21-02-2020

CosmoML part 2

- Decision Trees, Random Forests, Support Vector Machines, k Nearest Neighbors
- Evaluation metrics
- Hyperparameter tuning+pipelines in sklearn

Michele Mancarella

ML workflow

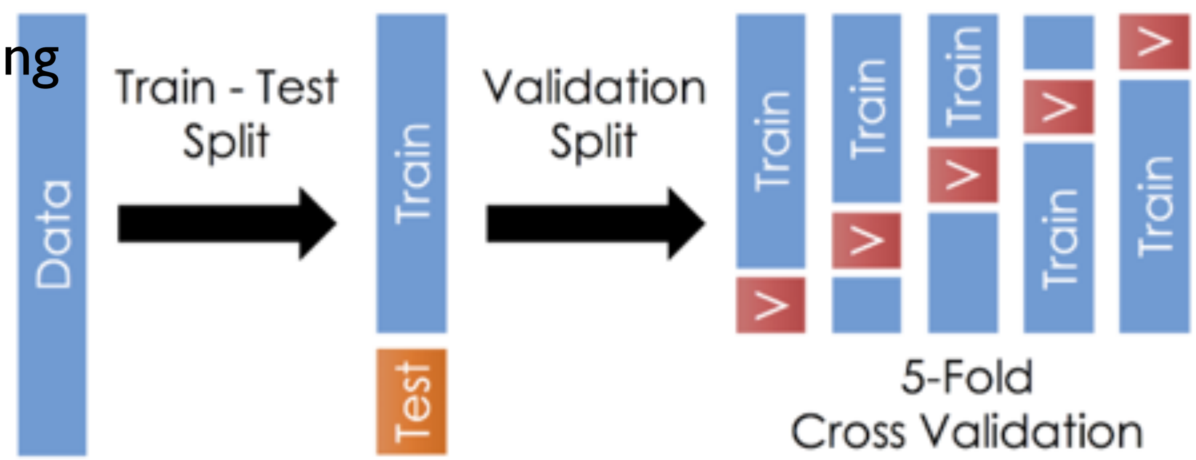
Data

Algorithm + Evaluation metric

Train

Split dataset

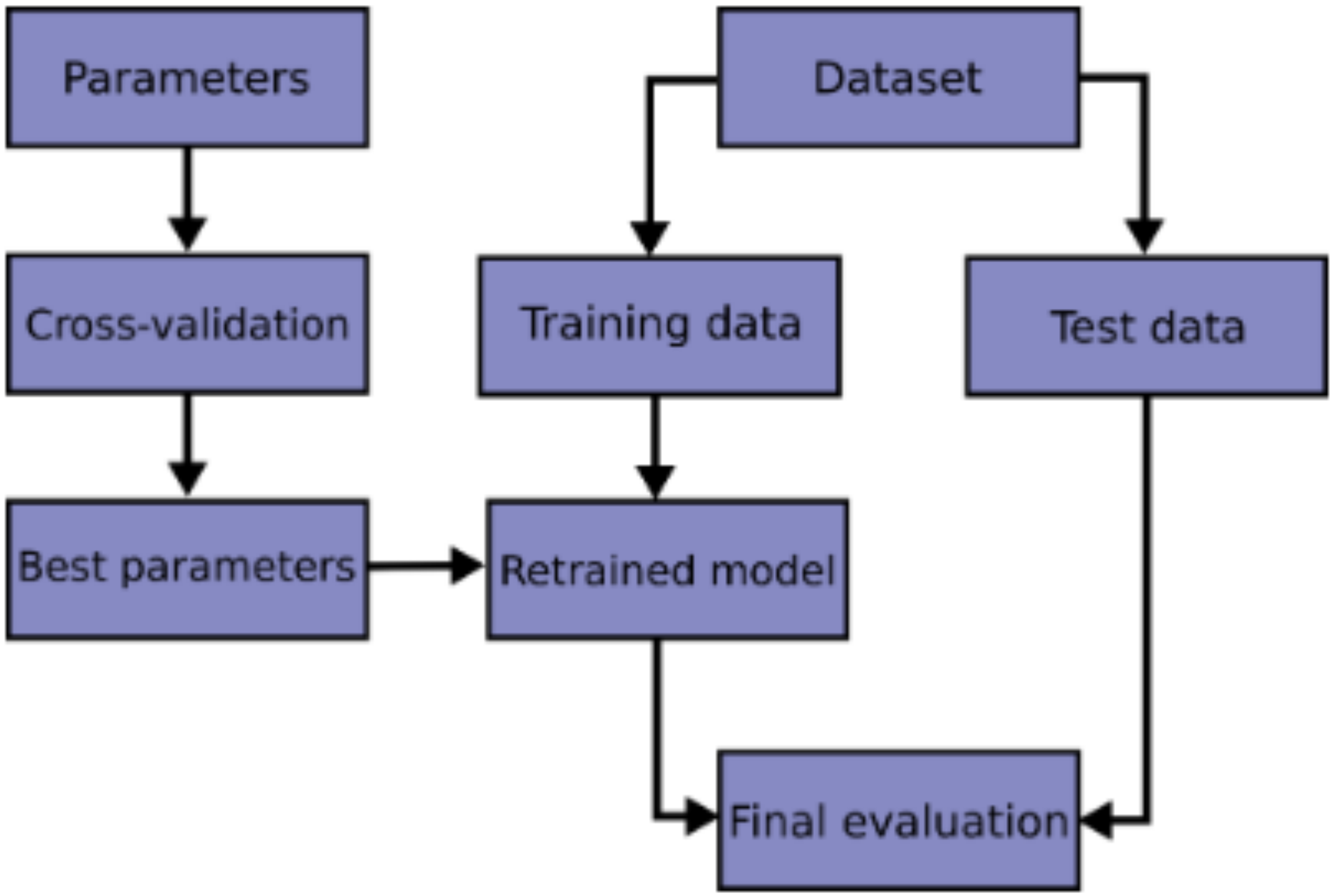
Hyperparameter tuning



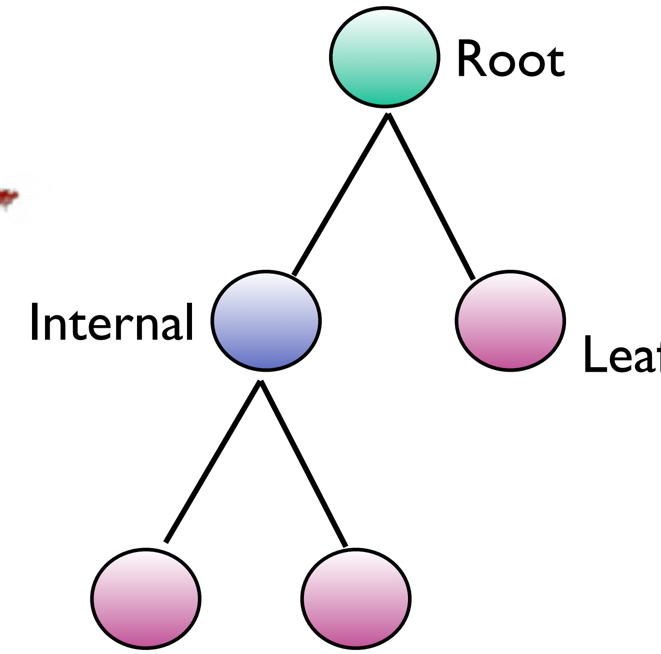
(Retrain on full test set)

Test

ML workflow


Parameters
Dataset
Cross-validation
Training data
Test data
Best parameters
Retrained model
Final evaluation

Decision trees



Tree = Oriented graph w. any two nodes connected by 1 edge

Algorithm 1 Decision Tree

Start at root node

repeat

$\forall$ partition $S(\theta)$, $\theta = (j, t_j)$ consisting of feature j and threshold t_j of data at node k:

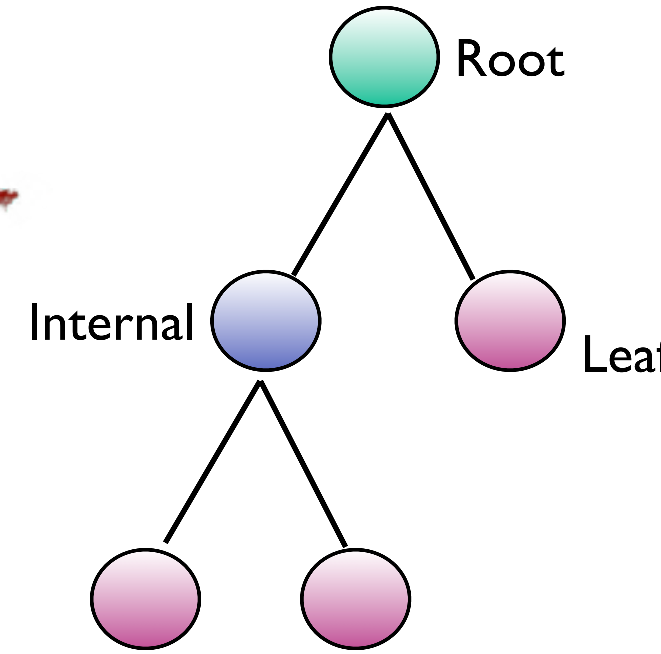
