CS484 - CS555: Introduction to Computer Vision

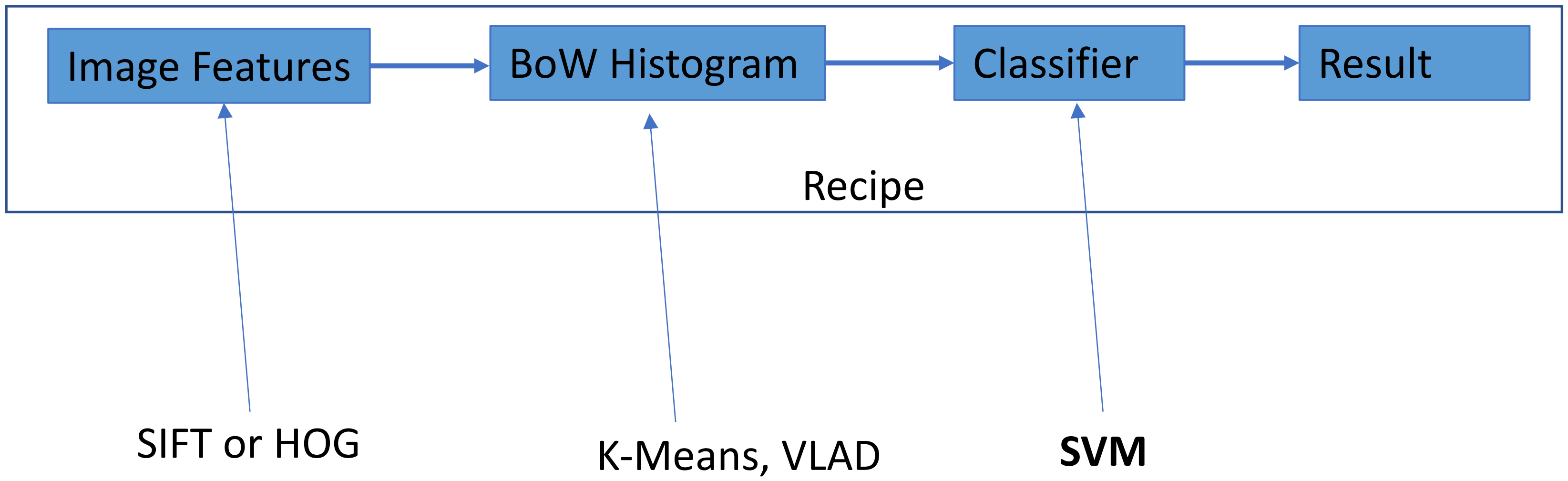
Shape Analysis with Kernel Machines

Dr. Sedat OZER

Today

Kernel Machines

Remember from the first lecture:



Outline

1. Fundamentals: Discussion
   - Machine Learning?
   - Supervised Learning?
   - Unsupervised Learning?
   - Pattern?

Outline

1. <u>Fundamentals: Discussion</u>
   - Machine Learning?
   - Supervised Learning?
   - Unsupervised Learning?

2. <u>Towards SVM… Even more discussion</u>
   - Classification vs Regression?
   - Structural Risk Minimization?

Outline

1. Fundamentals: Discussion
   - Machine Learning?
   - Supervised Learning?
   - Unsupervised Learning?
   - Pattern?
2. Towards SVM… Even more discussion
   - Classification vs Regression?
   - Structural Risk Minimization?
3. Finally… SVM Classification

Outline

1. Fundamentals: Discussion
   - Machine Learning?
   - Supervised Learning?
   - Unsupervised Learning?
   - Pattern?
2. Towards SVM… Even more discussion
   - Classification vs Regression?
   - Structural Risk Minimization?
3. Finally… SVM Classification
4. Similarity Domains Machine and MKL

Support Vector Machines

- A machine learning algorithm
  - A classifier (also a regression) algorithm

- The era of (mostly): between late 1990s and late 2000s

Remember: You do not always need “deep” algorithms!

Questions:

- What is supervised learning?
- What is unsupervised learning?
- What is classification?
- What is regression?
- What is structural risk minimization?

Fundamentals: Structural Risk Minimization

Empirical Risk Minimization

$$R(h) = \mathbf{E}[L(h(x), y)] = \int L(h(x), y)\, dP(x, y).$$

$$R_{\mathrm{emp}}(h) = \frac{1}{m}\sum_{i=1}^{m} L(h(x_i), y_i).$$

where h()=f(): the hypothesis function (the model)
L: Loss function, P: joint distribution

Fundamentals: Structural Risk Minimization (old school ☺)

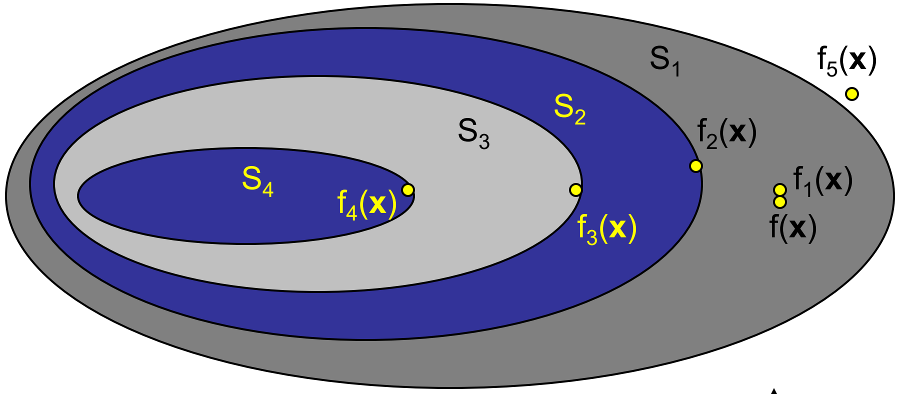


Figure 1: Dividing The Hypothesis Space into nested complexity subsets

Empirical Risk Minimization

$$R(h) = \mathbf{E}[L(h(x), y)] = \int L(h(x), y)\, dP(x, y).$$

$$R_{\text{emp}}(h) = \frac{1}{m}\sum_{i=1}^{m} L(h(x_i), y_i).$$

where h()=f(): the hypothesis function (the model)
L: Loss function, P: joint distribution

VC Confidence

$$R \le R_{emp} + \sqrt{\frac{n\ln(\frac{2m}{n}+1) - \ln(\frac{\delta}{4})}{m}}$$

With (1-δ) probability this inequality holds! Where n is the VC dimension.

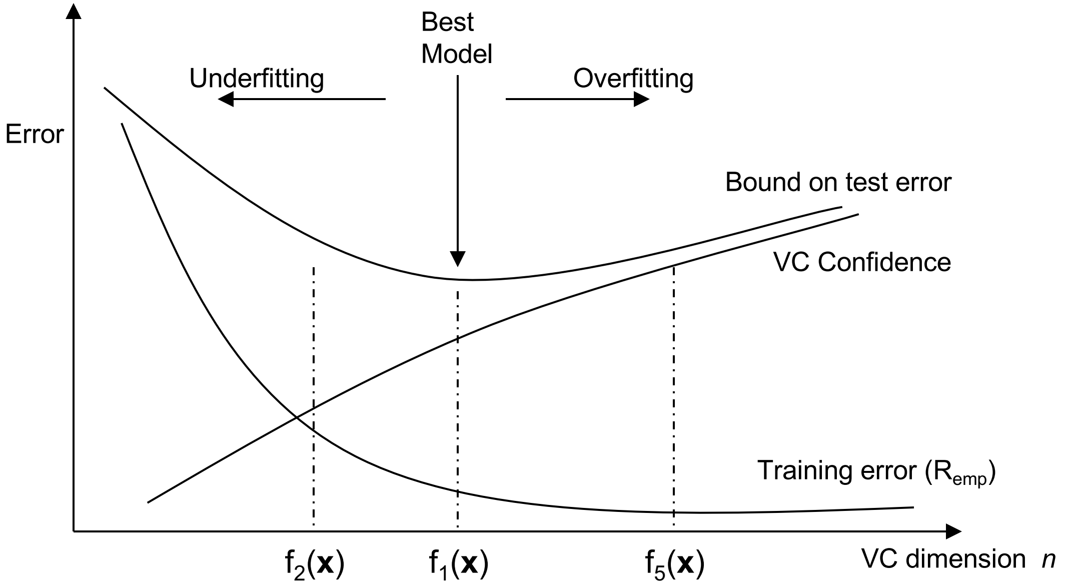
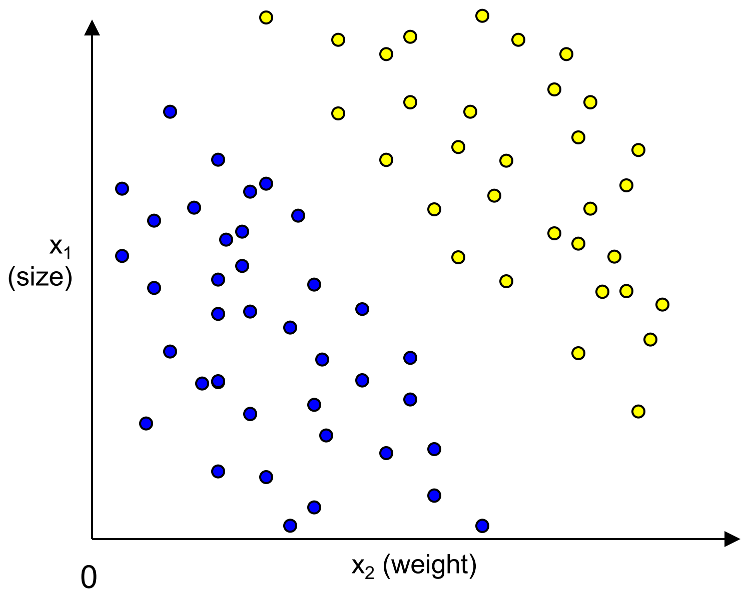


