Applying Machine Learning to Image Data

Data Science Detroit

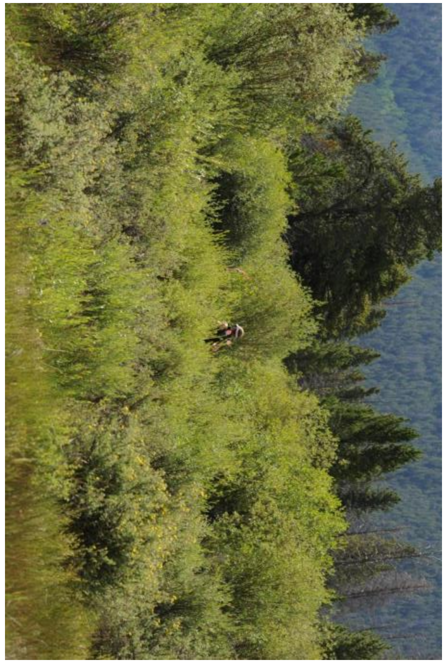
Jonathon Smereka

3/30/2017

Tonight’s Task: Image Orientation Detection

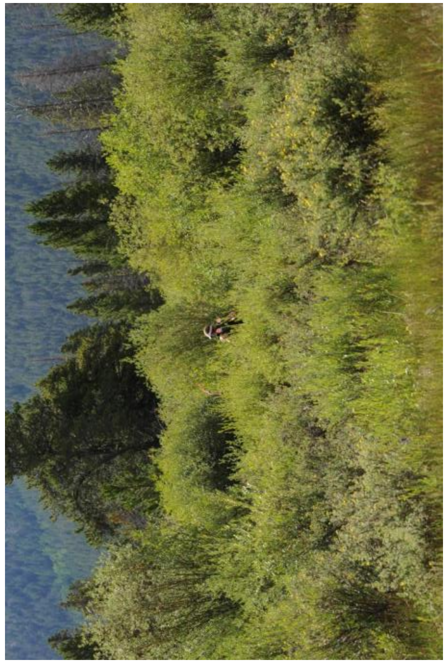
0°

90°



180°

270°



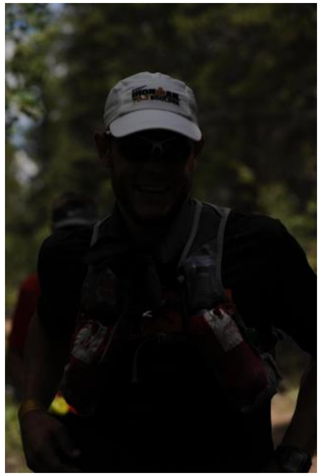
Tonight's Task: Image Orientation Detection

- 4 possible classes: $0^{\circ}, 90^{\circ}, 180^{\circ}, 270^{\circ}$
- Image features will be provided for you
  - Histogram of Oriented Gradients
  - Spatial color moments (3 mean and 3 variance values of L, U, and V)
  - Normalized spatial color moments
  - Principal Component Analysis
  - Linear Discriminant Analysis
- // TODO: Choose the combination of features and parameter values for training an SVM for classification

Training Data: 2149 images of people on bikes

Testing Data: 626 images of people running



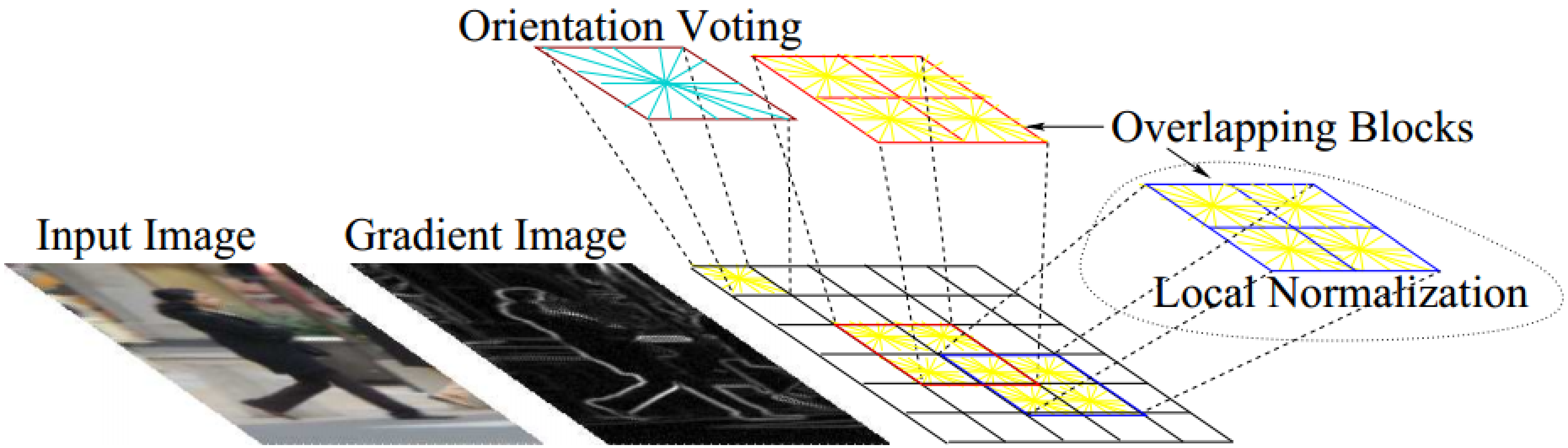




Tonight’s Task: What’s included?

