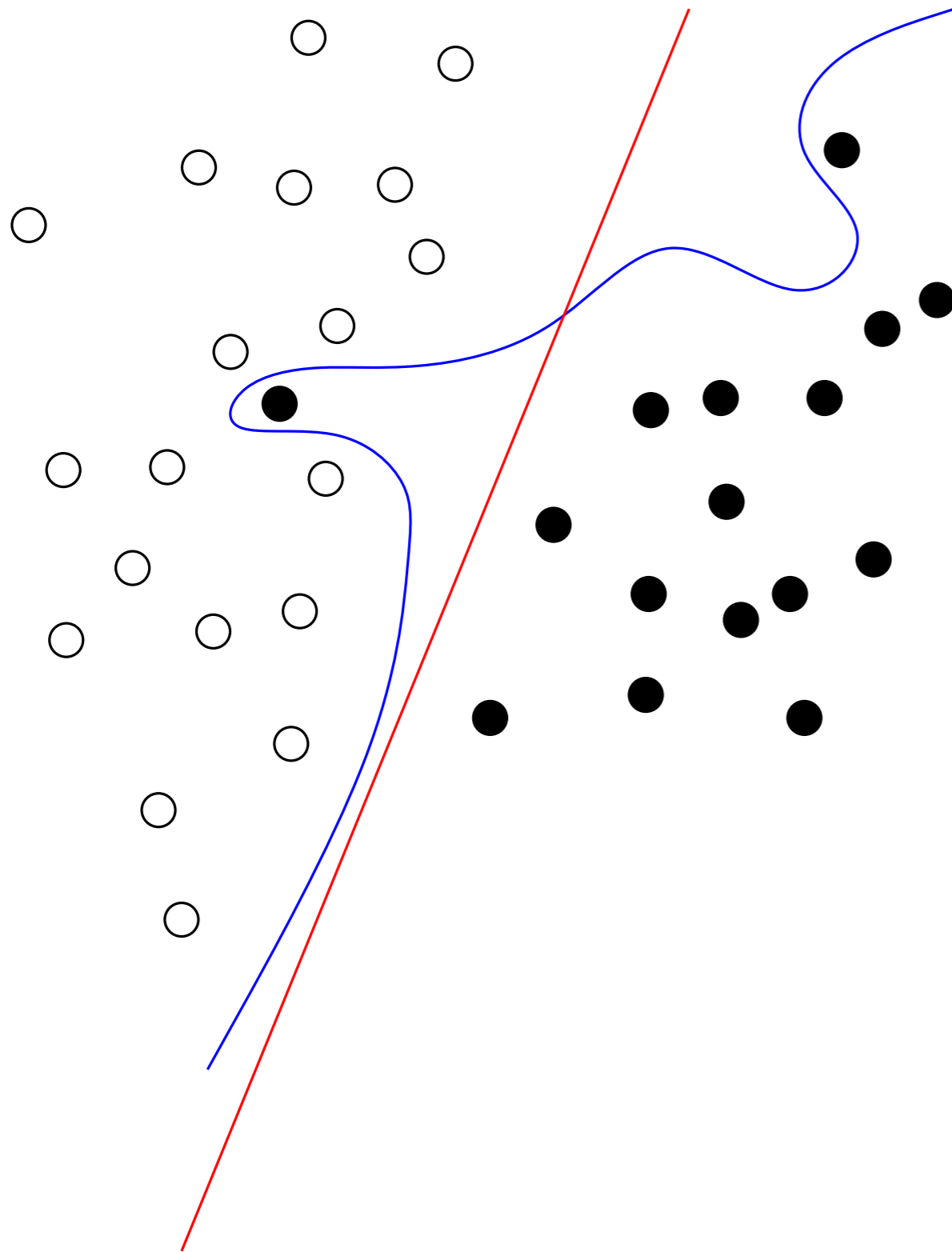
