Lecture 10: Kernel Machines

TTIC 31020: Introduction to Machine Learning

Instructor: Kevin Gimpel

TTI-Chicago

October 31, 2019

support vector is the vector lie on the margin

We can ignore the vector not on margin and fit the model again with only support vectors, the model will not change

Administrivia

- Problem set 3 due Thursday Nov 7 11:59pm
- Quiz 2
  - Thursday Nov 7
  - topic: linear classifiers, not including SVMs

Review: linear SVMs

- The objective, subject to perfect margin constraints:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t. } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 \geq 0 \quad \forall i$$

- Giving up hard margin constraints leads to the **soft margin** SVM:

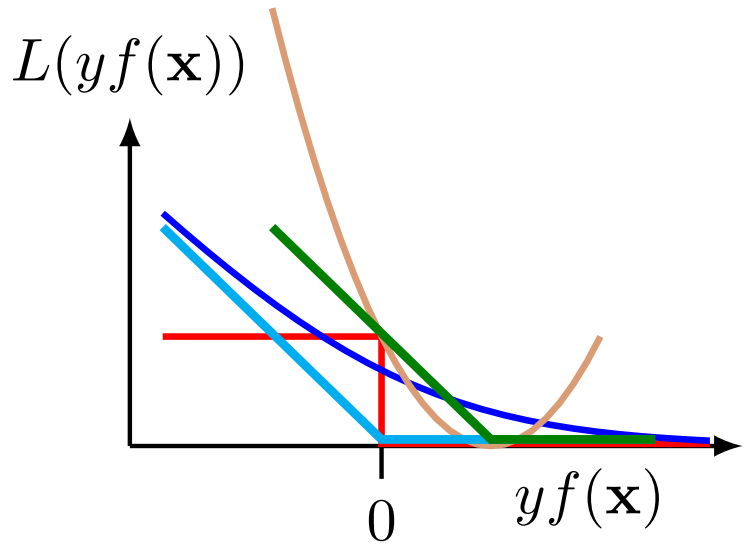
$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \quad \text{s.t. } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i \geq 0, \;\; \xi_i \geq 0 \quad \forall i$$

- Resulting optimization objective:

$$\min_{\mathbf{w},b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \left[1 - y_i\left(\mathbf{w}\cdot\mathbf{x}_i + b\right)\right]_+ \right\}$$

Review: surrogate losses

- We are considering learning classifiers $h(\mathbf{x}) = \text{sign}(\overbrace{\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) + b}^{f(\mathbf{x})})$
- Can not use 0/1 loss directly to train
- Choices of **surrogate loss** lead to different learning algorithms:

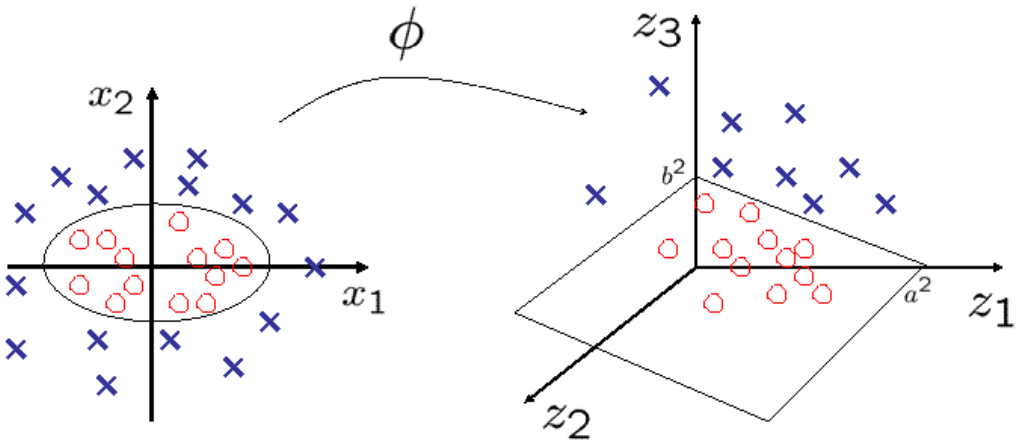


- Squared loss: least squares
- Logistic loss: logistic regression
- Hinge loss: perceptron/SVM

