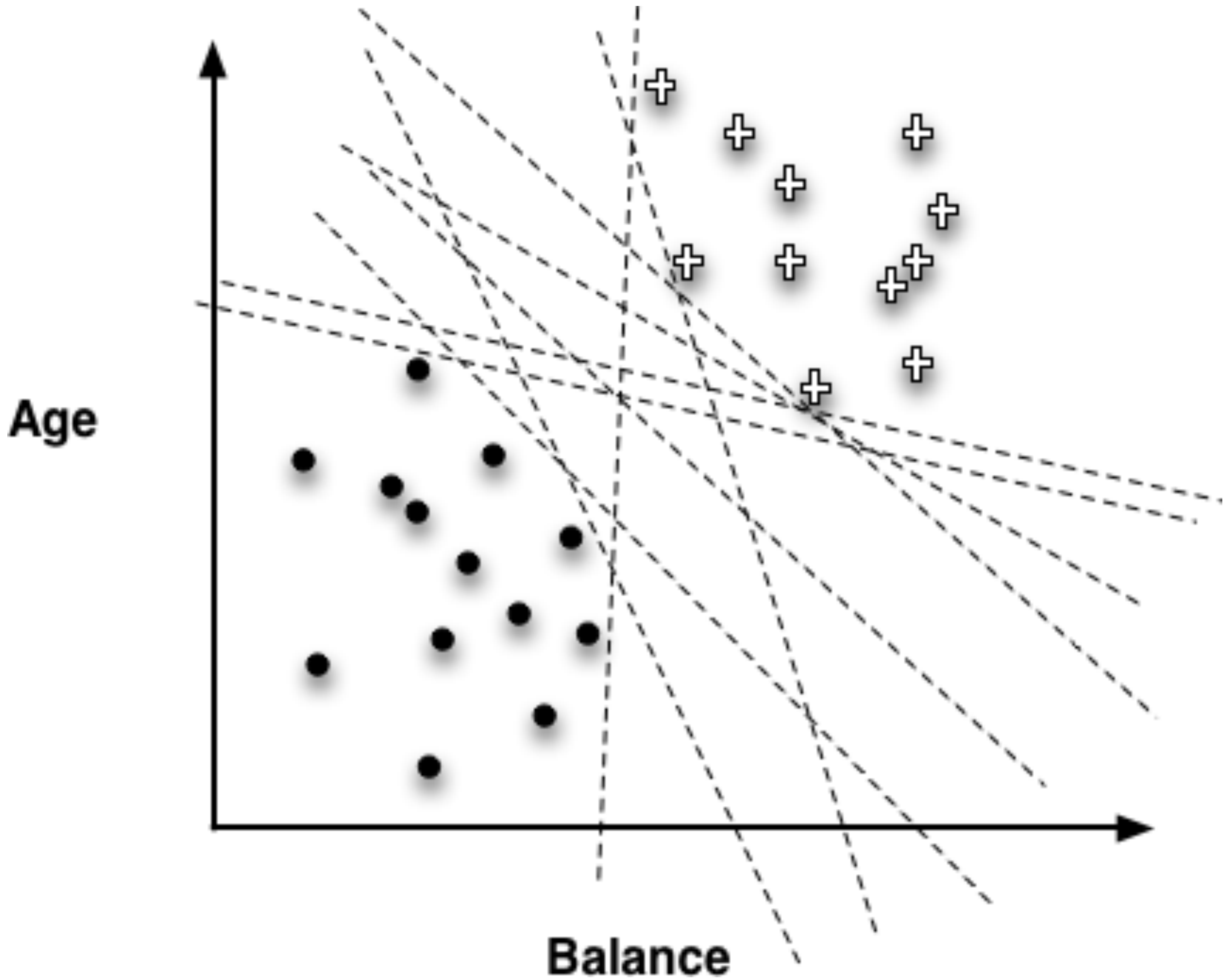
Choosing the "best" line



Objective Functions

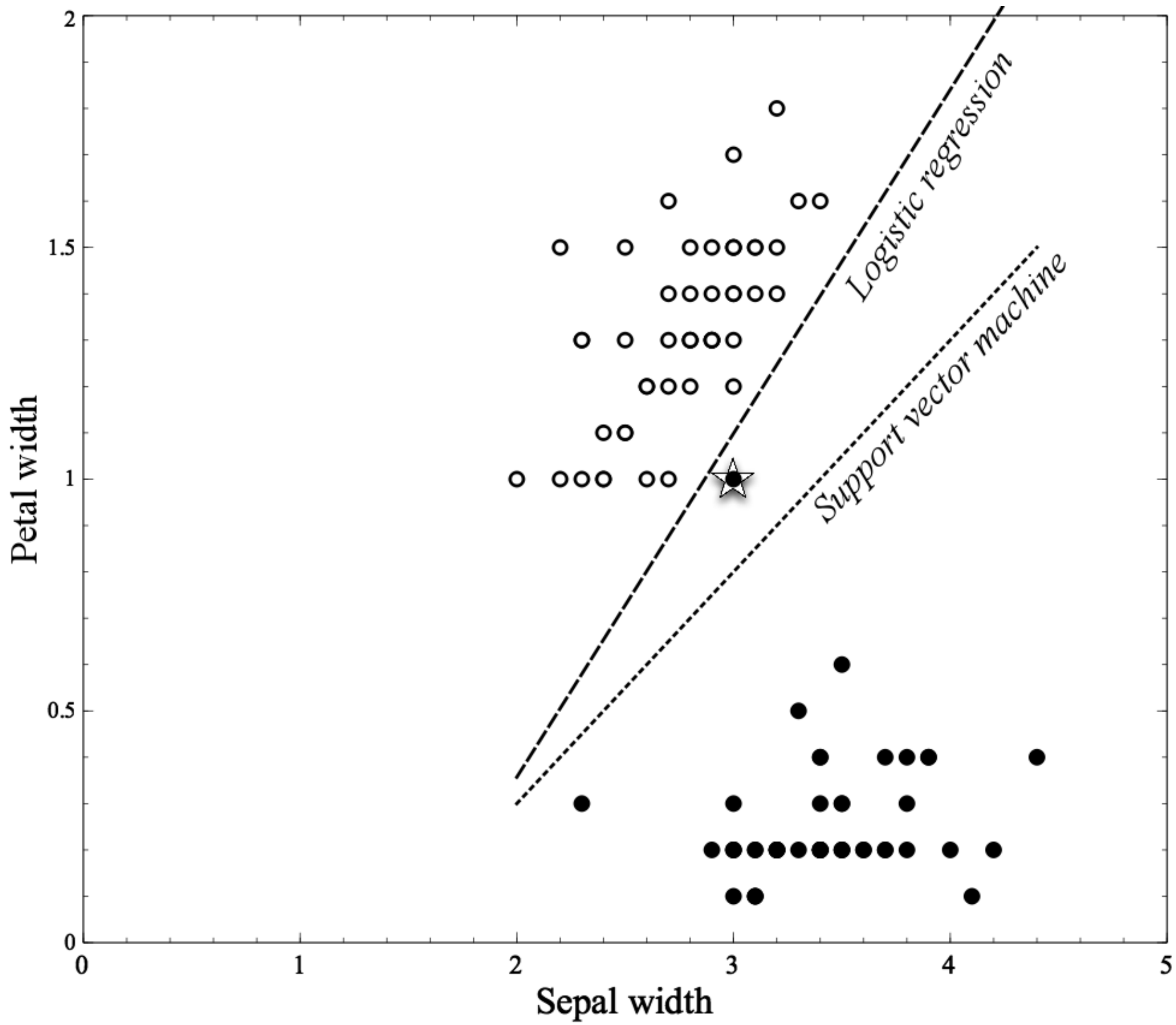
- "Best" line depends on the **objective (loss) function**
  - Objective function should represent our goal

- A loss function determines how much penalty should be assigned to an instance based on the error in the model's predicted value