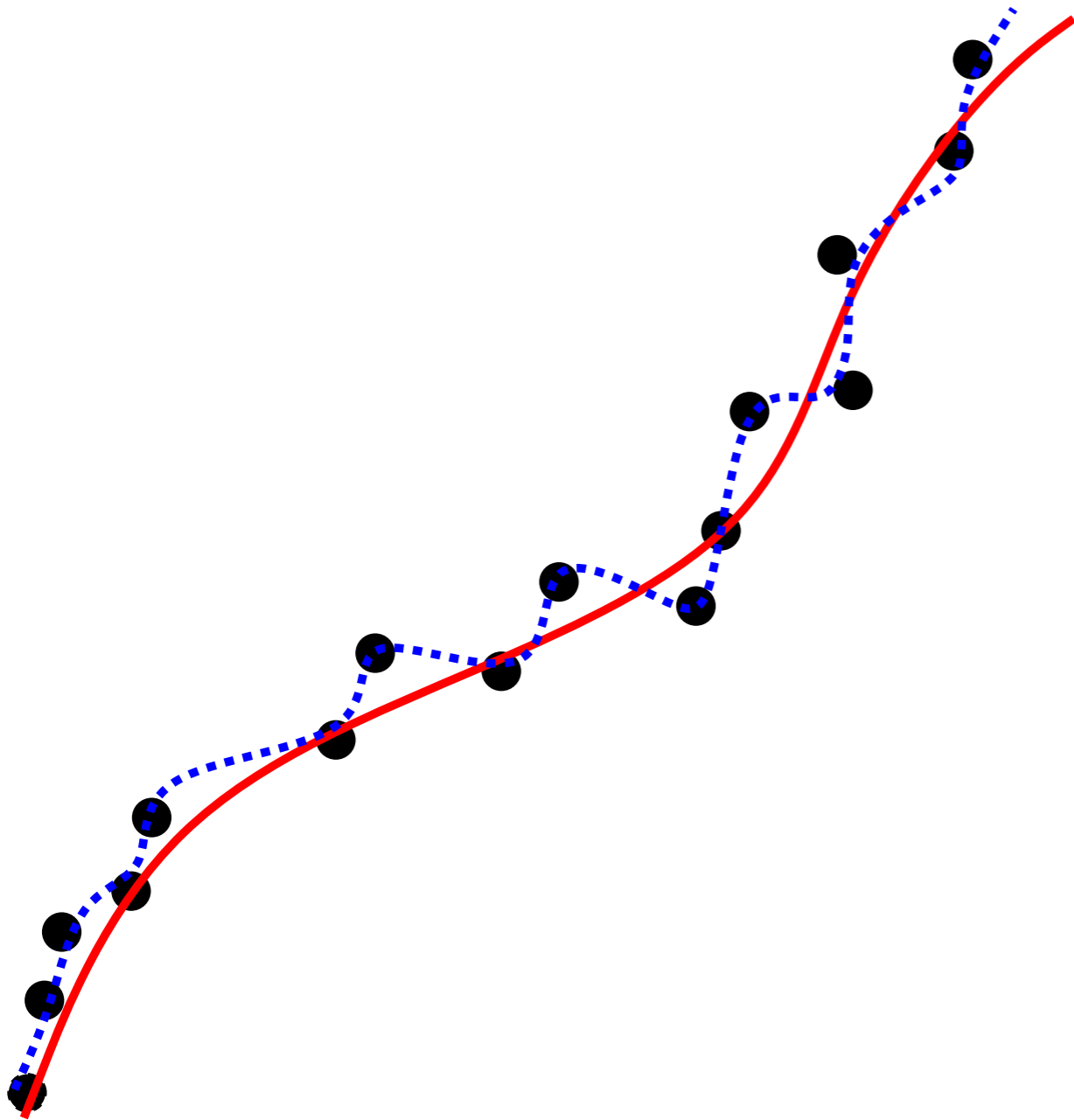