- So: same model class, but different algorithms for learning (selecting) a model given a dataset

Nonlinear features

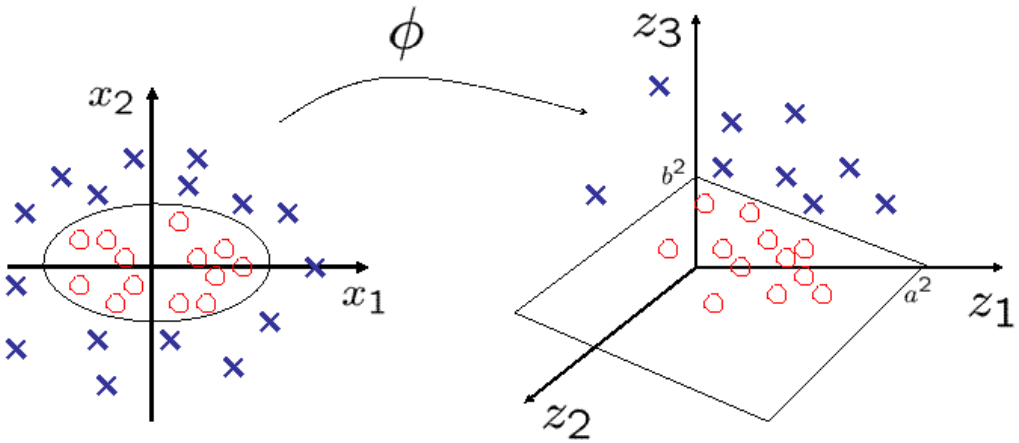
- As with logistic regression, we can move to nonlinear classifiers by mapping data into a nonlinear **feature space**. Example:



$$\phi : [x_1, x_2] \rightarrow [x_1^2, \sqrt{2}x_1x_2, x_2^2]$$

Nonlinear features

- As with logistic regression, we can move to nonlinear classifiers by mapping data into a nonlinear **feature space**. Example:



$$\phi : [x_1, x_2] \rightarrow [x_1^2, \sqrt{2}x_1x_2, x_2^2]$$

- Elliptical decision boundary in the input space becomes linear in the feature space $\mathbf{z} = \boldsymbol{\phi}(\mathbf{x})$:

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = c \Rightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = c.$$

Nonlinear SVM

- First, write the SVM objective in the feature space

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ C \sum_{i=1}^{n} \max\left(0, 1 - y_i \left(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b\right)\right) + \|\mathbf{w}\|^2 \right\}$$

- Prediction rule:

$$\widehat{y} = \operatorname{sign}\left(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) + b\right)$$

- After solving SVM, we have $\mathbf{w}^* = \sum_{i:\alpha_i>0} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i)$, so

$$\widehat{y} = \operatorname{sign}\left(\sum_{i:\alpha_i>0} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}) + b \right)$$

Xi is from training set

Example of nonlinear mapping

- Consider the feature mapping $\phi : \mathbb{R}^2 \to \mathbb{R}^6$:

$$\phi : \ [x_1, x_2] \ \to \ [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]$$

- The (linear) SVM classifier in the feature space:

$$\widehat{y} = \text{sign}\left(\sum_{i:\alpha_i>0} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}) + b\right)$$

X is a new one from test set

- The dot product in the feature space:

$$\begin{aligned} \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) &= 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \\ &= (1 + \mathbf{x} \cdot \mathbf{z})^2 \end{aligned}$$

z is the same space as x

like vectorization?

- So, we don't need to explicitly write expansions and compute dot product in 6D $\boldsymbol{\phi}(\mathbf{x})$; we can do everything in the original 2D space $\mathbf{x}$!