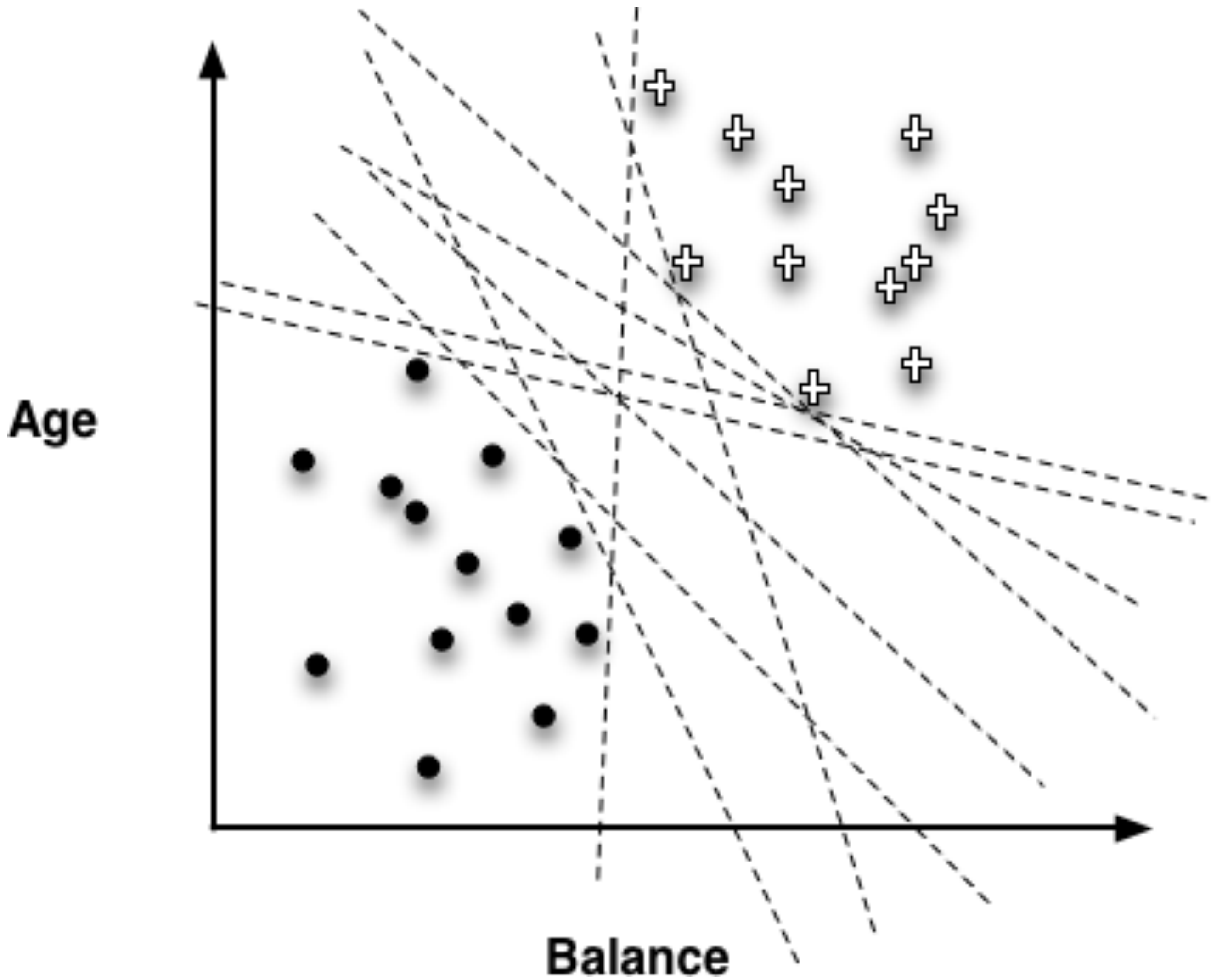
- Examples of objective (or loss) functions:
  - $\lambda(y; x) = |y - f(x)|$
  - $\lambda(y; x) = \left(y - f(x)\right)^2$ [convenient mathematically – linear regression]
  - $\lambda(y; x) = I(y \neq f(x))$

- **Linear regression**, **logistic regression**, and **support vector machines** are all very similar instances of our basic fundamental technique:
  - The key difference is that each uses **a different objective function**