# Recap of Classification Problem



- we wanted to pick a function  $f$  amongst a class  $\mathcal{F}$  of functions which hopefully not only classifies some given *sample*  $S$  (picked according to some probability distribution) but also future points that come along according to the same prob. dist.
- it seemed reasonable not to pick a very complicated function that fits the sample  $S$  exactly as this might decrease performance on future points that come along

# The Regression Problem



- we want to pick a function  $f$  amongst a class  $\mathcal{F}$  of functions which 'well approximates' the given *sample*  $S$  (picked according to some probability distribution) and also lies close to future points
- again it seems reasonable not to pick a very complicated function that fits the sample  $S$  exactly as this might decrease performance on future points that come along

# General plan

- same idea as for classification problem:
  - consider linear case first
  - apply kernel trick to deal with non-linear data

# Linear Regression SVM

- we want to determine a hyperplane of the form  $\mathbf{w} \cdot \mathbf{x} + \mathbf{b} = y$  which closely fits all data points, i.e.

$$y_i - (\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b}) \leq \epsilon$$

$$\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b} - y_i \leq \epsilon$$

assuming such a function exists which approximates the data up to  $\epsilon$ -precision

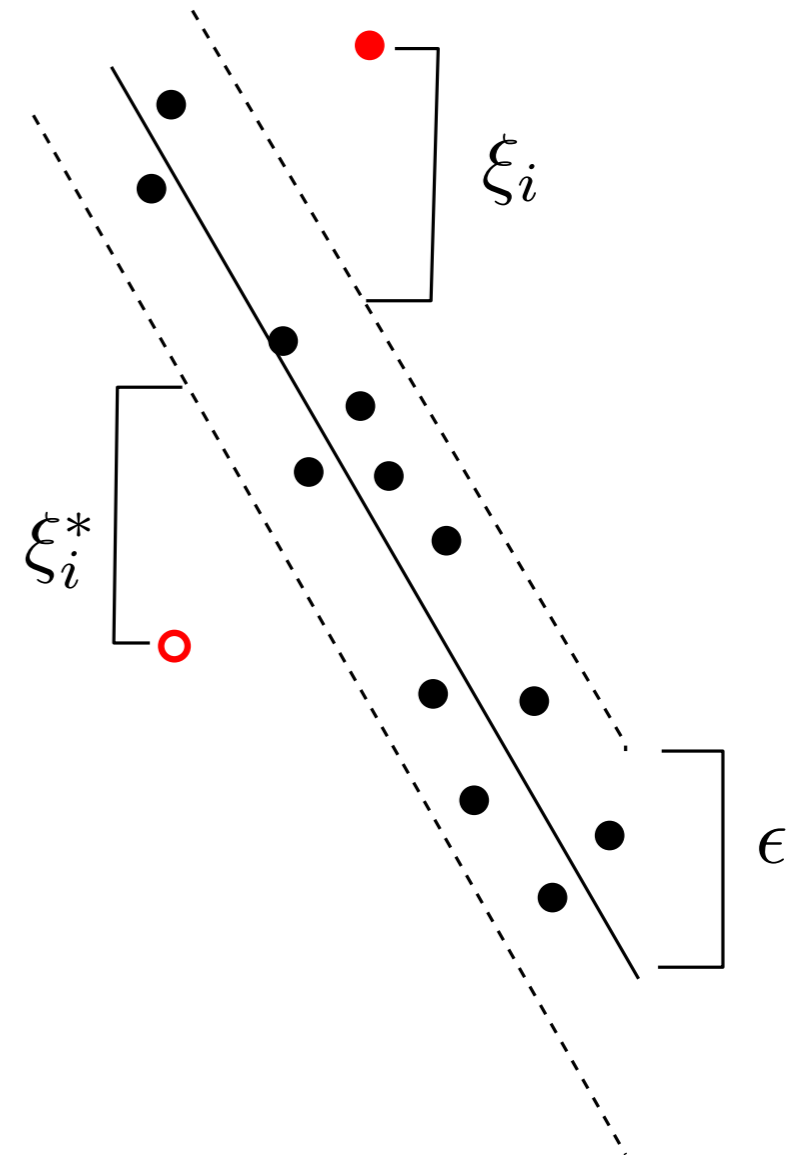
- analogously to the 'soft margin' classifier, we can also allow for 'outliers' with penalties

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum (\xi_i + \xi_i^*)$$

$$y_i - (\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b}) \leq \epsilon + \xi_i$$

$$\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b} - y_i \leq \epsilon + \xi_i^*$$