Compute *impurity* $I(S,\theta) = \frac{n_{\text{left}}}{N_k} H(S_{\text{left}}) + \frac{n_{\text{right}}}{N_k} H(S_{\text{right}})$

Choose the partition corresponding to $\hat{\theta} = \arg\min_\theta I(S,\theta)$

until Max depth is reached or $N_k = N_{min}$

$S_{\text{left}} =$ partition of data w. feature $j < t_j$, $\quad S_{\text{right}} = S \backslash S_{\text{left}}$

Decision trees



Gini $$H(x_k) = \sum_{i=1}^{N_{classes}} p_{ik}(1 - p_{ik})$$

$$p_{ik} = \text{fraction of points in node k in class i}$$

Entropy $$H(x_k) = - \sum_{i=1}^{N_{classes}} p_{ik} \log p_{ik}$$

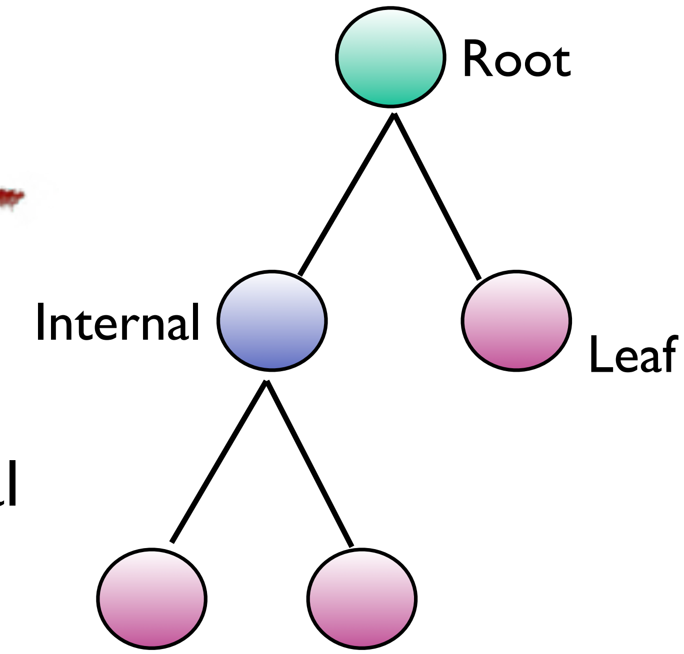
(For regression problems: MSE)

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.**DecisionTreeClassifier**(*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort='deprecated', ccp_alpha=0.0*) [source]

A decision tree classifier.

