ISML - AdaBoost

1 Rationale

- Building a highly accurate classifier is a difficult task.
- We could probably come up with many quick **rules of thumb**

Example of rules of thumb:

An example could be "if the subject line contains 'buy now' then classify as spam."

This certainly doesn't cover all spams, but it will be significantly better than random guessing.

2 Overview

Boosting refers to a general and provably effective method of producing a very accurate classifier by combining rough and moderately inaccurate rules of thumb.

It is based on the observation that finding many rough rules of thumb can be a lot easier than finding a single, highly accurate classifier.

The boosting algorithm repeatedly calls this weak learner, each time feeding it a different distribution over the training data (in AdaBoost).

Each call generates a weak classifier and we must combine all of these into a single classifier that, hopefully, is much more accurate than any one of the rules.

Voting (Ensemble Methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers that are good at different parts of the input space**
- **Output class:** (Weighted) vote of each classifier
  - Classifiers that are most "sure" will vote with more conviction
  - Classifiers will be most "sure" about a particular part of the space
  - On average, do better than single classifier!

3 Steps of AdaBoost

Given $\mathcal{D} = (\mathrm{x}_i, y_i), \ldots, (\mathrm{x}_m, y_m)$ as before.
Initialize the distribution D_1 to be uniform: $D_1(i) = \frac{1}{m}$.
Repeat for $t = 1, \ldots, T$:

1 Learn weak classifier h_t using distribution D_t.

- For the example given, this requires you to learn the threshold and the parity at each iteration given the current distribution D_t for the weak classifier h over each feature:

❶ Compute the weighted error for each weak classifier.

$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(i)\delta(h(\mathrm{x}_i) \neq y_i), \quad \forall h \tag{14}$$

❷ Select the weak classifier with minimum error.

$$h_t = \mathrm{argmin}_h\, \epsilon_t(h) \tag{15}$$

2 Set weight α_t based on the error:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} \right) \tag{16}$$

3 Update the distribution based on the performance so far:

$$D_{t+1}(i) = \frac{1}{Z_t} D_t(i) \exp\left[-\alpha_t y_i h_t(x_i)\right] \tag{17}$$

where Z_t is a normalization factor to keep D_{t+1} a distribution. Note the careful evaluation of the term inside of the exp based on the possible $\{-1, +1\}$ values of the label.

One chooses T based on some established error criterion or some fixed number.

5 Summary

- AdaBoost is a sequential algorithm that minimizes an upper bound of the empirical classification error by selecting the weak classifiers and their weights. These are "pursued" one-by-one with each one being selected to maximally reduce the upper bound of error.
- AdaBoost defines a distribution of weights over the data samples. These weights are updated each time a new weak classifier is added such that samples misclassified by this new weak classifier are given more weight. In this manner, currently misclassified samples are emphasized more during the selection of the subsequent weak classifier.
- The empirical error will converge to zero at an exponential rate.

6 Advantage

- It is fast to evaluate (linear-additive) and can be fast to train (depending on weak learner).
- T is the only parameter to tune.
- It is flexible and can be combined with any weak learner.
- It is provably effective if it can consistently find the weak classifiers (that do better than random).
- Since it can work with any weak learner, it can handle the gamut of data.

7 Caveats

- Performance depends on the data and the weak learner. It can fail if
- The weak classifiers are too complex and overfit.
- The weak classifiers are too weak, essentially underfitting.
- AdaBoost seems, empirically, to be especially susceptible to uniform noise.

4 Analysis of AdaBoost

- The selected weight for each new weak classifier is always positive.

$$\epsilon_t(h_t) < \frac{1}{2} \Rightarrow \alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} > 0 \tag{18}$$

- The smaller the classification error, the bigger the weight and the more this particular weak classifier will impact the final strong classifier.

$$\epsilon(h_A) < \epsilon(h_B) \Rightarrow \alpha_A > \alpha_B \tag{19}$$

- The weights of the data points are multiplied by $\exp\left[-y_i \alpha_t h_t(\mathbf{x}_i)\right]$.

$$\exp\left[-y_i \alpha_t h_t(\mathbf{x}_i)\right] = \begin{cases} \exp\left[-\alpha_t\right] < 1 & \text{if } h_t(\mathbf{x}_i) = y_i \\ \exp\left[\alpha_t\right] > 1 & \text{if } h_t(\mathbf{x}_i) \neq y_i \end{cases} \tag{20}$$

- The weights of correctly classified points are reduced and the weights of incorrectly classified points are increased. Hence, the incorrectly classified points will receive more attention in the next run.
- Key Idea: **AdaBoost minimizes an upper bound on the classification error.**
- **Therefore, after t steps, the error rate of the strong classifier is bounded on top by**

$$\text{Err}(H) \leq Z \leq \exp\left[-2 \sum_{t=1}^{T} \gamma_t^2\right] \tag{48}$$

- Hence, each step decreases the upper bound exponentially.
- And, a weak classifier with small error rate will lead to faster descent.
- AdaBoost takes a stepwise minimization scheme, which may not be optimal (it is greedy). When we calculate the parameter for the t^{th} weak classifier, the others remain set.
- We should stop AdaBoost if all of the weak classifiers have an error rate of $\frac{1}{2}$.

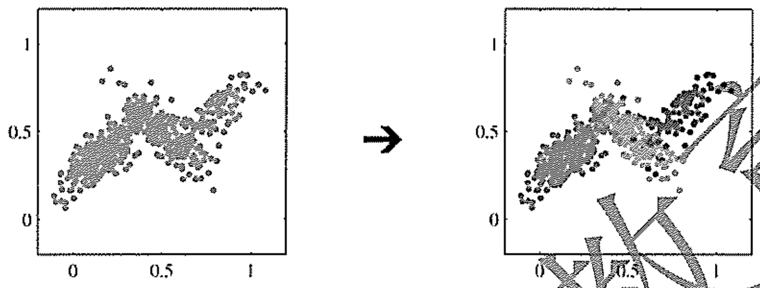
ISML - Clustering

1 Why do unsupervised learning?

- Raw data cheap. Labelled data expensive
- Save memory/computation
- Reduce noise in high-dimensional data
- Useful in exploratory data analysis
- Often a pre-processing step for supervised learning

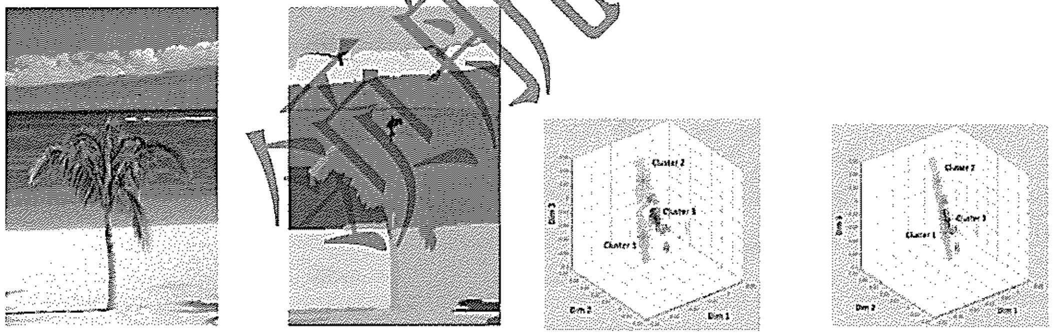
2 Cluster Analysis

Discover groups such that samples within a group are more similar to each other than samples across groups.



2.1 Applications

- Image Segmentation



- Vector quantization to compress images

2.2 Ingredients of Cluster Analysis

- A dissimilarity function between samples.
- A loss function to evaluate clusters.
- Algorithm that optimizes this loss function.

2.3 The Dissimilarity functions

- Choice of dissimilarity function is application dependent.
- Need to consider the type of features.
- Categorical, ordinal or quantitative.
- Possible to learn dissimilarity from data (later).

2.4 Dissimilarity based on features

- Data point x_i has features x_{ij}, j = 1,...,p.
- One choice of dissimilarity function is the Euclidean distance

$$D(x_i, x_{i'}) = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}$$

- Resulting clusters invariant to rotation and translation of features but not to scaling.
- If the features have different scales, standardize the data.