$\xi_i, \xi_i^* \geq 0$   $C$  determines the weight of the penalties for the outliers



# Lagrangian Formulation

- We replace the original problem by its Lagrangian formulation (penalizing violations of the constraints)

$$L_P \equiv \frac{1}{2} \|\mathbf{w}\|^2 + C \sum (\xi_i + \xi_i^*) - \sum_{i=1}^l \alpha_i (\epsilon + \xi_i - y_i + x_i \cdot \mathbf{w} + b) \\ - \sum_{i=1}^l \alpha_i^* (\epsilon + \xi_i^* + y_i - x_i \cdot \mathbf{w} - b) - \sum \eta_i \xi_i + \eta_i^* \xi_i^*$$

with dual variables  $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0$ .

which needs to be minimized (wrt  $\mathbf{w}, \mathbf{b}, \xi_i, \xi_i^*$ ; requiring that the partial derivatives wrt  $\alpha_i, \alpha_i^*, \eta_i, \eta_i^*$  vanish)

# Lagrangian Dual

- in the respective dual we *maximize*  $L_P$  s.t. the gradients wrt to  $w, b, \xi_i, \xi_i^*$  vanish, i.e. we obtain the conditions

$$\sum (\alpha_i^* - \alpha_i) = 0$$

$$w - \sum (\alpha_i - \alpha_i^*) x_i = 0$$

$$C - \alpha_i^{(*)} - \eta_i^{(*)} = 0$$

- Substitution yields

$$\max L_D = -\frac{1}{2} \sum_{i,j} (\alpha_i - \alpha_i^*) (\alpha_j - \alpha_j^*) x_i \cdot x_j$$

$$-\epsilon \sum (\alpha_i + \alpha_i^*) + \sum y_i (\alpha_i - \alpha_i^*)$$

subject to

$$\sum (\alpha_i - \alpha_i^*) = 0$$

$$\alpha_i, \alpha_i^* \in [0, C]$$

# Lagrangian Dual

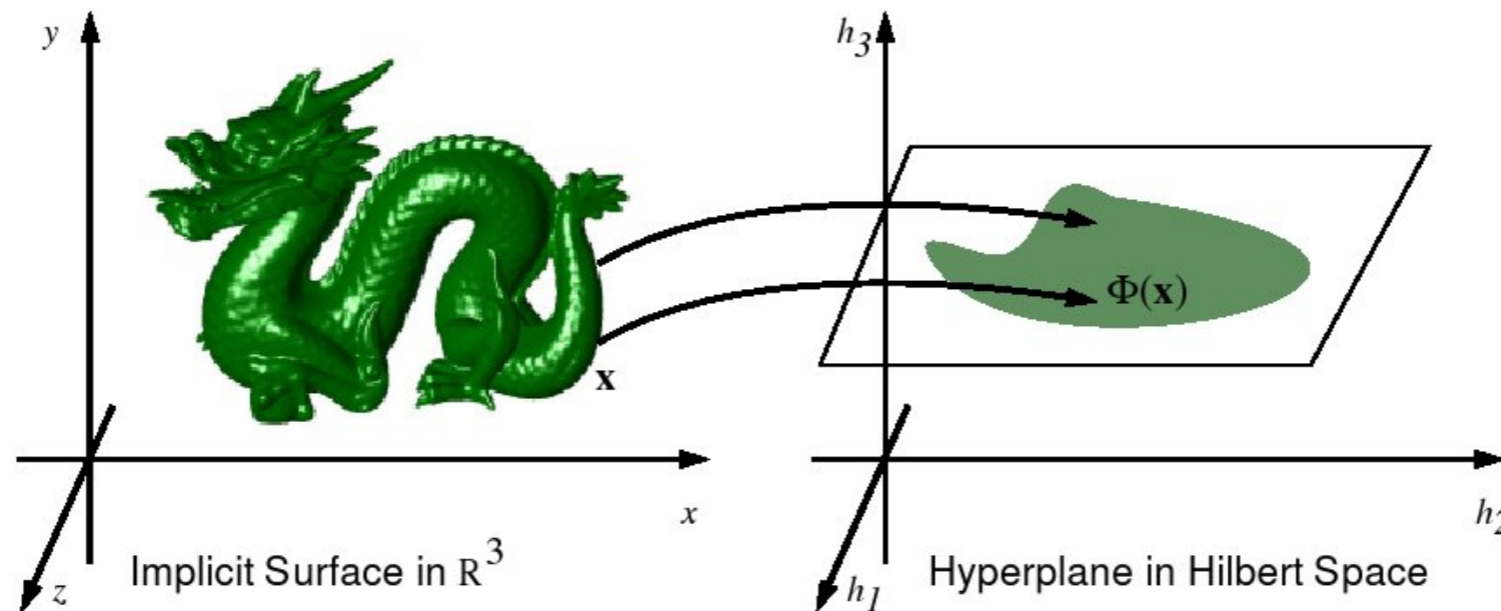
- the  $y$ -coordinate of new points can be predicted as

