School of AI

January 2020

Ensemble learning
Random forests
Dimension reduction

Ensemble learning

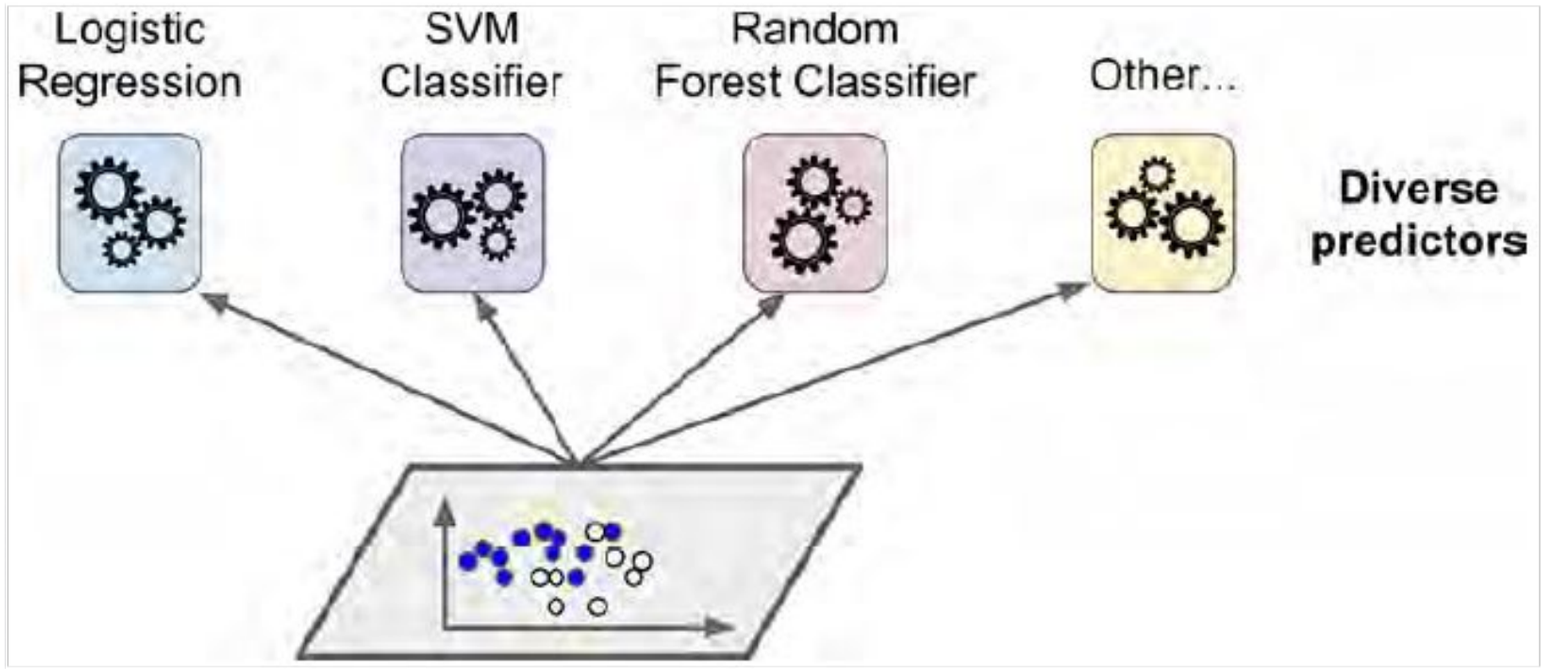
a group of predictors (ensemble)

wisdom of the crowd

one expert

(pictures: shutterstock.com)

Training multiple classifiers

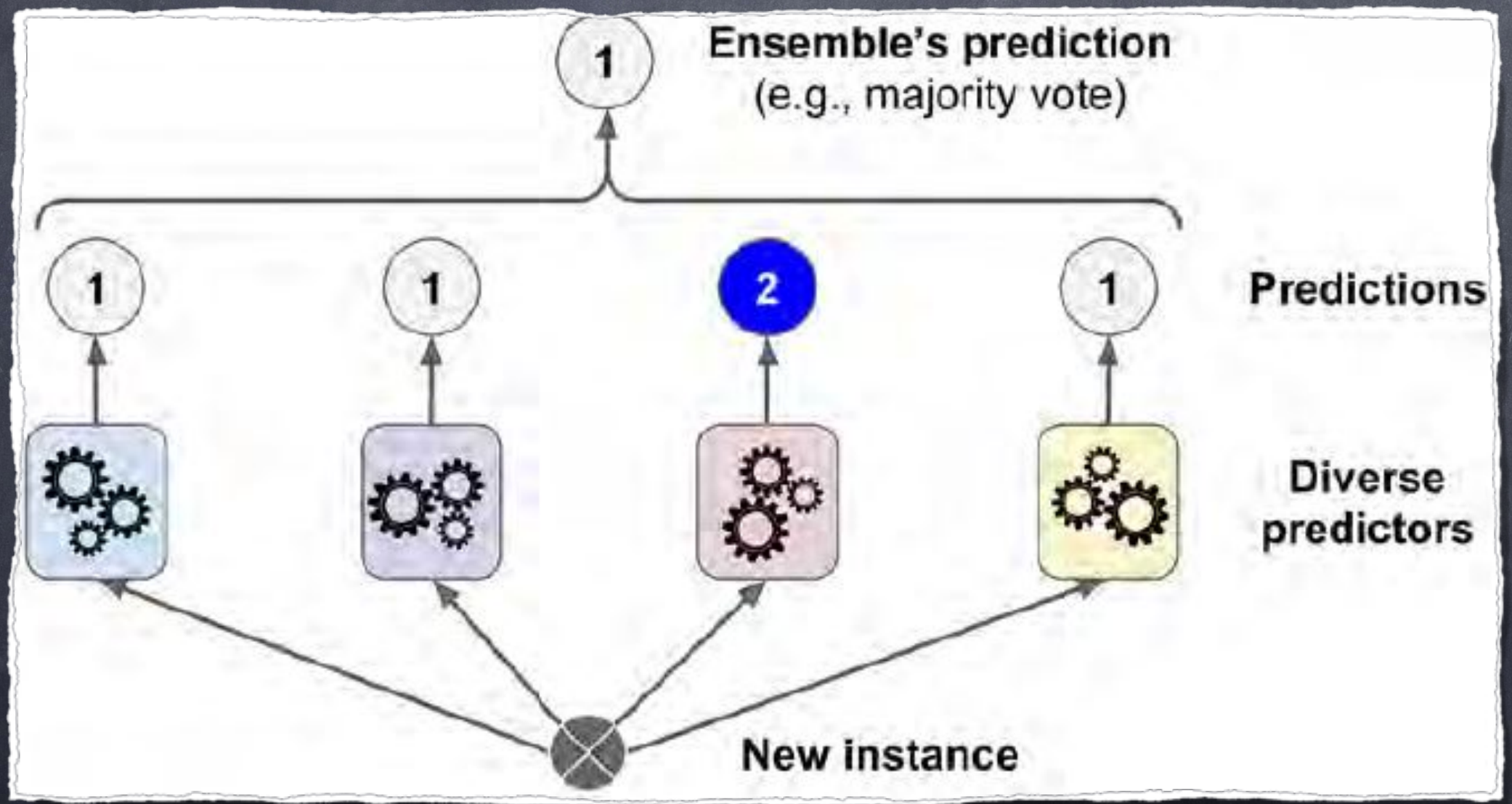


you may have trained:

- logistic regression classifier
- SVM
- Random forest
- K-nearest neighbors classifier
- or other