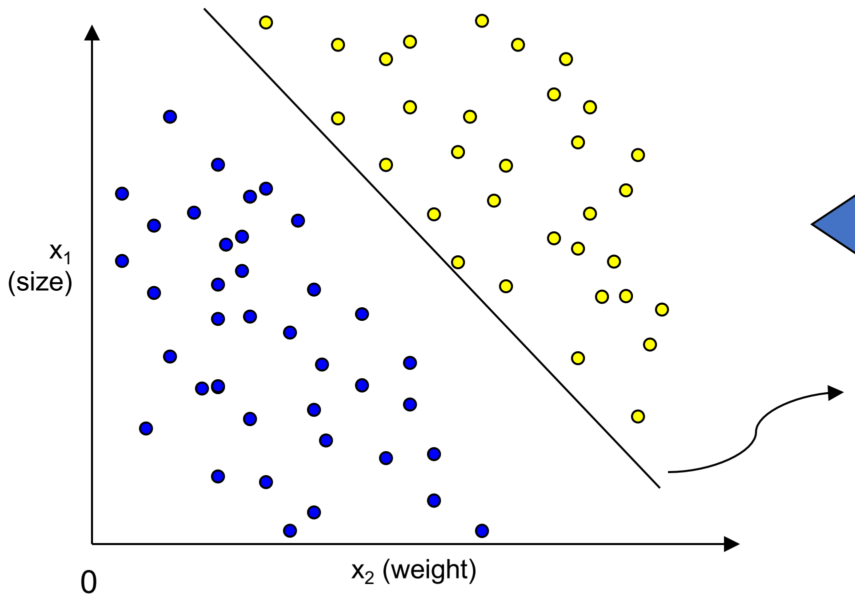
Figure 2: SRM principle

Fundamentals: Support Vector Machines



How to separate this data?

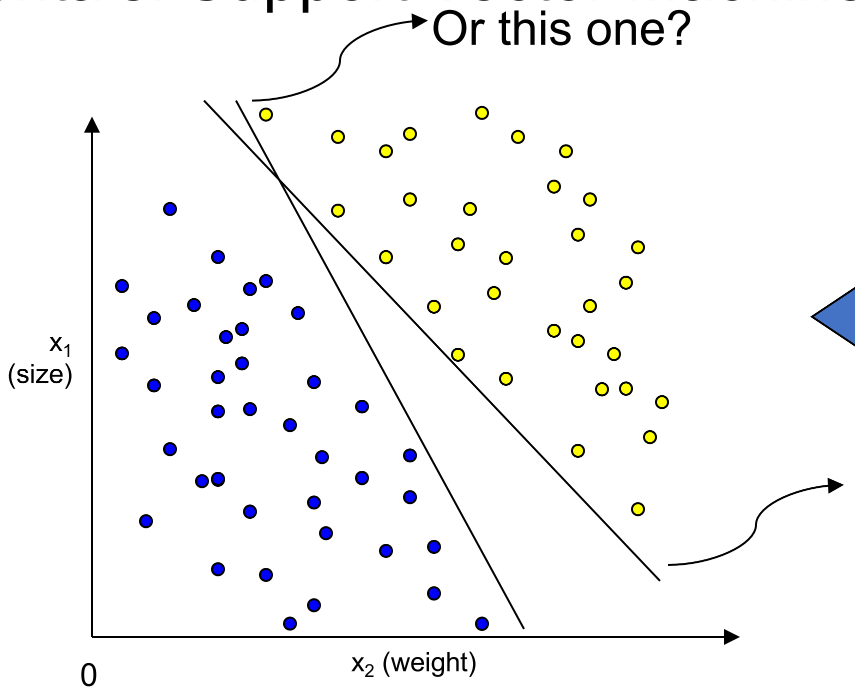
Fundamentals: Support Vector Machines



How to separate this data?

By drawing this line?

Fundamentals: Support Vector Machines



How to separate this data?

Fundamentals: Support Vector Machines

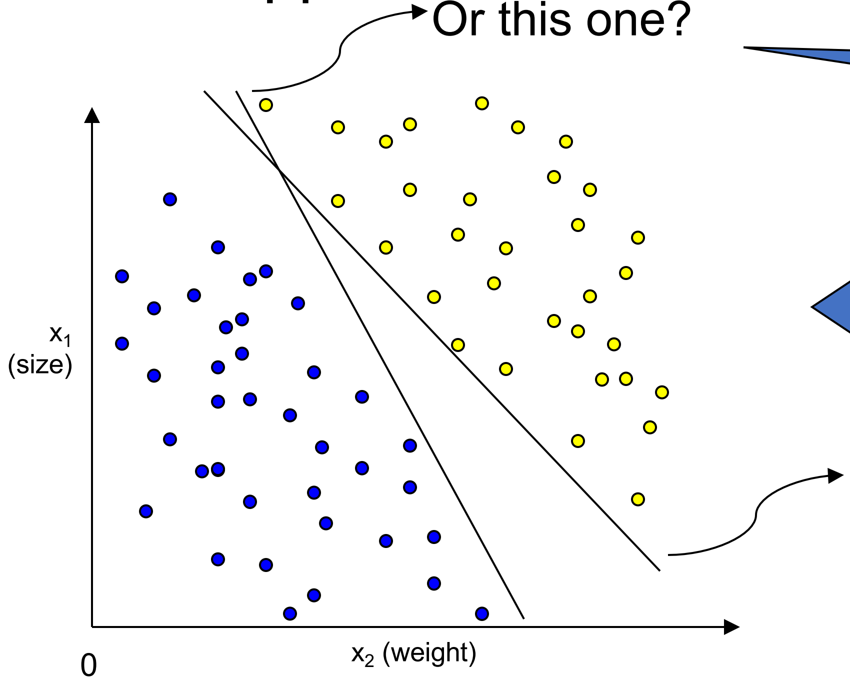


Figure: Which line is the optimal decision line

Errmm, c'mon
does it really matter?

How to separate this data?

By drawing this line?

Fundamentals: Support Vector Machines

Or this one?