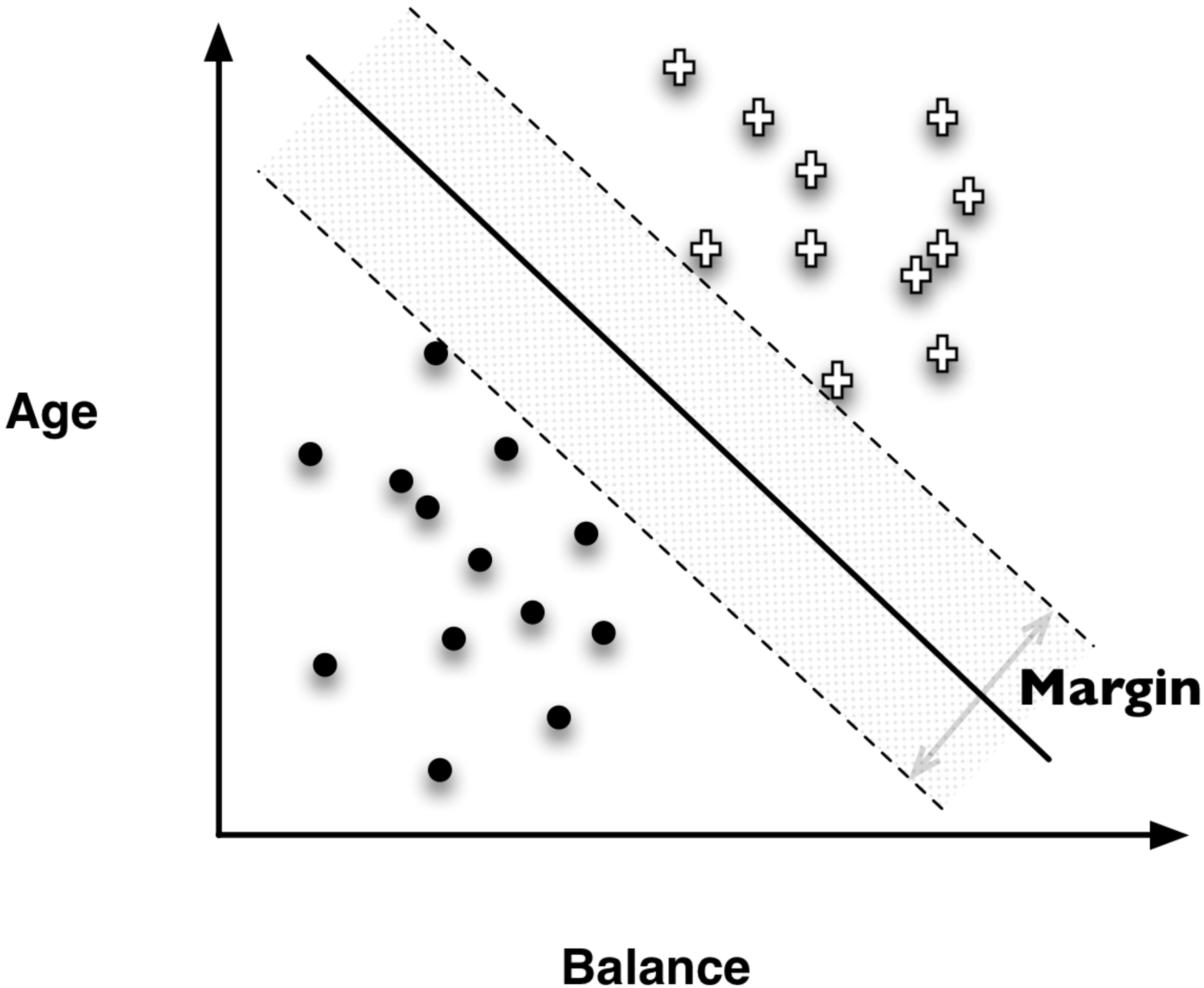
Classifying Flowers



Choosing the “best” line



Support Vector Machines (SVMs)



Support Vector Machines (SVMs)