Dot products and feature space

- We defined a specific nonlinear feature mapping $\boldsymbol{\phi}$, and saw that $\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z})$ can be computed using the **kernel** $K(\mathbf{x}, \mathbf{z})$:

$$K(\mathbf{x}, \mathbf{z}) \;=\; (1 + \mathbf{x} \cdot \mathbf{z})^2$$

  in the original input space
- This allows us to compute predictions of a linear classifier in the feature space without directly writing anything in it down:

$$\begin{aligned} \widehat{y} &= \operatorname{sign}\left(\sum_{i:\alpha_i > 0} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}) + b \right) \\ &= \operatorname{sign}\left(\sum_{i:\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \end{aligned}$$

- Can we use kernels to train an SVM too?

Training SVMs: take two

- Let's revisit the original SVM formulation

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t. } y_i\left(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i)+b\right)-1 \geq 0 \quad \forall i$$

- The constrained formulation, using Lagrangian multipliers $\boldsymbol{\alpha}$:

max over alpha is just like taking derivative over alpha step

$$\min_{\mathbf{w},b}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \max_{\alpha_i\geq 0} \alpha_i\left(1-y_i\left(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i)+b\right)\right)\right\}$$

?? why max

- Intuition: associate with each constraint the loss

$$\max_{\alpha_i\geq 0} \alpha_i\left(1-y_i(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i)+b)\right) = \begin{cases} 0 & \text{if } y_i\left(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i)+b\right)\geq 1, \\ \infty & \text{otherwise (constraint violated).} \end{cases}$$

The way to minimize a negative number is make alpha 0

Training SVMs: constrained optimization

- Primal formulation:

$$\min_{\mathbf{w},b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \max_{\alpha_i \geq 0} \alpha_i \left(1 - y_i \left(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b\right)\right) \right\}$$

- Dual formulation – equivalent due to convexity of problem (Slater's conditions allowing switching min/max order)

$$\max_{\boldsymbol{\alpha} \geq \mathbf{0}} \left\{ \min_{\mathbf{w},b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \alpha_i \left(1 - y_i \left(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b\right)\right) \right\} \right\}$$

Intuition: for any $\boldsymbol{\alpha} \geq \mathbf{0}$, minimize value of the inner function of $(\mathbf{w}(\boldsymbol{\alpha}), b(\boldsymbol{\alpha}))$; find $\boldsymbol{\alpha}$ for which that (minimal) value is maximized.

Training SVMs: optimizing the dual

$$\max_{\boldsymbol{\alpha}\geq\mathbf{0}} \left\{ \min_{\mathbf{w},b} \left\{ \underbrace{\frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \alpha_i \left(1 - y_i \left(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i) + b\right)\right)}_{J(\mathbf{w},b;\boldsymbol{\alpha})} \right\} \right\}$$

- To minimize (convex) J, set derivatives to zero:

$$\frac{\partial J}{\partial \mathbf{w}} = 0 \;\Rightarrow\; \mathbf{w} = \sum_i \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i)$$

$$\frac{\partial J}{\partial b} = 0 \;\Rightarrow\; \sum_i \alpha_i y_i = 0$$

- Substituting these into J we get

$$\min_{\mathbf{w},b} J(\mathbf{w}, b; \boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\mathbf{x}_i)\cdot\boldsymbol{\phi}(\mathbf{x}_j)$$

SVM dual

- Putting it all together, our new optimization objective:

$$\max_{\boldsymbol{\alpha}} \left\{ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) \right\}$$

$$\text{s.t.} \sum_i \alpha_i y_i = 0,$$

$$\alpha_i \geq 0 \quad \forall i$$

- This is a quadratic program (QP) – the objective is quadratic in the unknowns (α_is) and the constraints are linear.
- Many solvers, with some advances in the last 20 years motivated by need to train SVMs!