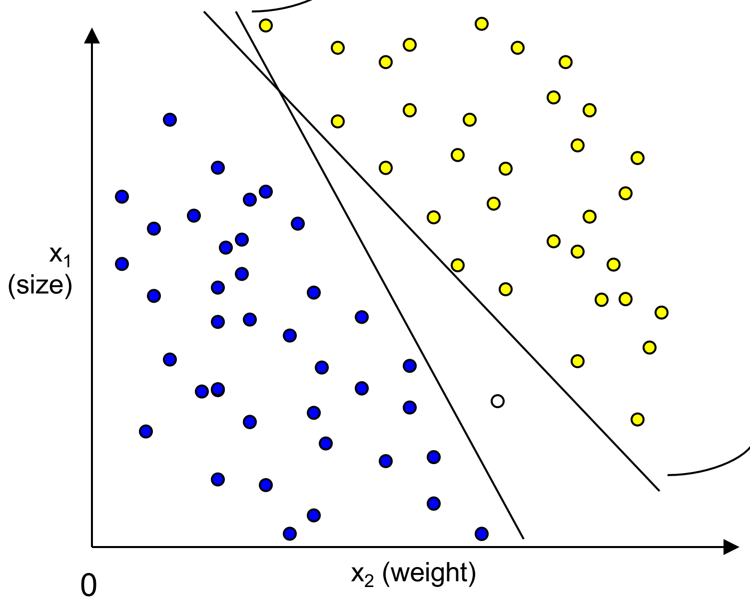


Figure: Which line is the optimal decision line

By drawi

d

Errmm, c'mon
does it really matter?

Answer: it may not ☺
if we are lucky. (lucky?
What is the probability
of being lucky?)

Fundamentals: Support Vector Machines

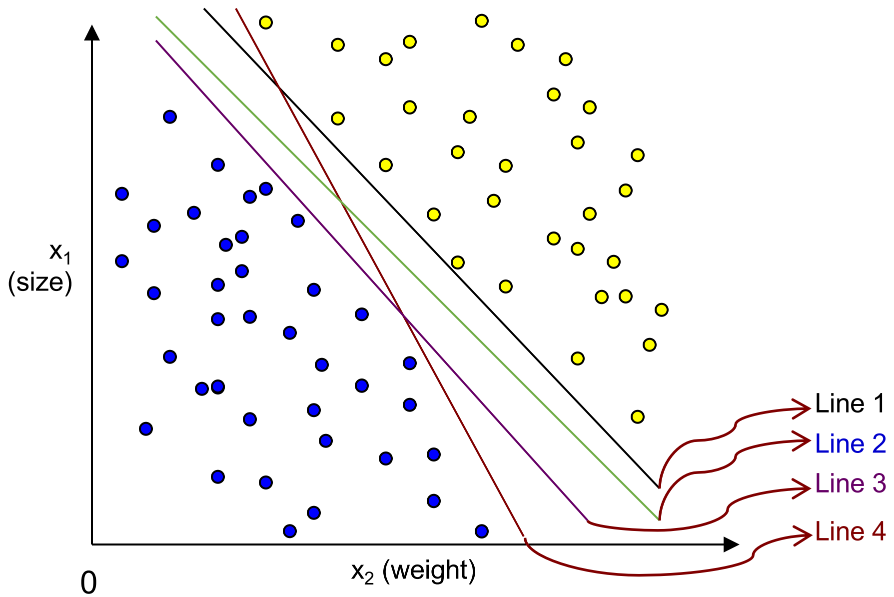


Figure: Which line is the optimal decision line

Which line?

Fundamentals: Support Vector Machines

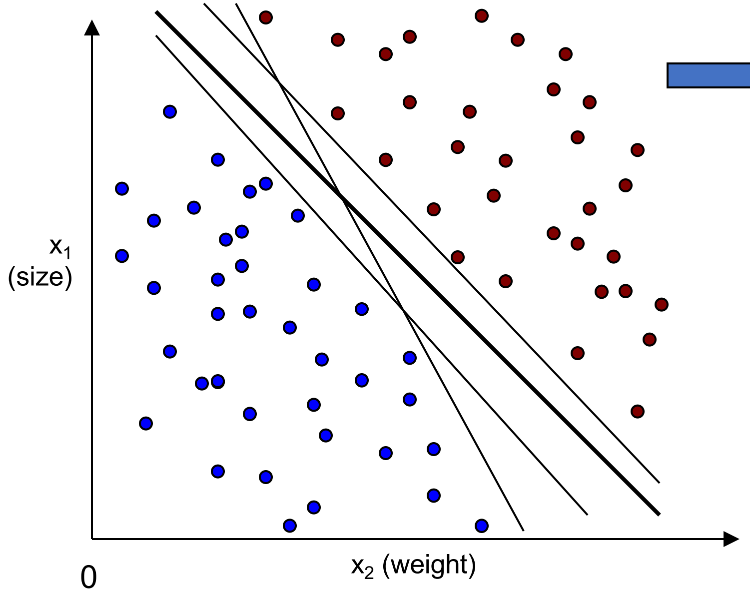


Figure: Which line is the optimal decision line

Which line?

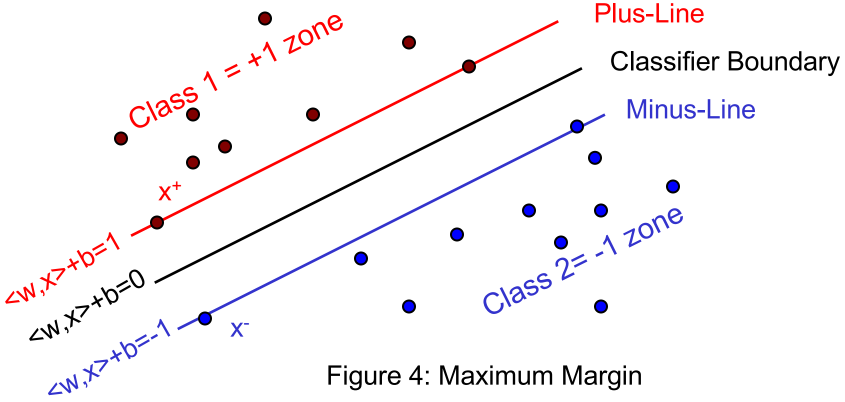


Figure 4: Maximum Margin

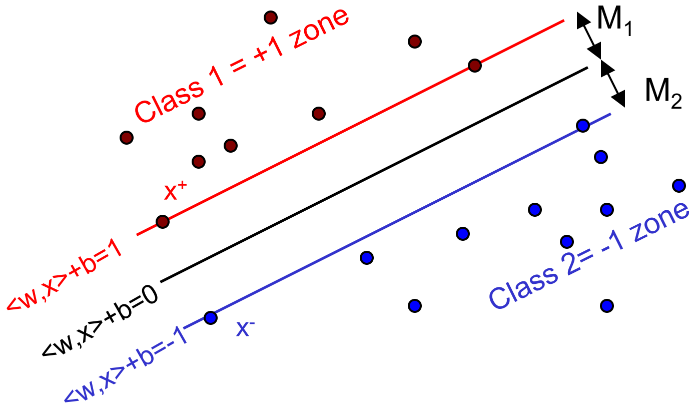


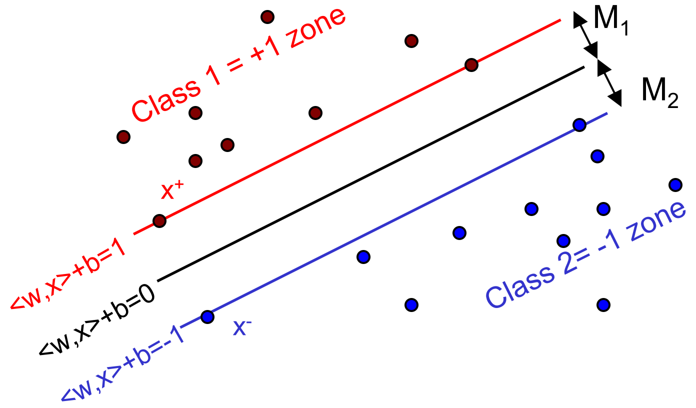
Figure 5: Maximum Margin Width

Fundamentals: Support Vector Machines

$$M = M_1 + M_2 = \frac{\left| < \mathbf{w}, \mathbf{x}^{+} > + b \right|}{\|\mathbf{w}\|} + \frac{\left| < \mathbf{w}, \mathbf{x}^{-} > + b \right|}{\|\mathbf{w}\|}$$

$$M = \frac{|1|}{\|\mathbf{w}\|} + \frac{|-1|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

$$y_i[< \mathbf{w}, \mathbf{x}_i > + b] \geq 1$$



Can you use these two conditions, ideas to find the parameters?

