Support Vector Machines

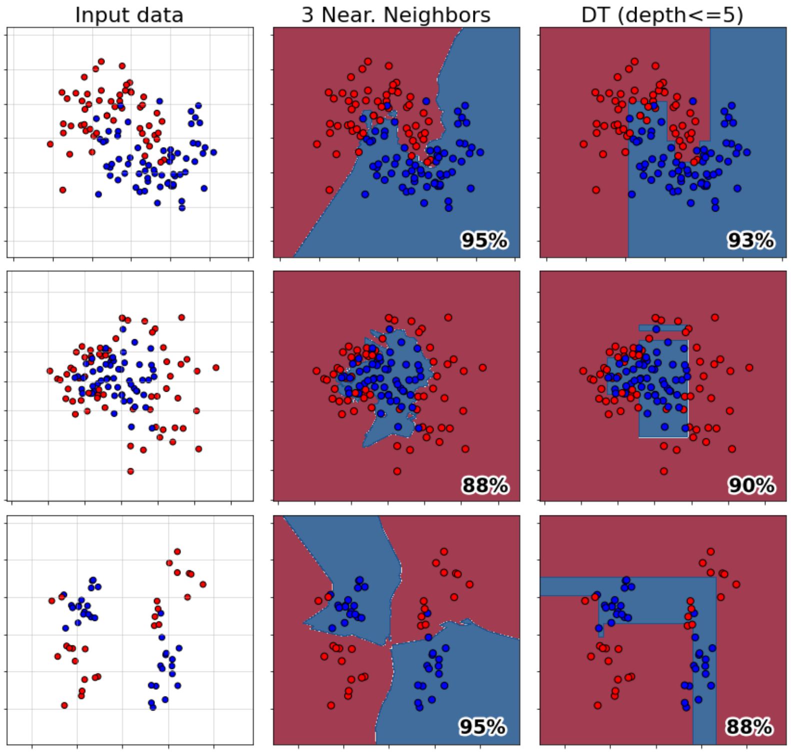
By Amir Feder & Itay Evron

Outline

- Hard SVM
- Soft SVM
- Kernel SVM

Decision boundaries

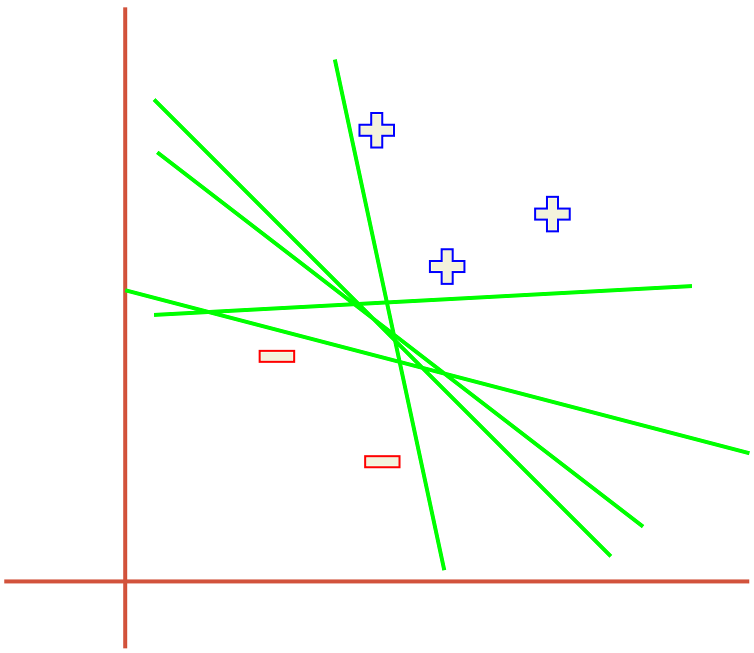
- Different models on 3 datasets.
- On the bottom right – the train accuracy.