3 K-means

3.1 Idea of K-means

- K clusters each summarized by a prototype μ_k
- Assignment of data xi to a cluster represented by responsibilities

$$r_{ik} \in \{0, 1\} \text{ with } \sum_{k=1}^{K} r_{ik} = 1$$

An example with 4 data points and 3 clusters

$$(r_{ik}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- The loss function of K-means

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|_2^2.$$

3.2 Minimizing the loss function

- If prototypes known, can assign responsibilities.
- If responsibilities known, can compute prototypes.
  (a chicken-and-egg problem)
- We use an iterative procedure - EM method

- Likelihood function (the dissimilarity function)

$$r_{ik} = E(z_{ik}) = P(z|x) = \frac{P(z)P(x|z)}{P(x)} = \frac{P(z)P(x|z)}{\sum_1^K P(z)P(x|z)}$$

- Loss function

$$-\log \Pr(x|\pi, \mu, \Sigma) = -\sum_{i=1}^{n} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \right\}$$

4.2 EM Method of GMM

- E-step: Given parameters, compute

$$r_{ik} \triangleq E(z_{ik}) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}$$

- M-step: Maximize the expected complete log likelihood

$$E\left[\log \Pr(x, z|\pi, \mu, \Sigma)\right] = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \left\{ \log \pi_k + \log \mathcal{N}(x_i|\mu_k, \Sigma_k) \right\}$$

  To update the parameters (set derivates equal to 0)

$$\pi_k = \frac{\sum_i r_{ik}}{n}, \mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}, \Sigma_k = \frac{\sum_i r_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i r_{ik}}$$

- Iterate till likelihood converges.
- Converges to local optimum of the log likelihood.

4.3 GMM: Relation to K-means

- E-step in GMM: a soft version of K-means. $r_{ik} \in [0,1]$, *instead of* $\{0,1\}$
- M-step in GMM: estimates the probabilities and the covariance matrix of each cluster in addition to the means.

4.4 K-means vs GMM

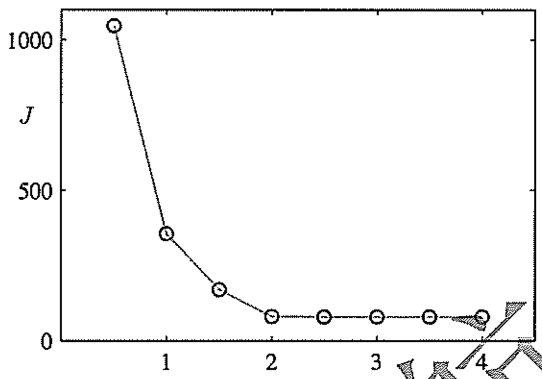
• Loss function: minimize sum of squared distance.	• Minimize negative log likelihood.
• Hard assignment of points to clusters.	• Soft assignment of points to clusters.
• Assumes spherical clusters with equal probability of a cluster.	• Can be used for non-spherical clusters with different probabilities.

3.3 The step of K-means (EM method applied)

- E-step: Fix μ_k, minimize J w.r.t. r_{ik}.
  - Assign each data point to its nearest prototype.
- M-step: Fix r_{ik}, minimize J w.r.t. μ_k.
  - Set each prototype to the mean of the points in that cluster, i.e., $\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$.

- This procedure is guaranteed to converge.
- Converges to a local minimum.
- Use different initializations and pick the best solution.
- May still be insufficient for large search spaces.

3.4 Loss function J after each iteration (converged)



3.5 Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster. (Solution: GMM)
- Assumes spherical clusters and equal probabilities for each cluster. (Solution: GMM)
- Clusters change arbitrarily for different K (Solution: Hierarchical clustering)
- Sensitive to outliers. (Solution: Use a robust loss function)
- Works poorly on non-convex clusters (Solution: Spectral clustering)

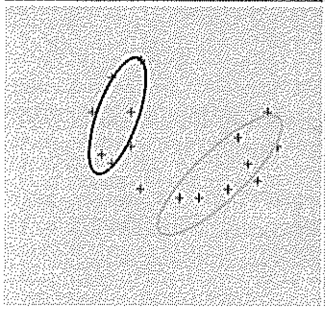
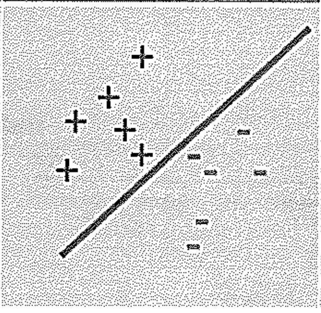
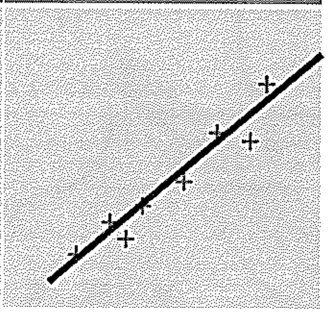
4 Gaussian Mixture Model

4.1 Idea of GMM

- Each cluster is associated with a Gaussian distribution. To generate data, randomly choose a cluster k with probability π_k and sample from its distribution
- $$\Pr(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \text{ where } \sum_{k=1}^{K} \pi_k = 1, 0 \le \pi_k \le 1$$
  $$P(z) = \sum_1^K \pi_k$$
  $$P(x|z) = \sum_1^K N(x|\mu_k, \Sigma_k)$$

ISML - Regression

1 Overview

CLUSTERING	CLASSIFICATION	REGRESSION (THIS TALK)
		
K-means	• Decision tree • Linear Discriminant Analysis • Neural Networks • Support Vector Machines • Boosting	• Linear Regression • Support Vector Regression
Group data based on their characteristics	Separate data based on their labels	Find a model that can explain the output given the input

2 Linear Regression

- Given data with n dimensional variables and 1 target-variable (real number):

  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$

  Where $\mathbf{x} \in \Re^n, y \in \Re$
- The objective: Find a function f that returns the best fit
- Assume that the relationship between X and y is approximately linear. The model can be represented as (w represents coefficients and b is an intercept)

  $$f(w_1, \ldots, w_n, b) = y = \mathbf{w} \cdot \mathbf{x} + b + \varepsilon$$
- To find the best fit, we minimize the sum of squared errors→Least square estimation

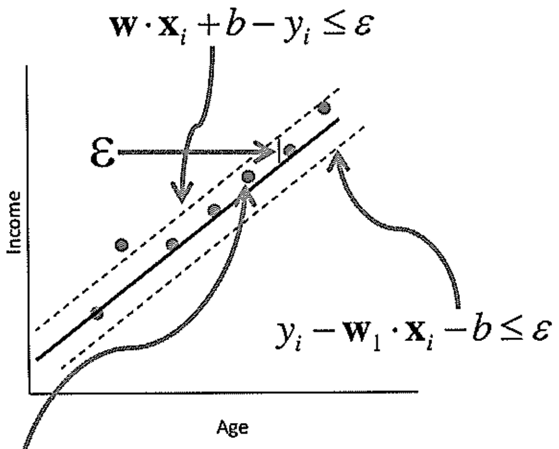
  $$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} (y_i - (\mathbf{w} \cdot \mathbf{x}_i + b))^2$$
- The solution can be found by solving (By taking the derivative of the above objective function w.r.t.w)

  $$\hat{\mathbf{w}} = (X^T X)^{-1} X^T Y$$
- To ovoid over-fitting, a regularization term can be introduced (minimize a magnitude of w)
  - LASSO: $\min \sum_{i=1}^{m} (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2 + C \sum_{j=1}^{n} |w_j|$
  - Ridge regression: $\min \sum_{i=1}^{m} (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2 + C \sum_{j=1}^{n} |\mathbf{w}_j^2|$

3 Support Vector Regression (SVR)

3.1 Hard margin SVR

- Find a function, f(x), with at most ε-deviation from the target y



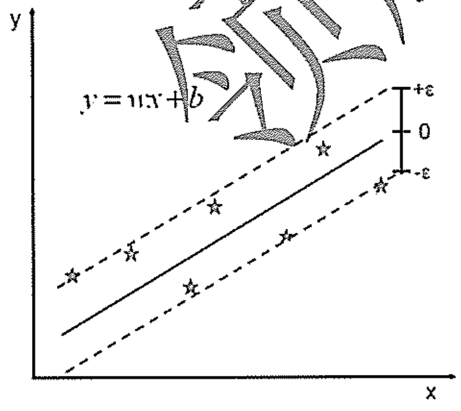
$\mathbf{w}\cdot\mathbf{x}_i + b - y_i \le \varepsilon$
ε
Income
Age
$y_i - \mathbf{w}_1\cdot\mathbf{x}_i - b \le \varepsilon$