Fundamentals: Support Vector Machines

$$M = M_1 + M_2 = \frac{\left| < \mathbf{w}, \mathbf{x}^{+} > + b \right|}{\|\mathbf{w}\|} + \frac{\left| < \mathbf{w}, \mathbf{x}^{-} > + b \right|}{\|\mathbf{w}\|}$$

$$M = \frac{|1|}{\|\mathbf{w}\|} + \frac{|-1|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

$$y_i[< \mathbf{w}, \mathbf{x}_i > + b] \geq 1$$

Can you use these two conditions, ideas to find the parameters?

Build a cost function and use optimization

Fundamentals: Support Vector Machines

$$\left.\begin{aligned} M &= \frac{|1|}{\|\mathbf{w}\|} + \frac{|-1|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \\ & y_i[<\mathbf{w},\mathbf{x}_i> + b] \geq 1 \end{aligned}\right\} L(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l}\alpha_i\left[y_i\left(<\mathbf{w},\mathbf{x_i}> + b\right) - 1\right]$$

Fundamentals: Support Vector Machines

Lagrange multiplier

$$\left.\begin{aligned} M &= \frac{|1|}{\|\mathbf{w}\|} + \frac{|-1|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \\ & y_i[<\mathbf{w},\mathbf{x}_i> + b] \geq 1 \end{aligned}\right\} \quad L(\mathbf{w}, b, \boxed{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \boxed{\alpha_i} \left[y_i \left(<\mathbf{w}, \mathbf{x}_\mathbf{i}> + b \right) - 1 \right]$$

Fundamentals: Support Vector Machines

$$\left.\begin{aligned} M &= \frac{|1|}{\|\mathbf{w}\|} + \frac{|-1|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \\ & y_i[<\mathbf{w},\mathbf{x}_i> + b] \geq 1 \end{aligned}\right\} \quad L(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i \left(<\mathbf{w},\mathbf{x_i}> + b \right) - 1 \right]$$

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial b} = 0 \Rightarrow 0 = \sum_{i=1}^{l} \alpha_i y_i$$

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$$

Fundamentals: Support Vector Machines

$$L(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l}\alpha_i\left[y_i\left(<\mathbf{w},\mathbf{x_i}> + b\right) - 1\right]$$

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial b} = 0 \Rightarrow 0 = \sum_{i=1}^{l}\alpha_i y_i$$

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{l}\alpha_i y_i \mathbf{x}_i$$

Fundamentals: Support Vector Machines

$$L(\mathbf{w},b,\alpha)=\frac{1}{2}\|\mathbf{w}\|^2-\sum_{i=1}^{l}\alpha_i\left[y_i\left(<\mathbf{w},\mathbf{x_i}>+b\right)-1\right]$$

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial b}=0\Rightarrow 0=\sum_{i=1}^{l}\alpha_i y_i$$

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial \mathbf{w}}=0\Rightarrow \mathbf{w}=\sum_{i=1}^{l}\alpha_i y_i \mathbf{x}_i$$

$$\phi(\alpha)=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j <\mathbf{x_i},\mathbf{x_j}>$$

Fundamentals: Support Vector Machines

Dual cost function (maximize)

Optimization part

(for **training**)

$$\phi(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j < \mathbf{x_i}, \mathbf{x_j} >$$

$$\boxed{\begin{gathered} \sum_{i=1}^{l} \alpha_i y_i = 0 \\ \alpha_i \geq 0 \end{gathered}}$$

Dual constraints

k: total number of Support Vectors (SV)

Decision function: $f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{k} \alpha_i y_i < \mathbf{x}, \mathbf{x}_i > +b \right)$

Fundamentals: Support Vector Machines

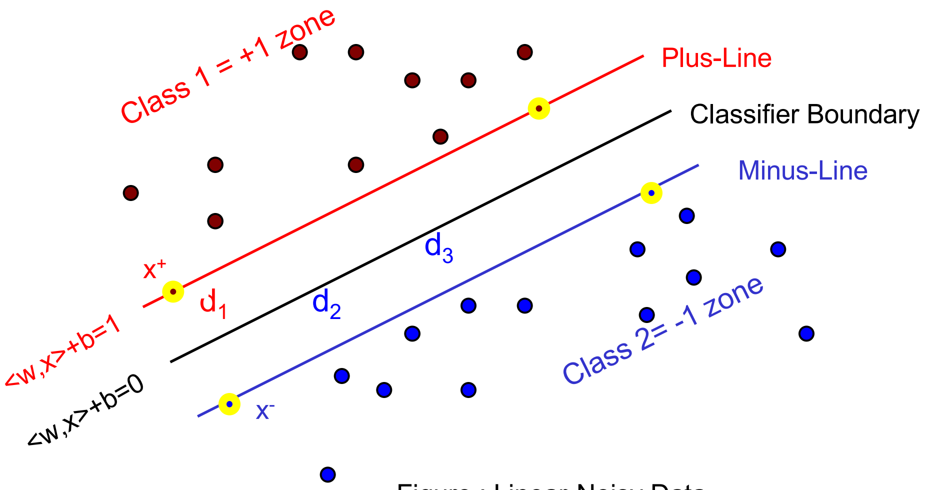


Figure : Linear-Noisy Data

Typically, SVs are laying on the planes…

Fundamentals: Support Vector Machines (C-SVM)

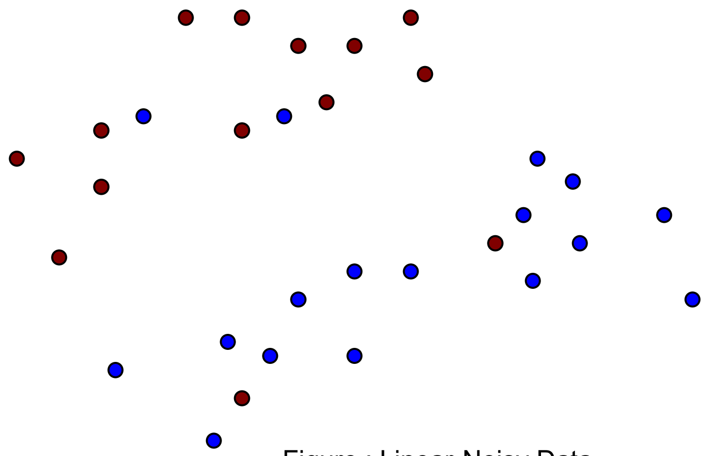


Figure : Linear-Noisy Data

$$d_i = \frac{\xi_i}{\|\mathbf{w}\|} \qquad y_i[< \mathbf{w}, \mathbf{x_i} > +b] \geq 1 - \xi_i$$

$$\phi(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} \xi_i$$

Fundamentals: Support Vector Machines (C-SVM)

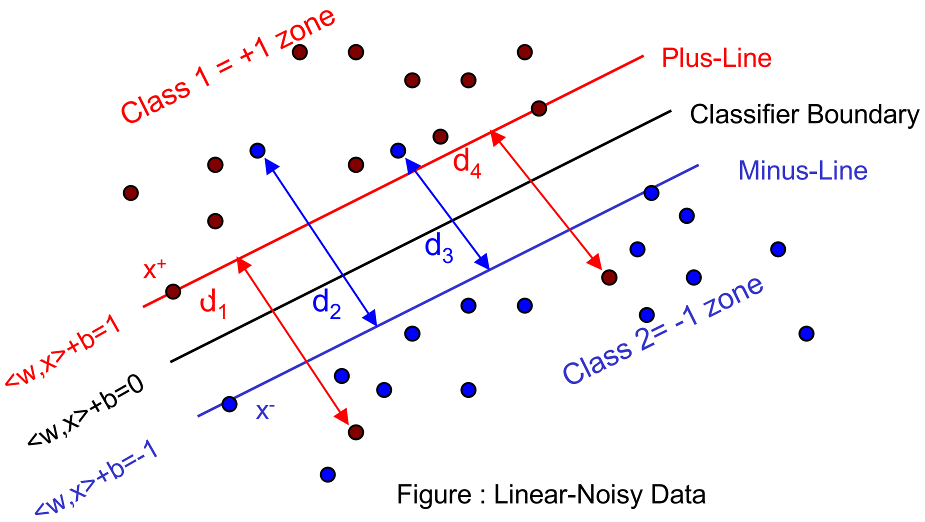


Figure : Linear-Noisy Data

$$d_i = \frac{\xi_i}{\|\mathbf{w}\|} \qquad y_i[<\mathbf{w},\mathbf{x_i}>+b] \geq 1-\xi_i$$

$$\phi(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i$$

Fundamentals: Support Vector Machines (C-SVM)

$$\phi(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j < \mathbf{x_i}, \mathbf{x_j} >$$

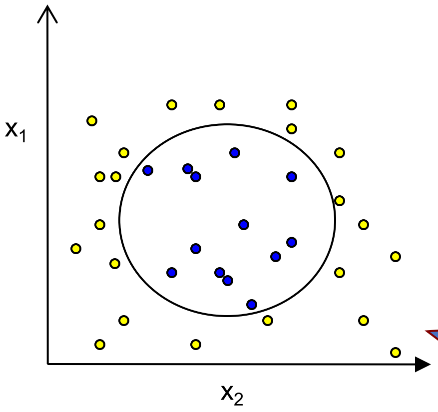
$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

Use optimization libraries (such as Matlab libraries, MOSEK, etc) to solve this quadratic constraint optimization problem.