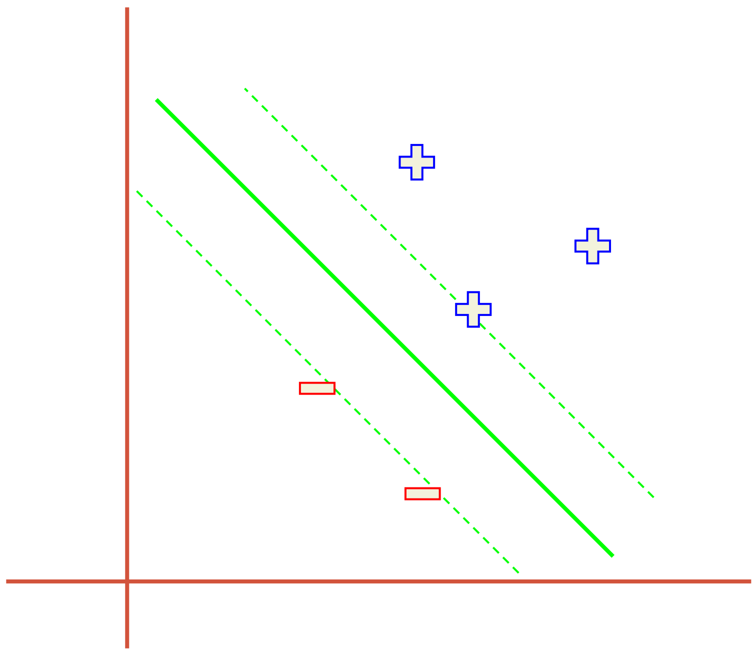
Based on: [sklearn's Classifier comparison]()

Linear separation

Infinitely many separators



Max-margin separator



Intuition: Hard SVM fits the “widest” possible strip between classes

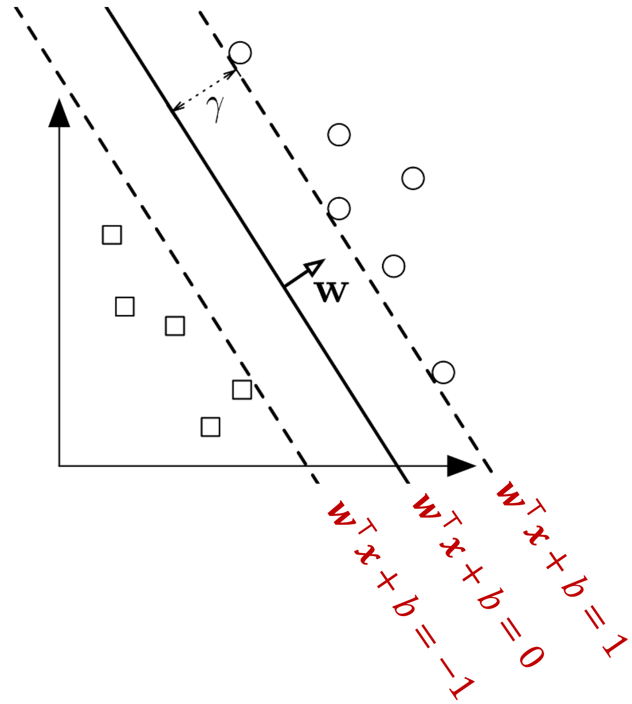
Hard SVM

Hard SVM: Recap

- We wanted a vector $\boldsymbol{w}$ that maximizes the margin.

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmax}} \; \underbrace{\min_{i\in[m]} \frac{|\boldsymbol{w}^\top \boldsymbol{x}_i|}{\|\boldsymbol{w}\|_2}}_{\text{margin, } \gamma(w;S)}$$

s.t. $y_i \cdot \boldsymbol{w}^\top \boldsymbol{x}_i \geq 0, \; \forall i \in [m]$



- We solved an equivalent optimization problem:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \; \|\boldsymbol{w}\|_2^2$$

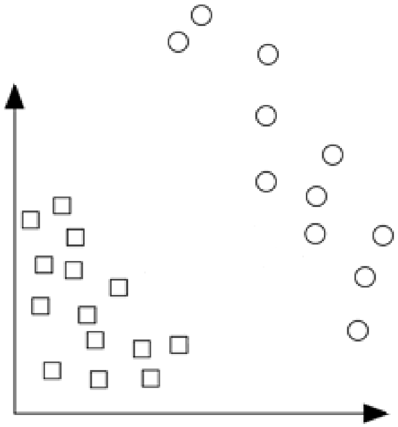
s.t. $y_i \cdot \boldsymbol{w}^\top \boldsymbol{x}_i \geq 1, \; \forall i \in [m]$