We do not care about errors as long as they are less than ε

- Primal for hard margin SVR

$$\min \frac{1}{2}\|\mathbf{w}\|^2$$
$$s.t.\ y_i - \mathbf{w}_1\cdot\mathbf{x}_i - b \le \varepsilon;$$
$$\mathbf{w}_1\cdot\mathbf{x}_i + b - y_i \le \varepsilon;$$

- A better graph



- Solution:

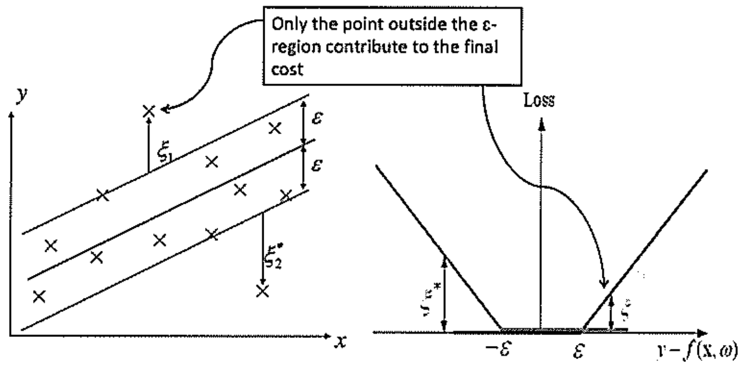
$$\min \frac{1}{2}\|w\|^2$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon$$
$$wx_i + b - y_i \le \varepsilon$$

3.2 Soft margin SVR

- What if the problem is not feasible? We can introduce slack variables (similar to soft margin loss function).



$$L_\varepsilon(y, f(\mathbf{x},\omega)) = \max\left(|y - f(\mathbf{x},\omega)| - \varepsilon, 0\right)$$

- Primal for soft margin SVR (l_1 norm)

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$

Under constraints
$$\begin{cases} y_i - (\mathbf{w}\cdot\mathbf{x}_i) - b \le \varepsilon + \xi_i \\ (\mathbf{w}\cdot\mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, i = 1, \ldots, m \end{cases}$$

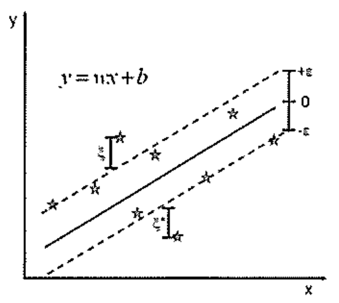
- Dual for soft margin SVR (l_1 norm)

$$\max\begin{cases} \frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\boxed{\langle x_i, x_j\rangle} \\ -\varepsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$s.t. \sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0;\quad 0 \le \alpha_i, \alpha_i^* \le C$$

- Primal vs. Dual

	Primal	Dual
Variables	w for each feature dim	a, a* for each data point
Complexity	the dim of the input space	Number of support vectors



• Minimize:
$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

• Constraints:
$$y_i - wx_i - b \le \varepsilon + \xi_i$$
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \ge 0$$

$$y = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)\cdot\langle x_i, x\rangle + b$$

- Primal & Dual for soft margin SVR (l_2 norm)

$$\begin{aligned} \text{minimize} - \mathbf{w}, \xi, \hat{\xi} \quad & \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_i (\xi_i^2 + \hat{\xi}_i^2) \\ \text{subject to} \quad & \mathbf{w}^T \Phi_i + b - y_i \le \varepsilon + \xi_i \ \forall i \\ & y_i - \mathbf{w}^T \Phi_i - b \le \varepsilon + \hat{\xi}_i \ \forall i \end{aligned}$$

The primal Lagrangian becomes

$$\mathcal{L}_P = \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_i (\xi_i^2 + \hat{\xi}_i^2) + \sum_i \alpha_i(\mathbf{w}^T\Phi_i + b - y_i - \varepsilon - \xi_i) + \sum_i \hat{\alpha}_i(y_i - \mathbf{w}^T\Phi_i - b - \varepsilon - \hat{\xi}_i)$$

Remark I: We could have added the constraints that $\xi_i \ge 0$ and $\hat{\xi}_i \ge 0$. However, it is not hard to see that the final solution will have that requirement automatically and there is no sense in constraining the optimization to the optimal solution as well. To see this, imagine some ξ_i is negative, then, by setting $\xi_i = 0$ the cost is lower and non of the constraints is violated, so it is preferred. We also note due to the above reasoning we will always have at least one of the $\xi, \hat{\xi}$ zero, i.e. inside the tube both are zero, outside the tube one of them is zero. This means that at the solution we have $\xi\hat{\xi} = 0$.
Remark II: Note that we don't scale $\varepsilon = 1$ like in the SVM case. The reason is that $\{y_i\}$ now determines the scale of the problem, i.e. we have not over-parameterized the problem.

We now take the derivatives w.r.t. $\mathbf{w}, b, \xi$ and $\hat{\xi}$ to find the following KKT conditions (there are more of course),

$$\mathbf{w} = \sum_i (\hat{\alpha}_i - \alpha_i)\Phi_i \quad (3)$$

$$\xi_i = \alpha_i / C \qquad \hat{\xi}_i = \hat{\alpha}_i / C \quad (4)$$

Plugging this back in and using that now we also have $\alpha_i\hat{\alpha}_i = 0$ we find the dual problem,

$$\begin{aligned} \text{maximize}_{\alpha,\hat{\alpha}} \quad & -\frac{1}{2}\sum_{ij}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)(K_{ij} + \frac{1}{C}\delta_{ij}) + \sum_i (\hat{\alpha}_i - \alpha_i)y_i - \sum_i (\hat{\alpha}_i + \alpha_i)\varepsilon \\ \text{subject to} \quad & \sum_i (\hat{\alpha}_i - \alpha_i) = 0 \\ & \alpha_i \ge 0, \ \hat{\alpha}_i \ge 0 \quad \forall i \end{aligned} \quad (5)$$

We now change variables to make this optimization problem look more similar to the SVM and ridge-regression case. Introduce $\beta_i = \hat{\alpha}_i - \alpha_i$ and use $\hat{\alpha}_i\alpha_i = 0$ to write $\hat{\alpha}_i + \alpha_i = |\beta_i|$,

$$\begin{aligned} \text{maximize}_{\beta} \quad & -\frac{1}{2}\sum_{ij}\beta_i\beta_j(K_{ij} + \frac{1}{C}\delta_{ij}) + \sum_i \beta_i y_i - \sum_i |\beta_i|\varepsilon \\ \text{subject to} \quad & \sum_i \beta_i = 0 \end{aligned} \quad (9)$$

3.3 Kernel trick

- Non-linear case
  - Linear: $\langle x, y \rangle$
  - Non-linear: $\langle \varphi(x), \varphi(y) \rangle = K(x, y)$

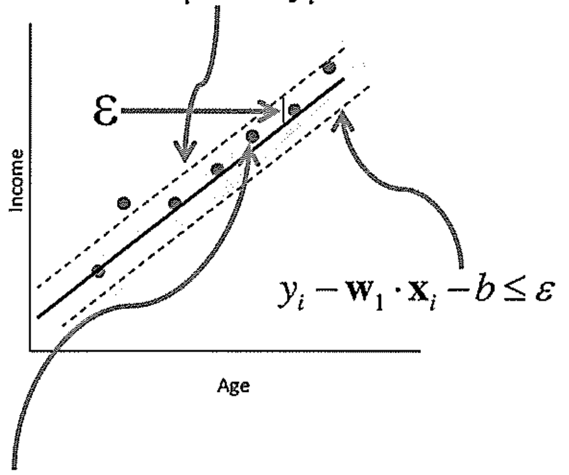
    Note: No need to compute the mapping function, φ(.), explicitly. Instead, we use the kernel function.