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Ensemble's prediction

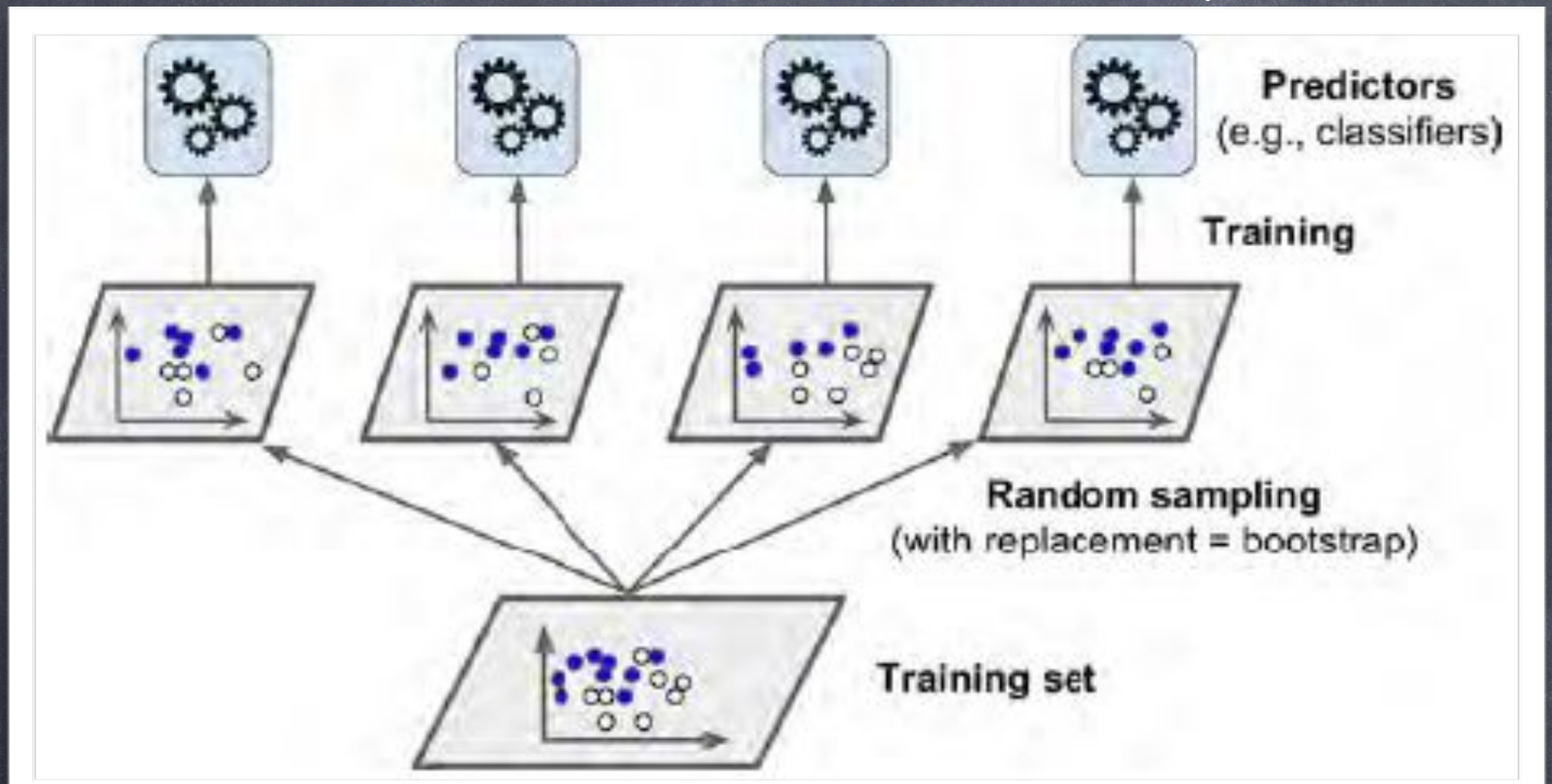


(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Interesting fact

- Ensemble methods work best when the predictors are as independent as possible. It is good idea to train them using very different algorithms. This would increase the chance that they make different types of errors, therefore improve the accuracy.

Bagging and pasting



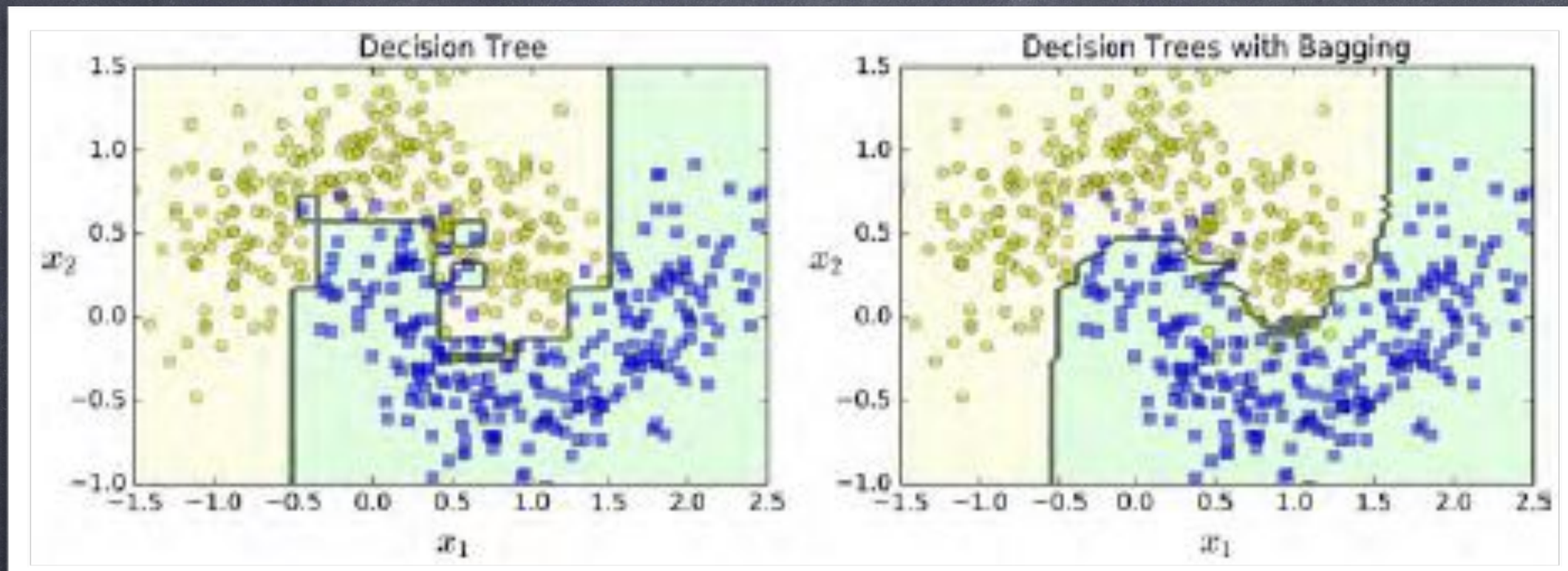
same classifiers

same data set

Bagging: subsamples with replacement
Pasting: subsamples without replacement

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Bagging and pasting



Left: a decision tree, Right: a bagging ensemble of 500 trees

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Random forests

```python
from sklearn.ensemble import RandomForestClassifier

rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1)
rnd_clf.fit(X_train, y_train)

y_pred_rf = rnd_clf.predict(X_test)
```