Decision trees



Advantages:

✦ No feature preparation - also w. mixed numerical&categorical
✦ "**White box**" ! (Visualizable, interpretable, simple). Feature importance:

$$\frac{n_k}{N}\left(I(S) - \frac{n_{\text{left}}}{N_k} I(S_{\text{left}}) - \frac{n_{\text{right}}}{N_k} I(S_{\text{right}})\right)$$

$$FI_{f_i} = \frac{\sum_{\text{j s.t. node j splits on i-th feature}} GI_j}{\sum_j GI_j}$$

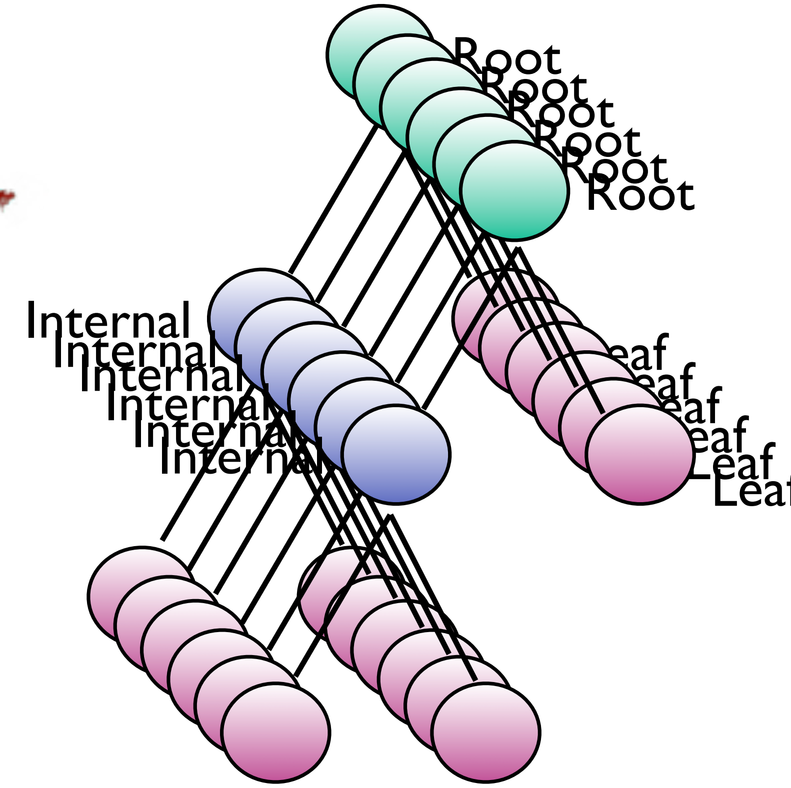
Disadvantages:

✦ Easily overfit ⟵ REGULARIZATION: max depth, max n. of leafs, min. sample split
✦ Handling of unbalanced classes
✦ Unstable to variations in training data
✦ Greedy ! (local minima)

Ensemble methods

✤ Random Forests - grow different trees w. bootstrap
✤ AdaBoost - re-weight error to reinforce points where classifier performs poorly (i.e. concentrate on difficult examples)
✤ Gradient Boosted Trees/XGBoost

Random Forests



Add randomness to prevent overfit (everywhere in ML, cfr dropout in NN)

(1) Grow different trees on bootstrap sample
(2) When splitting each node, partition using a random subset of features

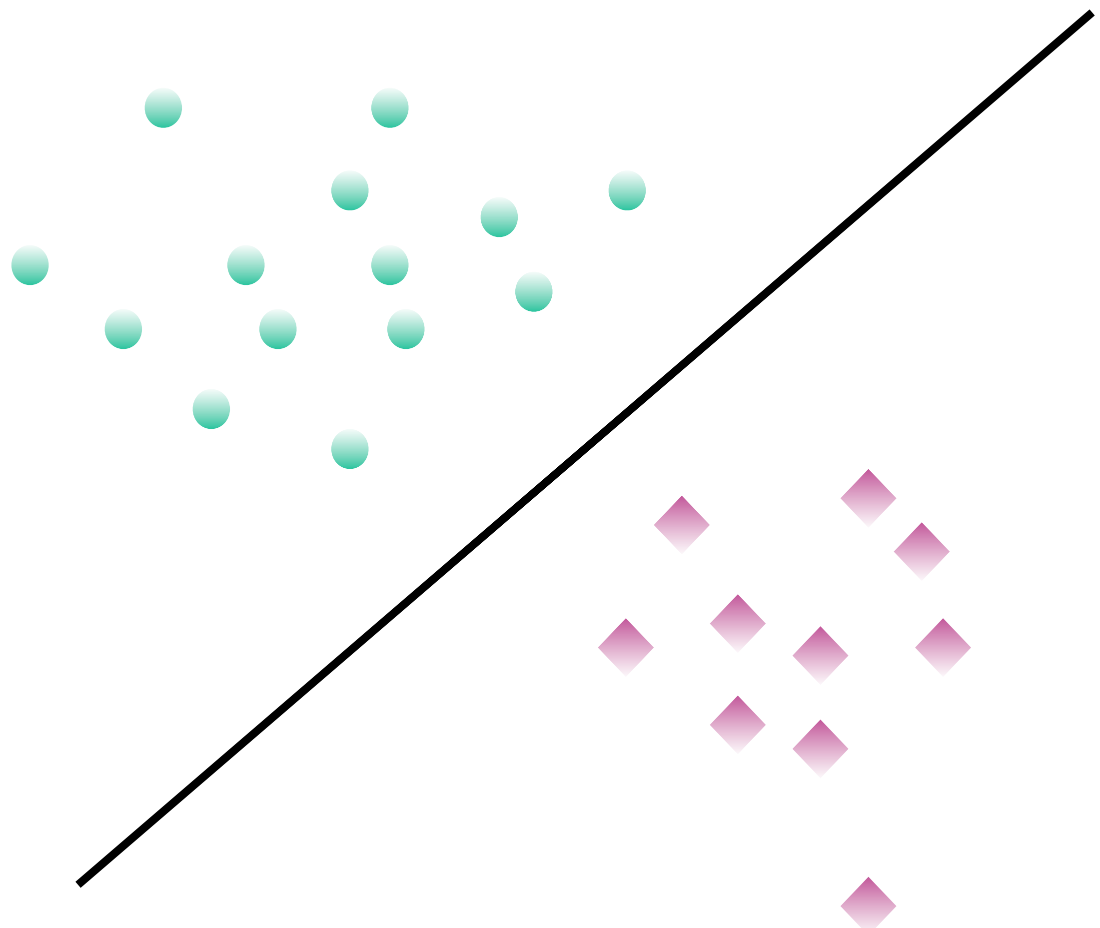
3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

class `sklearn.ensemble.`**`RandomForestClassifier`**(*n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

A random forest classifier.