3 Support Vector Regression (SVR)

3.1 Hard margin SVR

- Find a function, f(x), with at most ε-deviation from the target y

$$\mathbf{w}\cdot\mathbf{x}_i + b - y_i \le \varepsilon$$

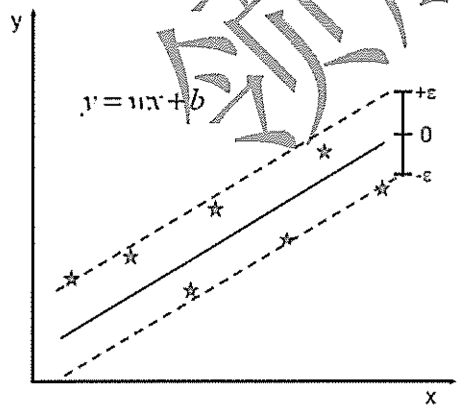


We do not care about errors as long as they are less than ε

- Primal for hard margin SVR

$$\min \frac{1}{2}\|\mathbf{w}\|^2$$
$$s.t.\ y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \le \varepsilon;$$
$$\mathbf{w}_1 \cdot \mathbf{x}_i + b - y_i \le \varepsilon;$$

- A better graph



- Solution:

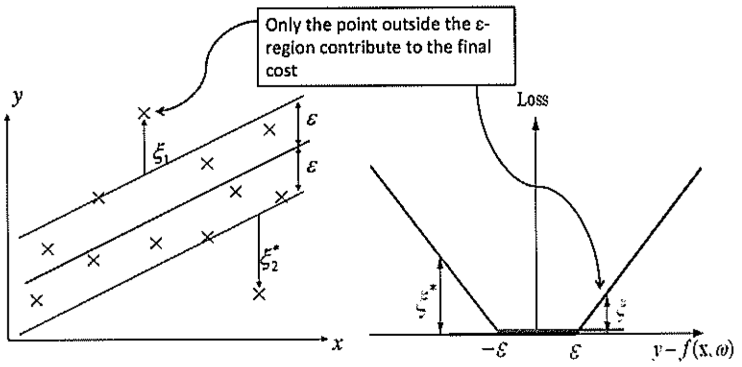
$$\min \frac{1}{2}\|w\|^2$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon$$
$$wx_i + b - y_i \le \varepsilon$$

3.2 Soft margin SVR

- What if the problem is not feasible? We can introduce slack variables (similar to soft margin loss function).


Only the point outside the ε-region contribute to the final cost
y
x
ξ_1
ξ_2^*
ε
ε
Loss
ξ^*
ξ
−ε
ε
$y-f(\mathbf{x},\omega)$

$$L_\varepsilon(y, f(\mathbf{x},\omega)) = \max(|\, y - f(\mathbf{x},\omega)\,| - \varepsilon, 0)$$

- Primal for soft margin SVR (l_1 norm)

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$

Under constraints
$$\begin{cases} y_i - (\mathbf{w}\cdot\mathbf{x}_i) - b \le \varepsilon + \xi_i \\ (\mathbf{w}\cdot\mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, i = 1,\ldots,m \end{cases}$$

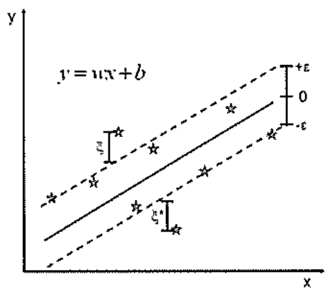
- Dual for soft margin SVR (l_1 norm)

$$\max\begin{cases} \frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\boxed{\langle x_i, x_j\rangle} \\ -\varepsilon\sum_{i=1}^{m}(\alpha_i+\alpha_i^*) + \sum_{i=1}^{m} y_i(\alpha_i-\alpha_i^*) \end{cases}$$

$$s.t. \sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0;\ 0 \le \alpha_i, \alpha_i^* \le C$$

- Primal vs. Dual

	Primal	Dual
Variables	w for each feature dim	a, a* for each data point
Complexity	the dim of the input space	Number of support vectors



• Minimize:
$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

• Constraints:
$$y_i - wx_i - b \le \varepsilon + \xi_i$$
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \ge 0$$

$$y = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)\cdot\langle x_i, x\rangle + b$$

- Primal & Dual for soft margin SVR (l_2 norm)

$$\begin{aligned} \text{minimize} - \mathbf{w}, \xi, \hat{\xi} \quad & \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_i (\xi_i^2 + \hat{\xi}_i^2) \\ \text{subject to} \quad & \mathbf{w}^T \Phi_i + b - y_i \leq \varepsilon + \xi_i \ \ \forall i \\ & y_i - \mathbf{w}^T \Phi_i - b \leq \varepsilon + \hat{\xi}_i \ \ \forall i \end{aligned}$$

The primal Lagrangian becomes

$$\mathcal{L}_P = \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_i (\xi_i^2 + \hat{\xi}_i^2) + \sum_i \alpha_i(\mathbf{w}^T\Phi_i + b - y_i - \varepsilon - \xi_i) + \sum_i \hat{\alpha}_i(y_i - \mathbf{w}^T\Phi_i - b - \varepsilon - \hat{\xi}_i)$$

Remark I: We could have added the constraints that $\xi_i \geq 0$ and $\hat{\xi}_i \geq 0$. However, it is not hard to see that the final solution will have that requirement automatically and there is no sense in constraining the optimization to the optimal solution as well. To see this, imagine some ξ_i is negative, then, by setting $\xi_i = 0$ the cost is lower and non of the constraints is violated, so it is preferred. We also note due to the above reasoning we will always have at least one of the $\xi, \hat{\xi}$ zero, i.e. inside the tube both are zero, outside the tube one of them is zero. This means that at the solution we have $\xi\hat{\xi} = 0$.
Remark II: Note that we don't scale $\varepsilon = 1$ like in the SVM case. The reason is that $\{y_i\}$ now determines the scale of the problem, i.e. we have not over-parameterized the problem.

We now take the derivatives w.r.t. $\mathbf{w}, b, \xi$ and $\hat{\xi}$ to find the following KKT conditions (there are more of course),

$$\mathbf{w} = \sum_i (\hat{\alpha}_i - \alpha_i)\Phi_i \tag{3}$$

$$\xi_i = \alpha_i / C \qquad \hat{\xi}_i = \hat{\alpha}_i / C \tag{4}$$

Plugging this back in and using that now we also have $\alpha_i\hat{\alpha}_i = 0$ we find the dual problem,

$$\begin{aligned} \text{maximize}_{\alpha,\hat{\alpha}} \quad & -\frac{1}{2}\sum_{ij}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)(K_{ij} + \frac{1}{C}\delta_{ij}) + \sum_i (\hat{\alpha}_i - \alpha_i)y_i - \sum_i (\hat{\alpha}_i + \alpha_i)\varepsilon \\ \text{subject to} \quad & \sum_i (\hat{\alpha}_i - \alpha_i) = 0 \\ & \alpha_i \geq 0, \ \hat{\alpha}_i \geq 0 \quad \forall i \end{aligned} \tag{5}$$

We now change variables to make this optimization problem look more similar to the SVM and ridge-regression case. Introduce $\beta_i = \hat{\alpha}_i - \alpha_i$ and use $\hat{\alpha}_i\alpha_i = 0$ to write $\hat{\alpha}_i + \alpha_i = |\beta_i|$,

$$\begin{aligned} \text{maximize}_{\beta} \quad & -\frac{1}{2}\sum_{ij}\beta_i\beta_j(K_{ij} + \frac{1}{C}\delta_{ij}) + \sum_i \beta_i y_i - \sum_i |\beta_i|\varepsilon \\ \text{subject to} \quad & \sum_i \beta_i = 0 \end{aligned} \tag{9}$$

3.3 Kernel trick

- Non-linear case
  - Linear: $\langle x, y \rangle$
  - Non-linear: $\langle \varphi(x), \varphi(y) \rangle = K(x, y)$

Note: No need to compute the mapping function, φ(.), explicitly. Instead, we use the kernel function.