$$f(x) = w^T x + b = \sum (\alpha_i - \alpha_i^*) x_i \cdot x + b$$

- where  $b$  can be computed via the Karush-Kuhn-Tucker complementary slackness conditions
- again, we have a formulation where the input only appears as dot products
- the kernel trick can be applied as in the case of the classifier SVMs

# Surface Reconstruction via Regression SVMs

- Paper by Steinke, Schölkopf, Blanz (EUROGRAPHICS'05)
- Goal: given a set of points (samples) from the surface of a volume, compute a function which implicitly represents the surface (surface  $\equiv$  zero set)
- Solution via Regression SVM by mapping the data to some space  $\mathcal{H}$ , where the original surface becomes a hyperplane



**Figure 1:** The mapping  $\Phi$  defined by the kernel function  $k$  (Equation 1) transforms the 3D surface to a hyperplane in a high-dimensional Hilbert space.

# Setting Up the SVM

- represent the surface as the zero set of

$$f(x) = w \cdot \Phi(x) + b$$

in the space  $\mathcal{H}$

- this hyperplane will approximately pass through the given surface points mapped to  $\mathcal{H}$  via  $\Phi$
- Idea: training data consists of points  $x_1, \dots, x_n$  in the original space with "offsets"  $y_1, \dots, y_n$  from the surface; ( $y_i = 0 \Rightarrow x_i$  is on the surface)
- Why not just use original point sample with  $y_i = 0$  ?

# Generating Training data

- Problem: this would support the 'trivial' function  $f(x) = 0$
- Rescue: Generate additional, artificial data points with non-zero  $y_i$  values by displacing surface points along their surface normals by distance  $y_i$
- $\Rightarrow$  prevents learning of the zero function
- Disadvantage: requires estimation/knowledge of surface normals and orientation
- apply standard SVM regression with a polynomial kernel

# Examples



**Figure 3:** An implicit surface reconstruction of the dragon of the Stanford 3D Scanning Repository. The original mesh has more than 400k data points. Our reconstruction uses 280k kernel centers.



**Figure 4:** Holes due to occlusions in the scanning process (left) are filled by the implicit surface (right).

- surprisingly fast: Dragon model (405k points) takes around 5min on a 2.2 GHz machine
- implicit representation to some degree repairs insufficient surface scans
- also behaves nicely wrt outliers/noise; smoothing can easily be incorporated

# Face detection via SVMs

- Paper by E. Osuna, R. Freund, and F. Girosi. at CVPR'97
- Goal: decide for small  $19 \times 19$  grayscale bitmaps 'face or not?'
- training using database of face/non-face pixel patterns after histogram normalization
- interpretation of  $19 \times 19$  pixel patterns as elements in 361-dimensional space
- polynomial kernel of degree 2
- retraining using misclassified images

# Results

- routine built into face recognition system, which scales and decomposes image into  $19 \times 19$  tiles searching for faces
- system is reported to achieve a detection rate between 75% and 97% with very few false positives
- compares apparently favourably to state of the art FR systems at the time
- running time not clearly stated