- Python code: function_list.py and student.py
- Random assignment and rotation of training and test images with extracted features:
  - /img_idx/trainingdata.pckl and /img_idx/testingdata.pckl – image indices
  - /features/trainingdata_*_8x8.pckl – training data features
  - /features/testingdata_*_8x8.pckl – testing data features
- Conference and journal papers which concern image orientation detection: /papers/*.pdf
- An overview SVMs & the feature extraction approaches: /ppt/features.pdf

Overview: Histogram of Oriented Gradients



Input Image → Normalize gamma → Compute gradients → Weighted vote into spatial & orientation cells → Normalize contrast over overlapping spatial blocks → Compute HOGs over detection Window

Preprocessing (Normalize gamma, Compute gradients)

Similar to SIFT (Weighted vote into spatial & orientation cells)

Adjust for local gradient variations (Normalize contrast over overlapping spatial blocks)

Image from Dalal and Triggs

HOG Preprocessing

- Normalize over gamma for luminance correction

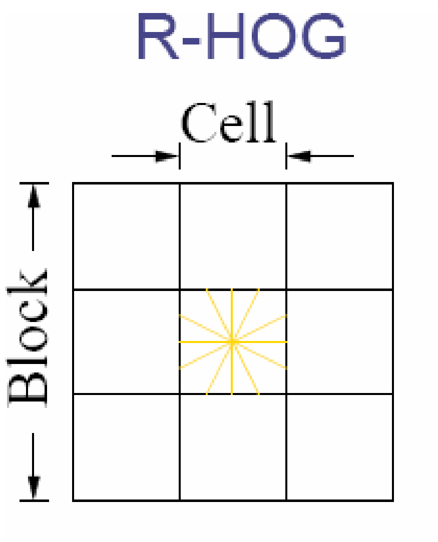
- Calculate gradient with mask $[-1,0,1]$
  - Larger masks (e.g., Sobel or 2-d Gaussian filtering) perform worse
  - Smoothing damages performance significantly
  - Color images have gradients per channel where the channel with the largest norm is used

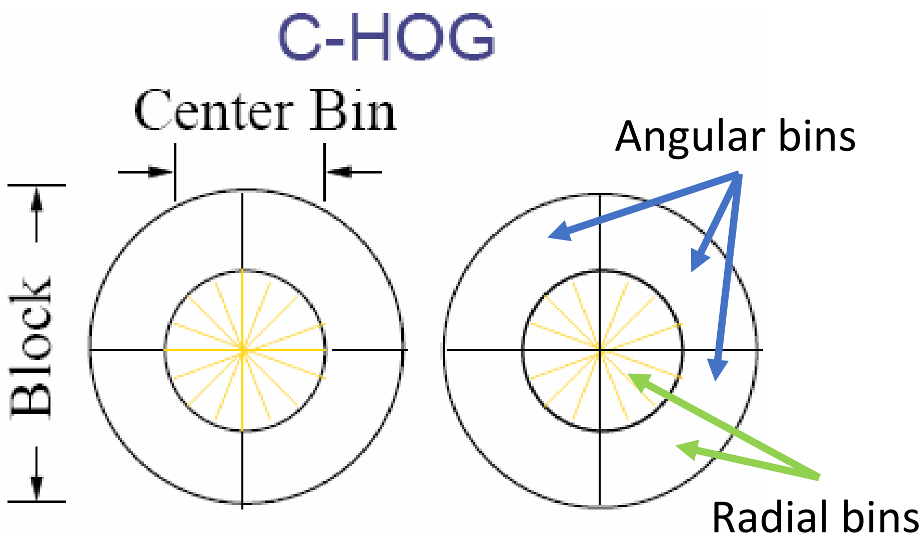
HOG Spatial and Orientation Cells

- Each pixel calculates a weighted vote based on the orientation of the gradient centered on it

- Calculate pixel orientation and magnitude
  - Orientation binned every $10°$ for 9 bins between $0° \rightarrow 180°$ (unsigned) or 18 bins between $0° \rightarrow 360°$ (signed)
  - Magnitude of the gradient = weight

HOG Contrast Normalization





3×3 cell with 6×6 pixels per cell works well, though depends on the size of the object being detected

Good parameters are 4 angular bins (adding more worsens results) with 2 radial bins (more is unnecessary), and a center bin has a radius of 4 pixels

Normalization

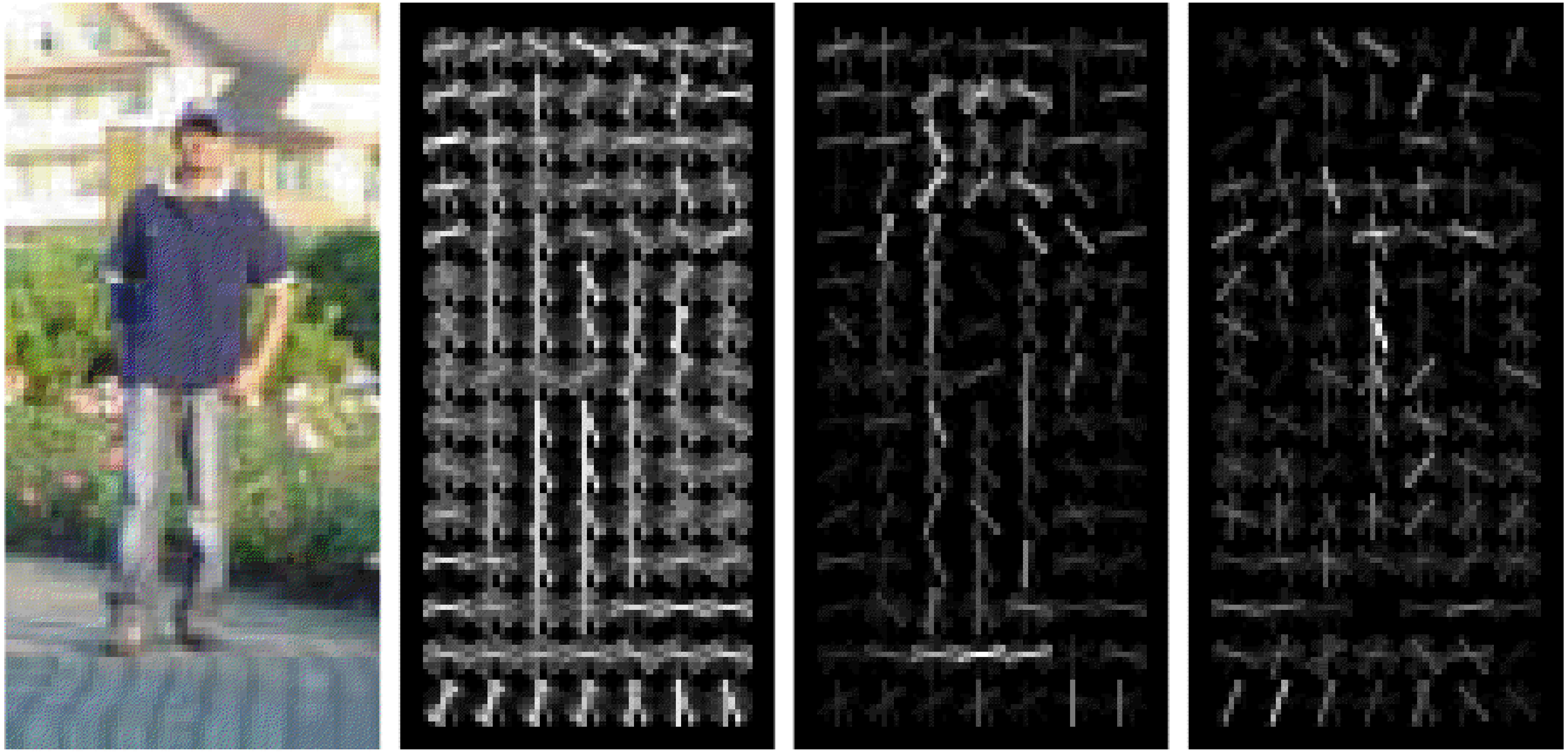
$$L1-norm : v \longrightarrow v/(||v||_1 + \epsilon)$$

