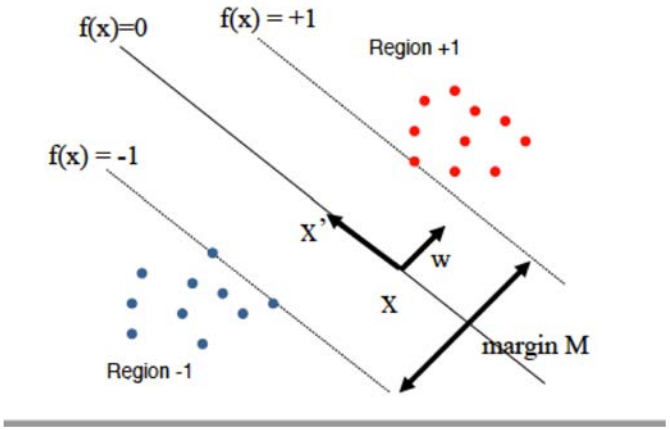
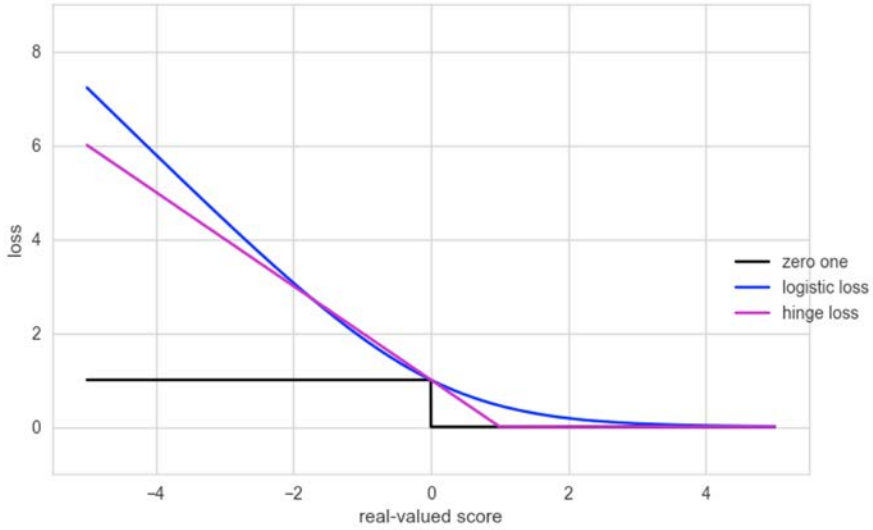
Tufts COMP 135: Introduction to Machine Learning
https://www.cs.tufts.edu/comp/135/2019s/

Support Vector Machines





Prof. Mike Hughes

Many ideas/slides attributable to:
Dan Sheldon (U.Mass.)
James, Witten, Hastie, Tibshirani (ISL/ESL books)

SVM Unit Objectives

Big ideas: Support Vector Machine

- Why maximize margin?
- What is a support vector?
- What is hinge loss?

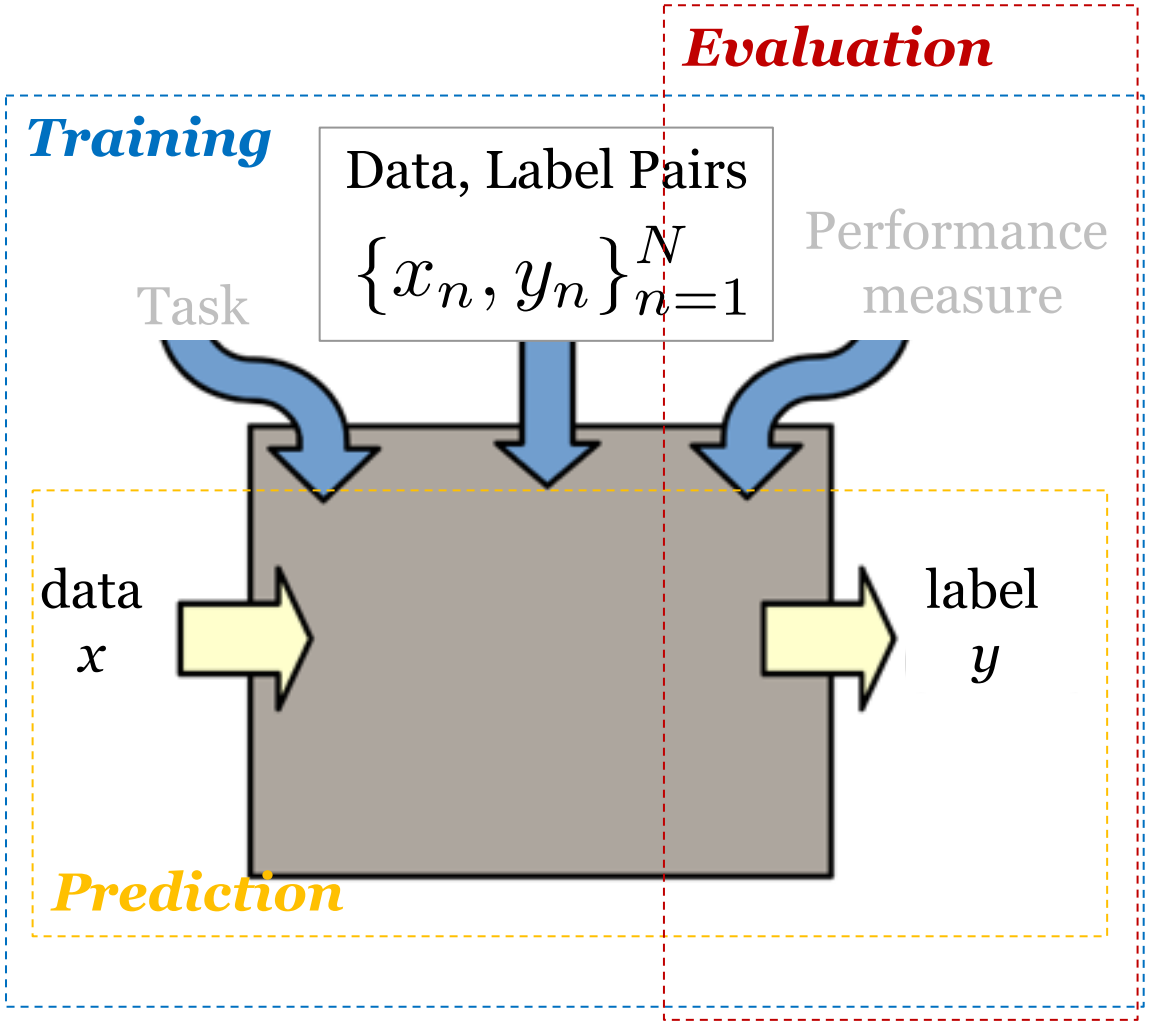
- Advantages over logistic regression
  - Less sensitive to outliers
  - Advantages from sparsity in when using kernels
- Disadvantages
  - Not probabilistic
  - Less elegant to do multi-class

What will we learn?