Classify **all** points correctly

with a margin at least ${}^{1}/_{\|\boldsymbol{w}\|_2^2}$

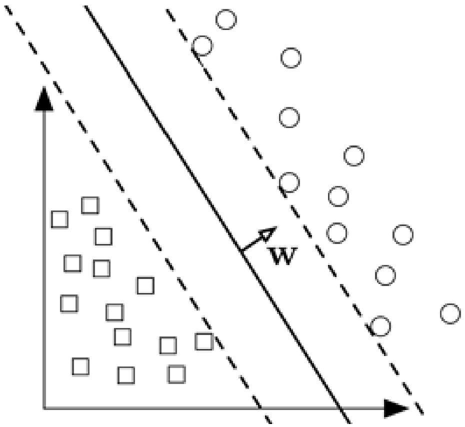
Sensitivity to outliers

Separable data

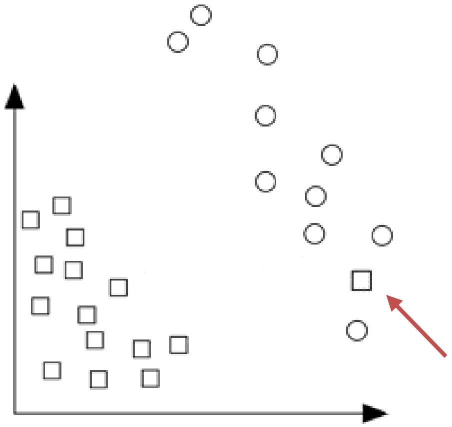


Hard SVM

Max-margin separator



One noised label



Hard SVM

?

Issues with Hard SVM

- Only works with linearly separable data
- Highly sensitive to outliers

- More flexible – Soft SVM:
  - Balances between max-margin and margin violations.

Soft SVM

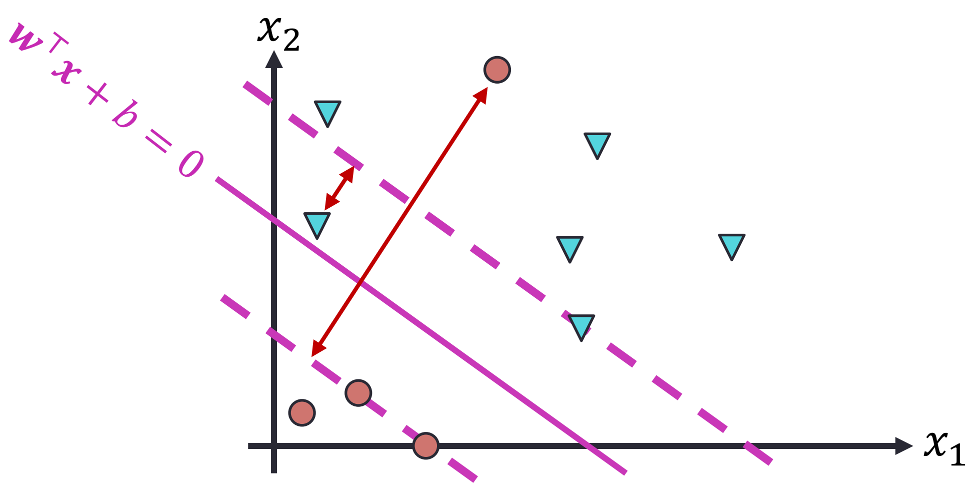
Soft SVM: Optimization problem

- Two conflicting objectives:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d, b\in\mathbb{R}}{\operatorname{argmin}} \ \lambda\|\boldsymbol{w}\|_2^2 + \frac{1}{m}\sum_{i\in[m]} \max\{0, 1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)\}$$

Complexity penalty (Regularization)
Hyperparameter λ controls the model complexity

Margin violation penalty
How much the i[th] example *violates* the margin



Soft SVM: Optimization problem

- Two conflicting objectives:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d, b\in\mathbb{R}}{\operatorname{argmin}} \ \lambda\|\boldsymbol{w}\|_2^2 + \frac{1}{m}\sum_{i\in[m]} \max\{0, 1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)\}$$