- Linear Discriminants
- Effective
- Use “hinge loss”
- Also, non-linear SVMs

Hinge Loss functions

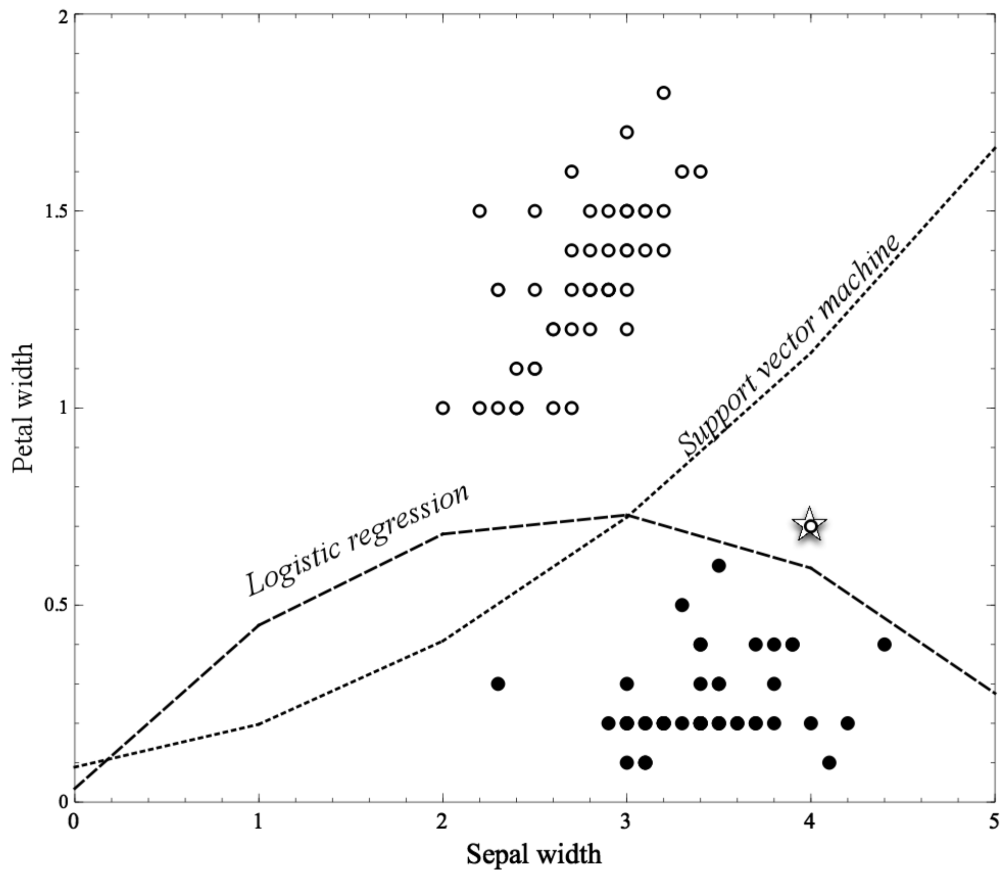
- Support vector machines use **hinge loss**
- Hinge loss incurs no penalty for an example that is not on the wrong side of the margin
- The hinge loss only becomes positive when an example is on the wrong side of the boundary and beyond the margin
  - Loss then increases linearly with the example's distance from the margin
  - Penalizes points more the farther they are from the separating boundary

Loss Functions

- **Zero-one loss** assigns a loss of zero for a correct decision and one for an incorrect decision
- **Squared error** specifies a loss proportional to the square of the distance from the boundary
  - Squared error loss usually is used for numeric value prediction (regression), rather than classification
  - The squaring of the error has the effect of greatly penalizing predictions that are grossly wrong

Non-linear Functions

- **Linear functions can actually represent nonlinear models**, if we include more complex features in the functions



Non-linear Functions

- Using “higher order” features is just a “trick”

