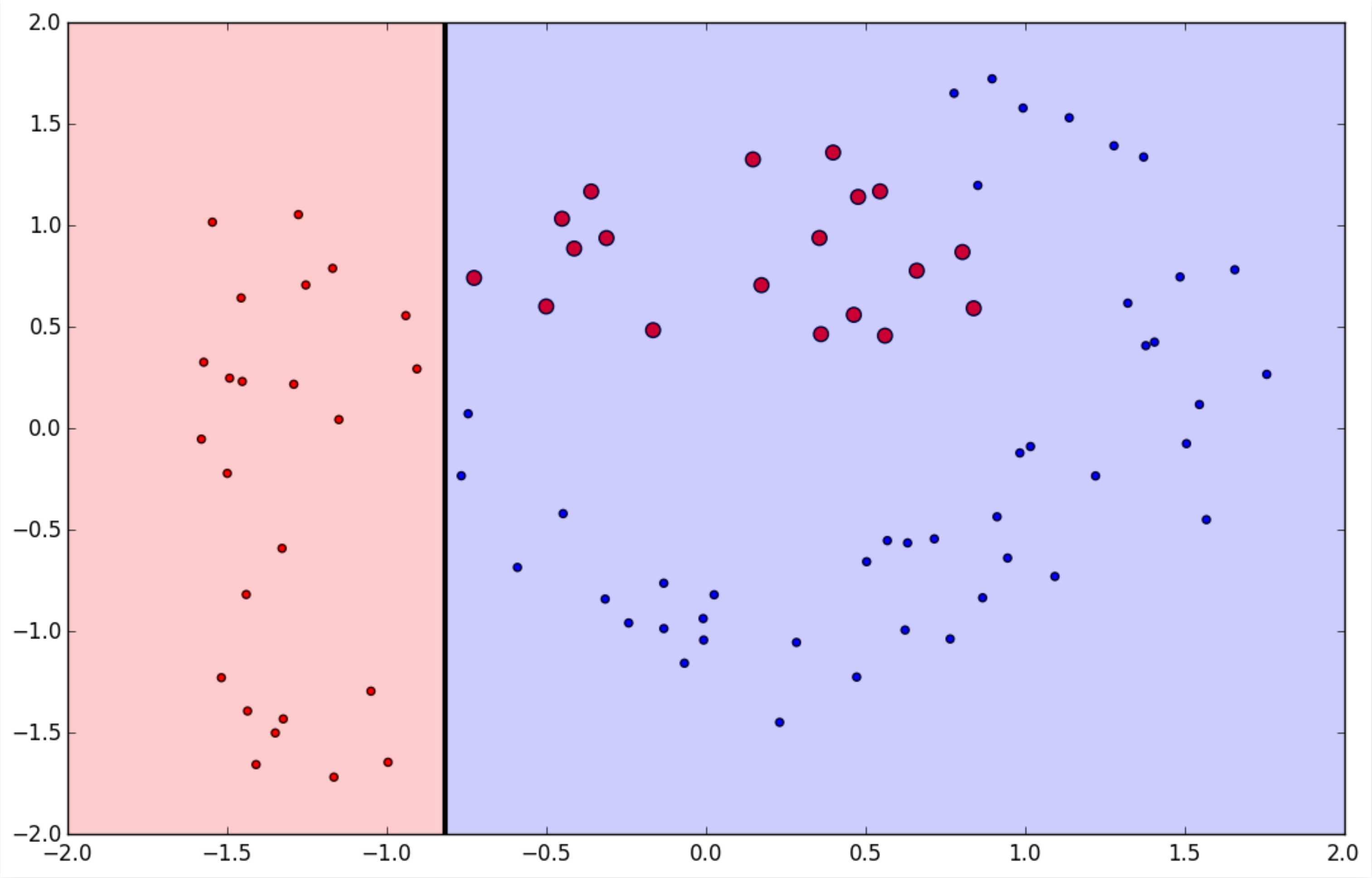
Boosting

- Stumping: AdaBoost with decision stumps (level 1 decision tree)
  - Choose one feature, split at one point
  - Use `DecisionTreeClassifier`

```
from sklearn.tree import DecisionTreeClassifier
hyps = []
hyp_ws = []
point_ws = np.ones(data.shape[0])/float(data.shape[0])
max_hyp = 50
for ix in range(max_hyp):
    stump = DecisionTreeClassifier(max_depth=1)
    stump.fit(data[:,:-1], data[:,-1], sample_weight = point_ws)
    pred = stump.predict(data[:,:-1])
    errs = (pred != data[:,-1]).astype(int)
    err = np.sum(errs*point_ws)
    alpha = np.log((1-err)/err)
    point_ws = point_ws*np.exp(alpha*errs)
    point_ws = point_ws/np.sum(point_ws)
    hyps.append(stump)
    hyp_ws.append(alpha)
```