- 500 trees, each limited to max 16 nodes

Feature importance

- Iris data set

Fisher's *Iris* Data

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*
4.4	2.9	1.4	0.2	*I. setosa*
4.9	3.1	1.5	0.1	*I. setosa*
5.4	3.7	1.5	0.2	*I. setosa*
4.8	3.4	1.6	0.2	*I. setosa*
4.8	3.0	1.4	0.1	*I. setosa*

Sort ascending

pictures: wikipedia.com

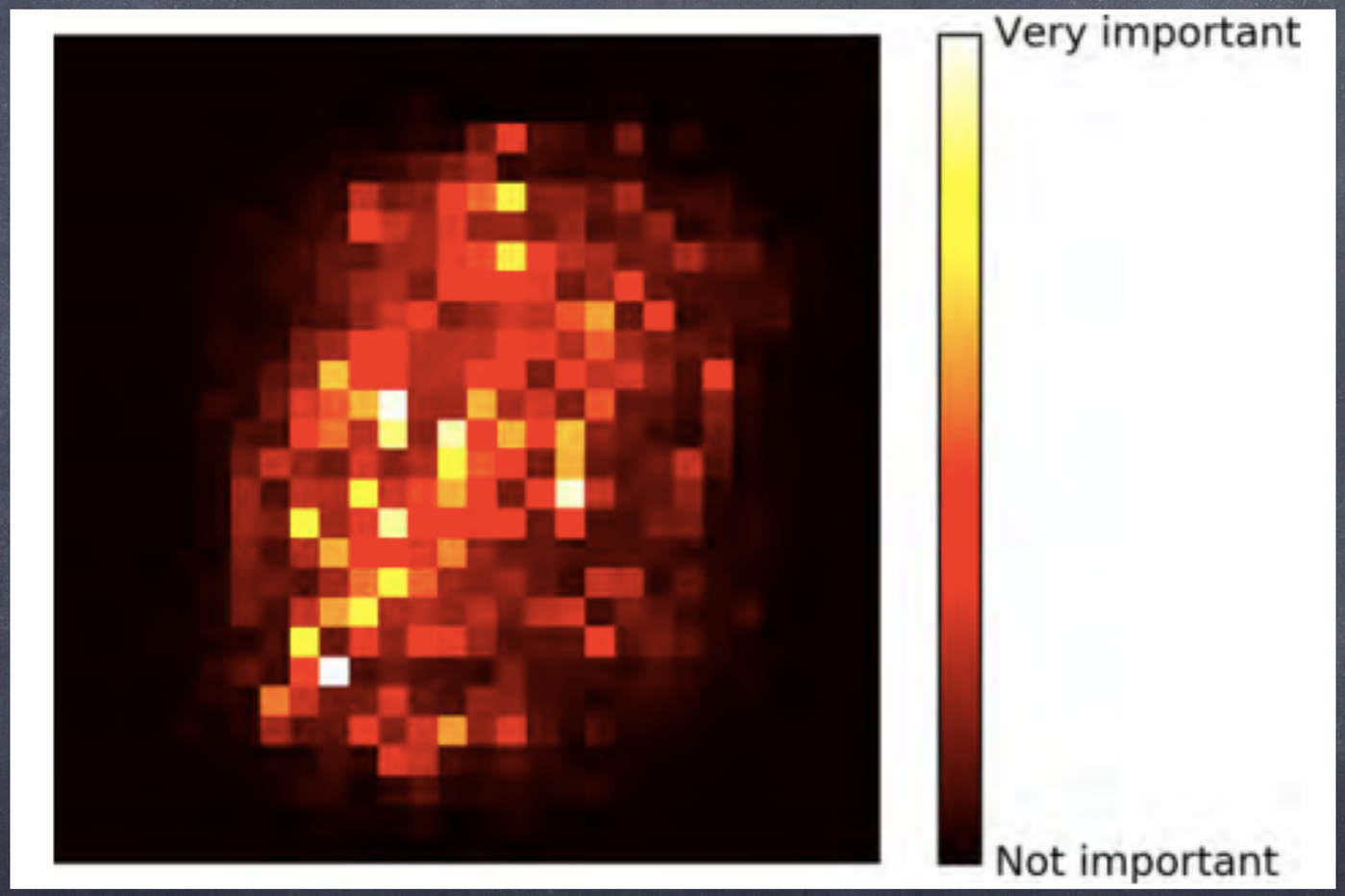
Feature importance

```
>>> from sklearn.datasets import load_iris
>>> iris = load_iris()
>>> rnd_clf = RandomForestClassifier(n_estimators=500, n_jobs=-1)
>>> rnd_clf.fit(iris["data"], iris["target"])
>>> for name, score in zip(iris["feature_names"], rnd_clf.feature_importances_):
>>>     print(name, score)
sepal length (cm) 0.112492250999
 sepal width (cm) 0.0231192882825
 petal length (cm) 0.441030464364
 petal width (cm) 0.423357996355
```

Which feature was not important at all?

Feature importance

- MNIST data set



pixel importance

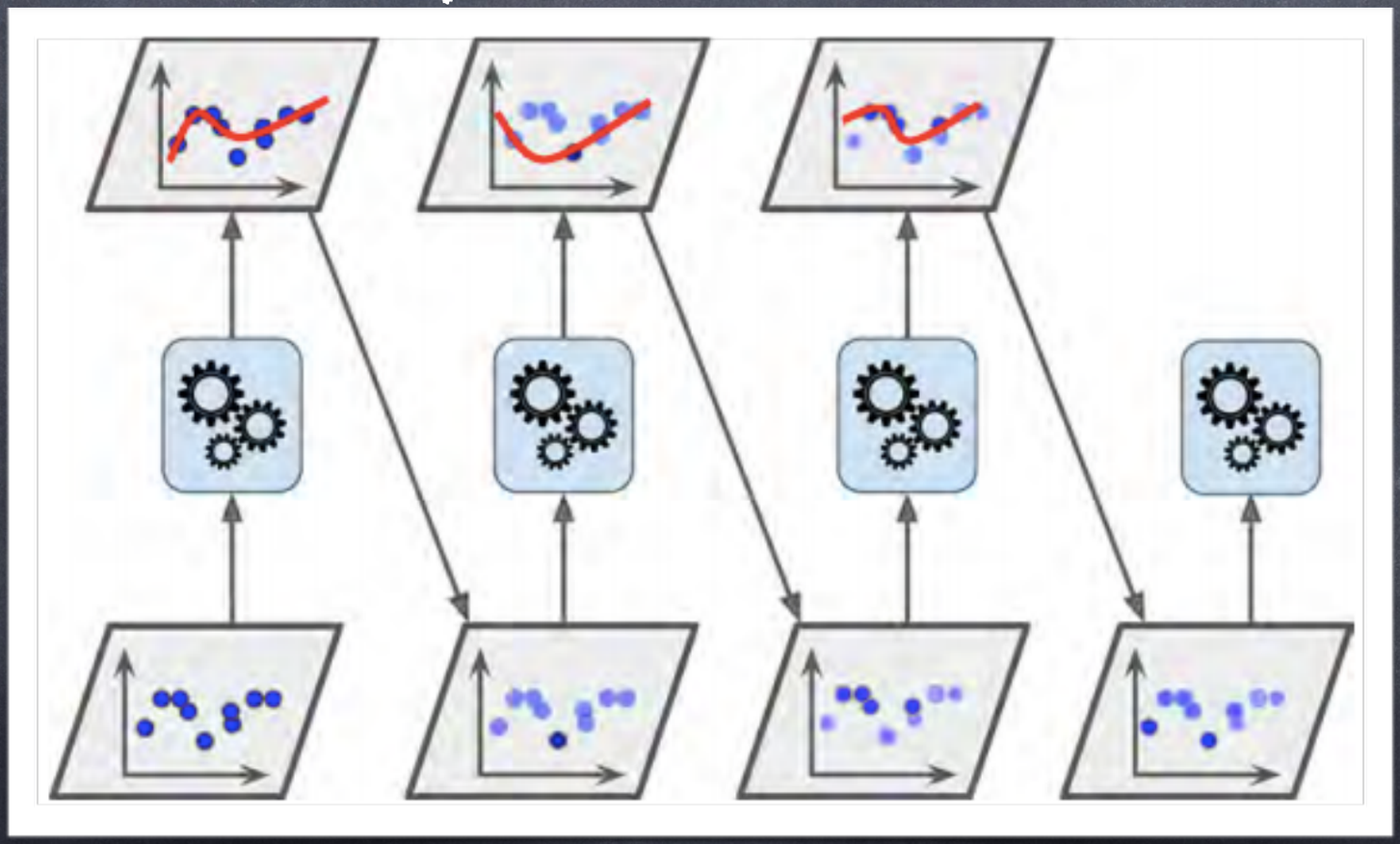
(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Boosting