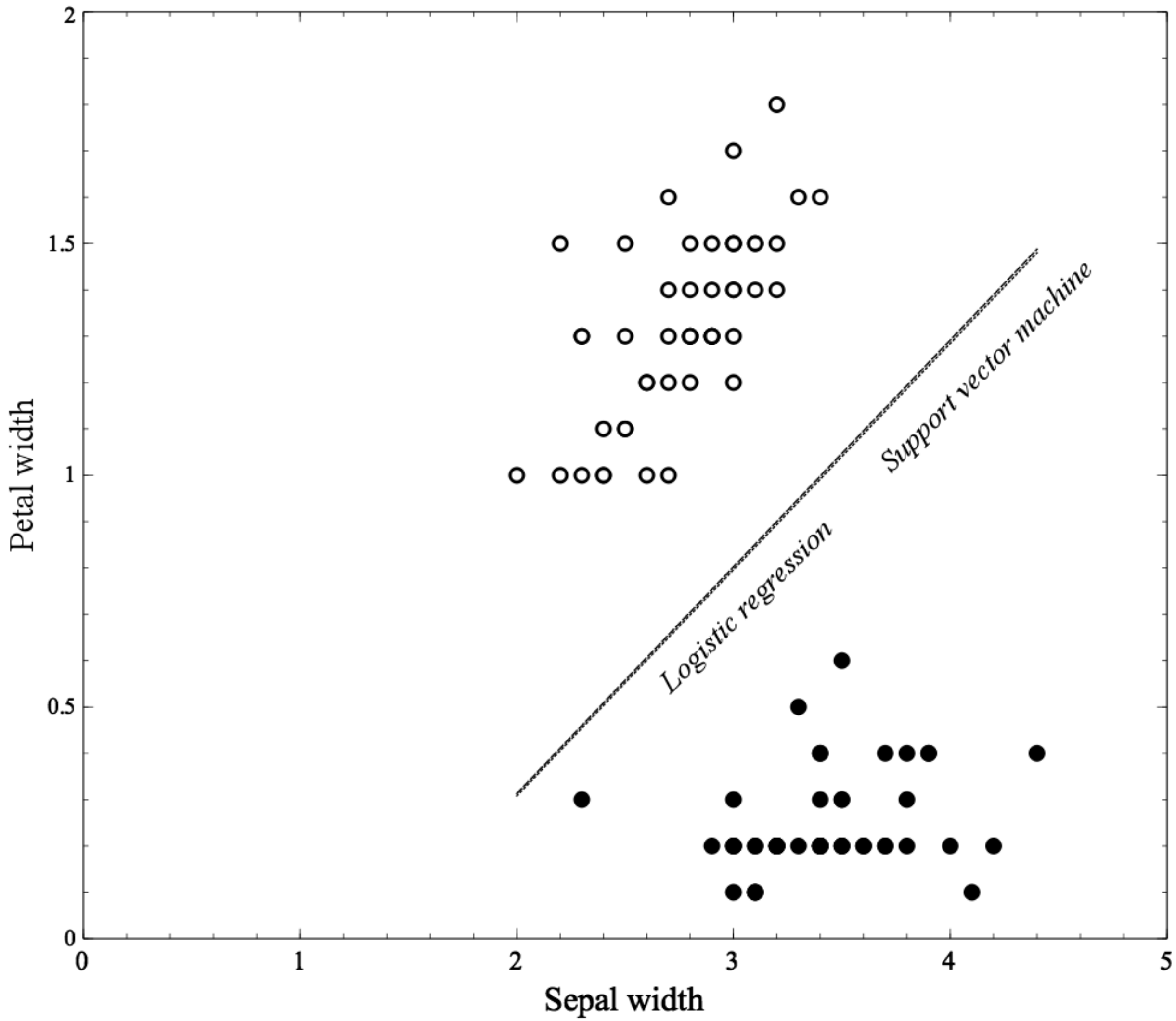
- Common techniques based on fitting the parameters of complex, nonlinear functions:
  - Non-linear support vector machines and neural networks

- **Nonlinear support vector machine** with a “polynomial kernel” consider “higher-order” combinations of the original features
  - Squared features, products of features, etc.