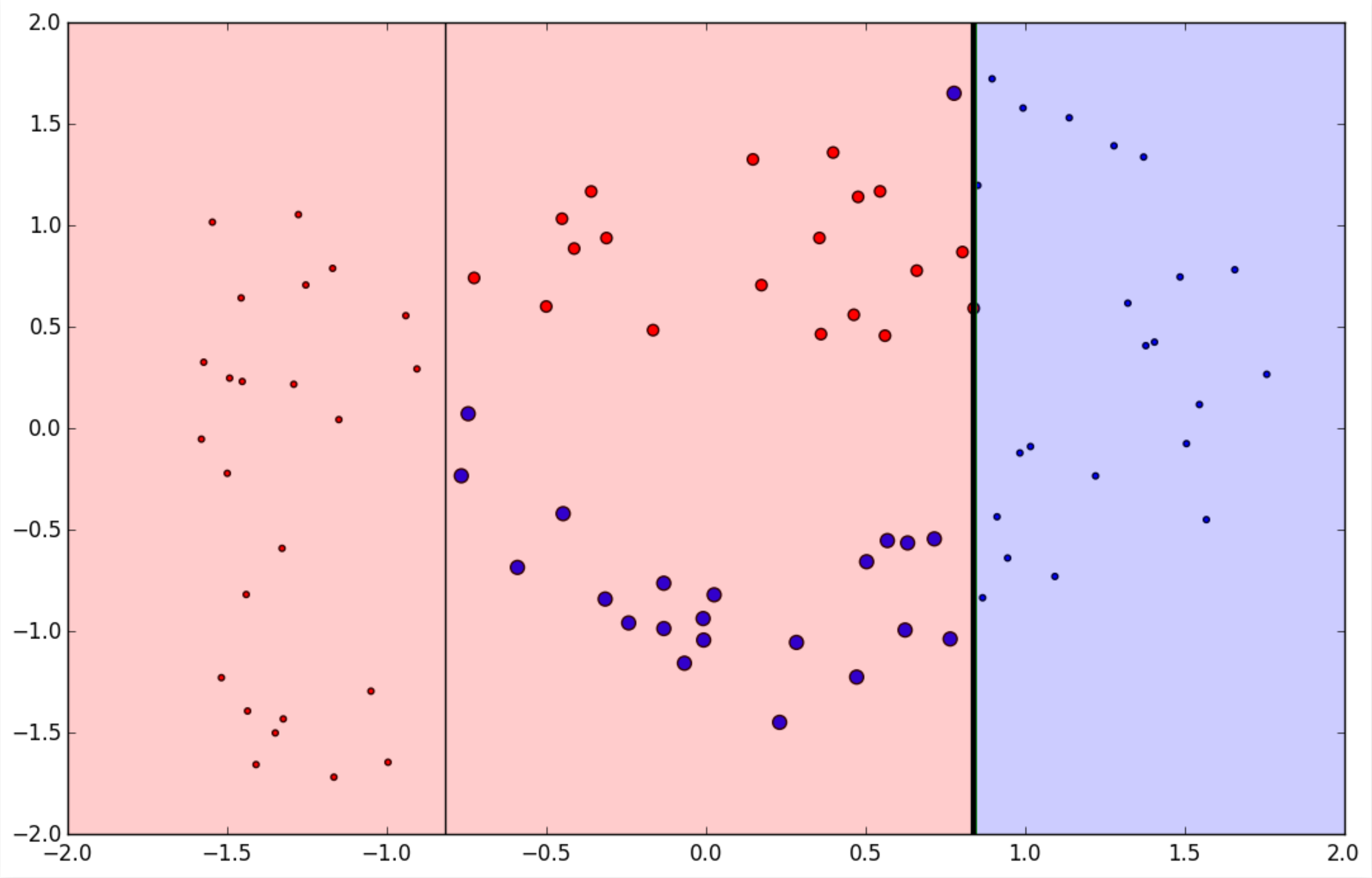
Boosting

- Stumping: AdaBoost with decision stumps (level 1 decision tree)