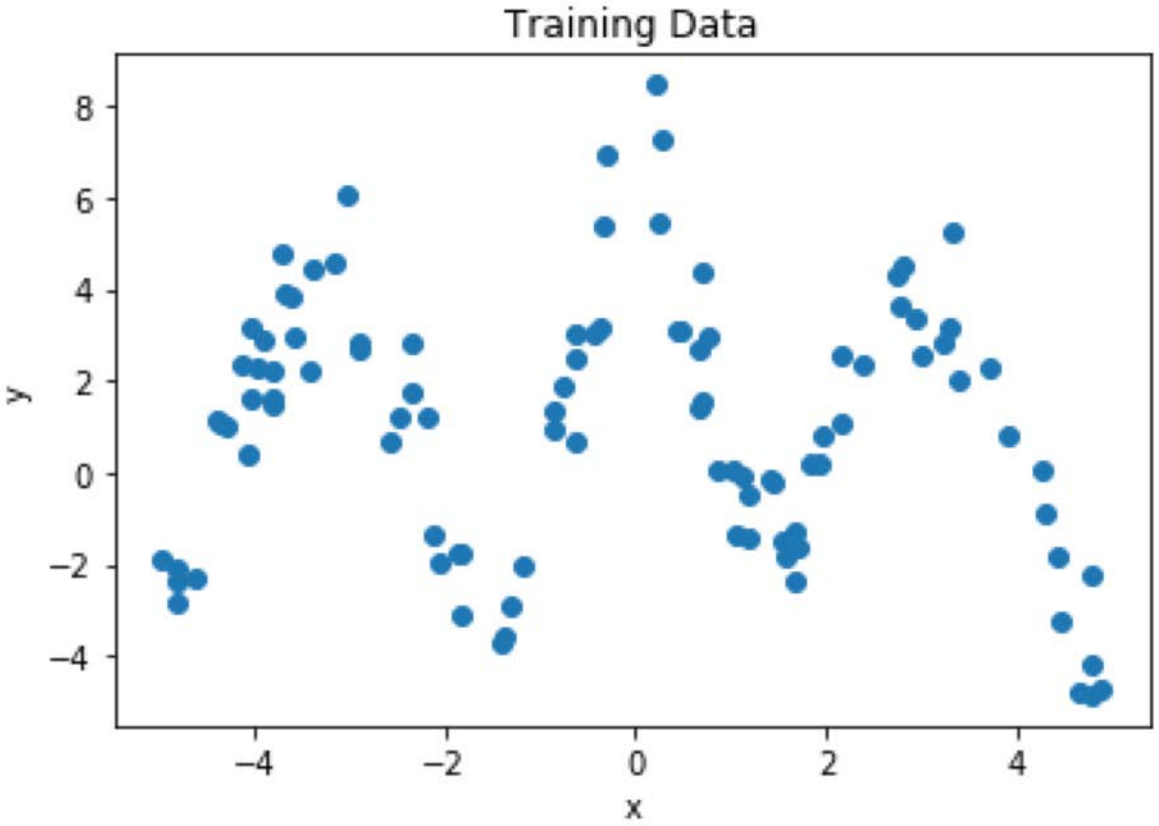
Supervised Learning

Unsupervised Learning

Reinforcement Learning

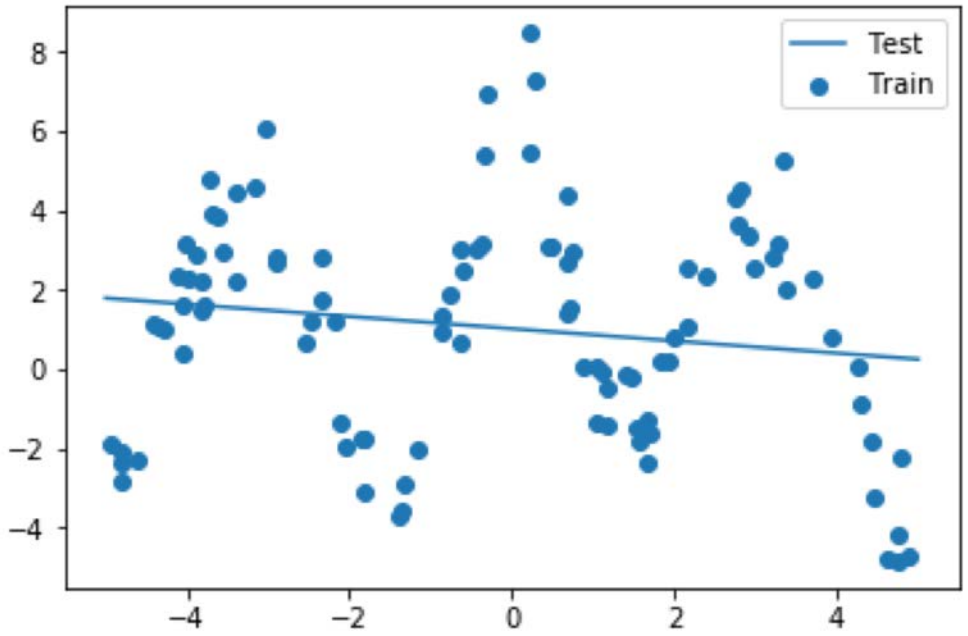


Recall: Kernelized Regression



Linear Regression

```
clf = sklearn.linear_model.LinearRegression()
clf.fit(x_train, y_train)
plot_model(x_test, clf)
```



Kernel Function

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Interpret: similarity function for x_i and x_j

Properties:

- Input: any two vectors
- Output: scalar real
- larger values mean more similar
- symmetric

Common Kernels

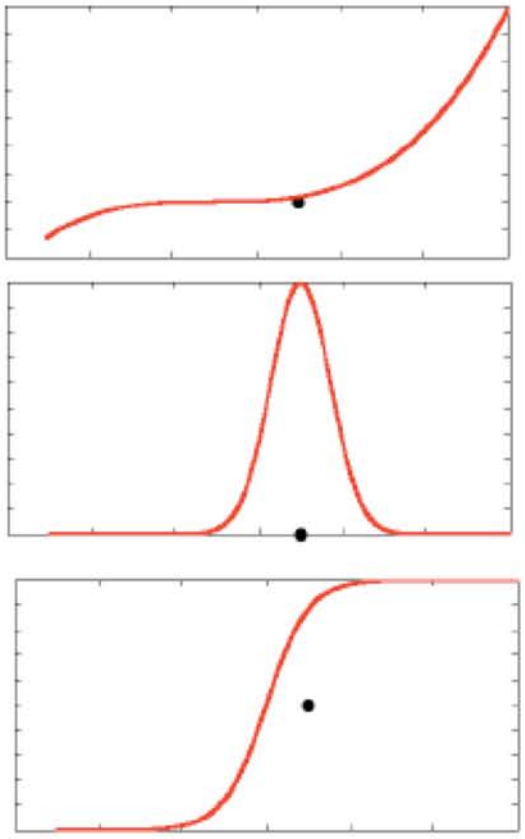
- Polynomial $K(a,b) = (1 + \sum_j a_j b_j)^d$
- Radial Basis Functions

$$K(a,b) = \exp(-(a-b)^2/2\sigma^2)$$

- Saturating, sigmoid-like:

$$K(a,b) = \tanh(ca^T b + h)$$

- Many for special data types:
  - String similarity for text, genetics



Kernel Matrices

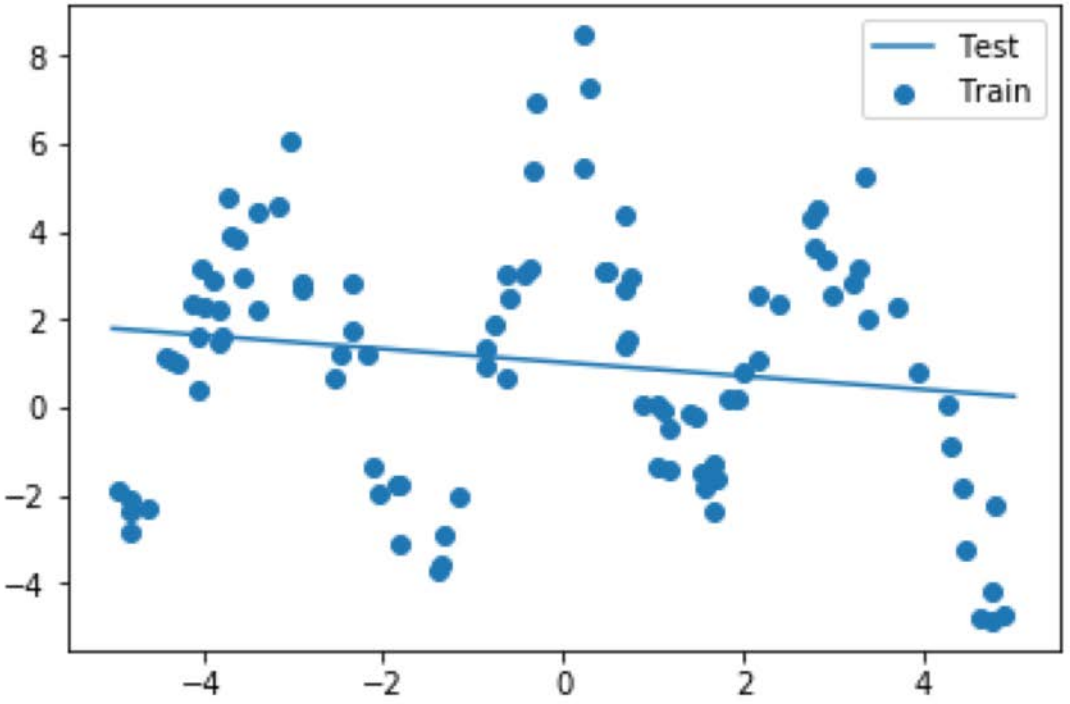
- **K_train : N x N symmetric matrix**

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) \dots k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) \dots k(x_2, x_N) \\ \vdots & \\ k(x_N, x_1) & k(x_N, x_2) \dots k(x_N, x_N) \end{bmatrix}$$