- Equivalently (convince yourselves), some sources formulate as:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d, b\in\mathbb{R}}{\operatorname{argmin}} \ \|\boldsymbol{w}\|_2^2 + C\sum_{i\in[m]} \max\{0, 1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)\}$$

Soft SVM: Separable case

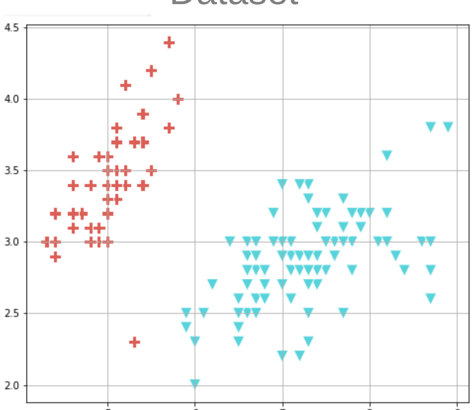
$$\underset{\boldsymbol{w}\in\mathbb{R}^d, b\in\mathbb{R}}{\operatorname{argmin}} \|\boldsymbol{w}\|_2^2 + C \sum_{i\in[m]} \max\{0, 1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)\}$$

- Given a linearly separable dataset, we train two models with: `C=1` and `1000`.

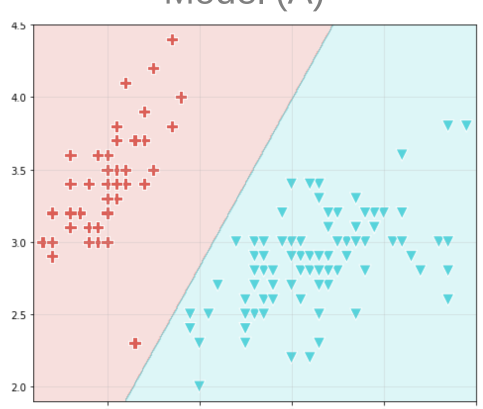
```
from sklearn.svm import LinearSVC
models = [LinearSVC(C=1, max_iter=10000), LinearSVC(C=1000, max_iter=10000)]
models = [h.fit(X, y) for h in models]
```

- Exercise: match the models to `C=1` and `1000`.

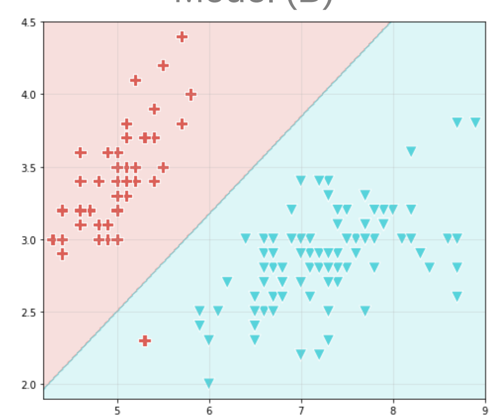
Dataset



Model (A)



Model (B)



Tuning the regularizer λ

- λ balances between model complexity and the violation penalty.

$$\underset{\boldsymbol{w}\in\mathbb{R}^d, b\in\mathbb{R}}{\operatorname{argmin}} \; \lambda \|\boldsymbol{w}\|_2^2 + \frac{1}{m}\sum\nolimits_{i\in[m]} \max\{0, 1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)\}$$

Complexity penalty (Regularization)
Hyperparameter λ controls the model complexity