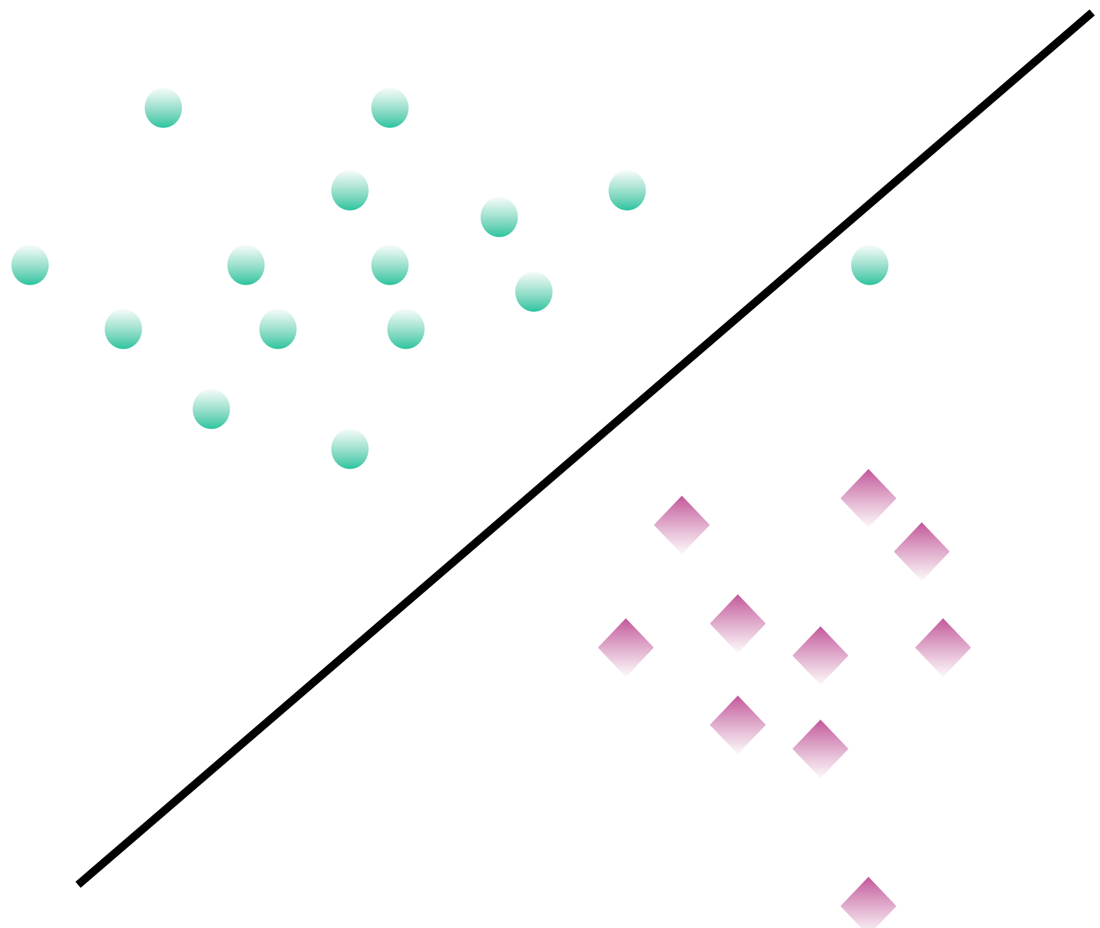
Support Vector Machines

Basic idea: find optimal separating hyperplane (then generalize to non-linear boundaries)