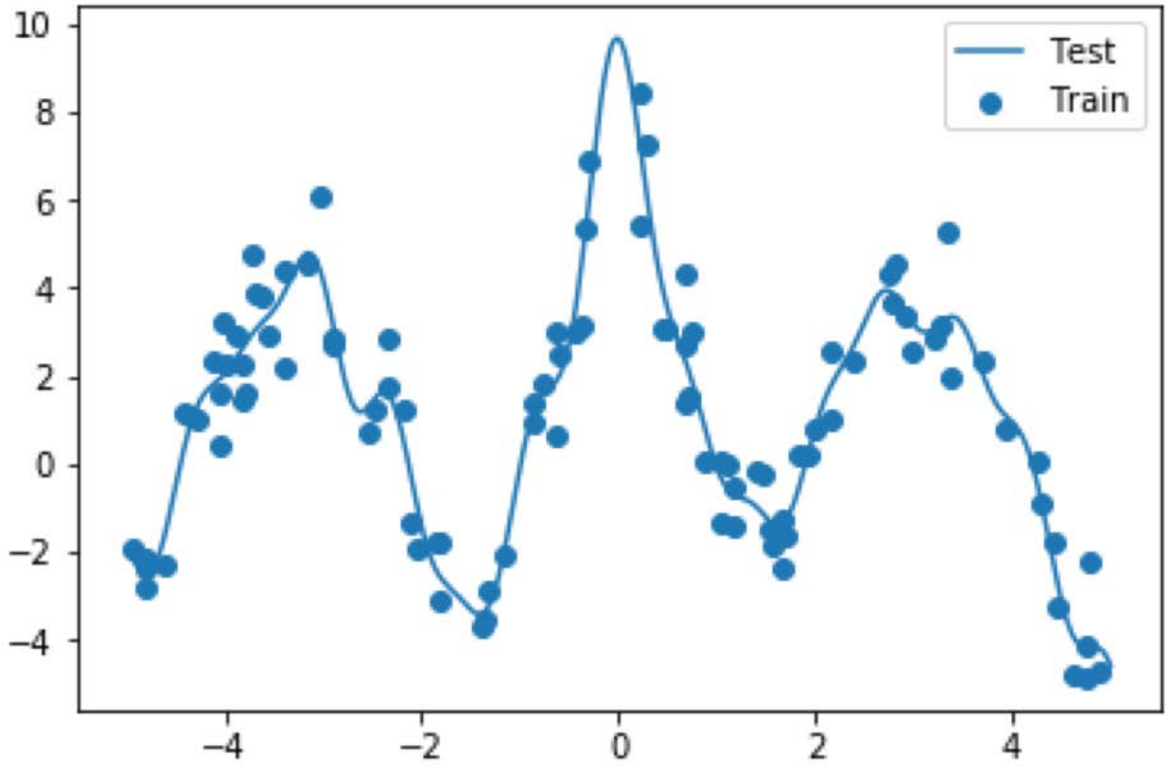
- **K_test : T x N matrix for test feature**

Linear Kernel Regression

```
clf = sklearn.linear_model.LinearRegression()
clf.fit(K_train, y_train)
plot_model(K_test, clf)
```



Radial basis kernel
aka Gaussian
aka Squared Exponential



Compare: Linear Regression

Prediction: Linear transform of G-dim features

$$\hat{y}(x_i, \theta) = \theta^T \phi(x_i) = \sum_{g=1}^{G} \theta_g \cdot \phi(x_i)_g$$

Training: Solve G-dim optimization problem

$$\min_{\theta} \sum_{n=1}^{N} \left(y_n - \hat{y}(x_n, \theta)\right)^2$$

+ L2 penalty (optional)

Kernelized Linear Regression

- **Prediction**:

$$\hat{y}(x_i, \alpha, \{x_n\}_{n=1}^N) = \sum_{n=1}^{N} \alpha_n k(x_n, x_i)$$
$$= X$$

- **Training:** Solve N-dim optimization problem

$$\min_{\alpha} \sum_{n=1}^{N} \left(y_n - \hat{y}(x_n, \alpha, X)\right)^2$$

Can do all needed operations with only access to kernel
(no feature vectors are created in memory)

Math on board

- Goal: Kernelized linear prediction reweights each training example

Can kernelize any linear model

Regression: Prediction

$$\hat{y}(x_i, \alpha, \{x_n\}_{n=1}^N) = \sum_{n=1}^{N} \alpha_n k(x_n, x_i)$$

Logistic Regression: Prediction

$$p(Y_i = 1 | x_i) = \sigma(\hat{y}(x_i, \alpha, X))$$

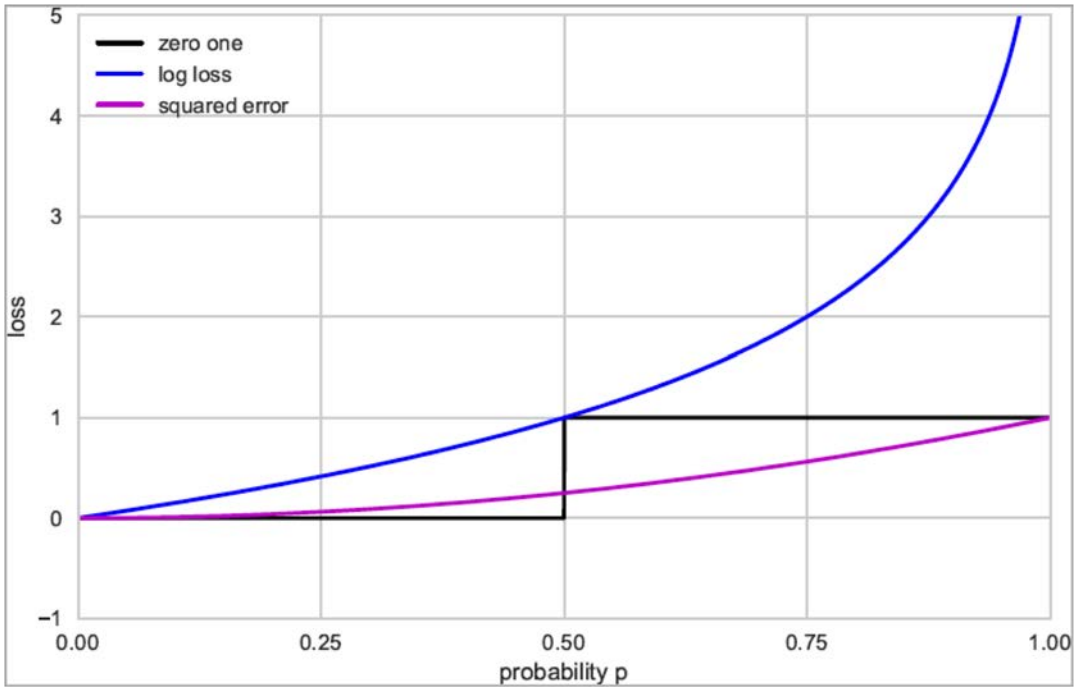
Training : Reg. Vs. Logistic Reg.

$$\min_{\alpha} \sum_{n=1}^{N} \left(y_n - \hat{y}(x_n, \alpha, X)\right)^2$$

$$\min_{\alpha} \sum_{n=1}^{N} \text{log\_loss}(y_n, \sigma(\hat{y}(x_n, \alpha, X)))$$