SVM dual: adding the slack

- What about non-separable data (using slack)?
- The only change: add upper bound on the Lagrange multipliers

$$\max_{\boldsymbol{\alpha}} \left\{ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) \right\}$$

$$\text{s.t.} \sum_i \alpha_i y_i = 0,$$

$$0 \leq \alpha_i \leq C \quad \forall i$$

  so we pay finite penalty for violating the constraints

- To derive this: need to introduce ξ_is, and set up the full Lagrangian form including the constraints on $\xi_i \geq 0$

Back to kernels

$$\max_{\mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \sum_i \alpha_i y_i = 0} \left\{ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) \right\}$$

- Now: the objective only depends on the data through dot products in feature space!

$$\max_{\mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \sum_i \alpha_i y_i = 0} \left\{ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right\}$$

- We need to compute the **Gram matrix** (kernel matrix) with an entry for every pair of training examples, then can solve the QP.
- What about solving in the primal, with gradient descent?

Training kernel SVMs: primal

- We know $\mathbf{w}^* = \sum_i \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i)$, so

$$\min_{\boldsymbol{\alpha}, b} \Big\{ C \sum_{i=1}^{n} \big[1 - y_i \big(\sum_j \alpha_j y_j \underbrace{\boldsymbol{\phi}(\mathbf{x}_j) \cdot \boldsymbol{\phi}(\mathbf{x}_i)}_{} + b \big) \big]_+ + \|\mathbf{w}\|^2 \Big\}$$

Training kernel SVMs: primal

- We know $\mathbf{w}^* = \sum_i \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i)$, so

$$\min_{\boldsymbol{\alpha}, b} \Big\{ C \sum_{i=1}^{n} \big[1 - y_i \big(\sum_j \alpha_j y_j \underbrace{\boldsymbol{\phi}(\mathbf{x}_j) \cdot \boldsymbol{\phi}(\mathbf{x}_i)}_{=K(\mathbf{x}_j, \mathbf{x}_i)} + b \big) \big]_+ + \|\mathbf{w}\|^2 \Big\}$$

- How can we write the regularizer?

$$\|\mathbf{w}\|^2 = \mathbf{w} \cdot \mathbf{w}$$

Training kernel SVMs: primal

- We know $\mathbf{w}^* = \sum_i \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i)$, so

$$\min_{\boldsymbol{\alpha}, b} \Big\{ C \sum_{i=1}^{n} \big[1 - y_i \big(\sum_j \alpha_j y_j \underbrace{\boldsymbol{\phi}(\mathbf{x}_j) \cdot \boldsymbol{\phi}(\mathbf{x}_i)}_{=K(\mathbf{x}_j, \mathbf{x}_i)} + b \big) \big]_+ + \|\mathbf{w}\|^2 \Big\}$$

alpha?b?min?

- How can we write the regularizer?

$$\|\mathbf{w}\|^2 = \mathbf{w} \cdot \mathbf{w} = \left[\sum_i \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \right] \cdot \left[\sum_j \alpha_j y_j \boldsymbol{\phi}(\mathbf{x}_j) \right]$$

Training kernel SVMs: primal

- We know $\mathbf{w}^* = \sum_i \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i)$, so

$$\min_{\boldsymbol{\alpha}, b} \Big\{ C \sum_{i=1}^{n} \big[1 - y_i \big(\sum_j \alpha_j y_j \underbrace{\boldsymbol{\phi}(\mathbf{x}_j) \cdot \boldsymbol{\phi}(\mathbf{x}_i)}_{=K(\mathbf{x}_j, \mathbf{x}_i)} + b \big) \big]_+ + \|\mathbf{w}\|^2 \Big\}$$

- How can we write the regularizer?

$$\begin{aligned} \|\mathbf{w}\|^2 = \mathbf{w} \cdot \mathbf{w} &= \left[\sum_i \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \right] \cdot \left[\sum_j \alpha_j y_j \boldsymbol{\phi}(\mathbf{x}_j) \right] \\ &= \sum_{i=1, j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

- So everything is a differentiable function of $\boldsymbol{\alpha}$, b

Primal (SGD) or dual (QP)?

- Both are possible. From [Chapelle, 2007]:

  > The historical reasons for which most of the research in the last decade has been about dual optimization are unclear. We believe that it is because SVMs were first introduced in their hard margin formulation [Boser et al., 1992], for which a dual optimization (because of the constraints) seems more natural. In general, however, soft margin SVMs should be preferred, even if the training data are separable: the decision boundary is more robust because more training points are taken into account [Chapelle et al., 2000].