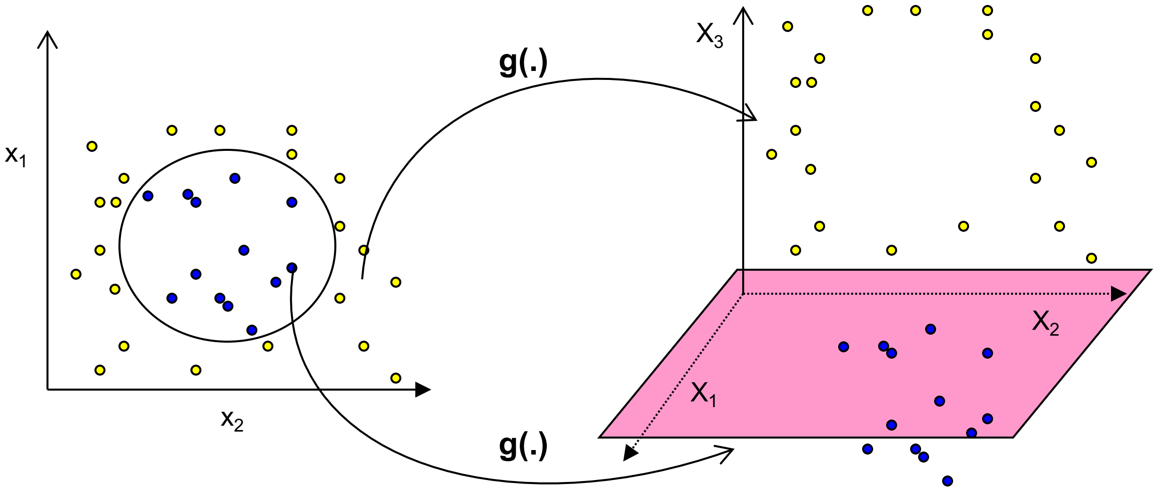
Or… Use sequential minimal optimization (SMO) method.
Or… Use gradient descent ☺
Or…

Fundamentals: Support Vector Machines: Nonlinear Data



Problem: SVM is designed to separate linear data, so how can we apply SVM onto non-linear data?

Fundamentals: Support Vector Machines: Nonlinear Data



(a):Nonlinear Data in 2 dimensional space

(b):Transformed Linearly Separable Data in 3dimensional space

Figure: Transforming the Data by using the transformation function **g(.)**

Fundamentals: Support Vector Machines: Kernel Functions

Decision Function: $$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i y_i < \mathbf{x}, \mathbf{x}_i > + b \right)$$

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i y_i < \mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{x}_\mathbf{i}) > + b \right)$$

Fundamentals: Support Vector Machines: Kernel Functions

Decision Function: $$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i y_i < \mathbf{x}, \mathbf{x}_i > +b \right)$$

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i y_i < \mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{x}_\mathbf{i}) > +b \right)$$

Kernel function

DEFINITION: $$\boxed{K}(\mathbf{x}, \mathbf{x}_\mathbf{i}) = \langle \mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{x}_\mathbf{i}) \rangle$$

Fundamentals: Support Vector Machines: Kernel Trick

Decision Function: $$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i y_i < \mathbf{x}, \mathbf{x}_i > +b\right)$$

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i y_i < \mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{x_i}) > +b\right)$$

Kernel function

$$\boxed{K}(\mathbf{x}, \mathbf{x_i}) = \langle \mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{x_i}) \rangle$$

Decision Function (New form) : $$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i y_i \boxed{K}(\mathbf{x}, \mathbf{x_i}) + b\right)$$

Kernel trick!

Fundamentals: Kernel Functions

$$K(\mathbf{x}, \mathbf{x_i}) \equiv \langle \mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{x_i}) \rangle$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$$

Fundamentals: Kernel Functions

$$K(\mathbf{x}, \mathbf{x_i}) \equiv \langle \mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{x_i}) \rangle$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$$

Mercer Kernels: Mercer Condition

$$\iint K(x, y) g(x) g(y) dx dy \geq 0$$

Fundamentals: Support Vector Machines (C-SVM)

$$\phi(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j < g(\mathbf{x_i}), g(\mathbf{x_j}) >$$

$$\phi(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j})$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

Use optimization libraries (such as Matlab libraries, MOSEK, etc) to solve this quadratic constraint optimization problem.

Or… Use sequential minimal optimization (SMO) method.

Fundamentals: Kernel Function Examples

1. Polynomial Kernel Functions:

$$K(\mathbf{x},\mathbf{x}_i) = \left(\frac{\langle \mathbf{x},\mathbf{x}_i \rangle}{c}\right)^d \qquad K(\mathbf{x},\mathbf{x}_i) = \left(\frac{\langle \mathbf{x},\mathbf{x}_i \rangle + 1}{c}\right)^d$$

2. Wavelet Kernel Functions:

$$K(\mathbf{x},\mathbf{y}) = \prod_{i=1}^{m} \left(Cos\left(1.75\frac{x_i - y_i}{a}\right) \exp\left(-\frac{\left\| x_i - y_i \right\|^2}{2a^2}\right)\right)$$

3. Gaussian Kernel Function: (Radial Basis Function)

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

Fundamentals: Valid Kernel Properties

1. Linear combinations of Kernels

$$K(\mathbf{x},\mathbf{y}) = a_1 K_1(\mathbf{x},\mathbf{y}) + a_2 K_2(\mathbf{x},\mathbf{y})$$

$$a_1, a_2 \geq 0$$

2. Kernel Products

$$K(\mathbf{x},\mathbf{y}) = K_1(\mathbf{x},\mathbf{y}) K_2(\mathbf{x},\mathbf{y})$$

3. Power Series Expansion:

$$K(\mathbf{x},\mathbf{y}) = K(<\mathbf{x},\mathbf{y}>) \qquad K(z) = \sum_{j=0}^{\infty} a_n z^n \qquad a_j \geq 0$$

Generalized Chebyshev Kernel Functions

Generalized Chebyshev Kernel Function:

$$K(\mathbf{x},\mathbf{y}) = \frac{\sum_{j=0}^{n} T_j(\mathbf{x}) T_j^T(\mathbf{y})}{\sqrt{m - <\mathbf{x},\mathbf{y}>}}$$

Sedat OZER, M.Sc. Thesis, Univ. of Massachusetts, 2007

Multiple Kernel Learning

Classical SVM

$$\max_{\alpha} Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}),$$

$$\text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } C \geq \alpha_i \geq 0$$