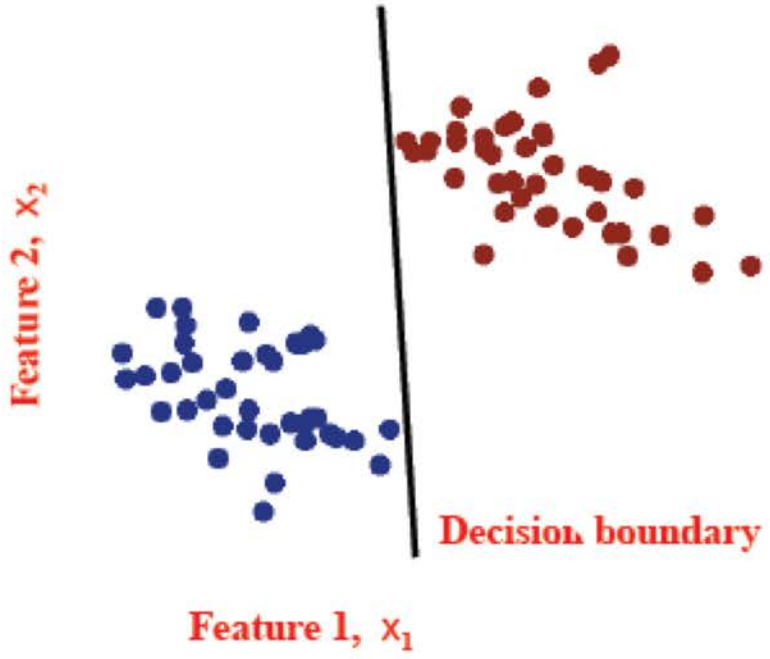
Downsides of LR

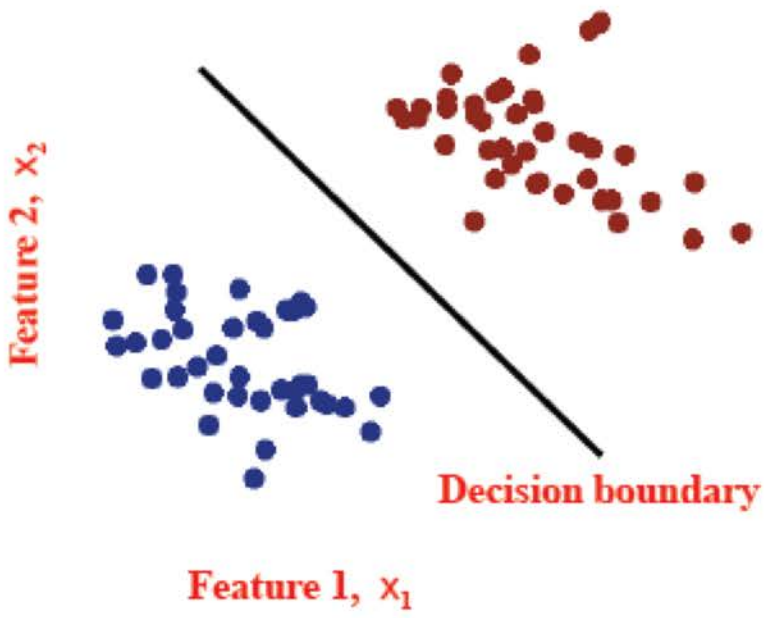
Log loss means any example misclassified pays a steep price, pretty sensitive to outliers



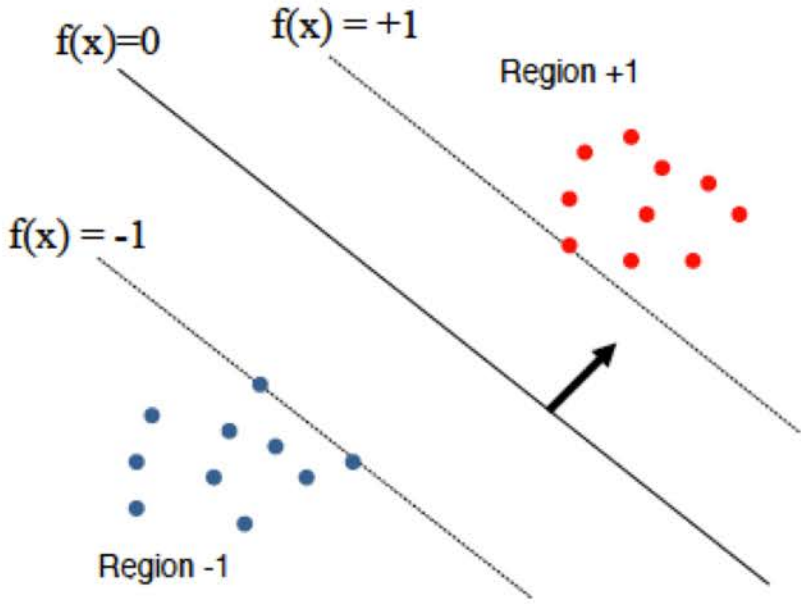
Stepping back

Which do we prefer? Why?





Idea: Define Regions Separated by Wide Margin



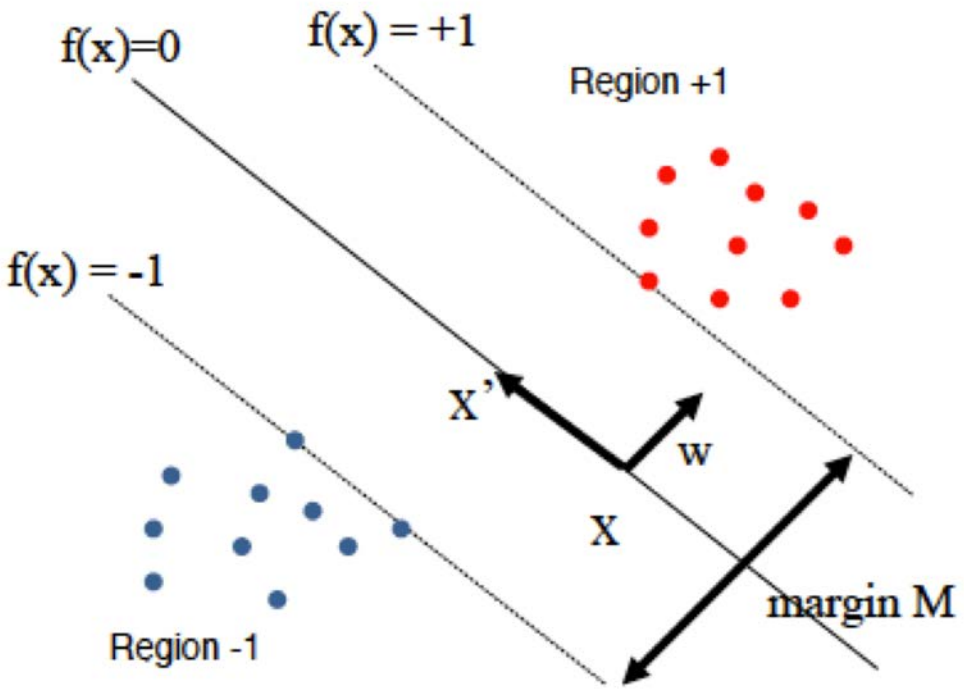
We could define such a function:

$f(x) = w*x' + b$

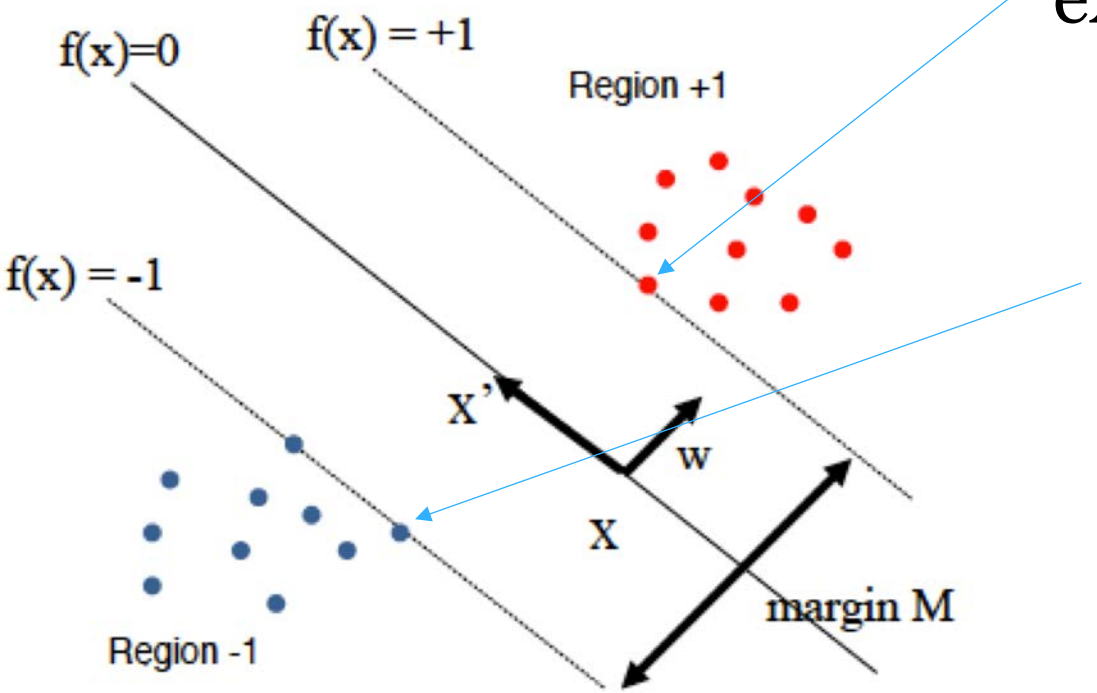
$f(x) > +1$ in region +1
$f(x) < -1$ in region –1