ISM - Deep Learning_1

Parametric Approach: Linear Classifier

Image

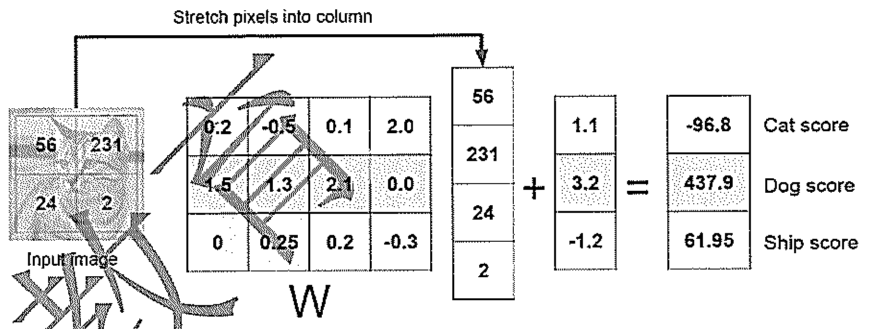
Array of **32x32x3** numbers
(3072 numbers total)

$$\underset{10\times1}{\boxed{f(x,W)}} = \underset{10\times3072}{\boxed{W}}\overset{3072\times1}{\boxed{x}} + \boxed{b}\ \mathbf{10x1}$$

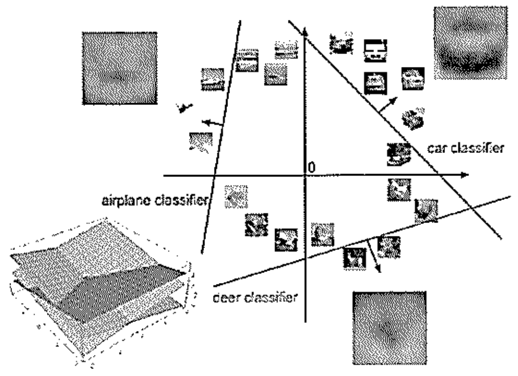
$$\xrightarrow{\quad} f(x,W) \xrightarrow{\quad}$$

10 numbers giving class scores

W
parameters
or weights



Interpreting a Linear Classifier



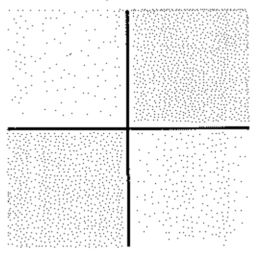
$f(x,W) = Wx + b$

Array of **32x32x3** numbers
(3072 numbers total)

Hard cases for a linear classifier

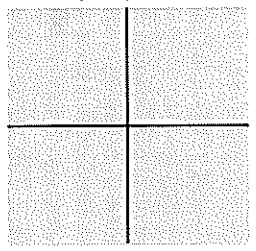
Class 1:
number of pixels > 0 odd

Class 2:
number of pixels > 0 even



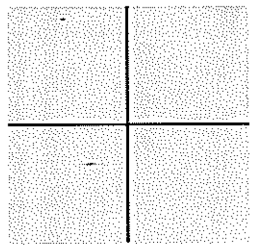
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

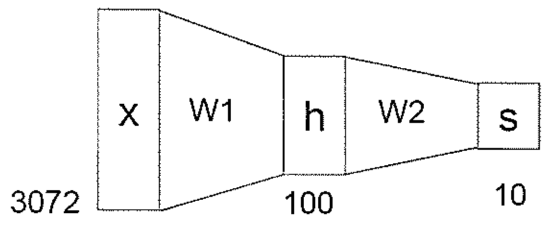
Class 2:
Everything else

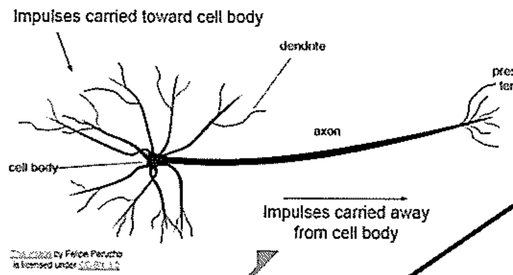


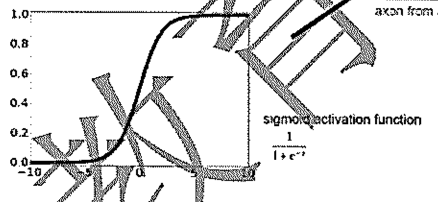
Neural networks

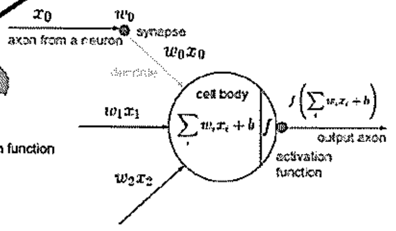
(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$



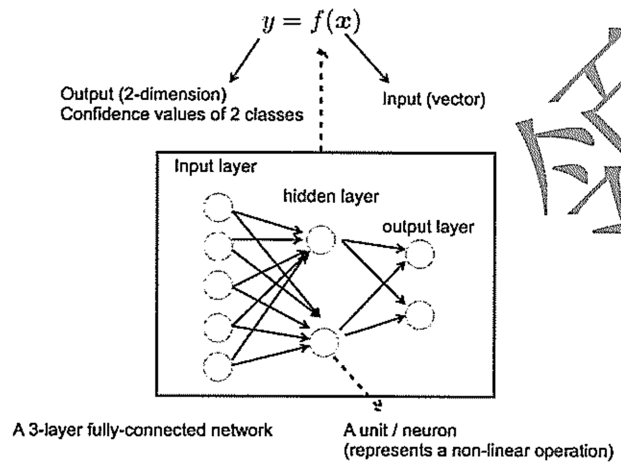

Impulses carried toward cell body
dendrite
presynaptic terminal
axon
cell body
Impulses carried away from cell body



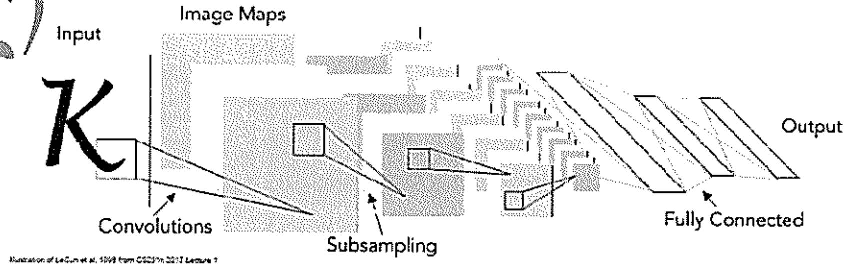


Multi-Layer Neural Network

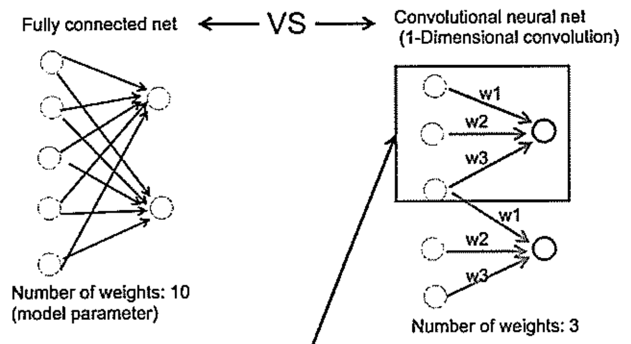
Example: classification of 2 classes.
A network is a complex non-linear mapping function:


$y = f(x)$
Output (2-dimension)
Confidence values of 2 classes
Input (vector)
Input layer
hidden layer
output layer
A 3-layer fully-connected network
A unit / neuron
(represents a non-linear operation)

Convolutional Neural Networks(CNNs)

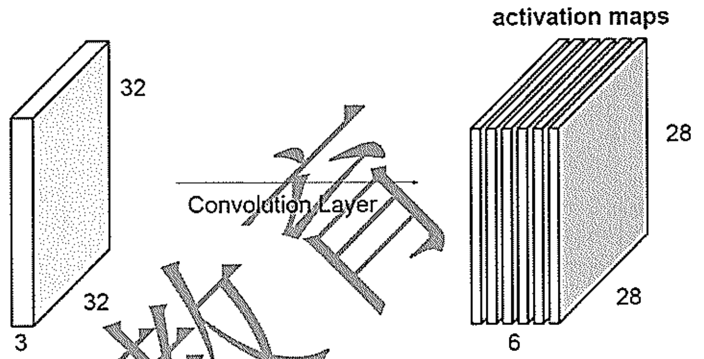


Convolutional Layer (1D)



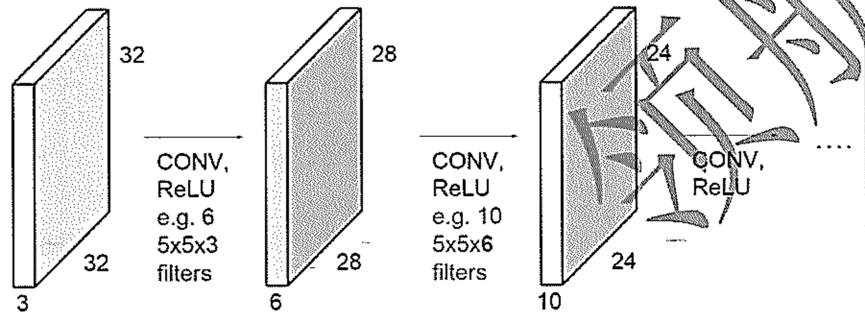
CNN layer properties:
1. local connectivity (spatially-local connection)
2. sharing weights
   (red and blue indicate two groups of edges which use the same set of weights)

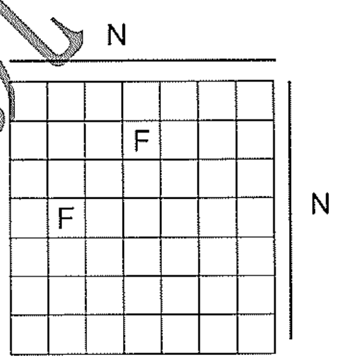
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

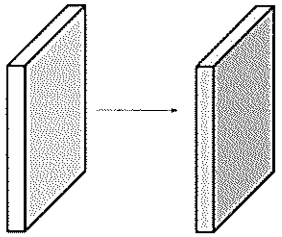

32
32
3
CONV, ReLU e.g. 6 5x5x3 filters
28
28
6
CONV, ReLU e.g. 10 5x5x6 filters
24
24
10
CONV, ReLU
....