Multiple Kernel Learning

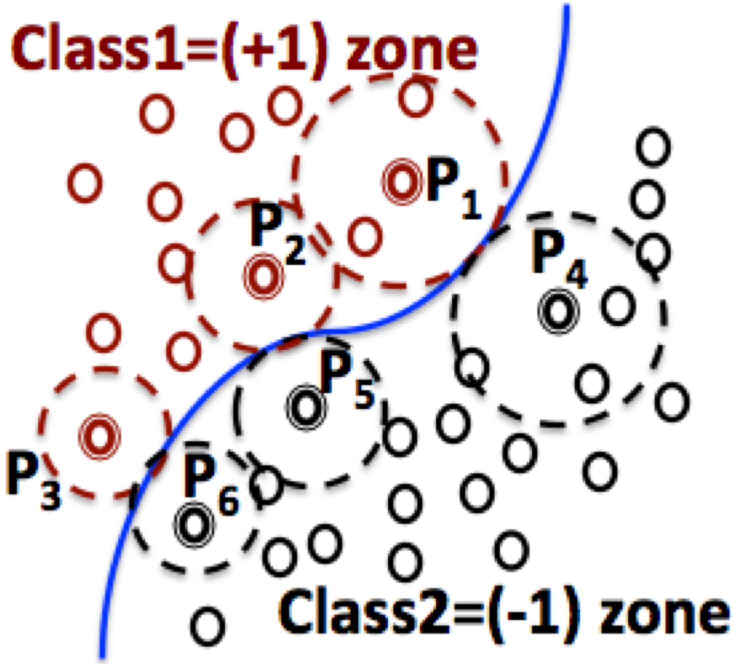
$$K(\mathbf{x_1}, \mathbf{x_2}) = \sum_{r=1}^{t} \beta_r K_r(\mathbf{x_1}, \mathbf{x_2}),$$

$$\max_{\alpha,\beta} Q(\alpha, \beta) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \sum_{r=1}^{t} \beta_r K_r(\mathbf{x_i}, \mathbf{x_j}),$$

$$\text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0, \ \sum_{r=1}^{t} \beta_r = 1, \ C \geq \alpha_i \geq 0 \ and \ \beta_r \geq 0$$

- Kernel functions (K_r) are given already.
- What if they are not the optimal kernel functions spanning the optimal solution space?
- Is it not costly to use two-step optimization? (Especially in the big data era)?

Similarity Domains Machine (2018)



$$\max_{\alpha} Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K_{\sigma ij}(\mathbf{x_i}, \mathbf{x_j}),$$

$$\text{subject to: } \sum_{i=1}^{n} \alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0 \text{ for } i = 1, 2, ..., n,$$

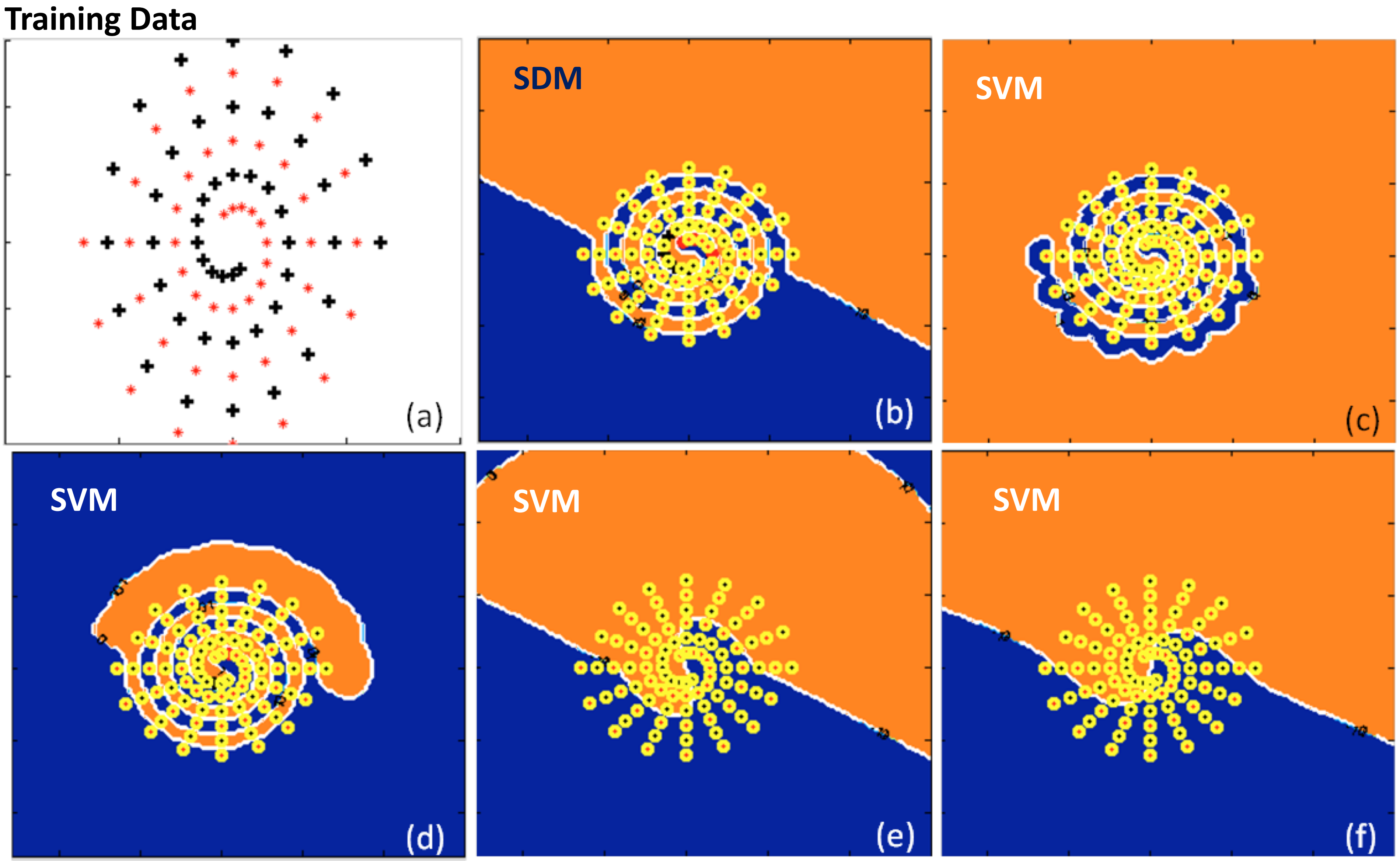
$$\text{and } K_{\sigma ij}(\mathbf{x_i}, \mathbf{x_j}) < T, \text{ if } y_i y_j = -1, \ \forall i, j$$

Paper: Sedat Ozer, "Similarity Domains Machine for Scale-invariant and Sparse Shape Modeling"
IEEE Transactions on Image Processing (2018-2019)

What is new?

- Kernel parameters are hard to choose from and can drastically change the classification results depending on the dataset. SDM solves that issue analytically by assigning different kernel parameters to different SVs!
- SDM uses Gaussian kernel as default!
- Define a local similarity domain in the feature space defined by the kernel parameter.
- Put each SV at the center of those local similarity domains!
  - (Each SV has its own kernel parameter)
- Use only one cost function to be optimized. No need to two-step optimization.
- Visualize and explain the kernel parameters for the first time without requiring an additional tool!