$$L2-norm : v \longrightarrow v/\sqrt{||v||_2^2 + \epsilon^2}$$

$$L1-sqrt : v \longrightarrow \sqrt{v/(||v||_1 + \epsilon)}$$

$L2-hys$: L2-norm, plus clipping at .2 and renomalizing

HOG Example



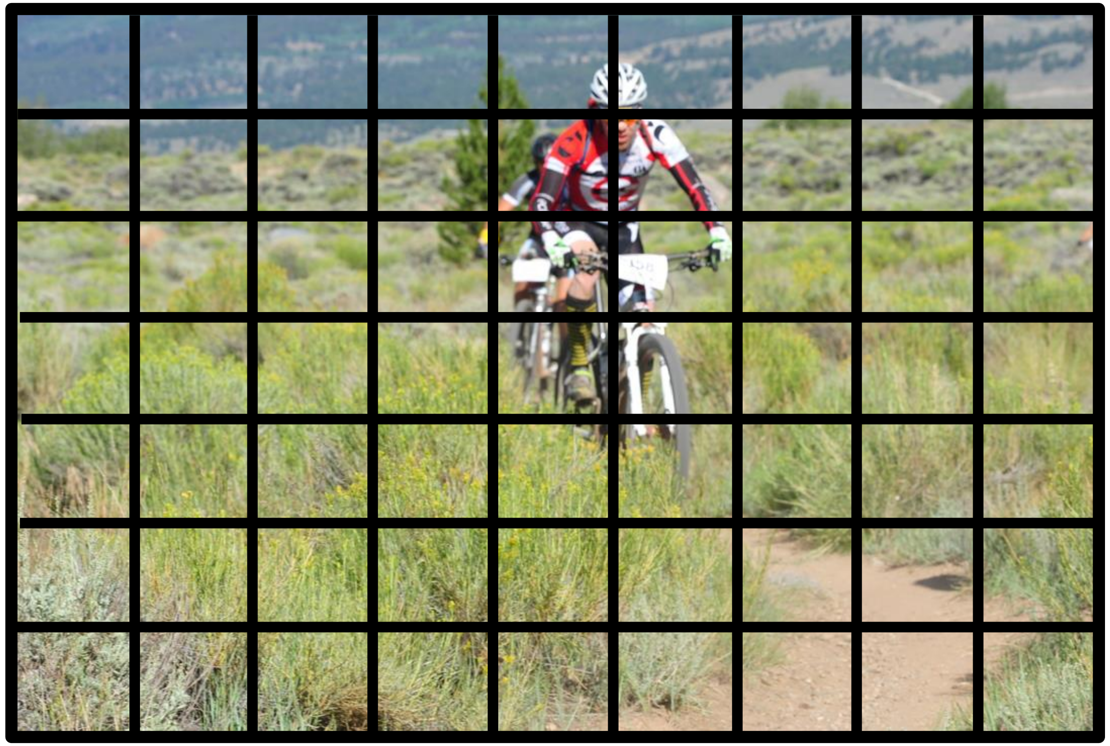
Image

R-HOG descriptor

Positive SVM weights

Negative SVM weights

Overview: Spatial Color Moments



- Divide into blocks
- Compute mean and variance of the pixel values within each block for each color channel
- Concatenate all mean and variance values into a vector

Overview: Normalized Spatial Color Moments

- Normalize the spatial color moment features to the same scale:

$$y_i' = \frac{y_i - min_i}{max_i - min_i}$$

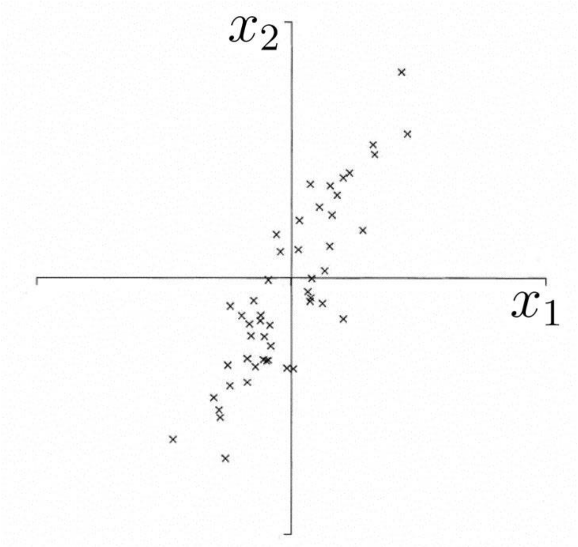
- y_i = the i-th feature component of a feature vector y