- Rule of thumb: with kernels, QP is often faster. Without kernels, almost always best to use primal (SGD).

Mercer kernels

- What kind of function K is a valid kernel, i.e., such that there exists a feature space $\Phi(\mathbf{x})$ in which $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z})$?
- Theorem due to Mercer (1909): K must be
  - continuous
  - symmetric: $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{z}, \mathbf{x})$
  - positive definite: for any n, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the **kernel matrix**

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & K(\mathbf{x}_1, \mathbf{x}_n) \\ \cdots & \cdots & \cdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

    must be positive semidefinite. (The terminology in common use can be confusing: a "positive definite kernel" is typically defined as having a positive *semidefinite* kernel matrix for any set of data points.)
- Change in outlook: design K instead of designing ϕ

Some popular kernels

- The linear kernel:

$$K(\mathbf{x}, \mathbf{z}) \;=\; \mathbf{x} \cdot \mathbf{z}$$

  This leads to the original, linear SVM.

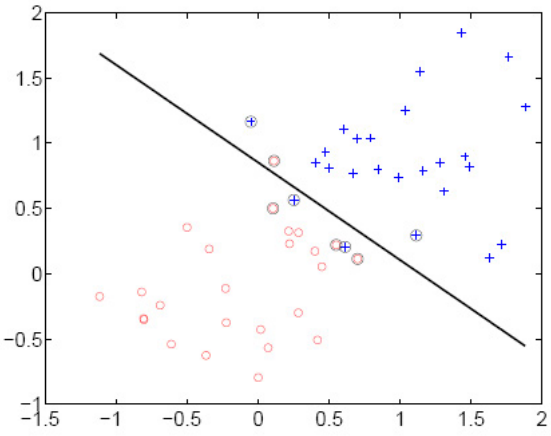
- The polynomial kernel:

$$K(\mathbf{x}, \mathbf{z};\, b, p) \;=\; (b + \mathbf{x} \cdot \mathbf{z})^p$$

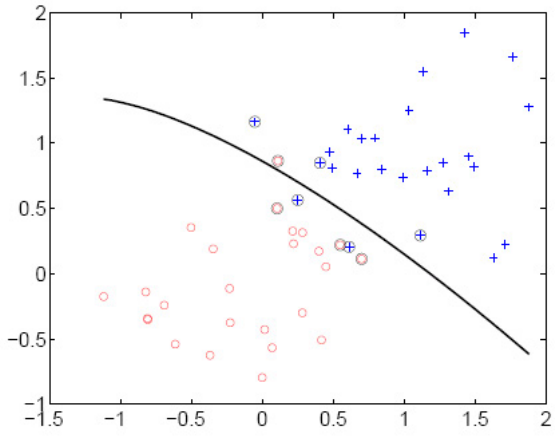
  We can write the expansion explicitly, by concatenating powers up to p and multiplying by appropriate weights.

- How many dimensions are in $\boldsymbol{\phi}(\mathbf{x})$? If $\mathbf{x} \in \mathbb{R}^d$, and $d \gg p$, number of terms grows as d^p.