- Trying to "boost" each weak classifier

Adaboost

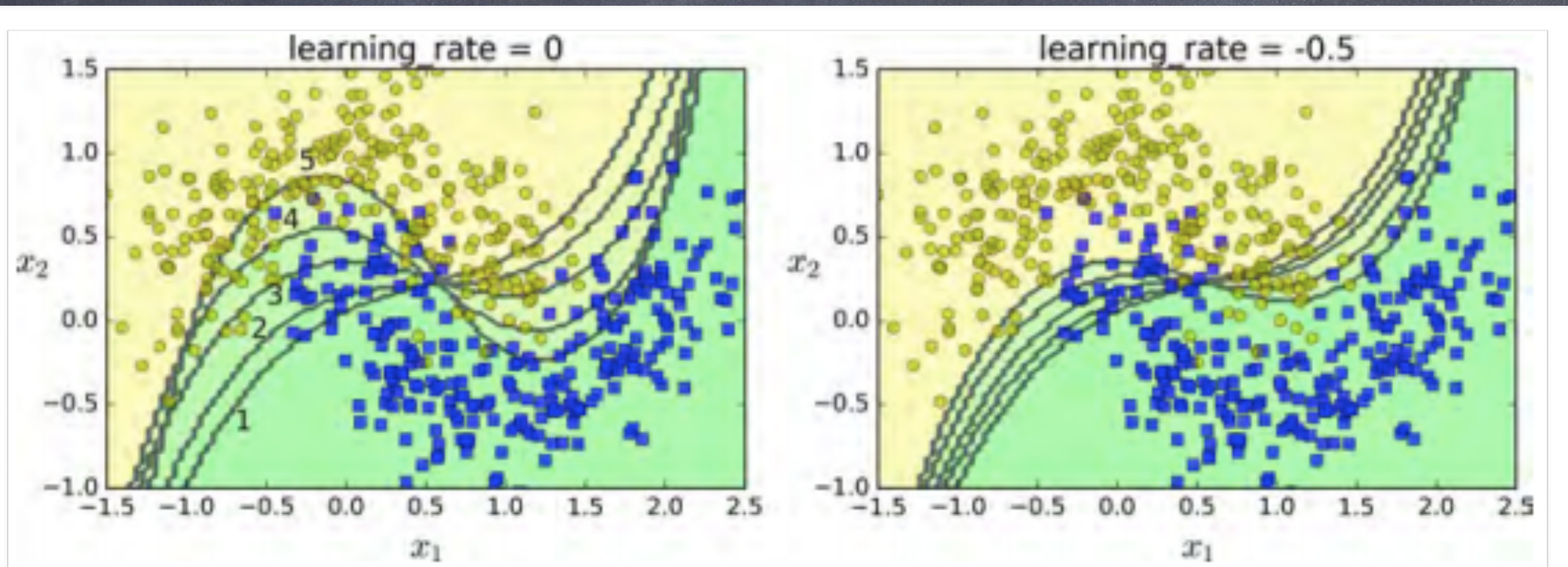
Adaptive Boosting



updating predictor weights iteratively

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Adaboost



(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Gradient boosting

- Like AdaBoost, works by sequentially adding predictors to an ensemble.
- Each predictor corrects its predecessor.
- Instead of tweaking the instance weights, this predictor tries to fit the new predictor to the residual errors made by the previous predictor.

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Gradient boosting

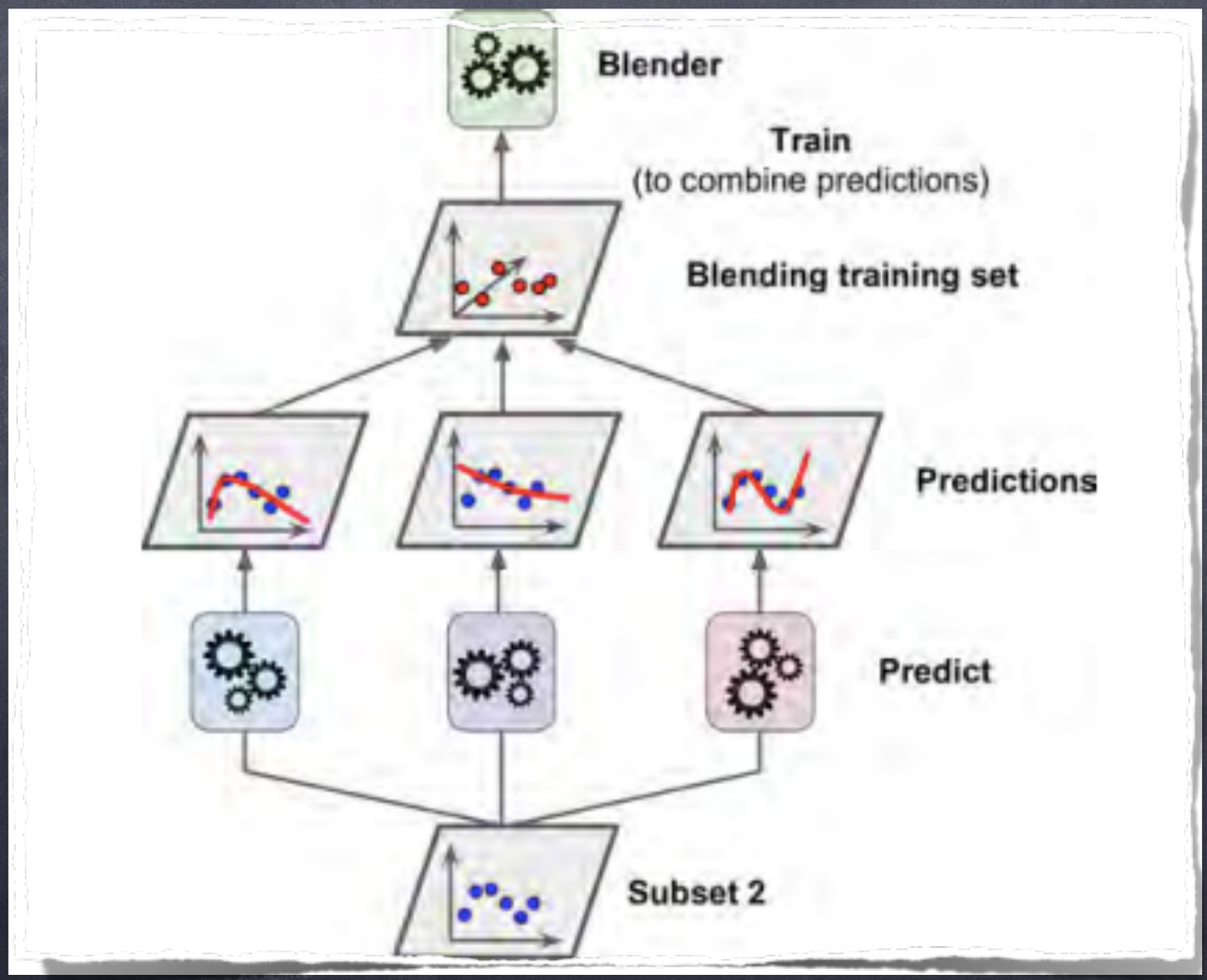
```python
from sklearn.ensemble import GradientBoostingRegressor

gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=3, learning_rate=1.0)
gbrt.fit(X, y)
```

- Learning rate hyperparameter scales the contribution of each tree.
- If it is too low (i.e. 0.1) you need more trees to fit into training set.
- But if it is low, then it generalises better.