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

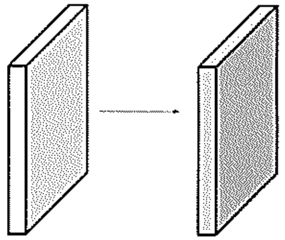
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Examples time:



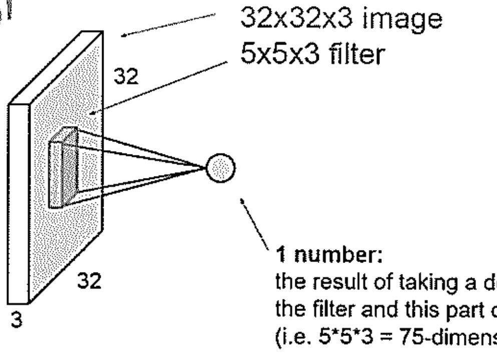
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

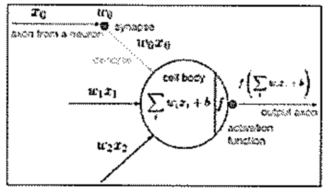
Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Summary: the Conv layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

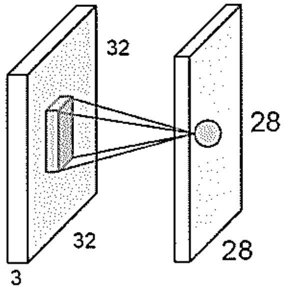
The neuron view of CONV Layer

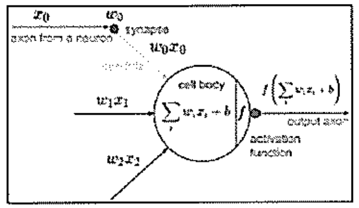

32x32x3 image
5x5x3 filter
32
32
3
1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)



It's just a neuron with local connectivity...

The neuron view of CONV Layer



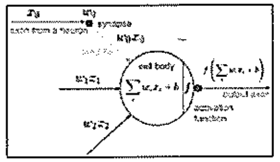


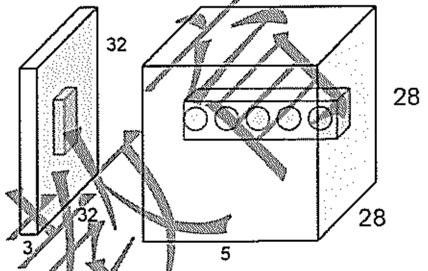
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

The neuron view of CONV Layer



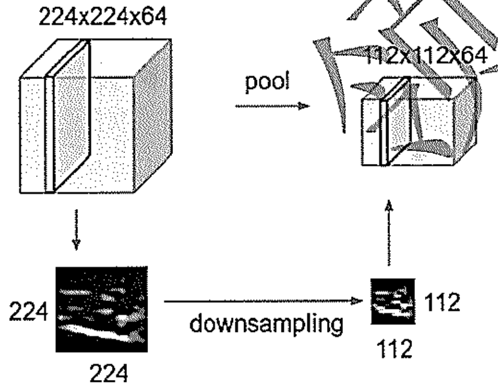


E.g. with 5 filters,
CONV layer consists of neurons arranged in a 3D grid (28x28x5)

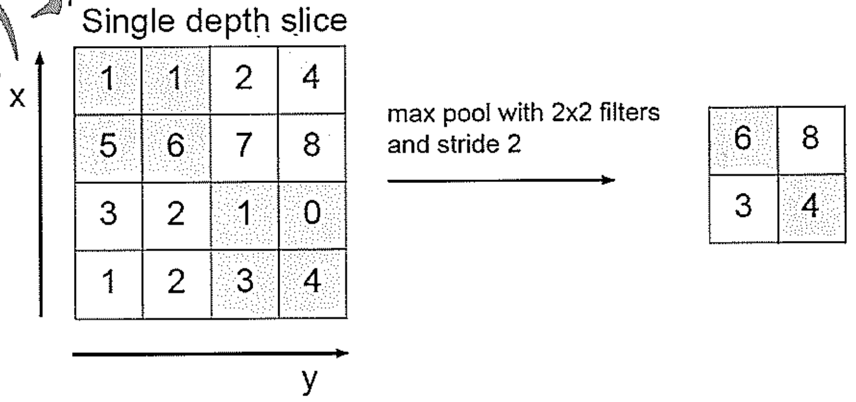
There will be 5 different neurons all looking at the same region in the input volume

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



Max Pooling



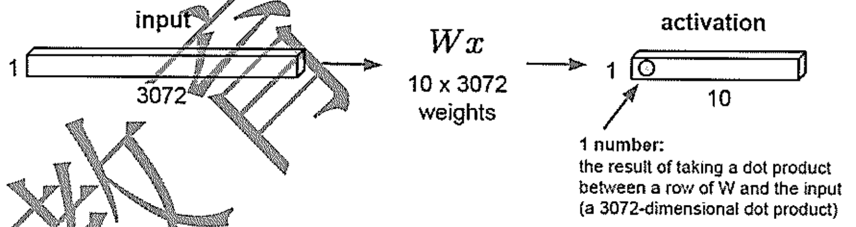
Pooling Layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume



input
1
3072
Wx
10 x 3072 weights
activation
1
10
1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

Summary

- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like

  [(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX
where N is usually up to ~5, M is large, 0 <= K <= 2.

  but recent advances such as ResNet/GoogLeNet, challenge this paradigm

[illegible]

ISML - Feature Extraction (PCA & LDA)

1 Dimensionality reduction

- Feature selection: select a subset of a given feature set
- Feature extraction: a linear or non-linear transform on the original feature space

2 Example of Feature Extraction

Each of the resulting images is represented by a point in the 100 x 100 = 10, OOO-dimensional data space. However, across a data set of such images, there are only three degrees of freedom of variability, corresponding to the vertical and horizontal translations and the rotations. The data points will therefore live on a subspace of the data space whose intrinsic dimensionality is three.

3 Applications of Feature Extraction

- **Visualization:** projection of high-dimensional data onto 2D or 3D
- **Data compression:** efficient storage, communication, and retrieval
- **Noise removal:** to improve accuracy by removing irrelevant features

4 Notations

- Observations and projected data

$$X' = A^T X \quad (x'_j = a_j^T x_j \text{ , for each sample})$$

- Mean

$$\mu = \bar{X} = \frac{1}{N}\sum X_i \qquad \mu' = \bar{X'} = \frac{1}{N}\sum X_i' = \frac{1}{N}\sum \alpha_i^T X_i$$

- Covariance

$$S = \frac{1}{N}\sum (X_i - \bar{X})(X_i - \bar{X})^T$$

$$\begin{aligned} S' &= \frac{1}{N}\sum (X_i' - \bar{X}')(X_i' - \bar{X}')^T \\ &= \frac{1}{N}\sum (\alpha_i^T X_i - \alpha_i^T \bar{X})(\alpha_i^T X_i - \alpha_i^T \bar{X})^T \\ &= \frac{1}{N}\sum [\alpha_i^T (X_i - \bar{X})][\alpha_i^T (X_i - \bar{X})]^T \\ &= \frac{1}{N}\sum \alpha_i^T (X_i - \bar{X})(X_i - \bar{X})^T \alpha_i \\ &= \alpha_i^T \left[\frac{1}{N}\sum (X_i - \bar{X})(X_i - \bar{X})^T\right] \alpha_i \\ &= \alpha_i^T S \alpha_i \end{aligned}$$

- Eigenvector and Eigenvalue

$$Ax = \lambda x$$

$x \to$ Eigenvector of A

$\lambda \to$ eigenvalue of x

5 PCA

5.1 Goal of PCA

Reducing the dimensionality of the data while preserving the variation present in the dataset as much as possible.