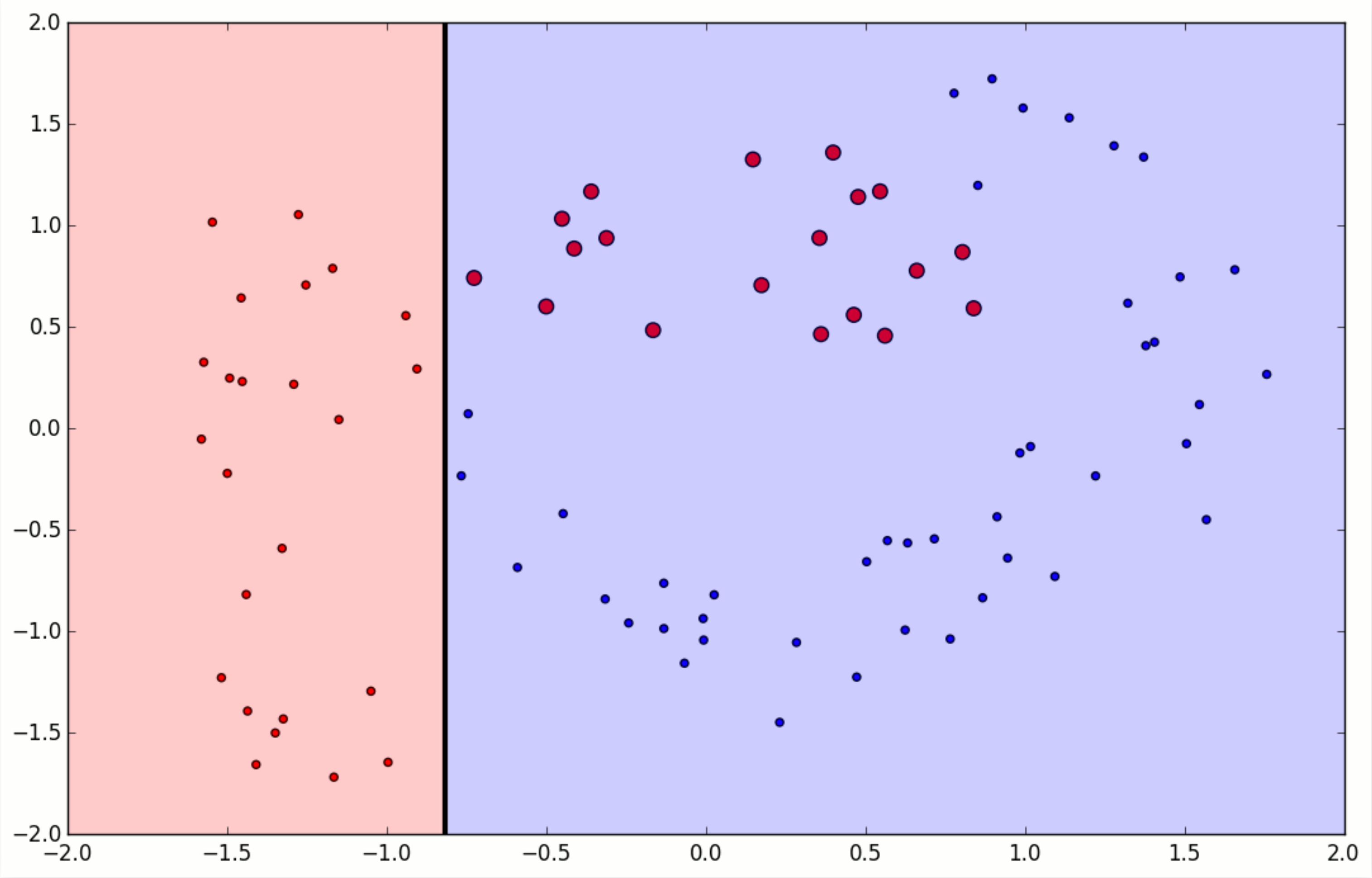
Boosting

- Stumping: AdaBoost with decision stumps (level 1 decision tree)



Boosting

- Stumping: AdaBoost with decision stumps (level 1 decision tree)



FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Boosting

- Stumping: AdaBoost with decision stumps (level 1 decision tree)
- Classifying data and computing error

```
net_pred = np.zeros(data.shape[0])
for ix in range(len(hyps)):
    pred_n = hyps[ix].predict(data[:,:-1])
    preds = preds+pred_n*hyp_ws[ix]
net_pred[preds<0] = -1
net_pred[preds>=0] = 1
errors = np.sum((net_pred !=data[:,-1]).astype(int))
```

AdaBoost, derivation

- We can see AdaBoost as a sequential mimization of the exponential error function:

$$E = \sum_{n=1}^{N} \exp(-t_n f_m(x_n))$$

- Where $f_m(x)$ is the weighted classification of the m classifiers:

$$f_m(x) = \frac{1}{2} \sum_{j=1}^{m} \alpha_j y_j(x)$$

- All $f_1 \ldots f_{m-1}$ are assumed constant
- Minimize only for the last one, $\alpha_m y_m(x)$

Boosting

- We can decompose the error in correctly and incorrectly classified:

$$E = \sum_{n=1}^{N} w_m^n \exp\left(-\frac{1}{2} t_n \alpha_m y_m(x_n)\right) = e^{-\alpha_m/2} \sum_{n \in \mathcal{T}} w_m^n + e^{\alpha_m/2} \sum_{n \in \mathcal{M}} w_m^n$$

$$= e^{-\alpha_m/2} \sum_{n=1}^{N} w_m^n + (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_m^n I(y_m(x^n) \neq t^n)$$

  - Minimizing with respect to y_m:

$$J_m = \sum_{n=1}^{N} w_m^n I(y_m(x^n) \neq t^n)$$