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Stacking

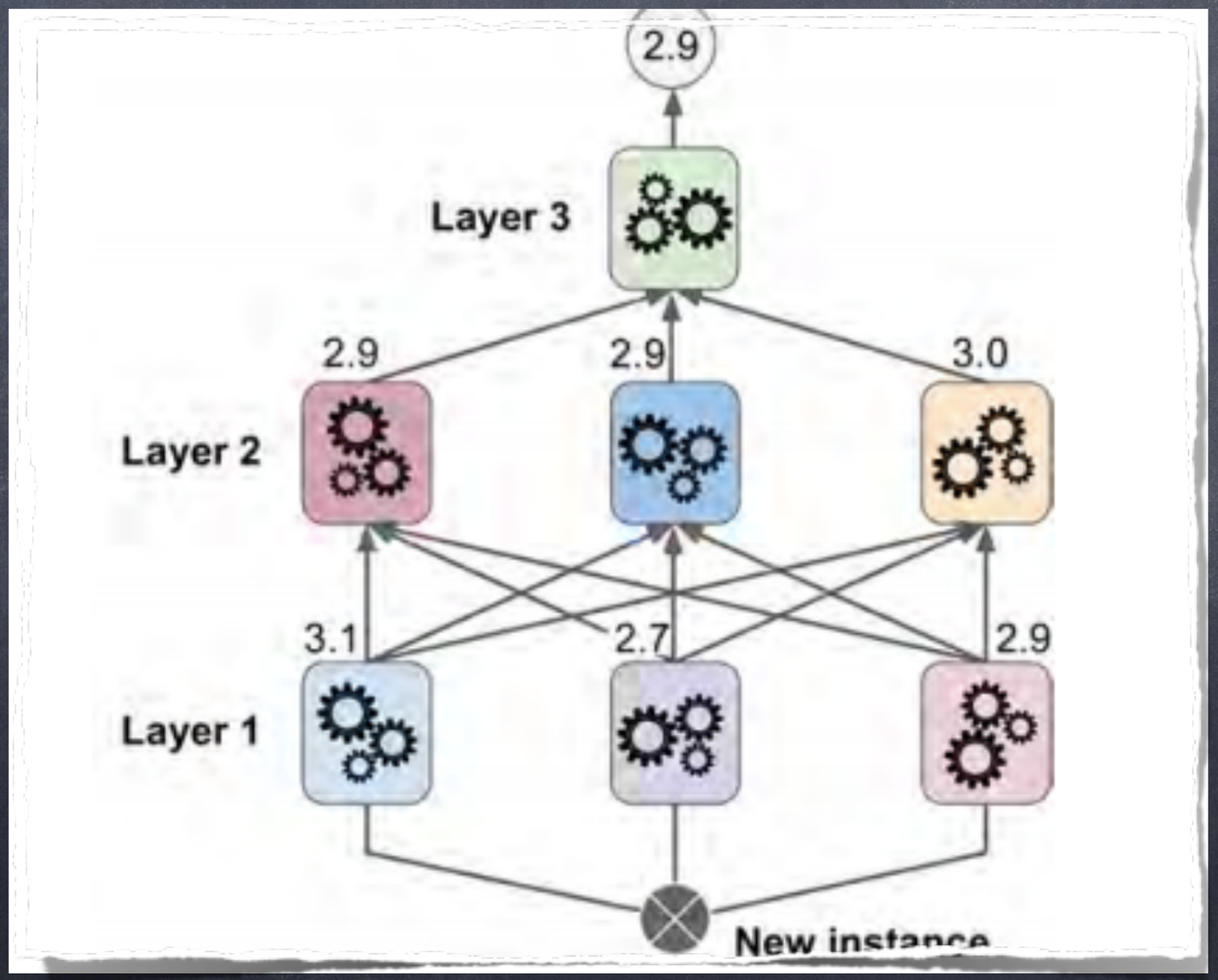
- Blending example



(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Stacking

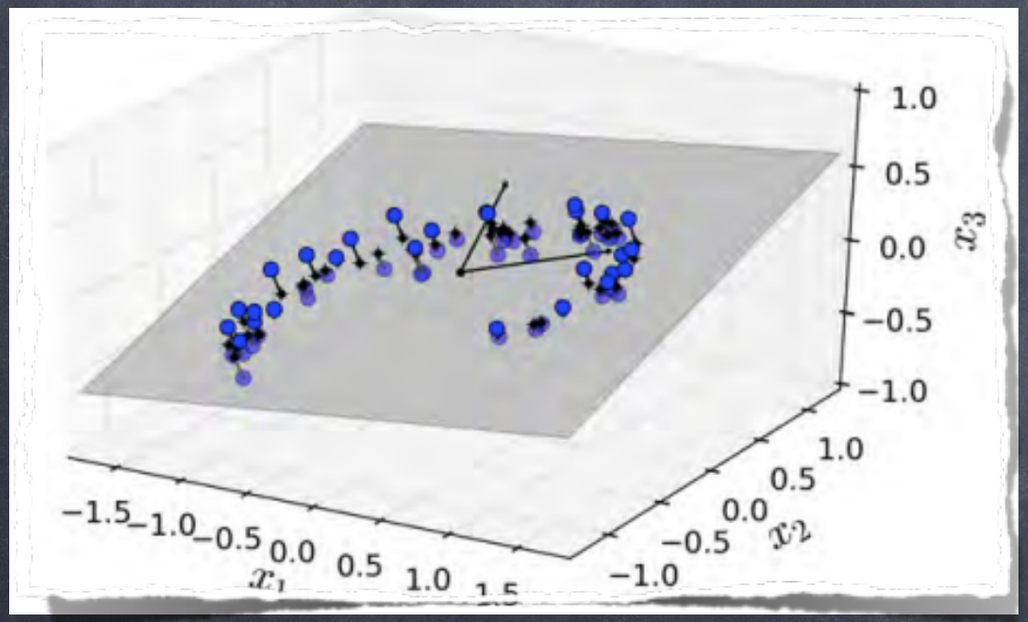
- Multi-layer stacking example

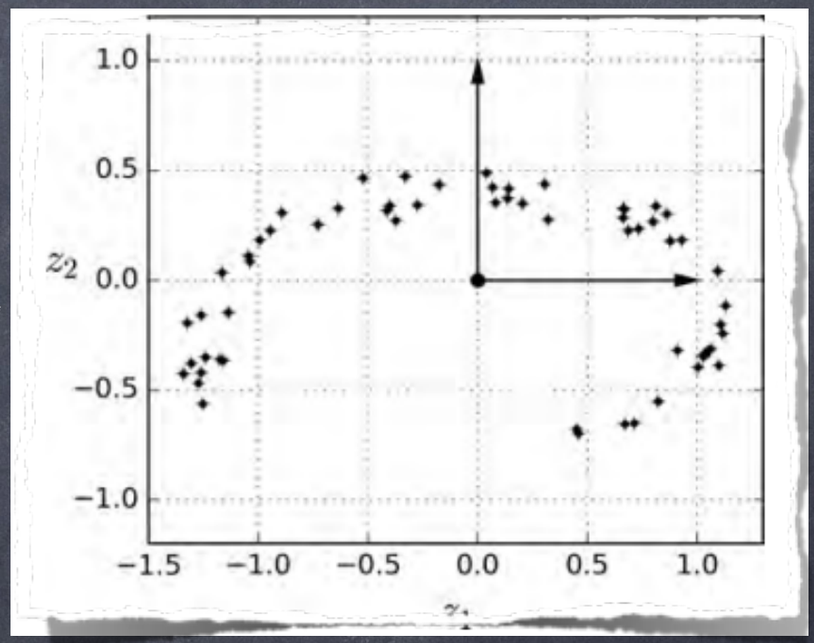


(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Dimensionality reduction

why dimension matters?