Example: SVM with polynomial kernel

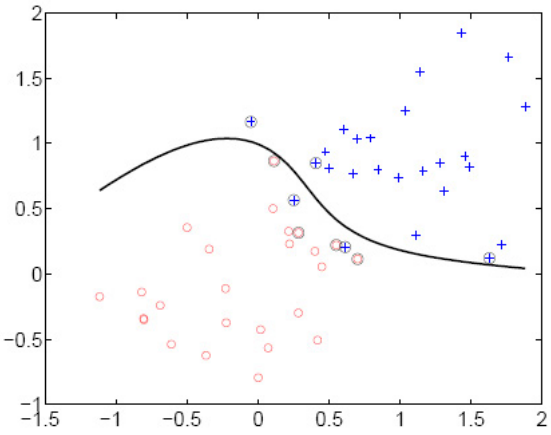


linear

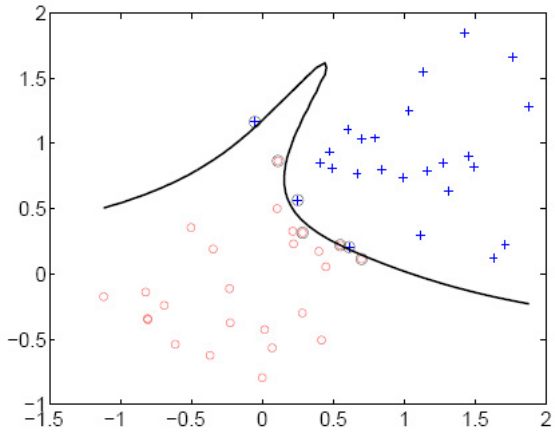


2^{nd} order polynomial

(using $C < \infty$)



4^{th} order polynomial



8^{th} order polynomial

Compare to the effect of model order in regression or logistic regression.

Radial basis function (RBF) kernel

$$K(\mathbf{x}, \mathbf{z}; \sigma) \;=\; \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{z}\|^2\right)$$

- The RBF kernel (also called the "Gaussian kernel") is a measure of similarity between two examples.
  - The feature space is infinite-dimensional!
- What is the role of parameter σ?

Radial basis function (RBF) kernel

$$K(\mathbf{x}, \mathbf{z}; \sigma) \;=\; \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{z}\|^2\right)$$

- The RBF kernel (also called the "Gaussian kernel") is a measure of similarity between two examples.
  - The feature space is infinite-dimensional!
- What is the role of parameter σ? Consider $\sigma \to 0$.

Radial basis function (RBF) kernel

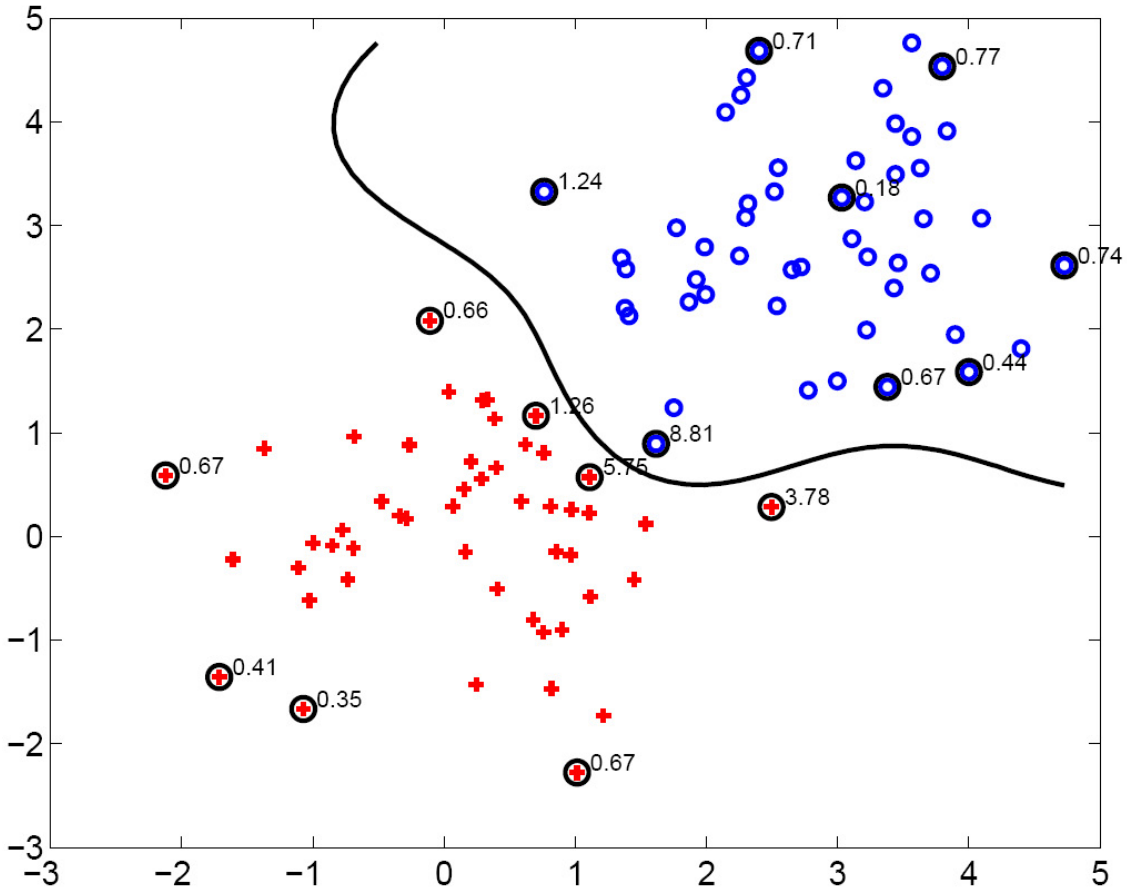
$$K(\mathbf{x}, \mathbf{z}; \sigma) \;=\; \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{z}\|^2\right)$$