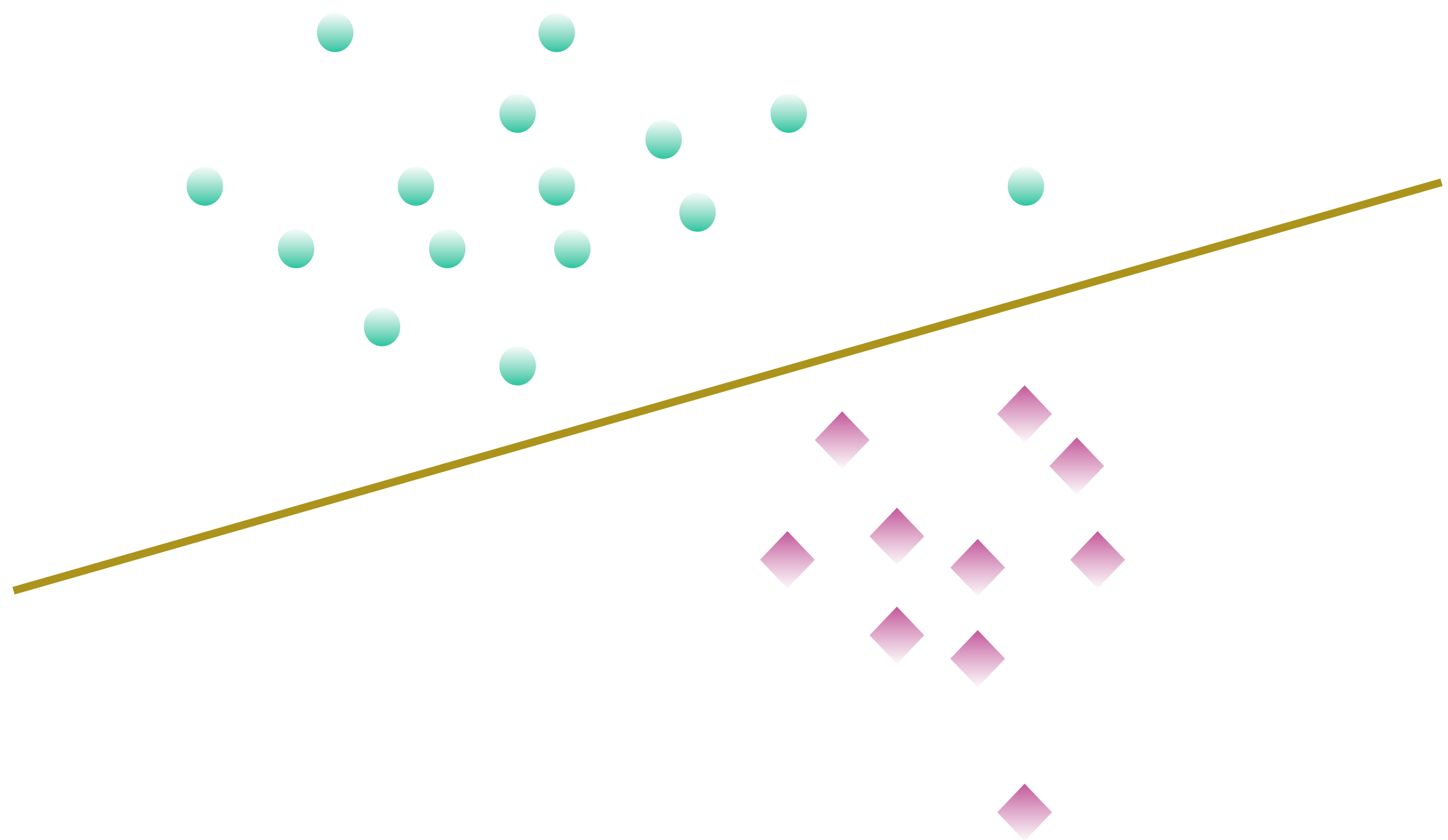
Support Vector Machines

Basic idea: find optimal separating hyperplane (then generalize to non-linear boundaries)



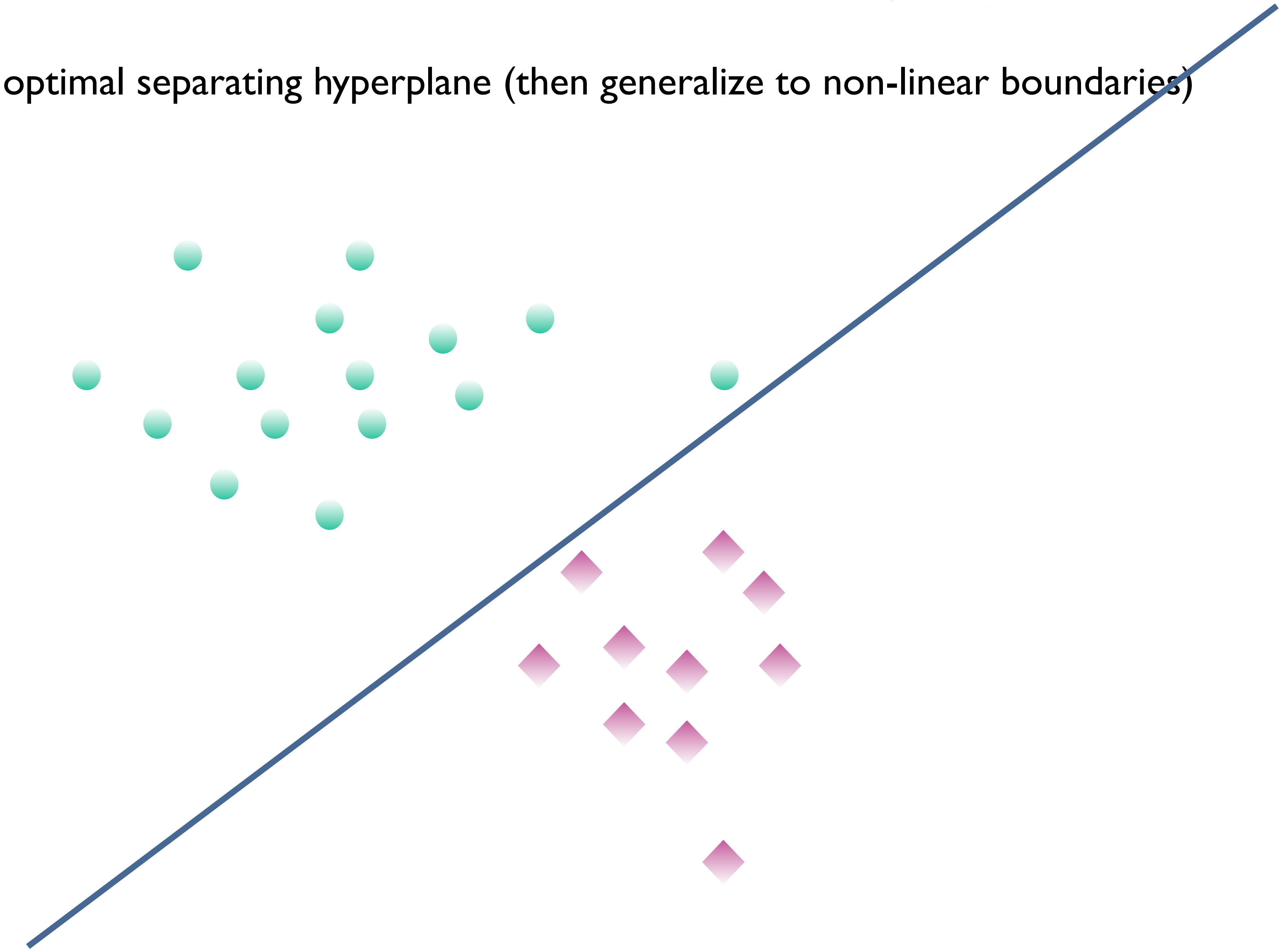
Support Vector Machines

Basic idea: find optimal separating hyperplane (then generalize to non-linear boundaries)