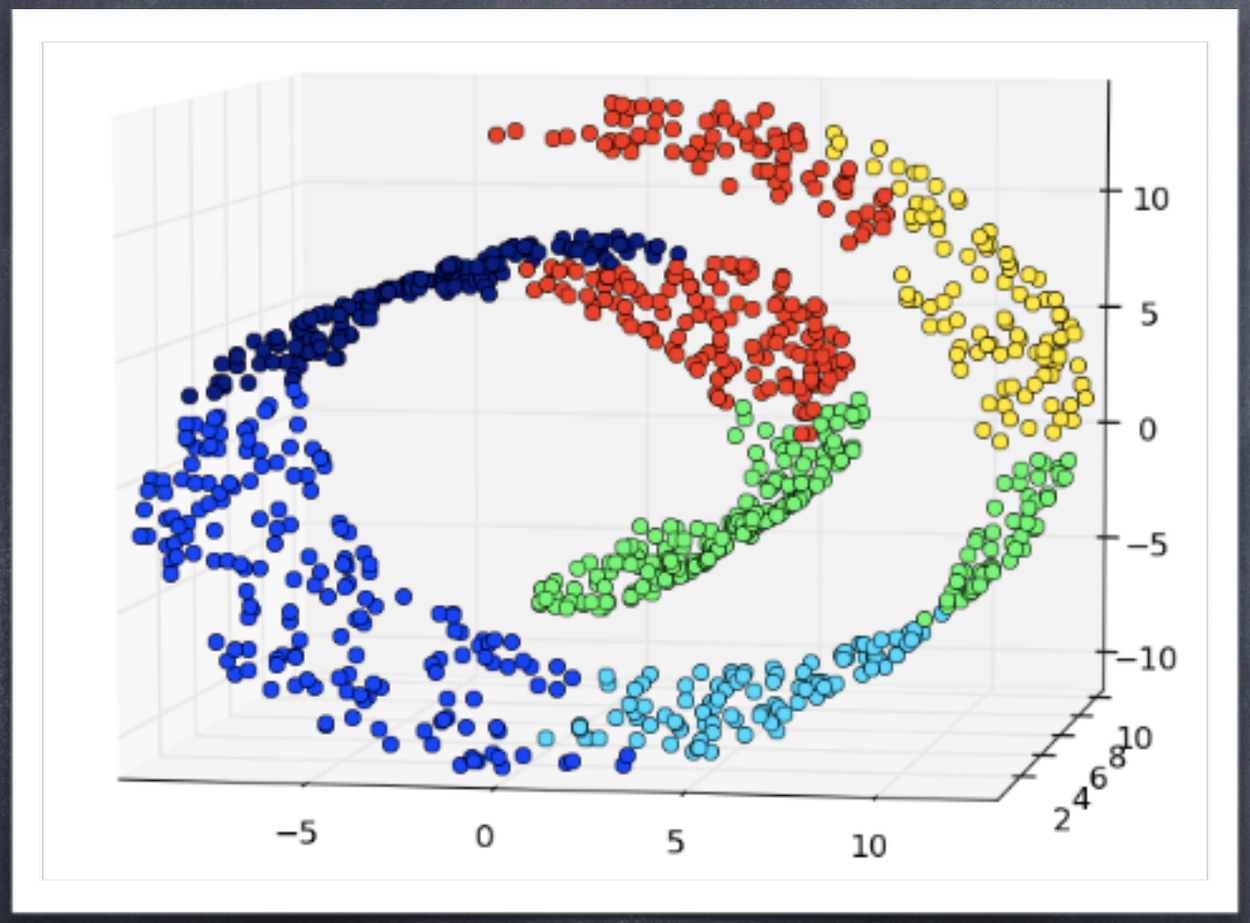




(projection of 3D to 2D)