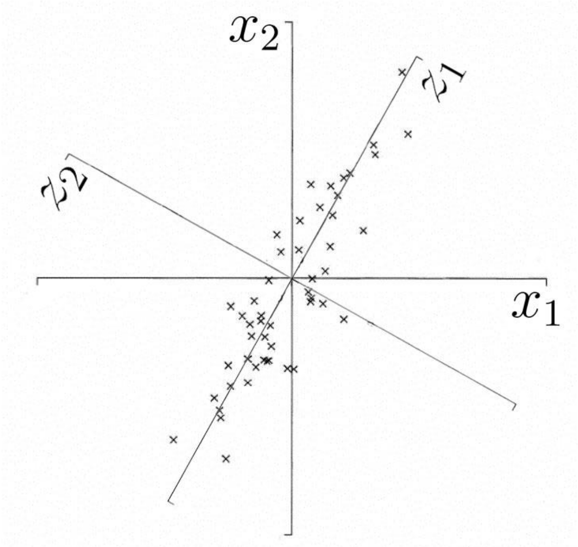
- min_i, max_i = the range of values for the i-th feature component over the training samples

- All values are computed using L, U, V color space

Overview: Principal Component Analysis

- Multivariate statistical procedure explaining covariance structure of a set of variables via small number of their linear combinations.
- Determine axis with greatest variation for a given data set.
- Final coordinate system best represents the variance.





PCA

- Algebra – Principal components are particular linear combinations of original variables, forming projection that best represents the data with low MSE
- Geometry – Linear combinations represent new coordinate system (via translations and rotations). New axes can determine directions of max variability of data set.
- Computational – Principal components found by calculating eigenvectors and eigenvalues of data covariance matrix. Eigenvector with largest eigenvalue reveals the direction of greatest variation.

PCA

- Formulate the mean adjusted data matrix.

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{x_{1,1} - \mu_1} & \cdots & \boldsymbol{x_{1,M} - \mu_M} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x_{N,1} - \mu_1} & \cdots & \boldsymbol{x_{N,M} - \mu_M} \end{pmatrix}$$

- Obtain the Covariance matrix.

$$\sum = \boldsymbol{U^T U}/(\boldsymbol{N} - \mathbf{1})$$

- Compute the projection matrix, each column being a direction of the new axes (an eigenvector).

$$\boldsymbol{\Phi} = [\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \boldsymbol{\varphi}_3, \dots, \boldsymbol{\varphi}_L]\ where\ L \leq M$$

PCA

- The set of **L** eigenvectors of the covariance corresponds to the **L** largest eigenvalues that minimizes the MSE of reconstruction over all choices of an orthonormal basis of size **L.**

- Much of the variability of data set can be accounted for in a smaller number of **L** principal components (pcs) as opposed to potentially large **M** variables.

- **N** examples of **M** variables has suddenly been *reduced* to **N** examples on **L** pcs.

PCA Example

PCA Example

- Our Mean Image:

- Eigenvectors (also known as eigenfaces):

PCA Example

- Reconstruct an Image from eigen-basis

PCA Summary

- Use eigenvectors of covariance matrix with largest eigenvalues – projects across dimensions of maximum variance
- Is a linear transform of high dimensional data to a lower dimension whose components are uncorrelated (if data is Gaussian, uncorrelated implies statistical independence)
- Is optimal when the you want to minimize the approximated mean square error of the projection