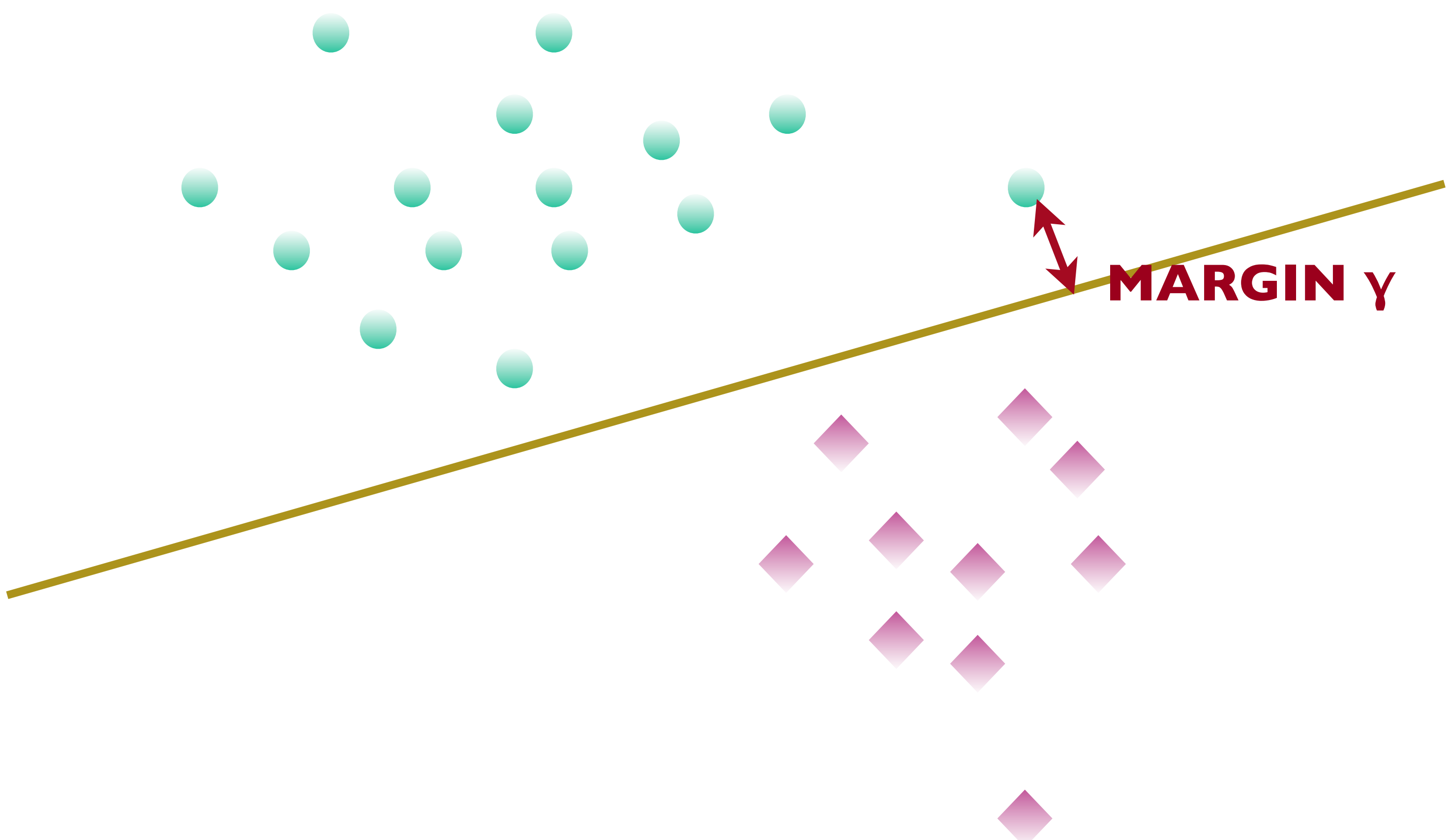
Support Vector Machines

Basic idea: find optimal separating hyperplane (then generalize to non-linear boundaries)



Support Vector Machines

Basic idea: find optimal separating hyperplane (then generalize to non-linear boundaries)



$$\gamma = \min_{i=1,\ldots,m} \gamma^{(i)}$$

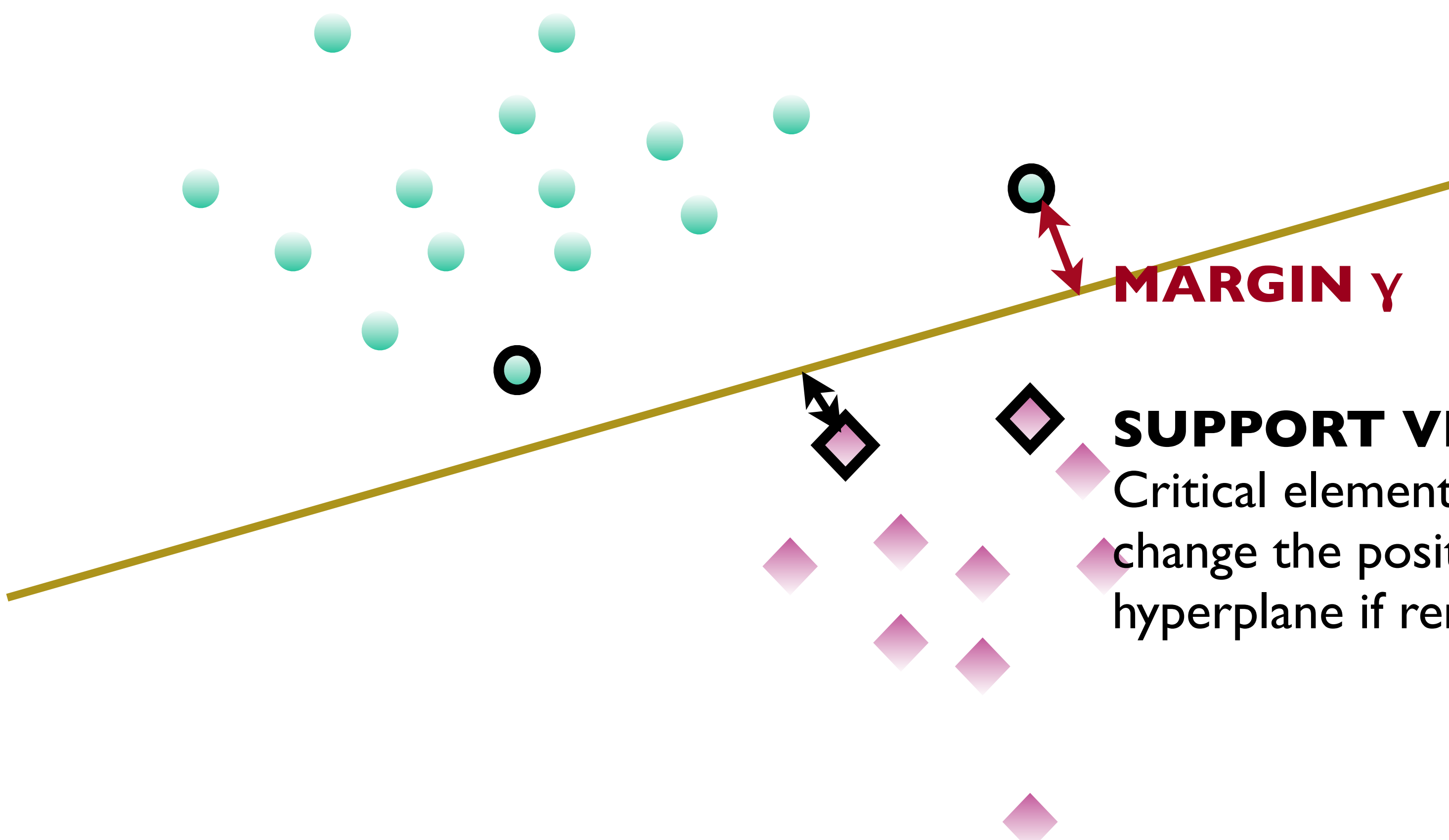
Intuitive formulation:

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq \gamma, \quad i = 1, \ldots, m \\ & ||w|| = 1. \end{aligned}$$

- Tough constraint to solve
- Far away points should not count

Support Vector Machines

Basic idea: find optimal separating hyperplane (then generalize to non-linear boundaries)



MARGIN γ $\gamma = \min_{i=1,\ldots,m} \gamma^{(i)}$

SUPPORT VECTORS

Critical elements of the set, those that would change the position of the separating hyperplane if removed

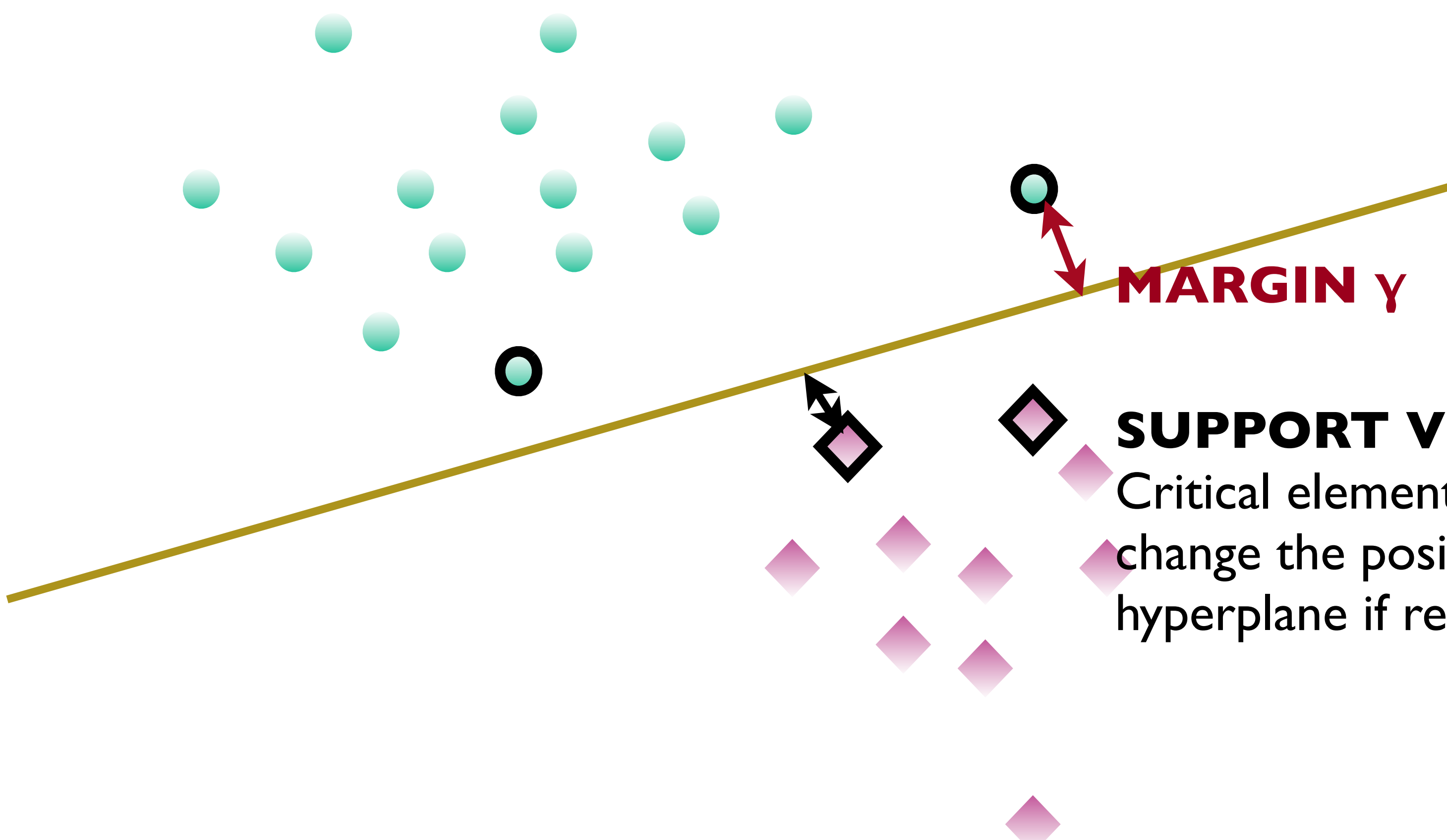
Less intuitive but optimal formulation:

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

- Use Lagrange multipliers α
- Solve the **dual** problem, i.e. maximize over α subject to relations implied by the constraints for **w** and **b** instead of maximizing over **w** and **b** subject to the constraint involving α

Support Vector Machines

Basic idea: find optimal separating hyperplane (then generalize to non-linear boundaries)