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

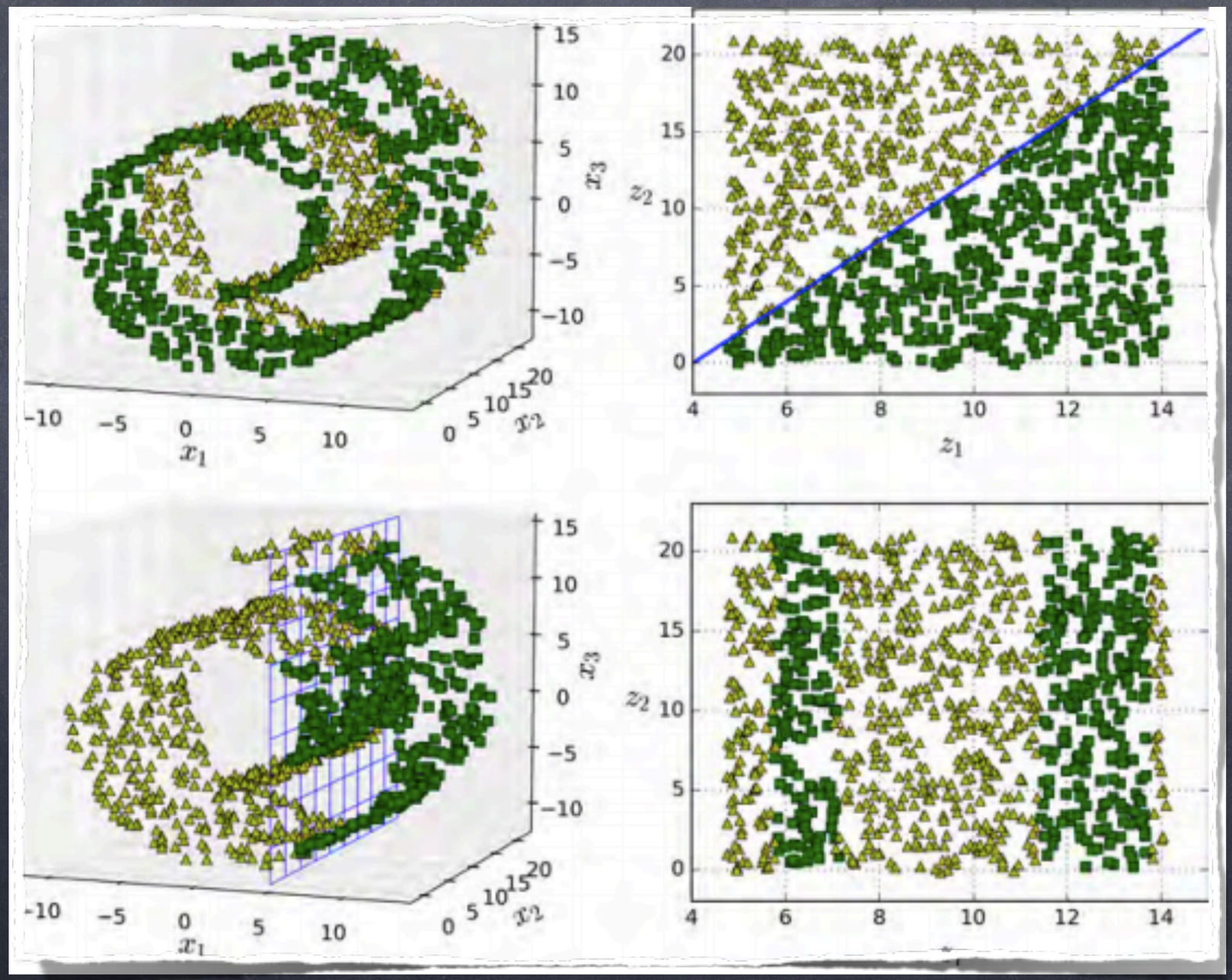
reduction



swiss roll data set

- projection cannot always solve the problem

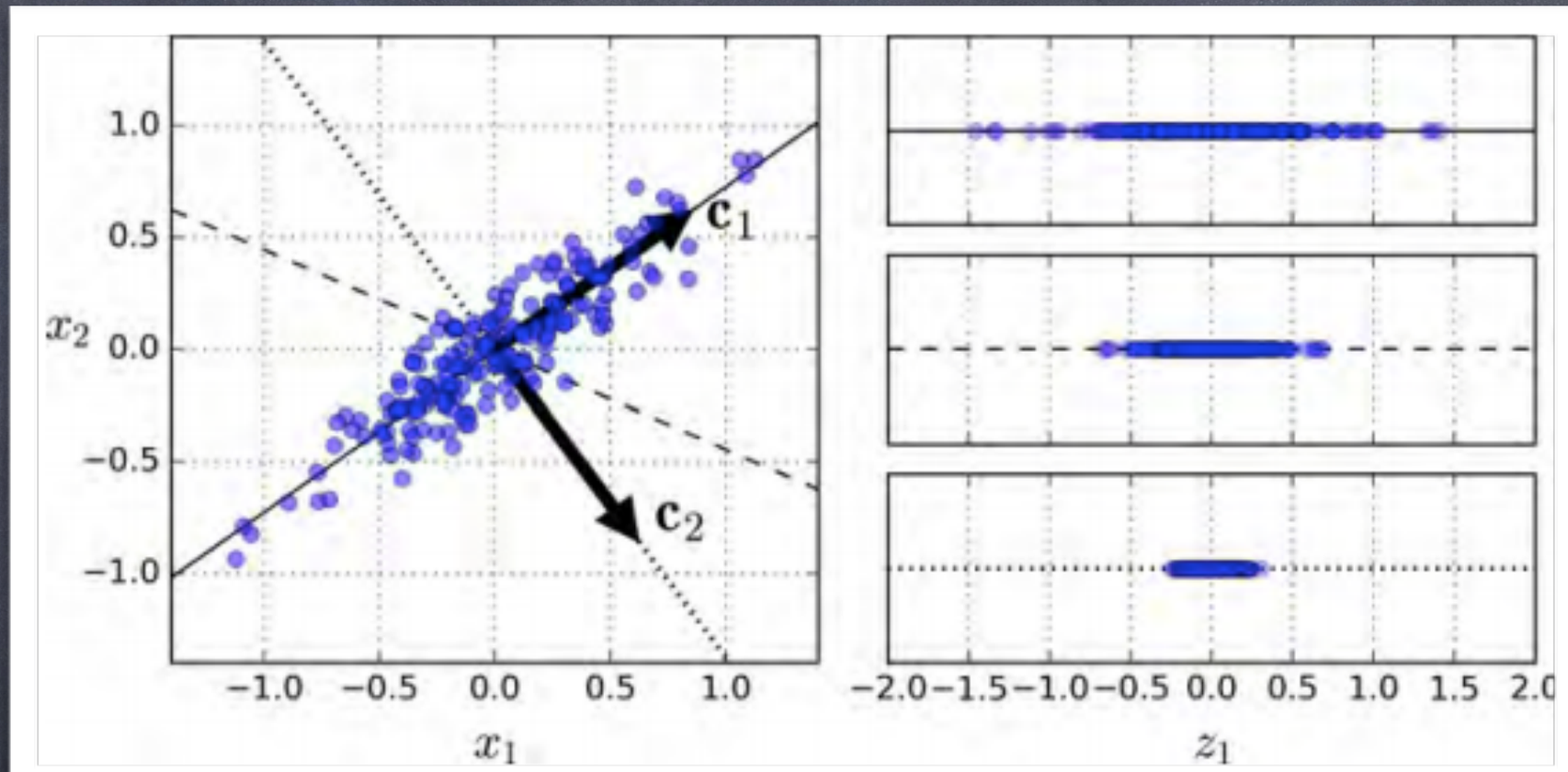
reduction



(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

PCA

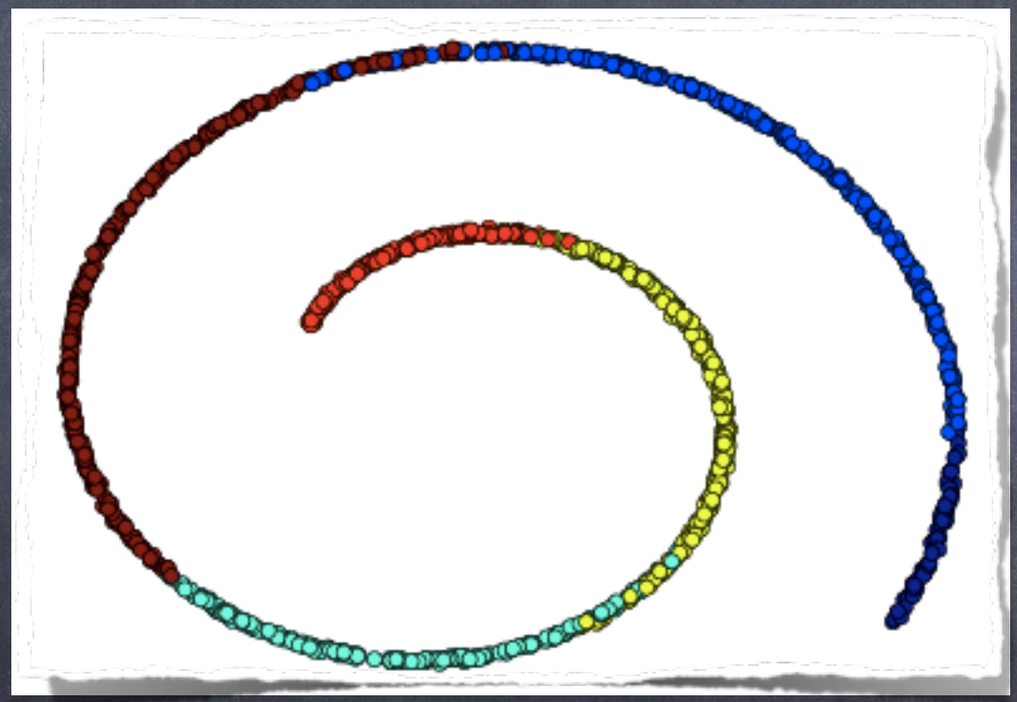
Principle Component Analysis



How many dimensions to take?