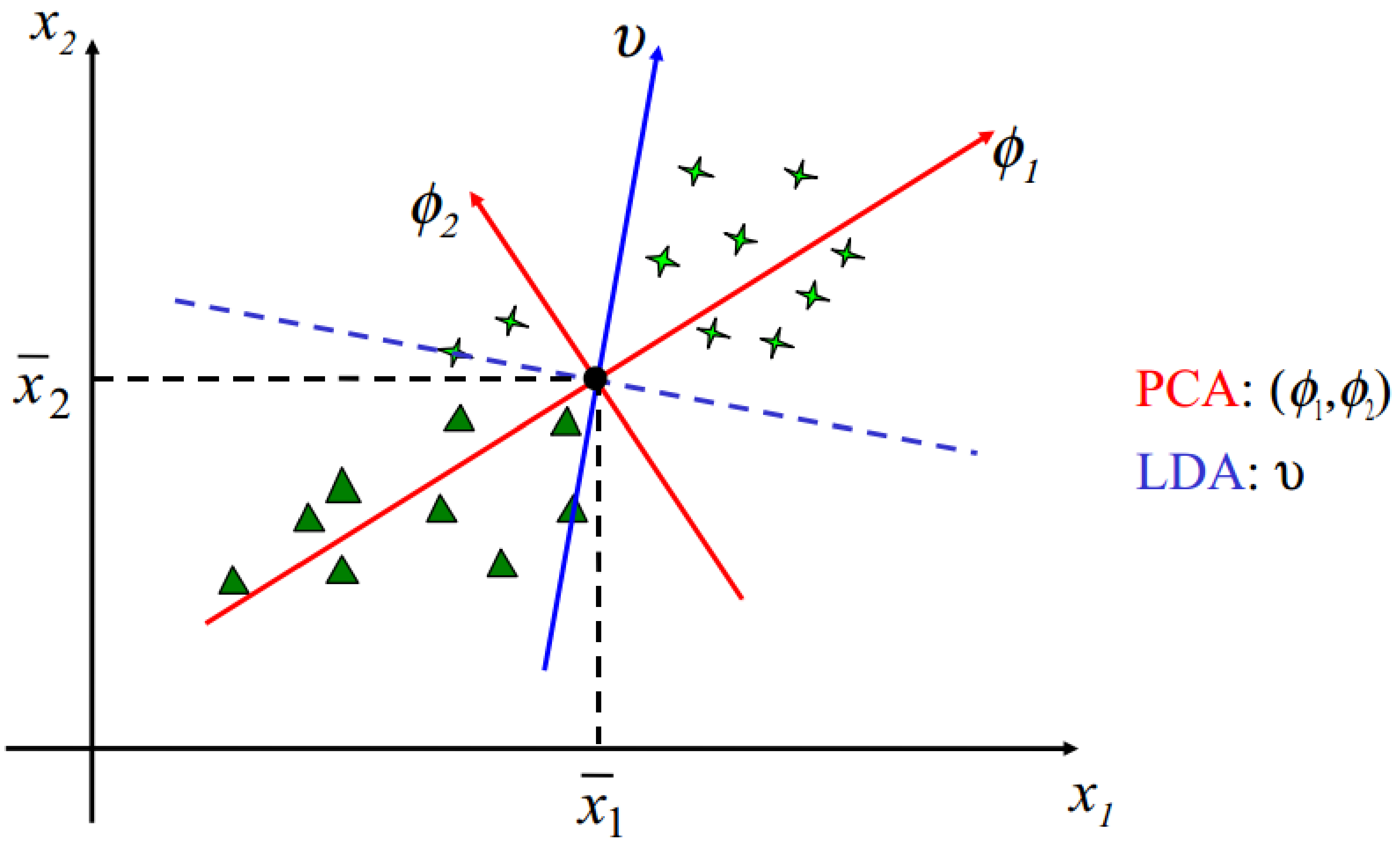
PCA Summary

- Works well if the data points are distributed throughout the hyperplane
- Does not necessarily lead to good class separability – no guarantee that the components will be discriminable features
- Will fail if data is highly non-linear or lies on more complicated manifolds

Overview: Linear Discriminant Analysis

- Statistical pattern recognition technique.
- Separates data into discrete groups via transformation into a space that *maximizes* 'between-class' separation while *minimizing* their 'with-in class' variability.
- Looks for a projection where new directions maximally separate data.

LDA Geometric View



LDA

- Within Class scatter matrix → $S_w = \sum_{i=1}^{c} \sum_{j=1}^{n_i} (Y_j - \mu_i)(Y_j - \mu_i)^T$
- Between Class scatter matrix → $S_b = \sum_{i=1}^{c} (\mu_i - \mu)(\mu_i - \mu)^T$
- The goal for LDA is to find a projection matrix that maximizes the ratio of determinant of S_b to the determinant of S_w (Fisher's Criterion)

$$\mathbf{\Phi}_{LDA} = \max |\mathbf{\Phi}^T S_b \mathbf{\Phi}| / |\mathbf{\Phi}^T S_w \mathbf{\Phi}|$$

- If S_w is non-singular then Fisher's Criterion is maximized when the projection matrix comprises of eigenvectors from ${S_w}^{-1} S_b$

LDA Summary

- LDA is a statistical pattern recognition technique for separating samples into discrete groups.

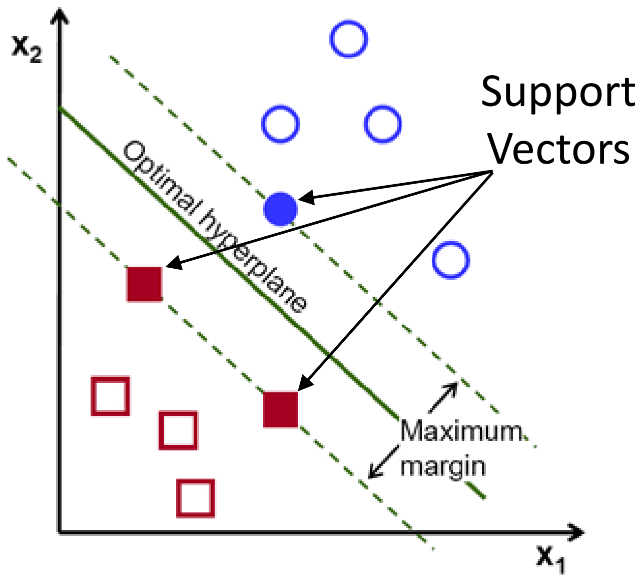
- LDA looks for directions that are efficient for ***discriminating*** data whereas PCA looks for data that are efficient for ***representing*** data.

- When the training data set is small, PCA can outperform LDA, also PCA is less sensitive to a different training data set.

Support Vector Machines

- Designed to find the maximum margin between two classes in feature space
- The optimal hyperplane (decision surface) is one that separates our data with the largest margin between the features of each class
- 'Support Vectors' are the data points that lie closest to the hyperplane

$$f(x) = \overset{\substack{\text{Weight vector} \\ \downarrow}}{w}^T \underset{\substack{\uparrow \\ \text{Input vector}}}{x} + \underset{\substack{\uparrow \\ \text{Bias}}}{\beta} \begin{cases} \geq 0 & \text{Class A} \\ < 0 & \text{Class B} \end{cases}$$



- Classifying a feature is simply labeling whether the new point falls on one side or the other of the trained margin

Support Vector Machines

- w is orthogonal to x when x is located on the hyperplane (i.e., $w^T x_\perp + \beta = 0$)
- By subtracting the two planes we can find our margin (separation between points x_m and x_p):