Passes through zero in center…

w is <u>perpendicular</u> to boundary



Examples that define the margin are called **support vectors**



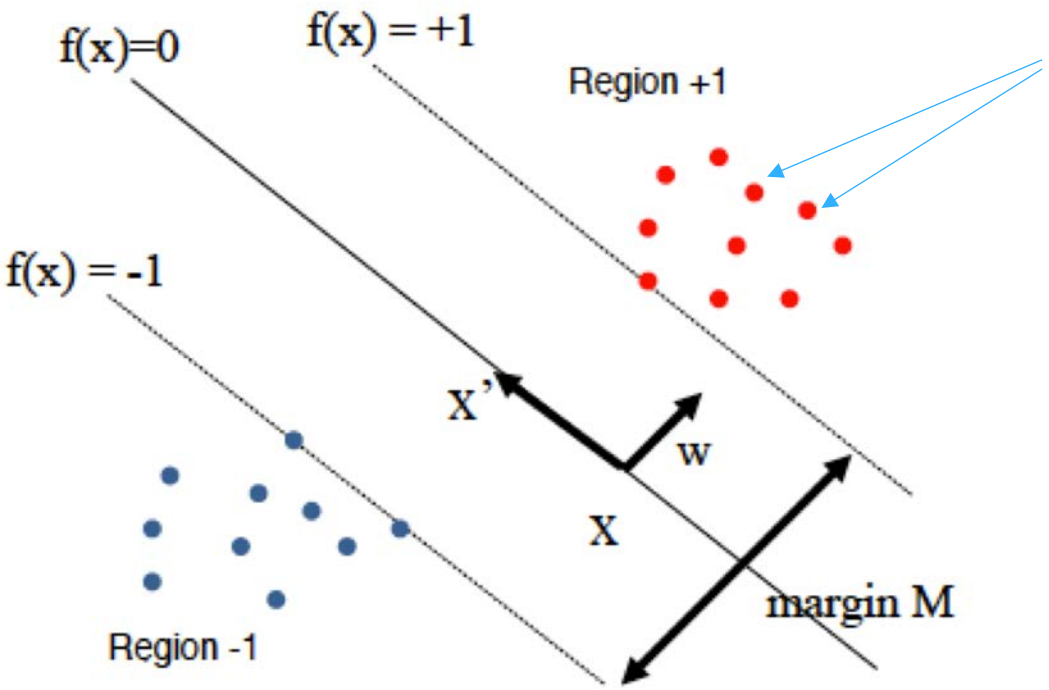
Nearest positive example

x_+

Nearest negative example

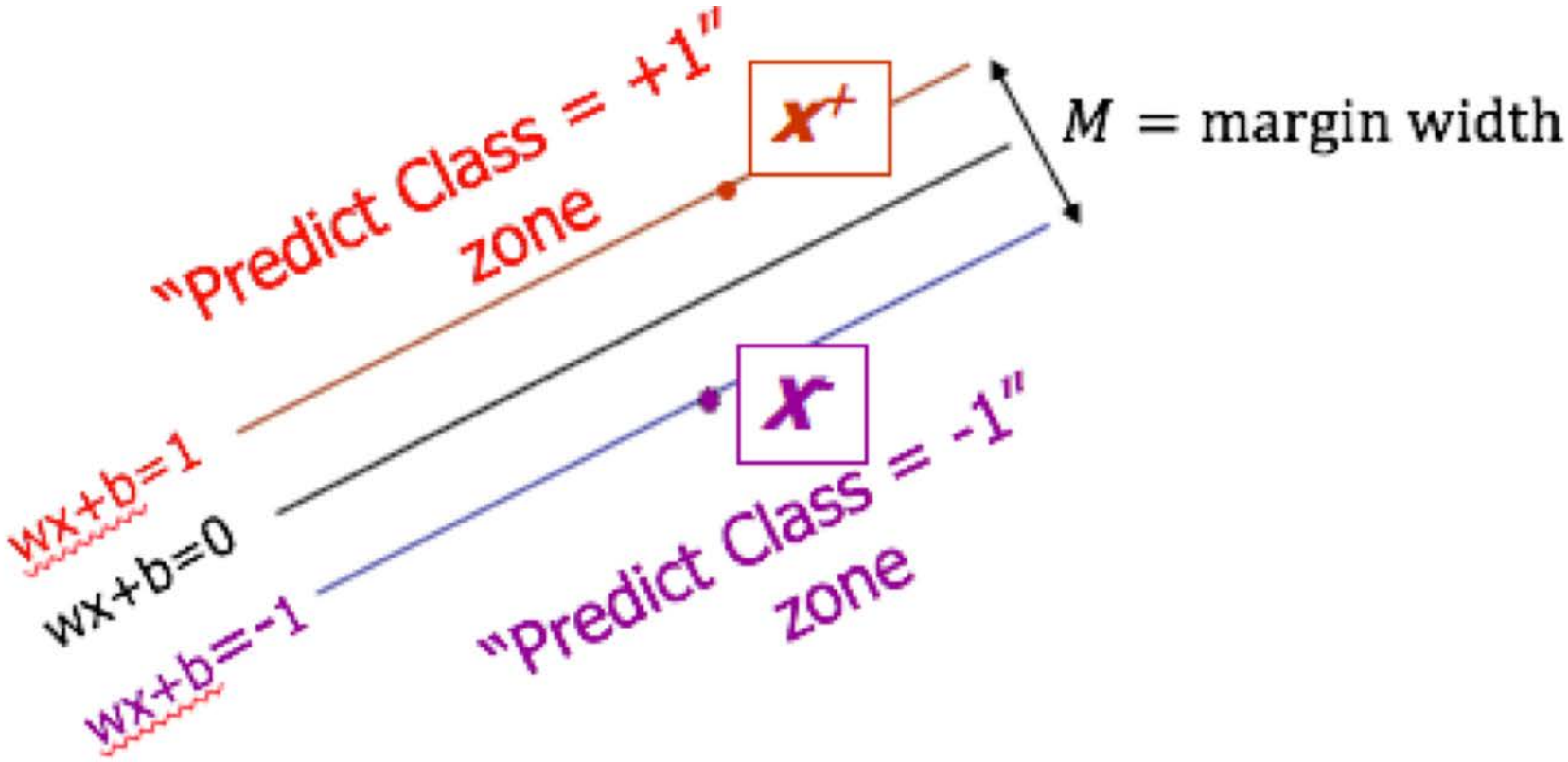
x_-

Observation: Non-support training examples do not influence margin ***at all***



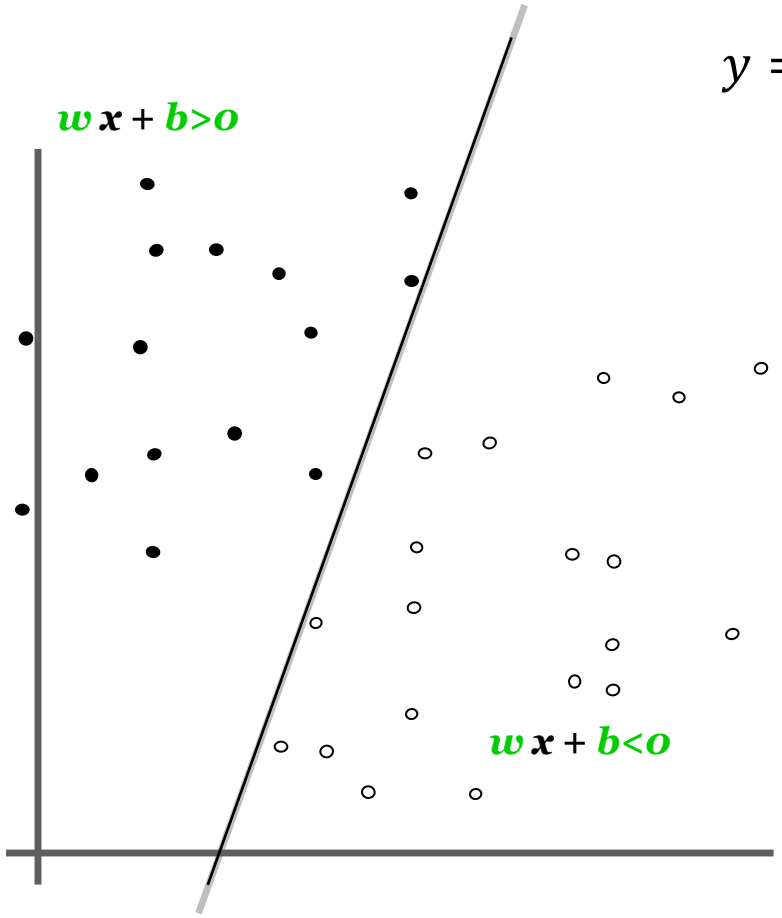
Could perturb these examples without impacting boundary

How wide is the margin?