- The RBF kernel (also called the "Gaussian kernel") is a measure of similarity between two examples.
  - The feature space is infinite-dimensional!
- What is the role of parameter σ? Consider $\sigma \to 0$.

$$K(\mathbf{x}_i, \mathbf{x}; \sigma) \;\to\; \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_i, \\ 0 & \text{if } \mathbf{x} \neq \mathbf{x}_i. \end{cases}$$

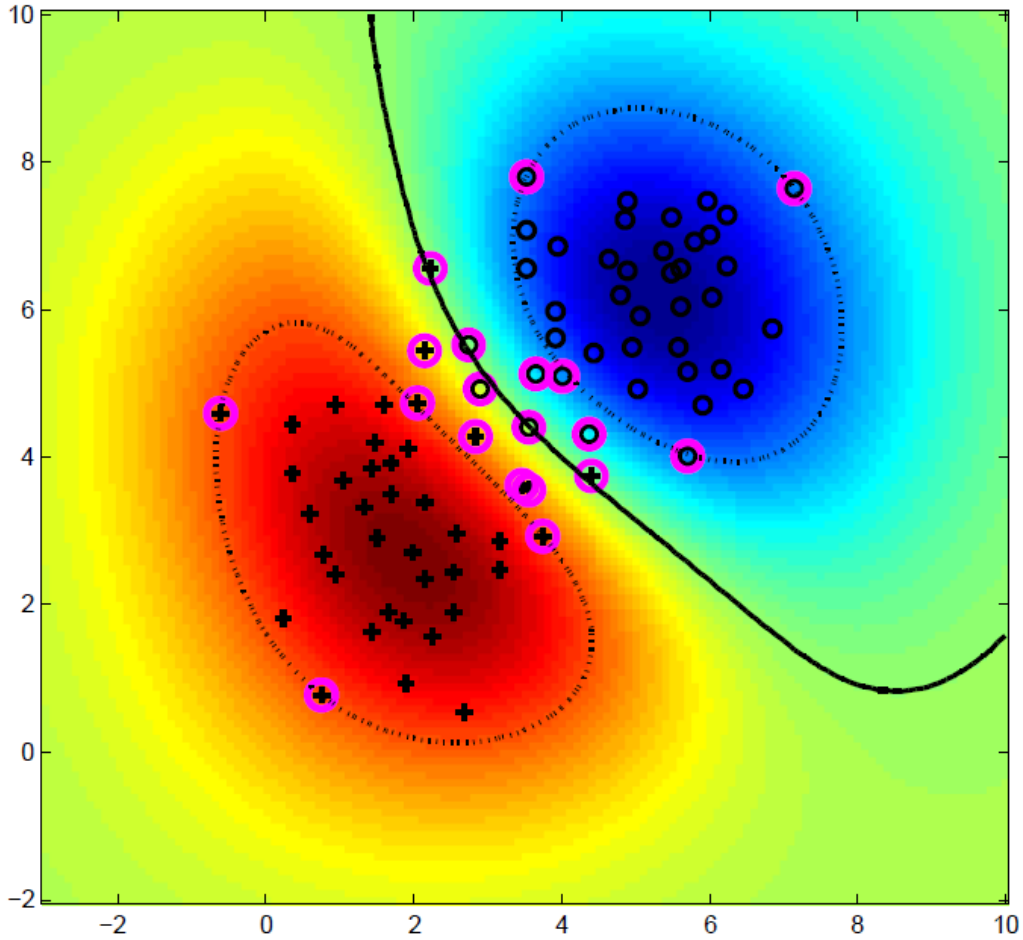
- All examples become SVs $\Rightarrow$ likely overfitting.