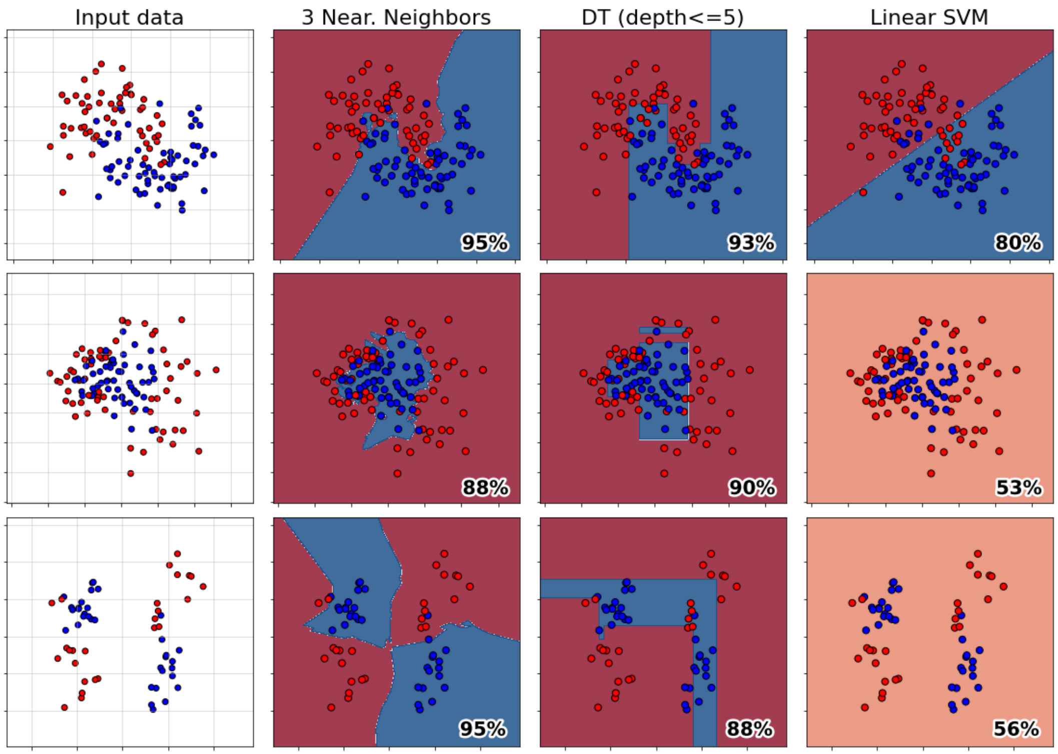
Margin violation penalty
How much the i^{th} example *violates* the margin

- Larger λ: more tolerance to violations $\Rightarrow$ lower complexity
- Smaller λ: less tolerance to violations $\Rightarrow$ overfitting
- As λ decreases, we get closer to Hard-SVM solution. Why?

  (if data is separable)

Decision boundaries

- Different models on 3 datasets.
- On the bottom right – the train accuracy.