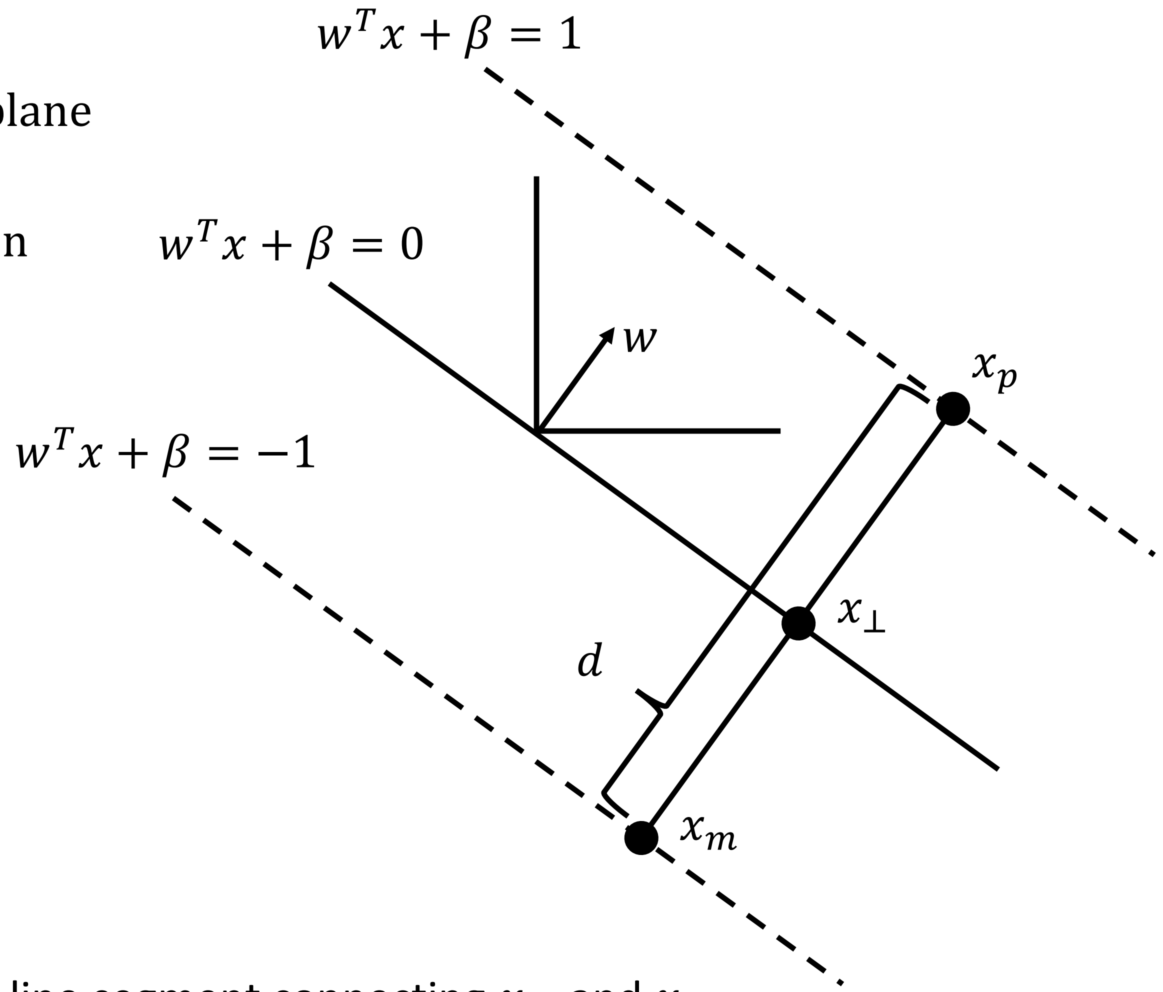
$$\begin{array}{r} w^T x_p + \beta = 1 \\ -\ w^T x_m + \beta = -1 \\ \hline w^T (x_P - x_m) = 2 \end{array}$$

Since w is perpendicular, the closest point to x_m can be defined as:

$$x_p = x_m + dw$$

$$\therefore dw^T w = 2 \rightarrow d = \frac{2}{w^T w} = \frac{2}{\|w\|_2^2}$$

- Distance $= d\|w\|_2 = \frac{2}{\|w\|_2^2}\|w\|_2 = \frac{2}{\sqrt{w^T w}}$



$dw =$ line segment connecting x_m and x_p

$d\|w\|_2 =$ distance between x_m and x_p

Support Vector Machines

- We want to maximize the margin, i.e. the distance between two class boundaries $\frac{2}{\sqrt{w^T w}}$, which is equivalent to minimizing $\frac{\sqrt{w^T w}}{2}$ (which is equivalent to minimizing $\frac{1}{2} w^T w$ since the square root function is monotonic)
- Additionally, we can define our labels $y_i = 1 \ \forall i \in \text{Class A}$ and $y_i = -1 \ \forall i \in \text{Class B}$:

$$\text{Class A: } w^T x_i + \beta \geq 1, y_i = 1 \rightarrow y_i(w^T x_i + \beta) \geq 1$$
$$\text{Class B: } w^T x_i + \beta \leq -1, y_i = -1 \rightarrow y_i(w^T x_i + \beta) \geq 1$$

- Accordingly, the convex optimization problem for a linear SVM with a hard margin:

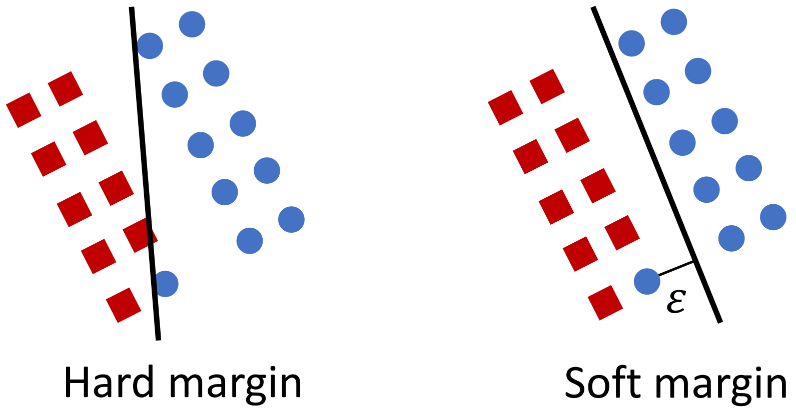
$$\min_{w,\beta} \frac{1}{2} w^T w \quad \text{subject to: } y_i(w^T x_i + \beta) \geq 1 \ \ \forall i$$

Support Vector Machines

- In practice, training samples may contain noise and are not linearly separable, giving no feasible solution to the hard margin problem
- The soft margin SVM uses a 'slack variable', ε_i, to represent the error of the i-th training sample (constrained optimization problem):