Small margin

- y positive
- y negative

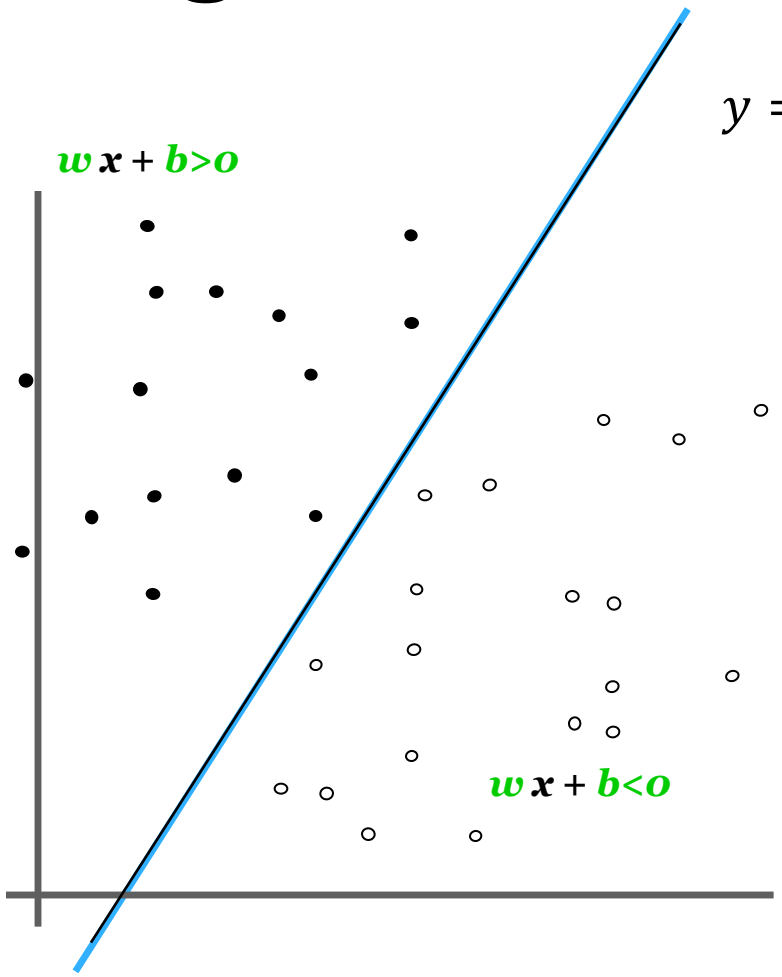


$$y = \begin{cases} +1 & if\ \boldsymbol{w}\boldsymbol{x} + b \geq 0 \\ -1 & if\ \boldsymbol{w}\boldsymbol{x} + b < 0 \end{cases}$$

Margin: distance to the boundary

Large margin

- y positive
- y negative



$$y = \begin{cases} +1 & if\ \boldsymbol{w}\boldsymbol{x} + b \geq 0 \\ -1 & if\ \boldsymbol{w}\boldsymbol{x} + b < 0 \end{cases}$$

Margin: distance to the boundary

How wide is the margin?

Distance from nearest positive example to nearest negative example along vector w:

$$M(w) = \frac{(x_+ - x_-)^T w}{||w||_2} = \frac{(x_+ - x_-)^T w}{\sqrt{w_1^2 + \ldots w_F^2}}$$

The scalar projection of $\bar{a}$ on $\bar{b}$ is the magnitude of the vector projection of $\bar{a}$ on $\bar{b}$.

$$|proj_{\bar{b}}\bar{a}| = \frac{\bar{a} \cdot \bar{b}}{|\bar{b}|}$$

How wide is the margin?

Distance from nearest positive example to nearest negative example along vector w:

$$M(w) = \frac{(x_+ - x_-)^T w}{||w||_2} = \frac{2}{||w||_2}$$

By construction, we assume

$$w^T x_+ + b = +1$$

$$w^T x_- + b = -1$$

$$w^T (x_+ - x_-) = 2$$

SVM Training Problem
Version 1: Hard margin

$$\max_{w,b} \frac{2}{||w||_2}$$

$$\text{subject to} \begin{cases} w^T x_n + b \geq +1 & \text{if } y_n = 1 \\ w^T x_n + b \leq -1 & \text{if } y_n = 0 \end{cases}$$

For each n = 1, 2, ... N

Requires all training examples to be correctly classified

This is a constrained quadratic optimization problem. There are standard optimization methods as well methods specially designed for SVM.

SVM Training Problem
Version 1: Hard margin

$$\min_{w,b} \frac{1}{2}||w||_2$$

Minimizing this equivalent to maximizing the margin width in feature space

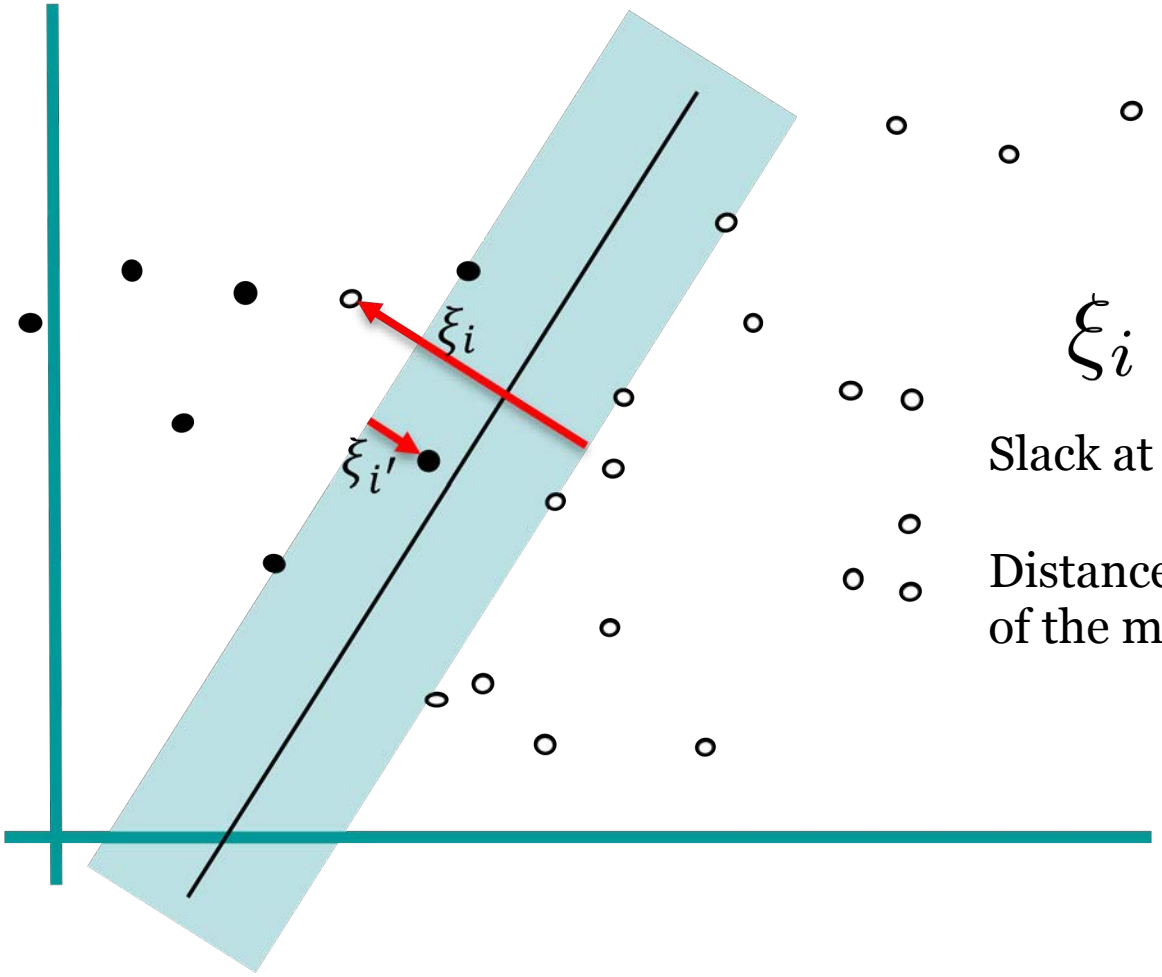
$$\text{subject to} \begin{cases} w^T x_n + b \geq +1 & \text{if } y_n = 1 \\ w^T x_n + b \leq -1 & \text{if } y_n = 0 \end{cases}$$

For each n = 1, 2, …. N

Requires all training examples to be correctly classified

This is a constrained quadratic optimization problem. There are standard optimization methods as well methods specially designed for SVM.