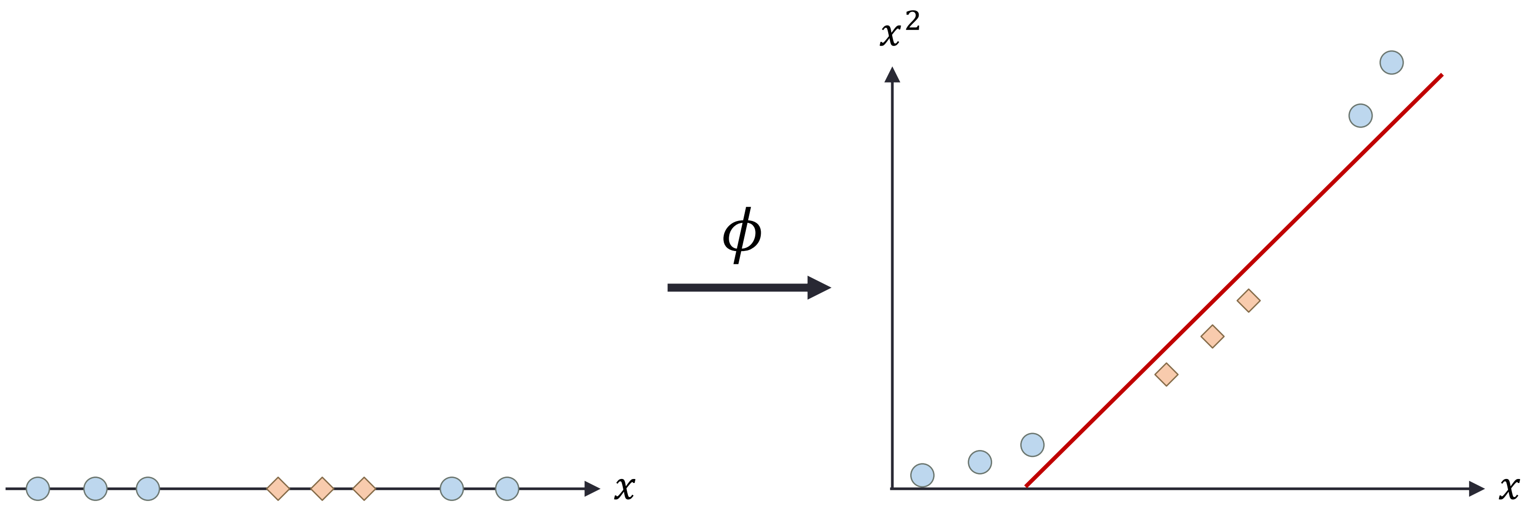


Based on: sklearn's Classifier comparison

Kernel SVM

Feature mappings

- Linear SVMs are efficient and often work well
- However, many problems are not linearly separable
    - One approach is to use a feature mapping ϕ to add more features:

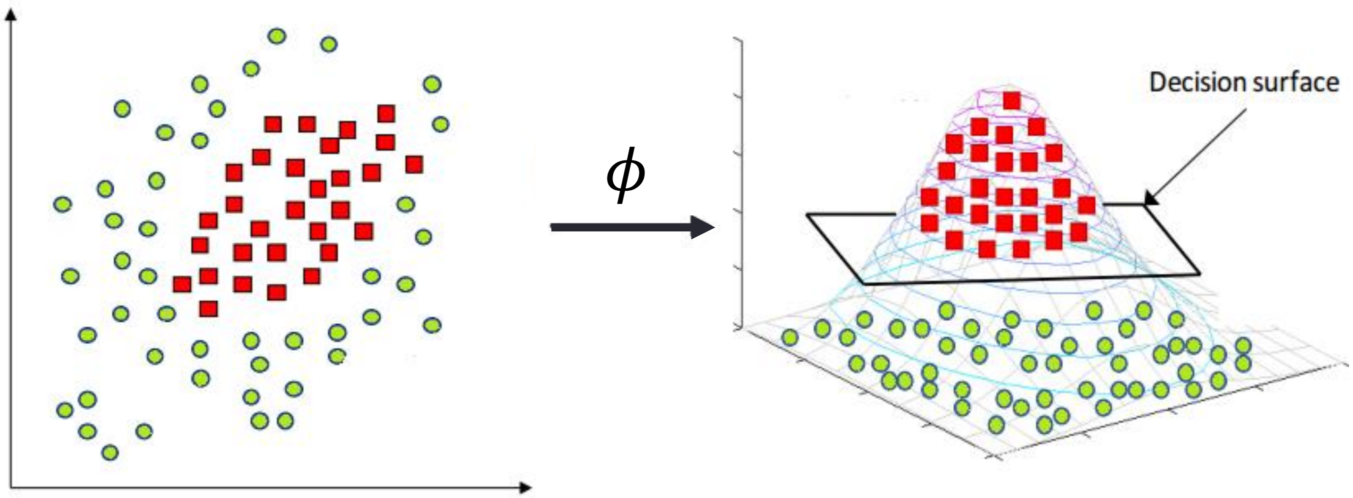


Not linearly separable: x

Linearly separable: $\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$

Feature mappings

- Linear SVMs are efficient and often work well
- However, many problems are not linearly separable
  - One approach is to use a feature mapping ϕ to add more features:



The polynomial mapping

- The 2nd-deg. polynomial mapping: $\phi\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ \sqrt{2}u_1 \\ \sqrt{2}u_2 \\ \sqrt{2}u_1u_2 \\ u_1^2 \\ u_2^2 \end{bmatrix}$
- Creates decision boundaries of the form:

$$0 \le \boldsymbol{w}^\top \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = w_1 + \sqrt{2}w_2x_1 + \sqrt{2}w_3x_2 + \sqrt{2}w_4x_1x_2 + w_5x_1^2 + w_6x_2^2$$

- Let us visualize it

What shapes can we create?

linear, circles, ellipsis, hyperbolas

The polynomial mapping

- The 2nd-deg. polynomial mapping: $\phi\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ \sqrt{2}u_1 \\ \sqrt{2}u_2 \\ \sqrt{2}u_1 u_2 \\ u_1^2 \\ u_2^2 \end{bmatrix}$
- Creates decision boundaries of the form:

$$0 \le \boldsymbol{w}^\top \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = w_1 + \sqrt{2}w_2 x_1 + \sqrt{2}w_3 x_2 + \sqrt{2}w_4 x_1 x_2 + w_5 x_1^2 + w_6 x_2^2$$