  - Minimizing with respect to α_m:

$$\alpha_{m+1} = \ln \frac{1 - \epsilon_m}{\epsilon_m} \qquad \epsilon_m = \sum_{n=1}^{N} w_m^n I(y_m(x^n) \neq t^n) \,/\, \sum_{n=1}^{N} w_m^n$$

- AdaBoost minimizes the exponential error of the linear combination of the base classifiers with a sequential optimization.

Ensemble methods

- Two examples, to illustrate solutions to different problems.

Bagging

- Averages predictions based on different datasets (bootstrapping)
- Good for models with low bias and high variance (overfitting)

Boosting

- Computes linear combination of weak classifiers (changing example weights)
- Good for models with high bias and low variance (underfitting)

First test

First Test

First test

- Lectures 1-12 (this one).
- Next 2 session (lectures 13-16) not for first test.
- Session of November 5 for questions and revisions
- You can bring 1 handwritten A4 sheet, written on both sides
  - With identification (name and number).
- Exam will be scored in two independent parts.
- Test will include questions for Assignment 1

Summary

Ensemble methods

Summary

- Bagging: reduce variance by averaging
  - Useful for models with large variance
  - Useful for unstable models, otherwise there is too much correlation
- Boosting: reduce bias by linear combination of classifiers
  - Useful for combining weak classifiers (large bias)
  - Note: must be able to weigh samples

Further reading

- Alpaydin, Sections 17.6, 17.7
- Marsland, Chapter 7
- Bishop, Sections 14.2, 14.3