SVM with RBF (Gaussian) kernels



- Data are linearly separable in the (infinite-dimensional) feature space
- We don't need to explicitly compute dot products in that feature space – instead we simply evaluate the RBF kernel.

SVM with RBF kernels: geometry



- positive margin: level set

$$\{\mathbf{x} : \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) = 1\}$$

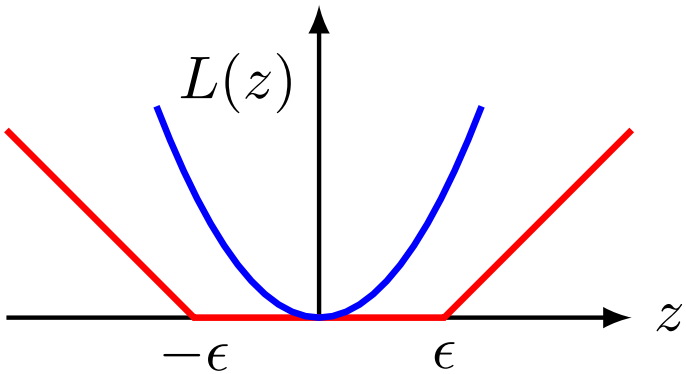
- negative margin: level set

$$\{\mathbf{x} : \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) = -1\}$$

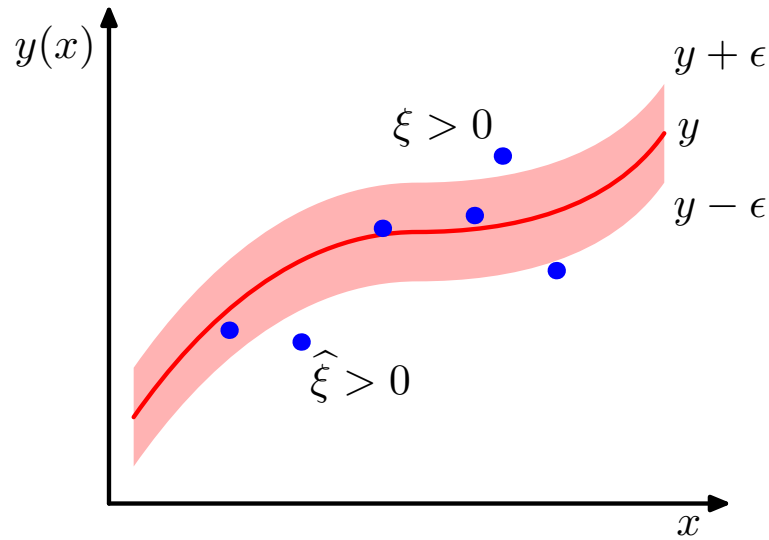
SVM regression

- The key ideas:

ϵ-insensitive loss



ϵ-tube



- Two sets of slack variables:

$$y_i \leq f(\mathbf{x}_i) + \epsilon + \xi_i,$$
$$y_i \geq f(\mathbf{x}_i) - \epsilon - \tilde{\xi}_i, \qquad \xi_i \geq 0,\ \tilde{\xi}_i \geq 0.$$

- Optimization: $\min C \sum_i \left(\xi_i + \tilde{\xi}_i\right) + \frac{1}{2}\|\mathbf{w}\|^2$