- Let us visualize it
- Works great with ML algorithms
- Higher polynomials can fit more complex data
  - But more features might cause computational (and statistical) problems

The kernel trick

Hard-SVM

$$\operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} \|\boldsymbol{w}\|_2^2$$
$$\text{s.t.} \quad y_i \cdot \boldsymbol{w}^\top \boldsymbol{x}_i \geq 1, \quad \forall i \in [m]$$

Feature mapping

$$\operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^{d'}} \|\boldsymbol{w}\|_2^2$$
$$\text{s.t.} \quad y_i \cdot \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1, \quad \forall i \in [m]$$

Dual problem uses only inner products

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}_+^m} \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^\top \boldsymbol{x}_j$$

Kernel trick

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}_+^m} \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \underbrace{\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)}_{=K(\boldsymbol{x}_i,\boldsymbol{x}_j)}$$

- If $d' \gg d$, optimization is expensive.
- The kernel trick: solve SVM for any feature mapping ϕ, by simply computing $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for every two training samples.

Valid kernels

- A kernel must hold $K(\boldsymbol{u}, \boldsymbol{v}) = \phi(\boldsymbol{u})^\top \phi(\boldsymbol{v})$ for some feature mapping ϕ.

$u: \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{bmatrix} u_1 & u_2 \end{bmatrix} = u_1^2 + 2u_1u_2 + u_2^2$

- Prove: the 2nd-deg. polynomial mapping $\phi\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ \sqrt{2}u_1 \\ \sqrt{2}u_2 \\ \sqrt{2}u_1u_2 \\ u_1^2 \\ u_2^2 \end{bmatrix}$

1, $2u_1v_1$, $2u_2v_2$, $2u_1u_2v_1v_2$, $u_1^2v_1^2$, $u_2^2v_2^2$

$u^2v^2 + 2uv + 1$

$(u^2, \sqrt{2}u, 1)$

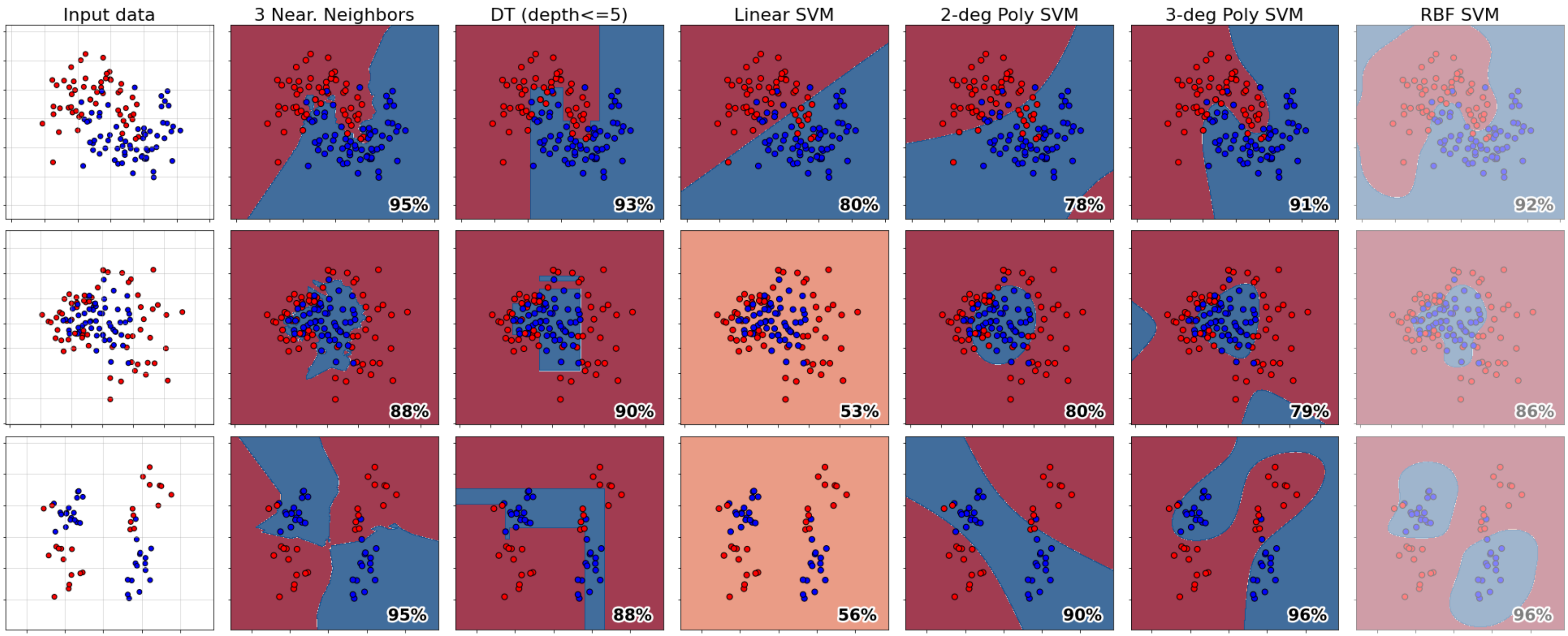
defines a (polynomial) kernel $K(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^\top \boldsymbol{v} + 1)^2 = \phi(\boldsymbol{u})^\top \phi(\boldsymbol{v})$.