MARGIN γ $\gamma = \min_{i=1,\ldots,m} \gamma^{(i)}$

SUPPORT VECTORS
Critical elements of the set, those that would change the position of the separating hyperplane if removed

Less intuitive but optimal formulation:

scalar product

$$\max_\alpha \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle.$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, m$$

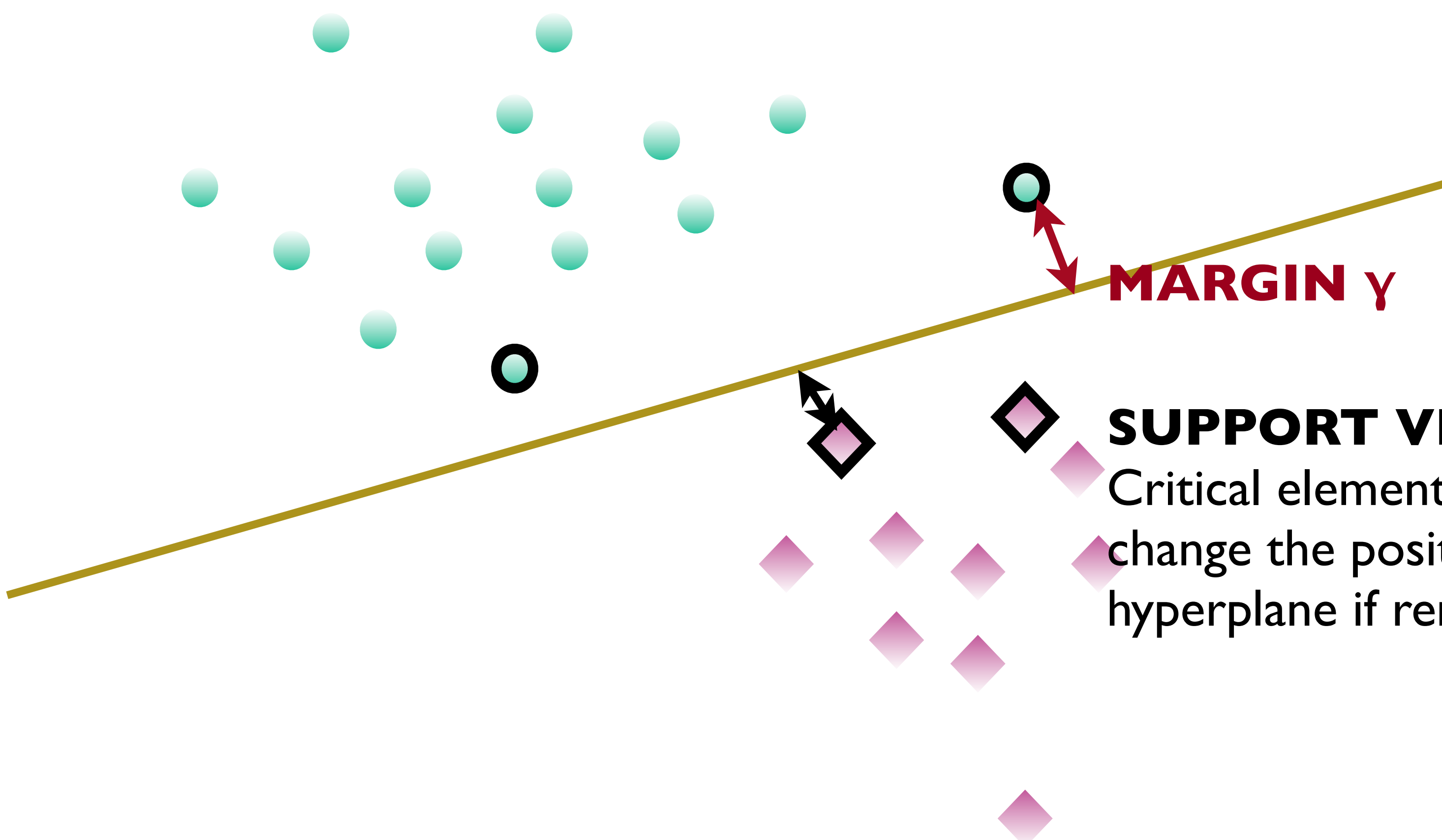
$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0,$$

Only true for SV !!

- Use Lagrange multipliers α
- Solve the **dual** problem, i.e. maximize over α subject to relations implied by the constraints for **w** and **b** instead of maximizing over **w** and **b** subject to the constraint involving α

Support Vector Machines

Basic idea: find optimal separating hyperplane (then generalize to non-linear boundaries)



MARGIN γ $\gamma = \min_{i=1,\ldots,m} \gamma^{(i)}$

SUPPORT VECTORS
Critical elements of the set, those that would change the position of the separating hyperplane if removed

Less intuitive but optimal formulation:

scalar product

$$\max_\alpha \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle.$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0,$$

Only true for SV !!

Non-linear boundaries

- Replace scalar product w. $\langle \phi(x), \phi(z) \rangle$ for any non-linear mapping
- Or directly define a KERNEL $K(x,z) = \phi(x)^T \phi(z)$

Support Vector Machines

Regularization:

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

Advantages:

- ✦ Nicely generalizes to non-linear boundaries, versatile
- ✦ Good for higher-dimensional problems
- ✦ Quadratic optimization problem

Disadvantages:

- ✦ No probability estimates!
- ✦ Don’t forget normalization !

k-Nearest Neighbors

Basic idea: assign as label the most frequent label among the closest k point (k nearest neighbors) in the feature space

- ✦ Not a minimization problem/nonparametric : simply stores instances of training data
- ✦ Can tune k and distance

Algorithm 1 kNN

for $i = 1, .., N_{new}$ **do** Compute distance $d(X_i, x_{new})$
end for
Find set S of points with k smallest distances $d(X_i, x_{new})$, $i \in S$
return Majority label in S

Advantages:

- ✦ Simple but effective
- ✦ Can handle complex boundaries

Disadvantages:

- ✦ No probability estimates!
- ✦ Don’t forget normalization !
- ✦ O(N^2) if you don’t use smart algorithms