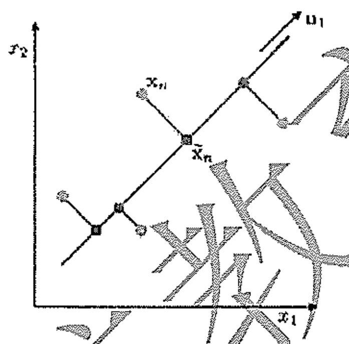
5.2 Methods (interpretations) of PCA

- Maximize variance

The orthogonal projection of the data onto a lower dimensional linear space, known as the principal subspace, such that the variance of the projected data is maximized

- Minimize the MSE (mean squared distance)

The linear projection that minimizes the average projection cost, defined as the mean squared distance between the data points and their projections



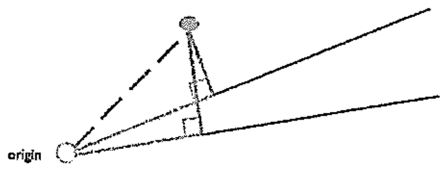
5.3 Steps of PCA

- Input: $N \times d$ data matrix X (each row contain a d dimensional data point)
  - $\mu = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$
  - $\widetilde{X} \leftarrow$ Mean value of data points is subtracted from rows of X
  - $\Sigma = \frac{1}{N}\widetilde{X}^T\widetilde{X}$ (Covariance matrix)
  - Calculate eigenvalue and eigenvectors of Σ
  - Pick d' eigenvectors corresponding to the largest eigenvalues and put them in the columns of $A = [\underset{\substack{\downarrow \\ \text{First PC}}}{v_1}, \dots, \underset{\substack{\downarrow \\ \text{d'-th PC}}}{v_{d'}}]$
  - $X' = A^T X$

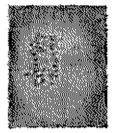
5.4 Topics about PCA

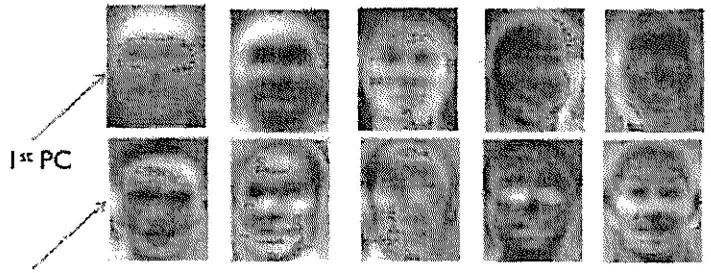
- Least squares error = Maximum variance

Minimizing sum of square distances to the line is equivalent to maximizing the sum of squares of the projections on that line.



⑤


Average face

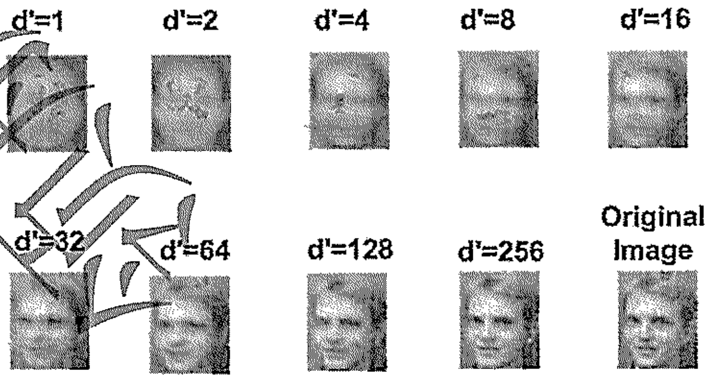


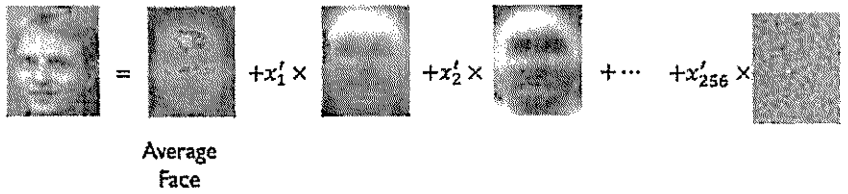
For eigen faces
"gray" = 0,
"white" > 0,
"black" < 0

x is a $112 \times 92 = 10304$ dimensional vector containing intensity of the pixels of this image

Feature vector=$[x_1', x_2', \dots, x_{d'}']$

$x_i' = PC_i^T x$ ⟶ The projection of x on the i-th PC

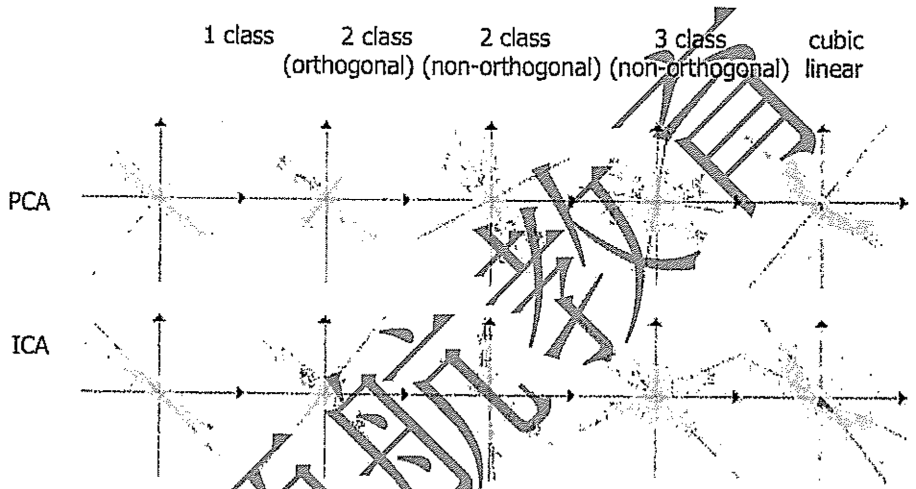




5.6 ICA (Independent Component Analysis)

- PCA:
  - The transformed dimensions will be uncorrelated from each other
  - Orthogonal linear transform
  - Only uses second order statistics (i.e., covariance matrix)

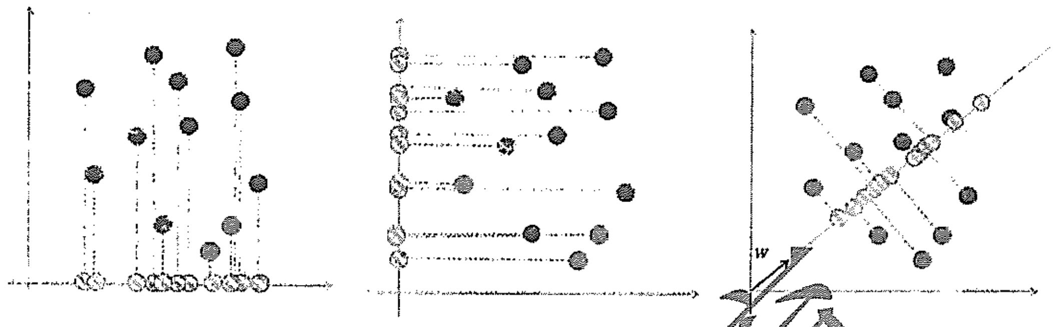
- ICA:
  - The transformed dimensions will be as independent as possible.
  - Non-orthogonal linear transform
  - High-order statistics are used



6 Linear Discriminant Analysis (LDA)

- Supervised feature extraction
- Dimensionality reduction

  Finds linear combinations of features with large ratios of between-groups to within-groups scatters
- Classification