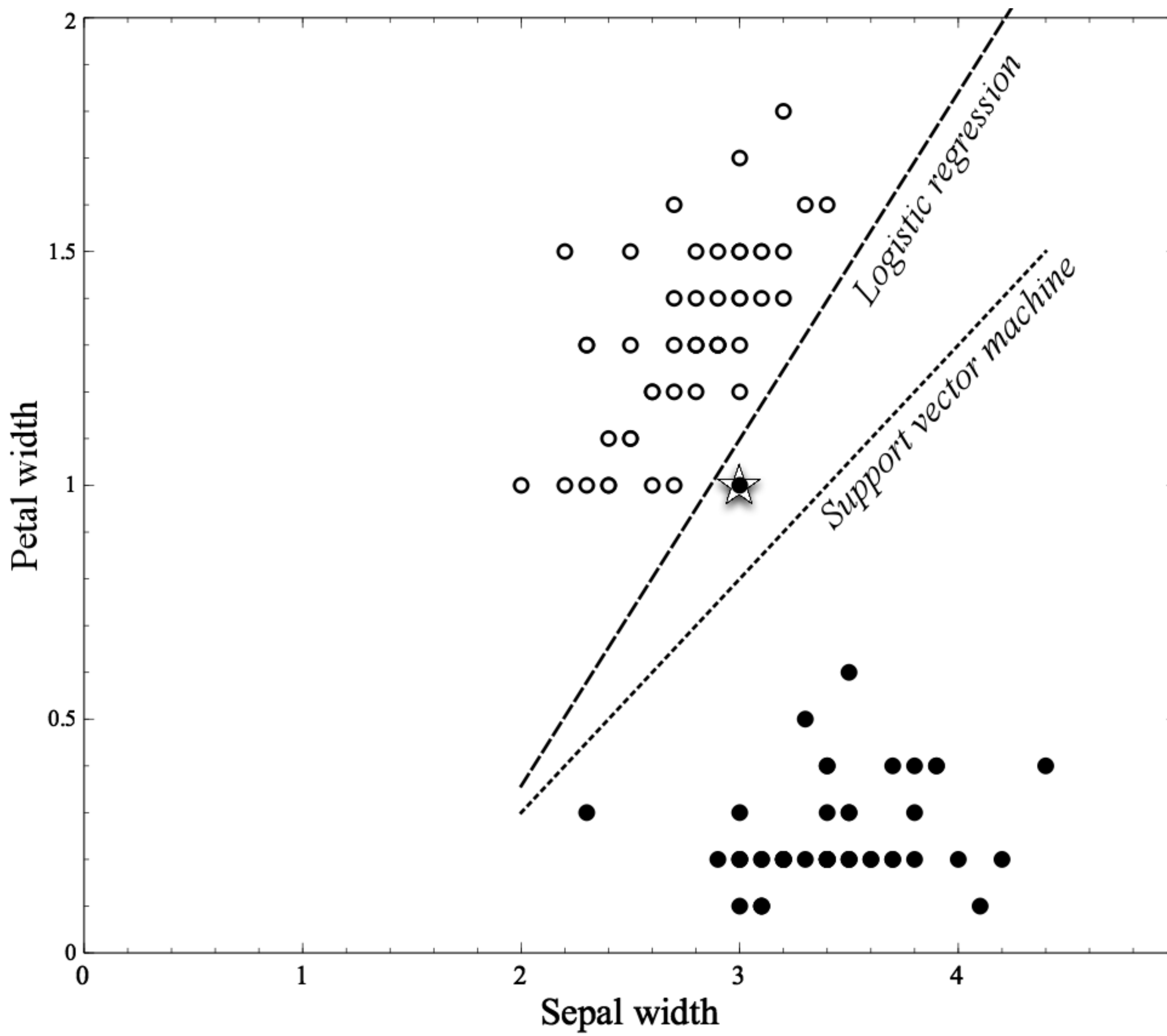
- Think of a **neural network** as a “stack” of models
  - On the bottom of the stack are the original features
  - Each layer in the stack applies a simple model to the outputs of the previous layer

- Might fit data *too* well (..to be continued)