Comparison: SVM vs SDM



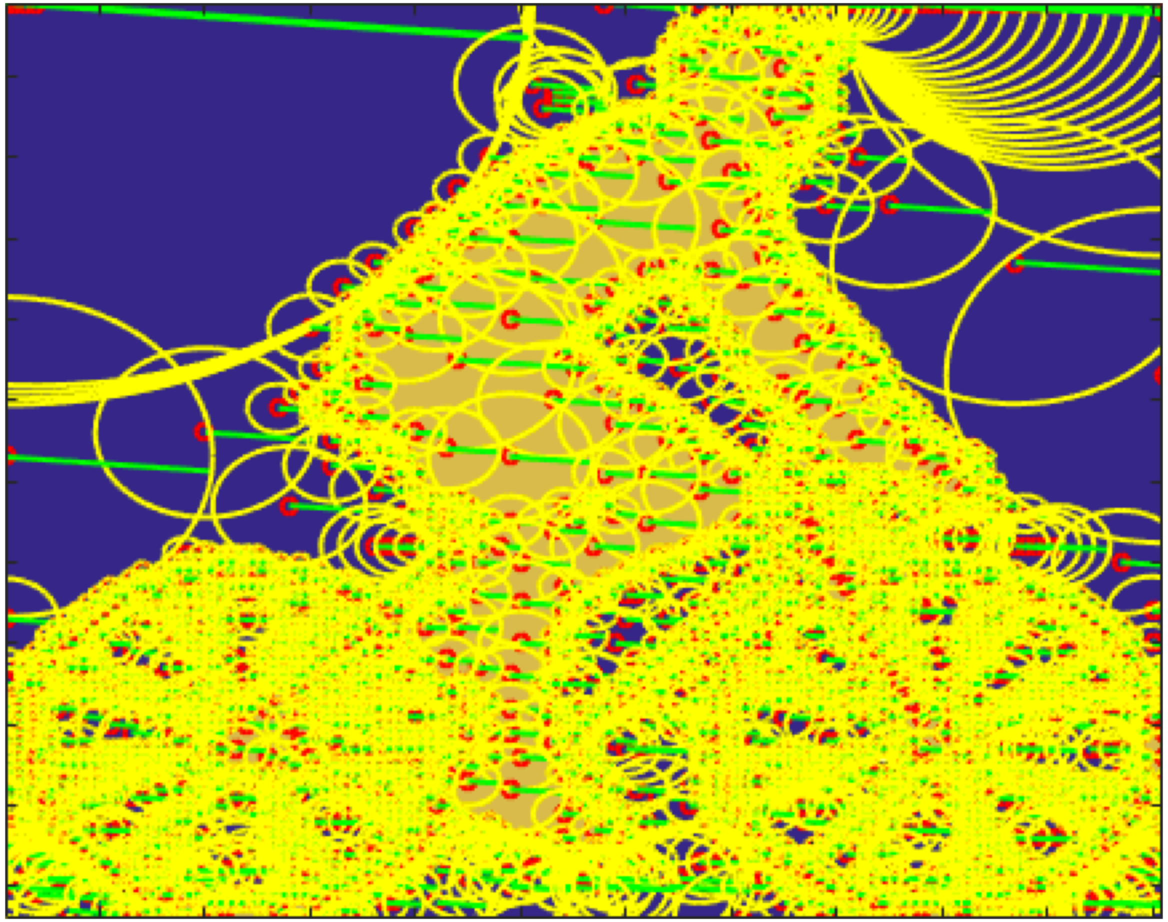
SVM with Gaussian kernel at different kernel parameters

Image source: [Ozer 2018]

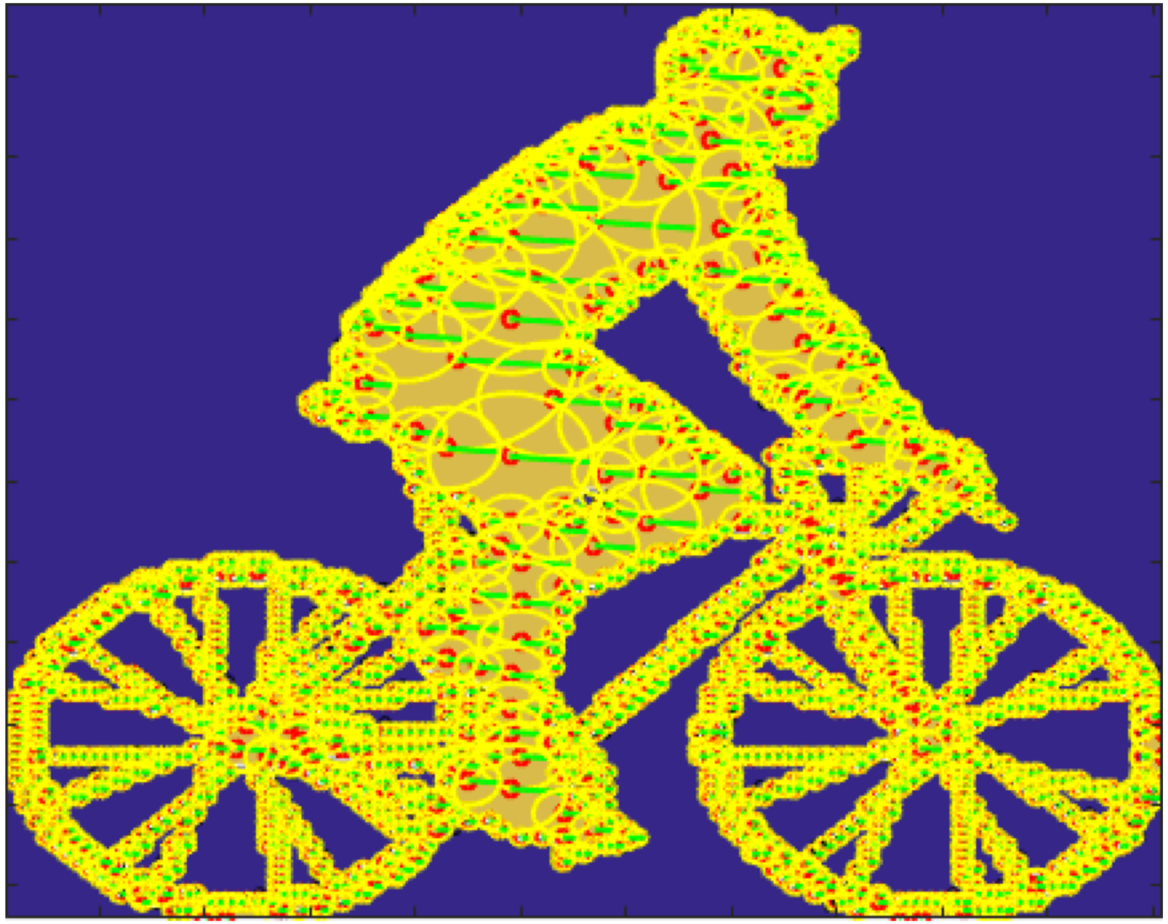
Kernel parameters? What are they looking like?

Original (binary) image:

$r^2 = a\sigma^2$



All the computed kernel parameters are visualized



All of the foreground kernel params are visualized

Image source: [Ozer 2018, "Similarity Domains Machine]

Can we use these parameters and do something with them?