  - Computation shortcut:
    - Computing $\phi(\boldsymbol{u}), \phi(\boldsymbol{v})$: $\mathcal{O}(d^2)$
    - Computing $\phi(\boldsymbol{u})^\top \phi(\boldsymbol{v})$ directly: $\mathcal{O}(d^2)$
    - Computing $(\boldsymbol{u}^\top \boldsymbol{v} + 1)^2$: $\mathcal{O}(d)$

  - Can now classify using all its monomials

Decision boundaries

- Different models on 3 datasets.
- On the bottom right – the train accuracy.



(in Major HW2)

Based on: sklearn's Classifier comparison

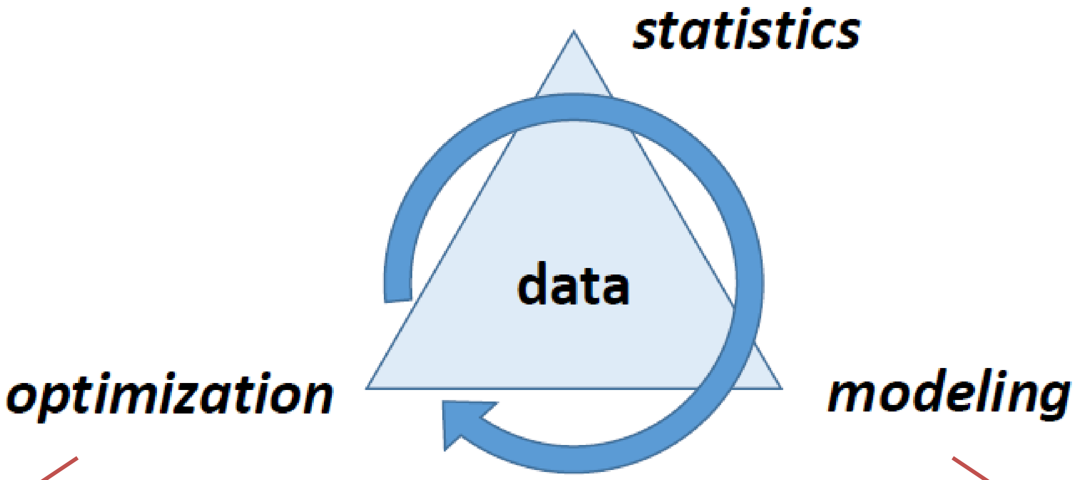
Which should we use?

- Rule of thumb: try the **linear kernel** first
  - Especially if training set is large (dual becomes expensive!)
  - In sklearn, `LinearSVC` is much faster than `SVC(kernel="linear")`

- If training set not too large, try **Gaussian RBF**

- If not satisfied, try others

Summary



- Heavy optimization and math
  - Reread lecture if needed
- Global convergence
- Primal vs. Dual
- Struggles with higher dimensions and/or many samples

- SVM is flexible, high-performing
  - Linear and nonlinear
- Controlling model complexity
  - Through hyperparameter λ or C
  - Kernel choice