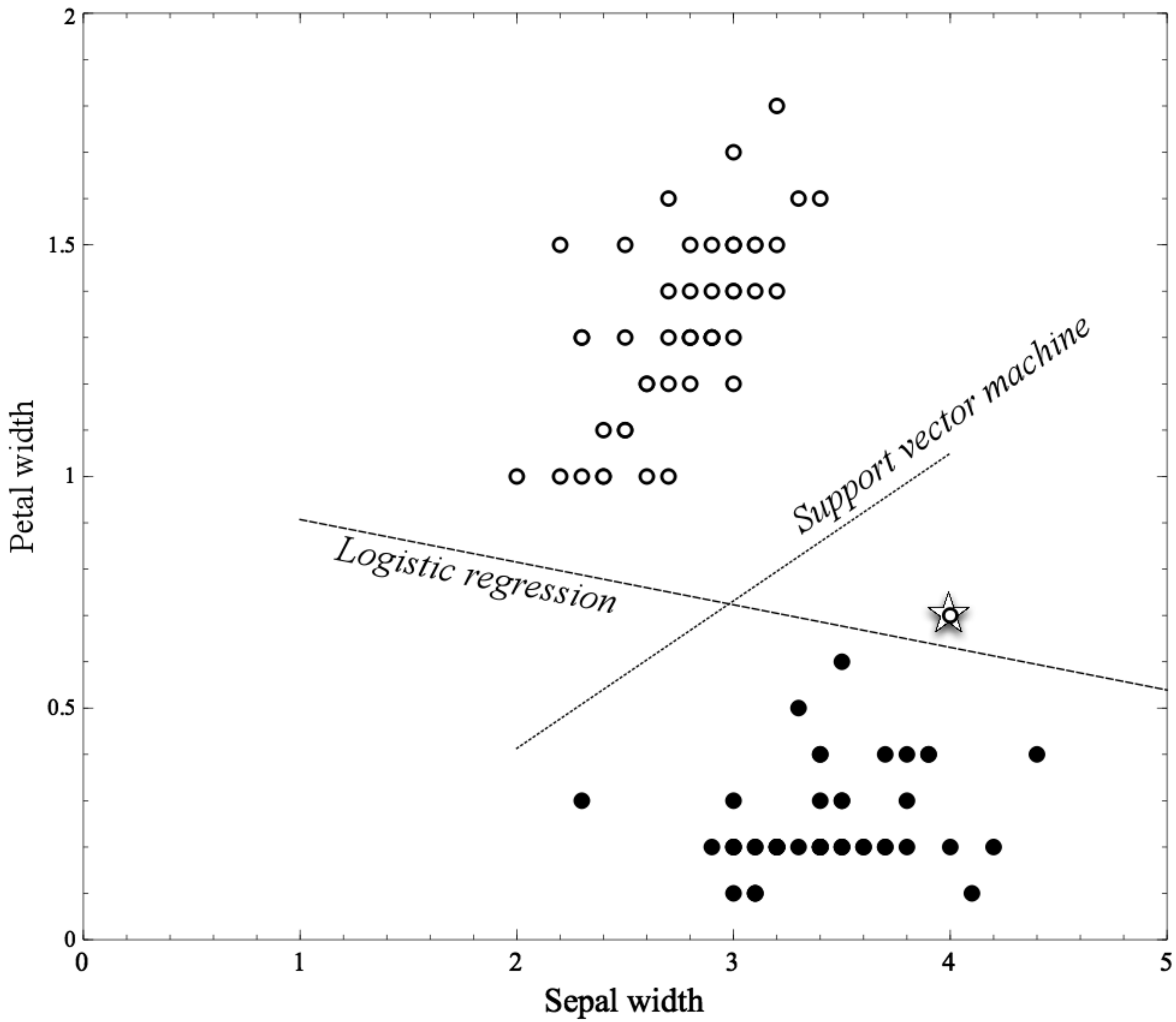
Example: Classifying Flowers



Example: Classifying Flowers



Example: Classifying Flowers



Avoiding Over-fitting

Tree Induction:

- Post-pruning
  - takes a fully-grown decision tree and discards unreliable parts
- Pre-pruning
  - stops growing a branch when information becomes unreliable

Linear Models:

- Feature Selection
- Regularization
  - Optimize some combination of fit and simplicity

Regularization

Regularized linear model:

$$\underset{\boldsymbol{w}}{\operatorname{argmax}}[\operatorname{fit}(\boldsymbol{x}, \boldsymbol{w}) - \lambda * \operatorname{penalty}(\boldsymbol{w})]$$

- "L2-norm"
  - The sum of the *squares* of the weights
  - L2-norm + standard least-squares linear regression = ridge regression
- "L1-norm"
  - The sum of the *absolute values* of the weights
  - L1-norm + standard least-squares linear regression = lasso
  - Automatic feature selection