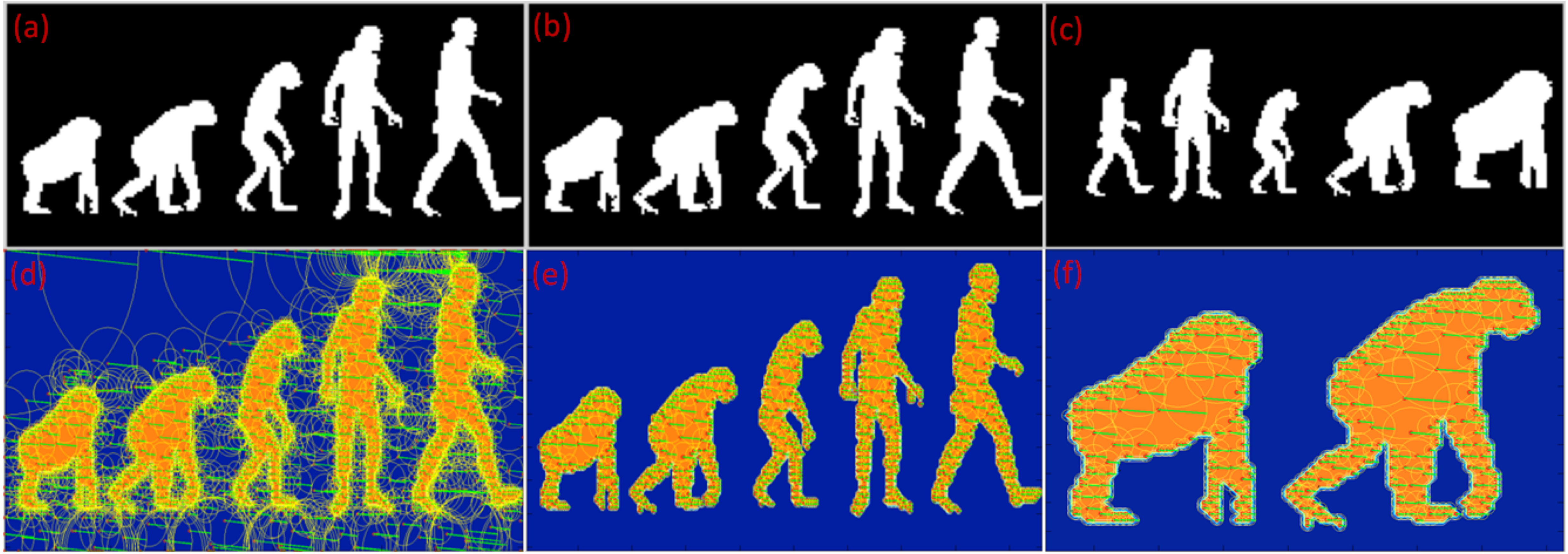
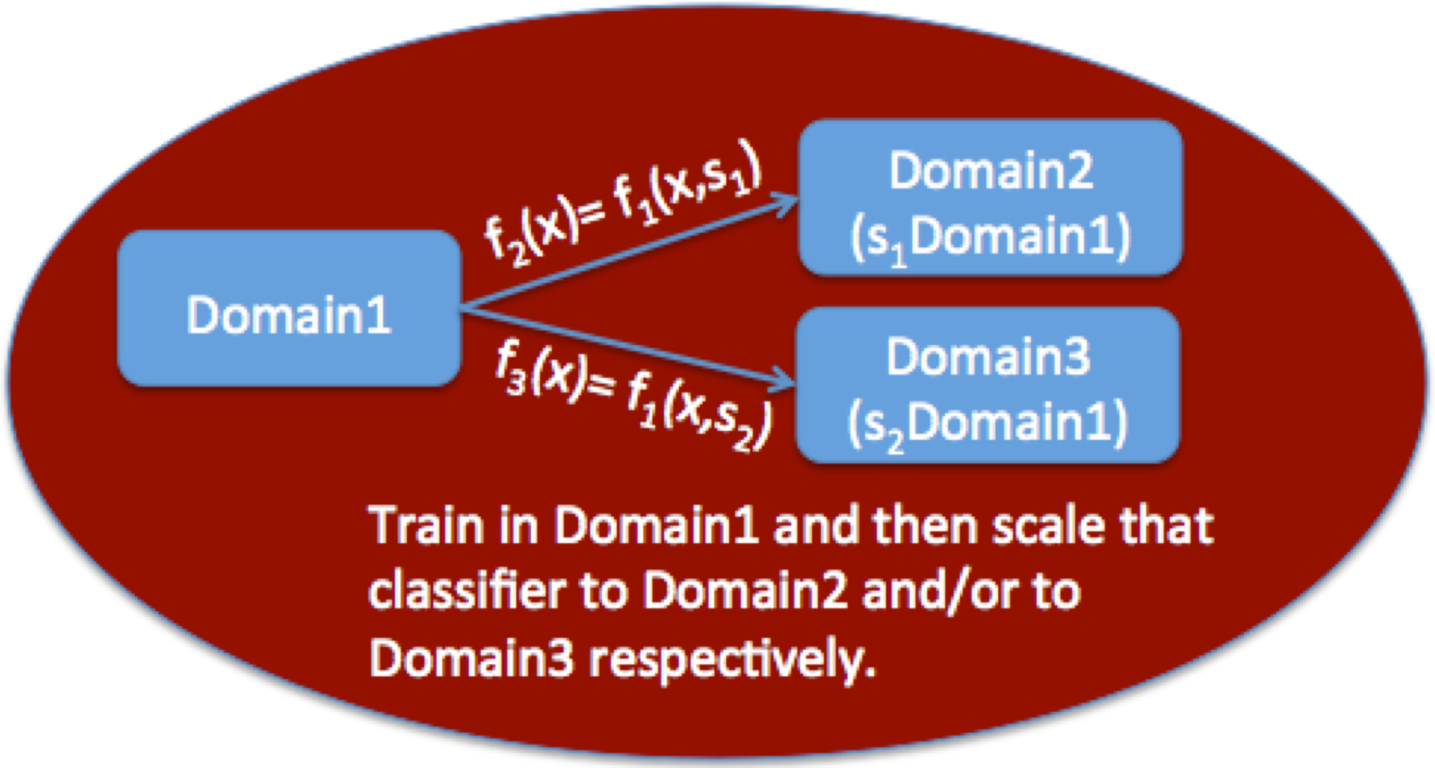


Image source: Sedat Ozer

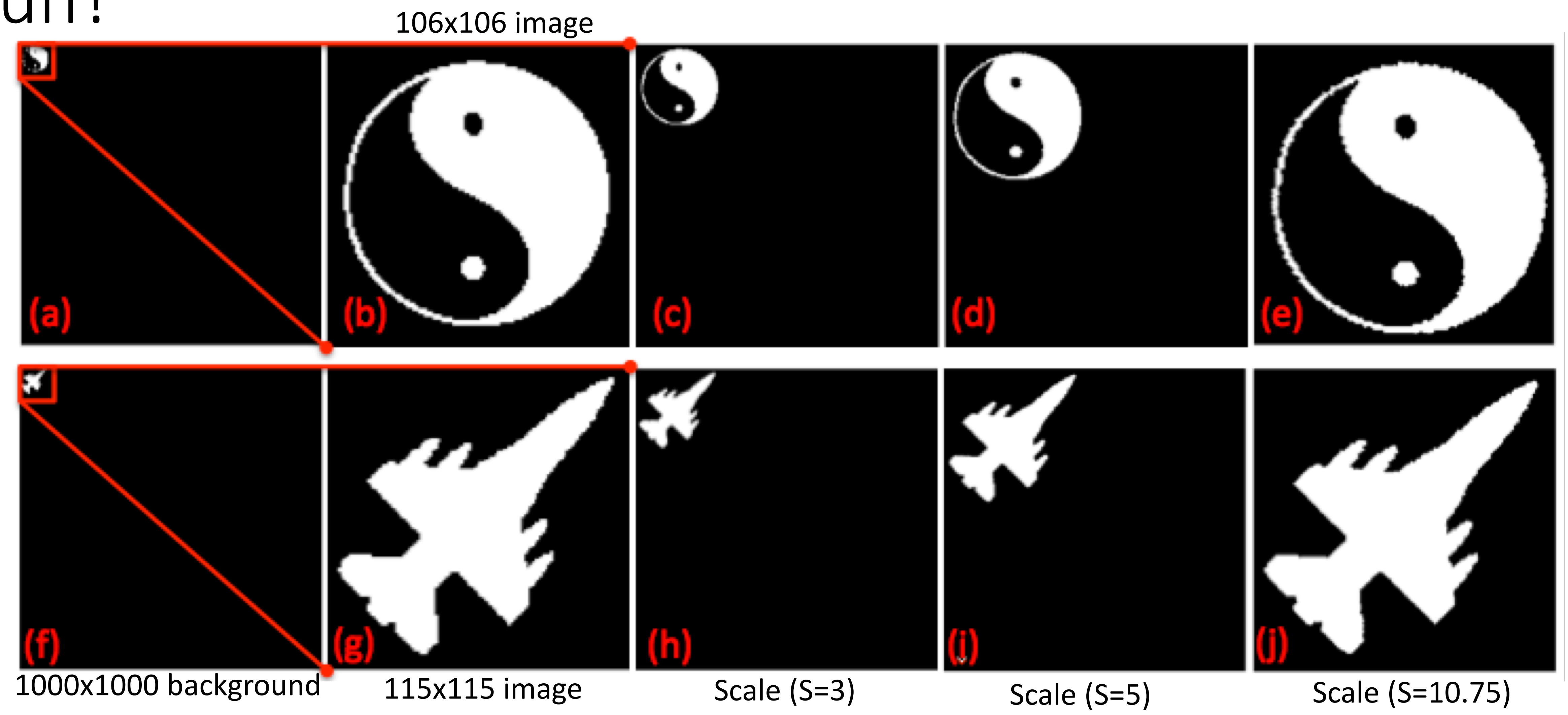
Scale-invariance in machine learning parameters? Huh?



Domain adaptation

Image source: [Ozer 2018]

Scale-invariance in machine learning parameters? Huh?



What we can do with scale-invariant machine learning algorithm: SDM results.

Image source: Sedat Ozer

How about computation time?

- SDM has multiple optimization techniques.
  - Some focused on speed and some focused on accuracy!

- SDM beats the best SVM implementations in terms of the computation time with particular optimization techniques while maintaining high accuracy!

Summary

- SVM is, fundamentally, a binary classifier.
- SVs are the training vectors that have nonzero alfa values.
- SVM is still one of the state of the art learning algorithms among the “shallow” techniques. ☺
- SVM idea has been heavily modified for different applications and purposes. Many versions are available. (See Similarity Domains Machine for example ☺)