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

PCA
Principle Component Analysis



95% variance PCA of the Swiss roll

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

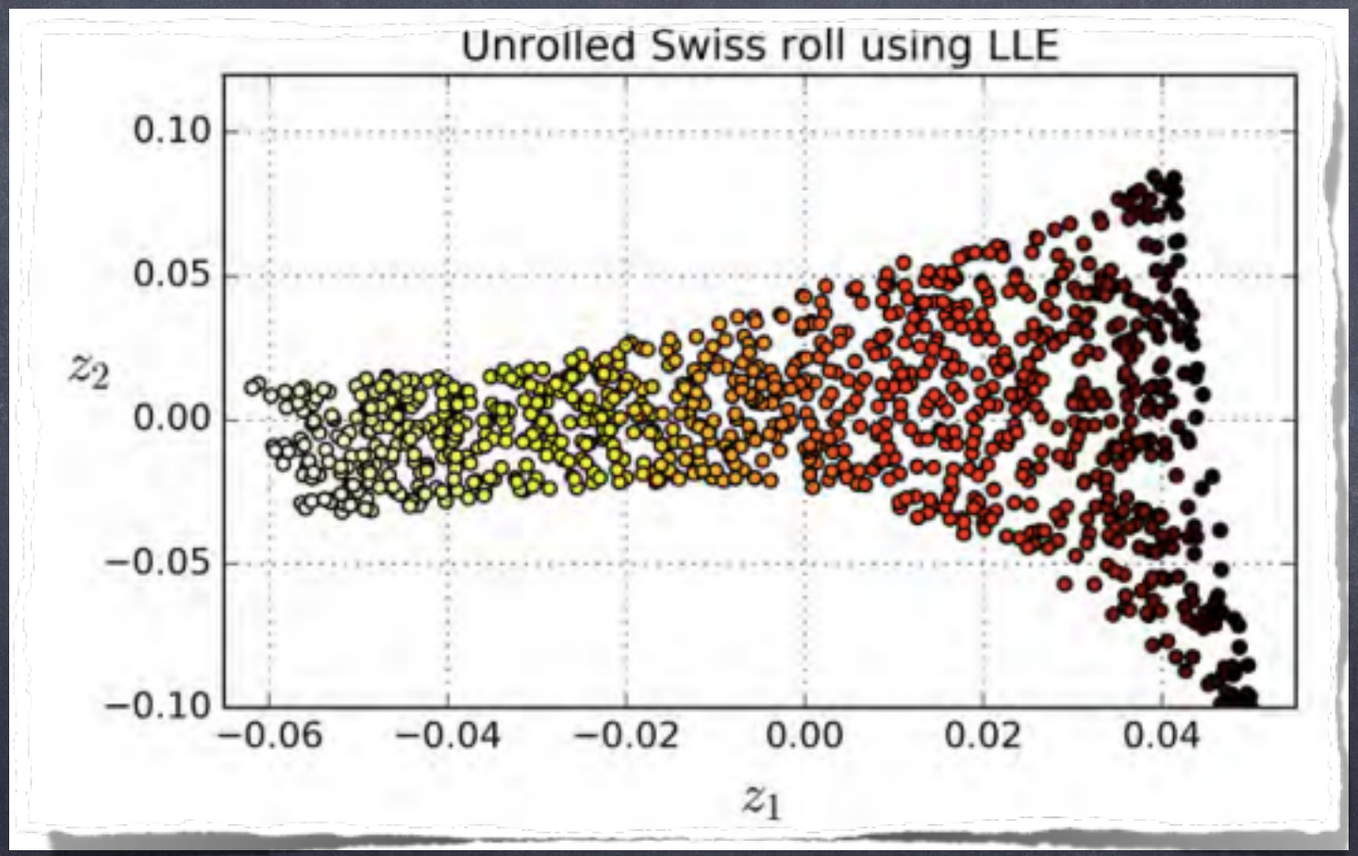
Incremental PCA

- you can split the training set into mini-batches
- IPCA takes one mini-batch at a time
- Useful for large training sets, or for applying PCA online (on the fly)

LLE

Locally linear Embedding

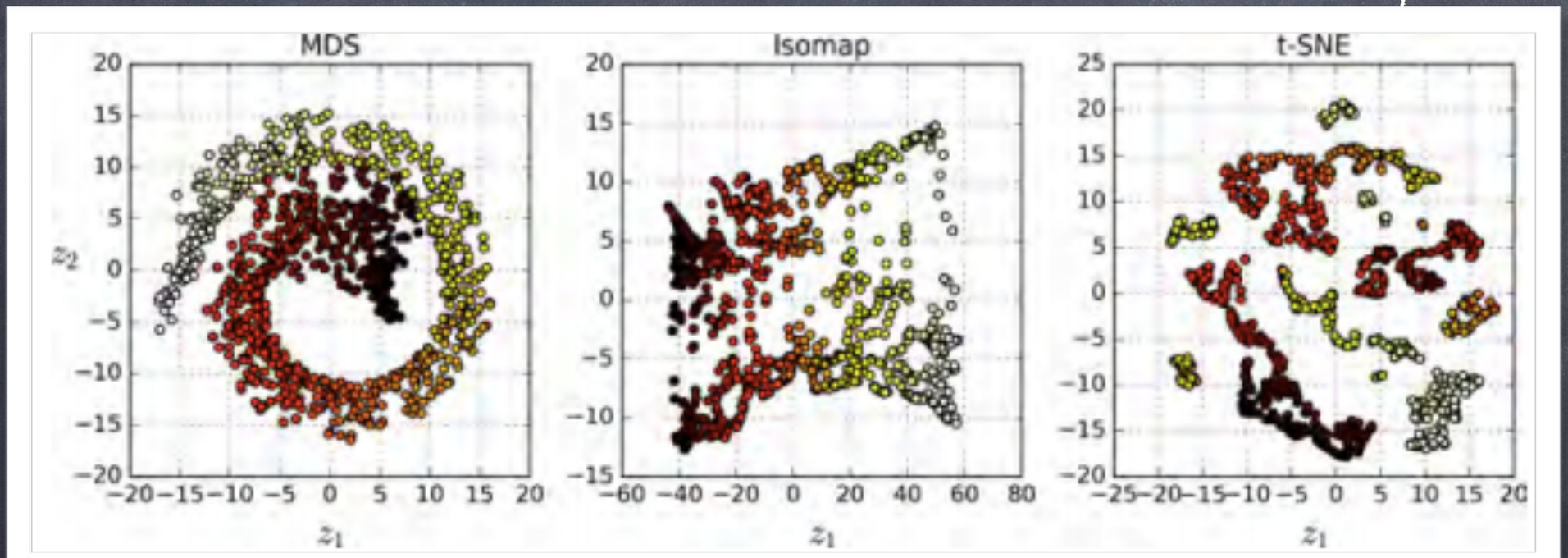
- Measures how each point linearly relates to its closest neighbours and looks for a low-dimensional representation of this relationship.



(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Other techniques

reducing the Swiss roll to 2D using various techniques



Multidimensional scaling

trying to reduce dimensionality while preserving the distances

Isomap

creates a graph by connecting each point to its nearest neighbour, tries to keep the geodesic distances

t-distributed stochastic neighbour embedding

reduces dimensionality while trying to keep similar points close and dissimilar points apart

(pictures: Hands-on machine learning with scikit-learn & tensorflow, Aurelien Geron, O'Reilly)

Question 1

- You have 5 classifiers, they are trained with the same data set, all of them give 95% classification performance on the same test data set.

- Is there any chance to get better performance by ensemble learning?

Question 2

- Your gradient boosting ensemble overfits the training data set.
- Should you increase or decrease the learning rate?