Soft margin:
Allow some misclassifications



$$\xi_i \geq 0$$

Slack at example i

Distance on wrong side of the margin

Hard vs. soft constraints

HARD: All positive examples to satisfy

$$w^T x_n + b \geq +1$$

SOFT: Want each positive examples to satisfy

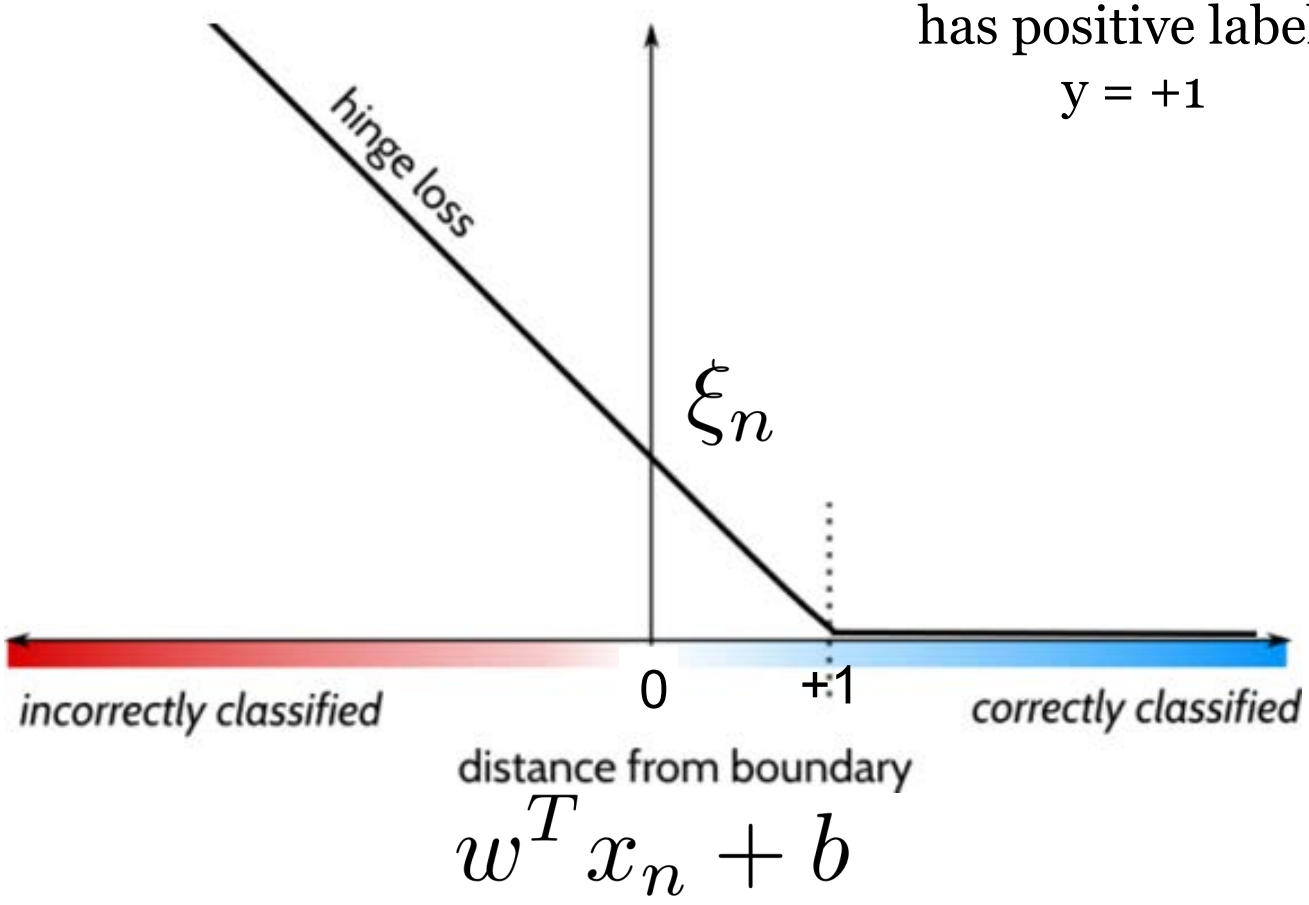
$$w^T x_n + b \geq +1 - \xi_n$$

with slack as small as possible
(minimize **absolute value**)

Hinge loss for positive example

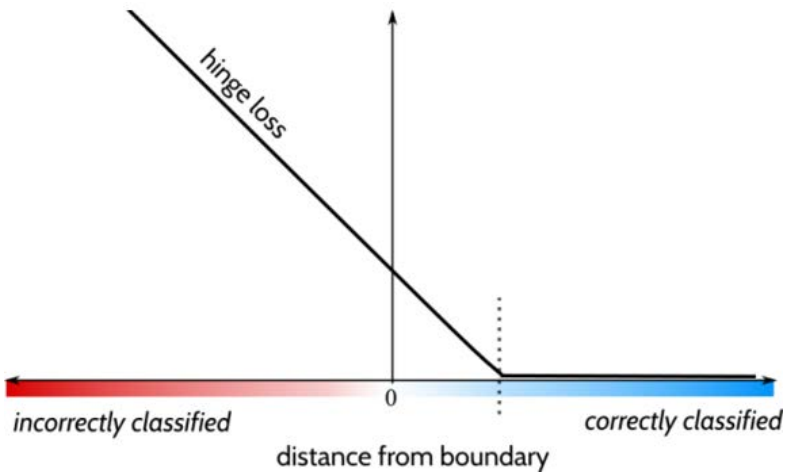
Assumes current example has positive label

$y = +1$



SVM Training Problem
Version 2: Soft margin

$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{n=1}^{N} \text{hinge\_loss}(y_n, w^T x_n + b)$$



Tradeoff parameter C
controls model complexity

Smaller C: Simpler model, encourage large margin even if we make lots of mistakes

Bigger C: Avoid mistakes

SVM vs LR: Compare training

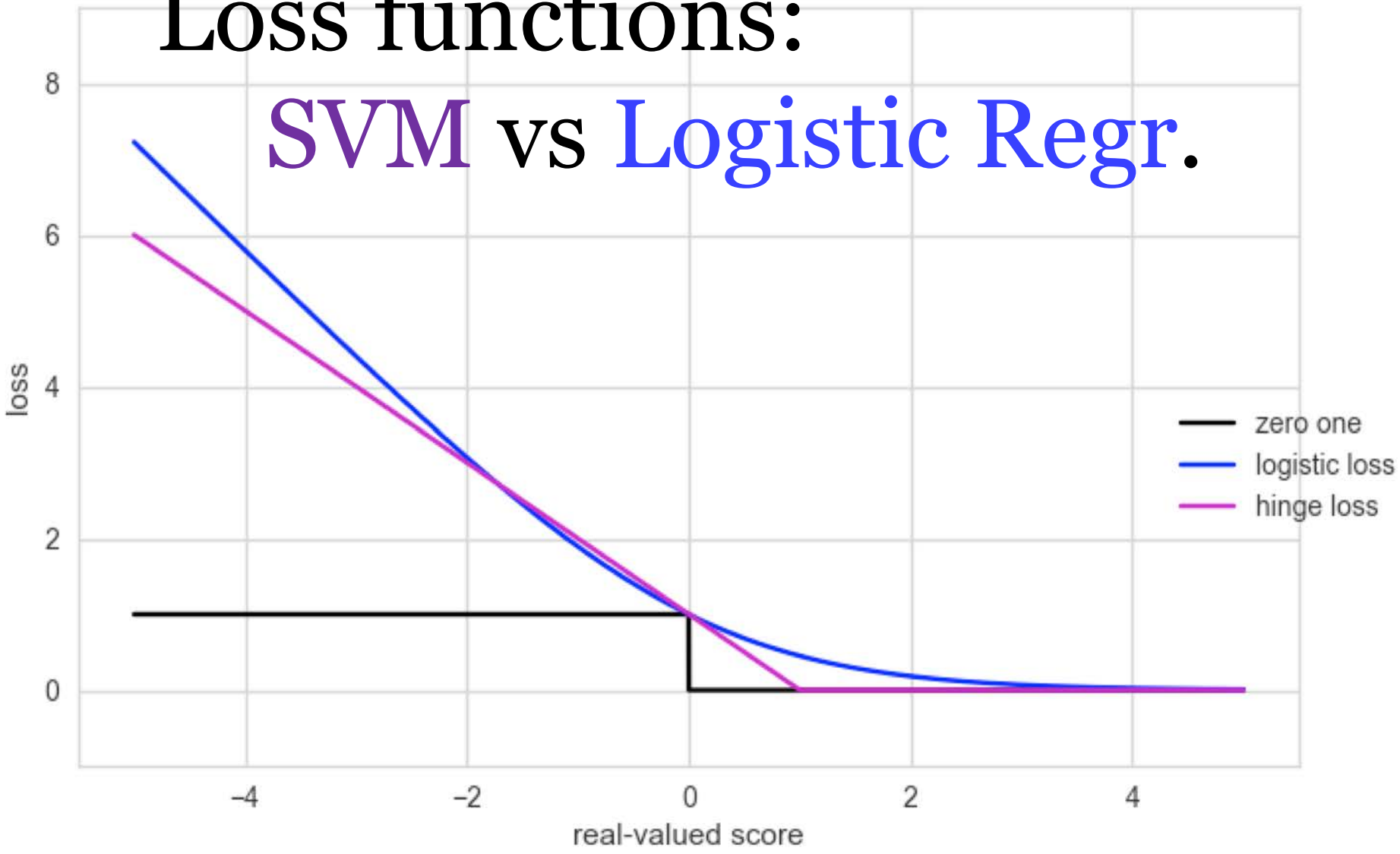
$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{n=1}^{N} \text{hinge\_loss}(y_n, w^T x_n + b)$$

$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{n=1}^{N} \text{log\_loss}(y_n, \sigma(w^T x_n + b))$$

Both require tuning complexity hyperparameter C to avoid overfitting

Loss functions:

SVM vs Logistic Regr.