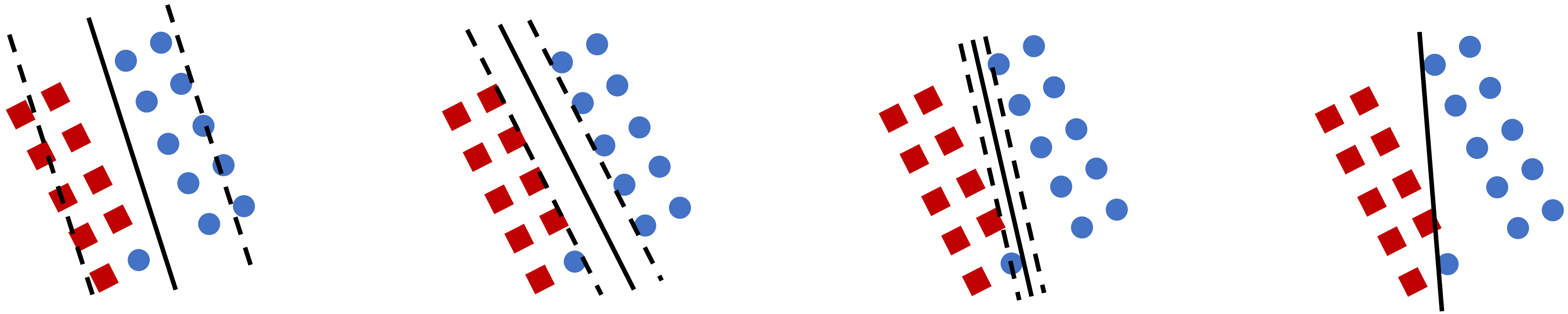
$$\min_{w,\beta,\varepsilon} w^T w + 2C \sum_i \varepsilon_i \quad \text{subject to: } y_i(w^T x_i + \beta) \geq 1 - \varepsilon_i \text{ and } \varepsilon_i \geq 0 \ \forall i$$

A regularization parameter that weighs the cost of the penalty for misclassification. Determined using cross-validation ($C \to \infty$ = hard margin)



Support Vector Machines

- As a regularization parameter, C, determines the penalty for misclassifications
  - Large C makes constraints hard to ignore (narrow margin) and will lead to a lower bias, higher variance SVM (i.e., overfitting)
  - Small C allows for constraints to be ignored (large margin) and will lead to a higher bias, lower variance SVM (i.e., underfitting)



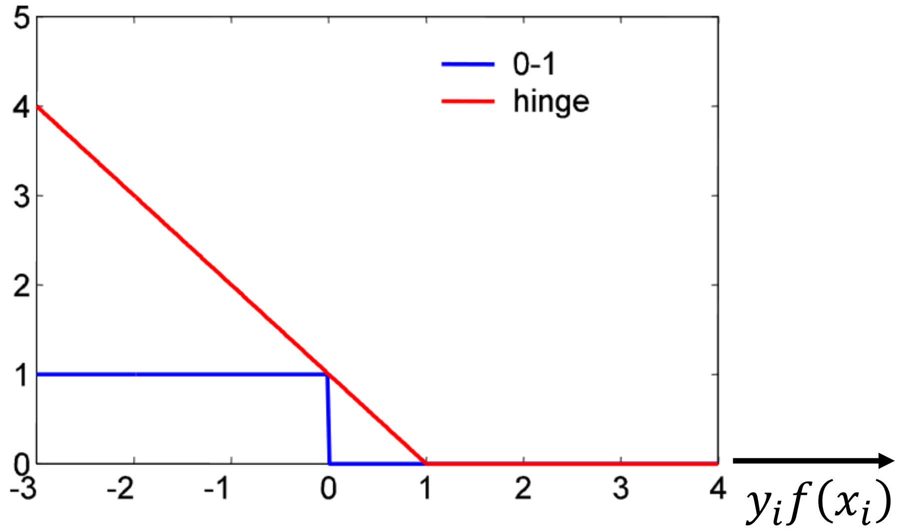
Small C (Large Margin) | Good Fit | Large C (Narrow Margin) | $C \rightarrow \infty$

Support Vector Machines

- Learning an SVM was previously formulated as a constrained optimization problem over w and ε, however, $y_i(w^T x_i + \beta) \geq 1 - \varepsilon_i = y_i f(x_i) \geq 1 - \varepsilon_i$ and with $\varepsilon_i \geq 0$, can be more concisely written as: $\varepsilon_i = \max\big(0, 1 - y_i f(x_i)\big)$

Known as the 'Primal Problem'

- The equivalent learning problem is unconstrained over w: $$\min_w \underbrace{\|w\|_2^2}_{\text{L2 regularization}} + C \sum_i^N \underbrace{\max\big(0, 1 - y_i f(x_i)\big)}_{\text{"hinge" loss function}}$$



- The function takes the form of $\arg\min \sum \mathcal{L}\big(y, f(x)\big) + \lambda R(f)$
- The graph represents two margin based classifiers which induce a decision rule via $sign(f)$:
  - Misclassification (0-1) loss: $\mathcal{L}\big(y, f(x)\big) = I(yf(x) \leq 0)$
  - Hinge loss (SVM): $\mathcal{L}\big(y, f(x)\big) = \max\big(0, 1 - yf(x)\big)$

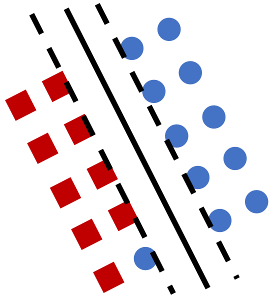
Support Vector Machines

- Problem: the primal problem is convex but not differentiable
- Solution: the 'Dual problem' can be derived using Lagrangian multipliers

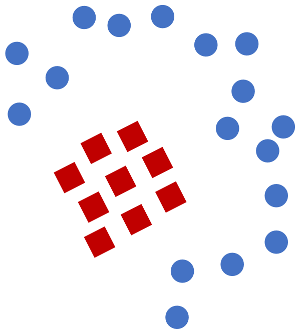
$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \left(x_j^T x_k\right) \ \text{ subject to: } 0 \leq a_i \leq C \text{ and } \sum_i \alpha_i y_i = 0$$

- The dual form benefits:
  - Requires to need to learn N parameters, while the primal form requires learning δ parameters ($x_i \in \mathbb{R}^{\delta}$ for $i = 1 \cdots N$, if $N \ll \delta$ then it's more efficient to solve for α than w)
  - Only involves $\left(x_j^T x_k\right)$ which is helpful for non-linear decision boundaries

Good Fit



????



Support Vector Machines

- Given a non-linear decision boundary, the data may be linearly separable in a different space, whereby $\phi(x)$ is a feature map ($\phi: x \rightarrow \phi(x)$ $\mathbb{R}^{\delta} \rightarrow \mathbb{R}^{D}$):

$$f(x) = w^{T}\phi(x) + \beta$$

- Map x to $\phi(x)$ where the data is separable (solving for w in the high dimensional space $\mathbb{R}^{D}$)

- If $D \gg \delta$? Use the dual formulation: $k\left(x_j, x_k\right) = \phi\left(x_j\right)^{T}\phi(x_k)$ where $k\left(x_j, x_k\right)$ is the 'Kernel'

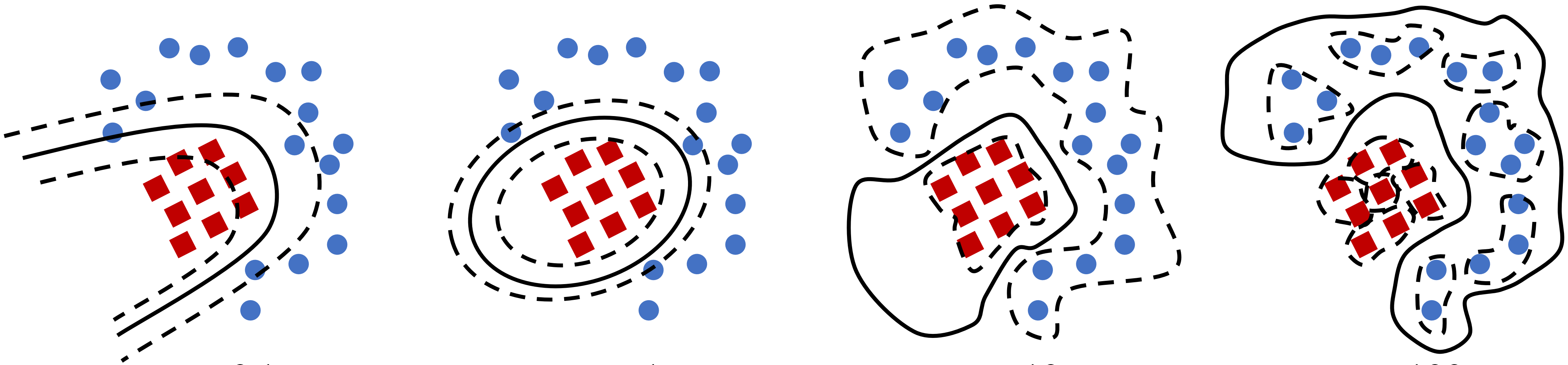
Support Vector Machines

Common Kernels:

- Linear kernel: $k(x, y) = x^T y$ (no mapping)
- Polynomial kernel: $k(x, y) = (\gamma x^T y + r)^\rho$ for $\gamma, \rho > 0$ and $r \geq 0$ (polynomial terms up to degree ρ)
- Radial basis function (RBF) kernel (Gaussian): $k(x, y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right) = \exp(-\gamma\|x - y\|_2^2)$ where $\gamma = \frac{1}{2\sigma^2}$, the RBF kernel is shift invariant ($k(x + a, y + a) = k(x, y)\ \forall a$
- Sigmoid kernel: $k(x, y) = \tanh(\gamma x^T y + r)$

Support Vector Machines

RBF Kernel: $k(x, y) = \exp(-\gamma \|x - y\|_2^2)$

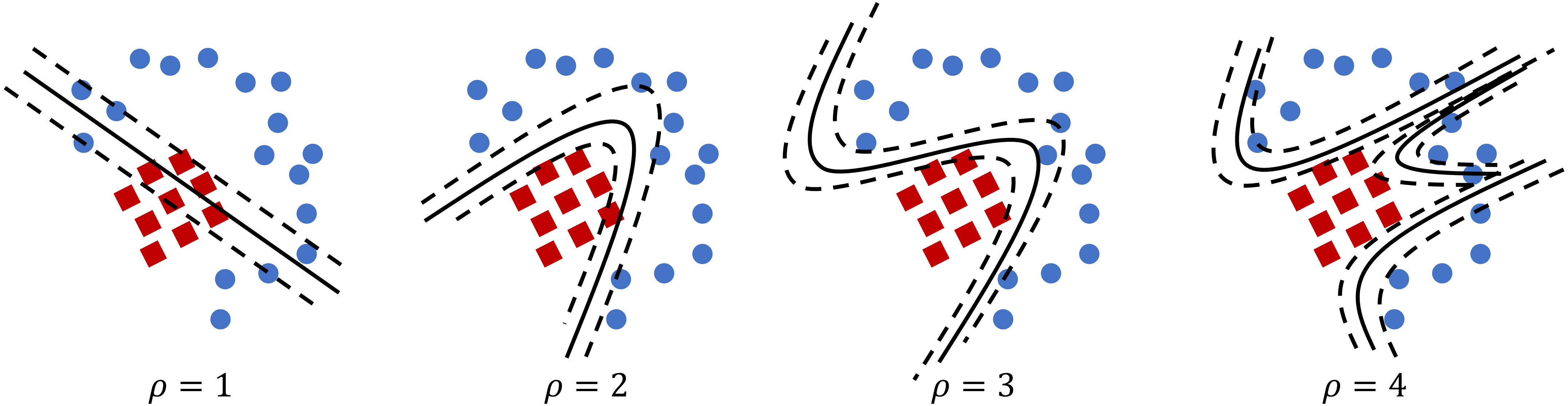


$\gamma = 0.1$ $\gamma = 1$ $\gamma = 10$ $\gamma = 100$

γ is known as the kernel bandwidth, as γ increases, the flexibility of the decision boundary increases. Small values produce a boundary that is near linear and as γ increases the SVM overfits.

Support Vector Machines

Polynomial kernel: $k(x, y) = (\gamma x^T y + r)^\rho$ ρ = degree, γ = scale, r = bias (offset)



The polynomial degree, ρ, controls the flexibility of the decision boundary increases. When $\rho = 1$ the boundary is linear, and as ρ increases the SVM will often overfit.

Grid Search

- As the number of hyperparameters increases, the complexity of choosing acceptable values also increases
- Grid search iterates over varying values of each hyperparameter while holding all others constant
- Grid points are usually chosen on a logarithmic scale and classifier accuracy is estimated for each point on the grid

Accuracy at $\gamma = 10^{-3}$ and $C = 10^{1}$