SVMs: Prediction

$$\hat{y}(x_i) = w^T x_i + b$$

Make binary prediction via hard threshold

$$\begin{cases} 1 & \text{if } \hat{y}(x_i) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

SVMs and Kernels: Prediction

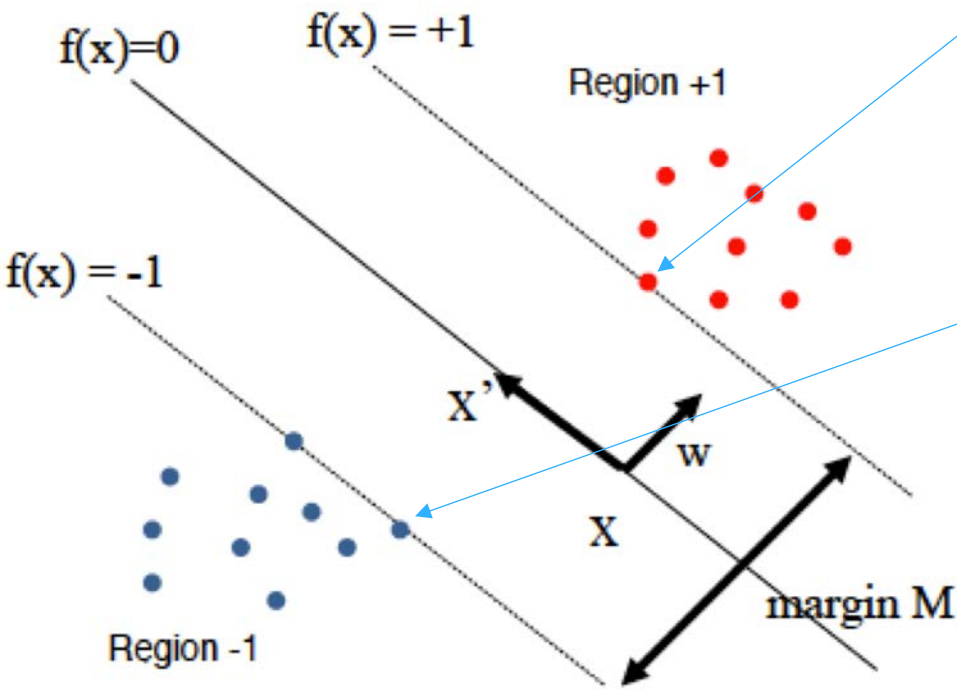
$$\hat{y}(x_i) = \sum_{n=1}^{N} \alpha_n k(x_n, x_i)$$

Make binary prediction via hard threshold

$$\begin{cases} 1 & \text{if } \hat{y}(x_i) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Efficient training algorithms using modern quadratic programming solve the dual optimization problem of SVM soft margin problem

Support vectors are often **small** fraction of all examples



Nearest positive example x_+

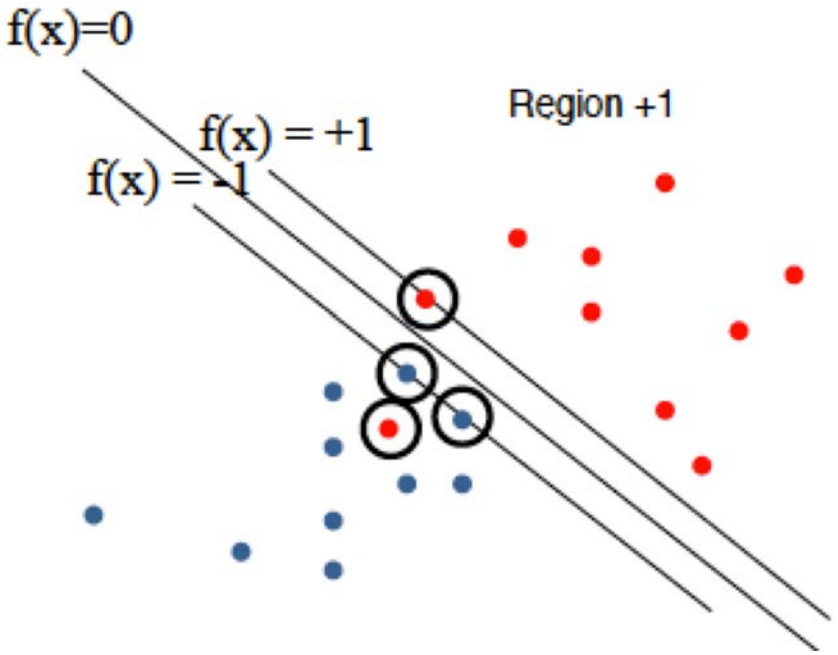
Nearest negative example x_-

Support vectors defined by **non-zero alpha** in kernel view

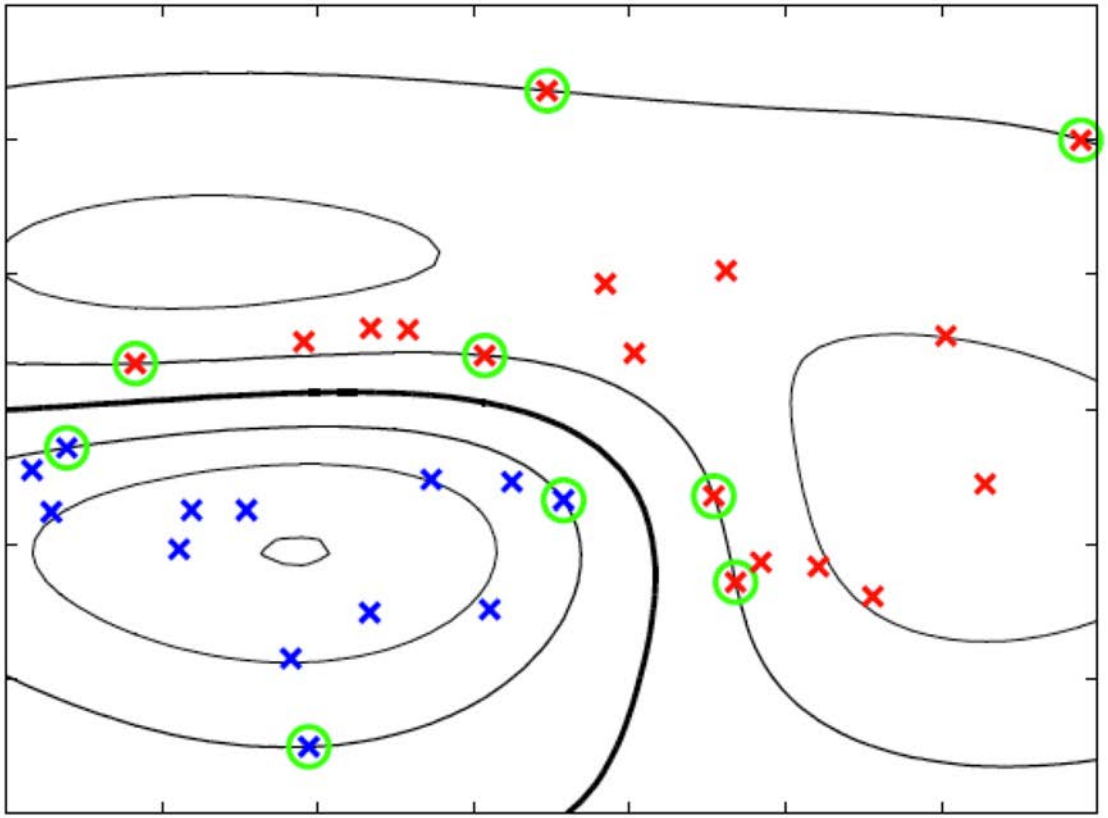
Data points *i* with non-zero weight α_i:

- Points with minimum margin (on optimized boundary)
- Points which violate margin constraint, but are still correctly classified
- Points which are misclassified

For all other training data, features have *no impact* on learned weight vector



SVM + Squared Exponential Kernel



Support vectors (green) for data separable by radial basis function kernels, and non-linear margin boundaries

	SVM	Logistic Regr
Loss	hinge	cross entropy (log loss)
Sensitive to outliers	**less**	more
Probabilistic?	no	**yes**
Kernelizable?	Yes, with speed benefits from **sparsity**	Yes
Multi-class?	Only via a separate one-vs-all for each class	**Easy**, use softmax

Activity

- KernelRegressionDemo.ipynb

- Scroll down to SVM vs logistic regression
  - Can you visualize behavior with different C?
  - Try different kernels?
  - Examine alpha vector?