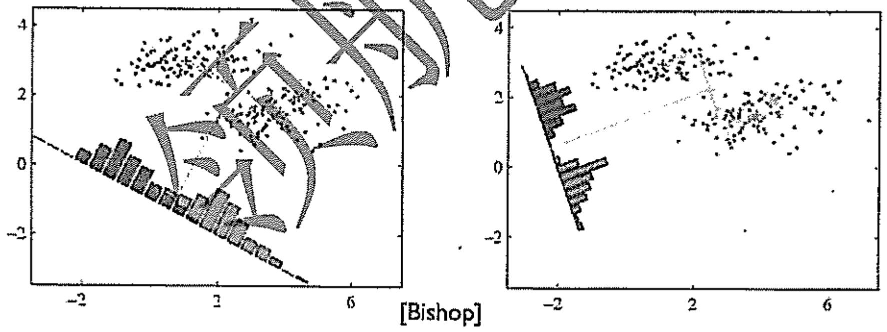


6.1 LDA Goal

- Finding the best direction w that we hope will enable accurate classification -> Maximum the separation of projected data

6.2 Measure of separation

- The projection of sample x onto a line in direction w is w^Tx
- Is the direction of the line jointing the class means is a good candidate for w? - No



[Bishop]

- Why? It does not consider the variances of the classes
  - The direction of the line jointing the class means is the solution of the following problem:
    - Maximizes the separation of the projected class means

$$\max_{w} J(w) = (\mu_1' - \mu_2')^2 \qquad \mu_1' = w^T \mu_1$$
$$\text{s.t.} \|w\| = 1 \qquad \mu_2' = w^T \mu_2$$

  - What is the problem with the criteria considering only $|\mu_1' - \mu_2'|$?
    - It does not consider the variances of the classes

5.7

mean of data : $\bar{X} = \frac{1}{N} \Sigma X_i$

mean of projected data : $\bar{X}' = a_i^T \bar{X} = \frac{1}{N} \Sigma a_i^T X_i$

covariance of data : $S = \frac{1}{N} \Sigma (X_i - \bar{X})(X_i - \bar{X})^T$

covariance of projected data

$$S' = a_i^T S a_i$$

Goal : max $S' = a_i^T S a_i$, st $a_i^T a_i = 1$

Apply Lagrange Multiplier

$$L = a_i^T S a_i + \Sigma \lambda_i (1 - a_i^T a_i)$$

$$\frac{\partial L}{\partial a_i} = 2 S a_i - 2 \lambda_i a_i = 0$$

$\Rightarrow S a_i = \lambda_i a_i \leftarrow$ eigenvalue problem

if we set a_i as a eigenvector of S

and λ_i as the eigenvalue of a_i

we get the max of $S' = a_i^T S a_i$

△ furthermore

$S a_i = \lambda_i a_i \Rightarrow a_i^T S a_i = a_i^T \lambda_i a_i = \lambda_i$

$\Rightarrow \lambda_i$ is the variance of projected data on a_i.

the largest eigenvalue of S is the first principle component

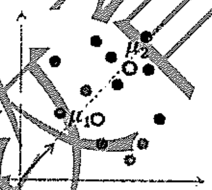
- Fisher idea (LDA Criteria)
  - Fisher idea: maximize a function that will give
    - large separation between the projected class means
    - while also achieving a small variance within each class, thereby minimizing the class overlap.

$$J(w) = \frac{|\mu_1' - \mu_2'|^2}{s_1'^2 + s_2'^2}$$

6.3 LDA Algorithm (for 2 classes)

- μ_1 and μ_2 ← mean of samples of class 1 and 2 respectively
- S_1 and S_2 ← scatter matrix of class 1 and 2 respectively
- $S_W = S_1 + S_2$
- $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$

- Feature Extraction
  - $w = S_W^{-1}(\mu_1 - \mu_2)$ as the eigenvector corresponding to the largest eigenvalue of $S_W^{-1}S_b$
- Classification
  - $w = S_W^{-1}(\mu_1 - \mu_2)$
  - Using a threshold on $w^T x$, we can classify x



6.4

Goal: max $J(w) = \frac{|\mu_1' - \mu_2'|^2}{s_1'^2 + s_2'^2}$

where $|\mu_1'-\mu_2'|^2 = |w^T\mu_1 - w^T\mu_2|^2 = (w^T\mu_1 - w^T\mu_2)(w^T\mu_1 - w^T\mu_2)^T$

$= [w^T(\mu_1-\mu_2)][w^T(\mu_1-\mu_2)]^T$

$= w^T(\mu_1-\mu_2)(\mu_1-\mu_2)^T w$

Scatter: $s_1'^2 = \sum \|w^T x_i - w^T\mu_1\|^2 = \sum (w^T x_i - w^T\mu_1)(w^T x_i - w^T\mu_1)^T$

$= w^T\left(\sum (x_i-\mu_1)(x_i-\mu_1)^T\right)w$

$s_2'^2 = w^T\left(\sum (x_i-\mu_2)(x_i-\mu_2)^T\right)w$

$\Rightarrow s_1'^2 + s_2'^2 = w^T\left(\sum(x_i-\mu_1)(x_i-\mu_2)^T + \sum(x_i-\mu_2)(x_i-\mu_2)^T\right)w$

set. $S_B = (\mu_1-\mu_2)(\mu_1-\mu_2)^T$

$S_W = \sum(x_i-\mu_1)(x_i-\mu_1)^T + \sum(x_i-\mu_2)(x_i-\mu_2)^T$

$\Rightarrow J(w) = \frac{w^T S_B w}{w^T S_W w}$

$\frac{\partial J}{\partial w} = \frac{w^T S_B w \times \frac{\partial (w^T S_W w)}{\partial w} - \frac{\partial (w^T S_B w)}{\partial w} \times w^T S_W w}{(w^T S_W w)^2}$

$= \frac{w^T S_B w \times 2 S_W w - 2 S_B w \times w^T S_W w}{(w^T S_W w)^2} = 0$

$\Rightarrow \quad w^T S_B w \times S_W w = S_B w \times w^T S_W w$

$\Rightarrow \quad S_B w = \lambda S_W w \quad \leftarrow$ eigenvalue problem

$S_W^{-1} S_B w = \lambda w \quad \swarrow$

w: eigenvector of $S_W^{-1} S_B$

λ: eigenvalue

How to solve w

$S_B w$ always in the direction of $\mu_1 - \mu_2$

$S_B w = \alpha(\mu_1-\mu_2) = \lambda S_W w$

$S_W^{-1}(\mu_1-\mu_2) = w$

6.5 Multi-class LDA (MDA)

Multi-Class LDA (MDA)

- $C > 2$: the natural generalization of LDA involves $C-1$ discriminant functions.
  - The projection from a d-dimensional space to a $(C-1)$-dimensional space (tacitly assumed that $d \geq C$).

$$S_W = \sum_{j=1}^{C} S_j$$

$$S_B = \sum_{j=1}^{C} N_j(\mu_j - \mu)(\mu_j - \mu)^T$$

$\mu_j = \frac{\sum_{x^{(i)} \in C_j} x^{(i)}}{N_j} \quad j = 1, \dots, C$

$\mu = \frac{\sum_{i=1}^{N} x^{(i)}}{N}$

$S_j = \sum_{x^{(i)} \in C_j} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T \quad j = 1, \dots, C$

Multi-Class LDA

- $\boldsymbol{W} = [\boldsymbol{w}_1\ \boldsymbol{w}_2\ \dots \boldsymbol{w}_{C-1}]$
- $\boldsymbol{x}' = \boldsymbol{W}^T \boldsymbol{x}$

- Means and scatters after transform $\boldsymbol{x}' = \boldsymbol{W}^T \boldsymbol{x}$:
  - $S'_B = W^T S_B W$
  - $S'_W = W^T S_W W$

Multi-Class LDA: Objective Function

- We seek a transformation matrix W that in some sense "maximizes the ratio of the between-class scatter to the within-class scatter".

- A simple scalar measure of scatter is the **determinant of the scatter matrix.**

Multi-Class LDA: Objective Function

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|} \longrightarrow \text{determinant}$$

- The solution of the problem where $W = [w_1\, w_2\, \dots w_{C-1}]$:

$$S_B w_i = \lambda_i S_W w_i$$

- It is a generalized eigenvectors problem.

Multi-Class LDA: $d' \leq C - 1$

- $rank(S_B) \leq C - 1$
  - S_B is the sum of C matrices $(\mu_j - \mu)(\mu_j - \mu)^T$ of rank (at most) one and only $C - 1$ of these are independent,
  - $\Rightarrow$ atmost $C - 1$ nonzero eigenvalues and the desired weight vectors correspond to these nonzero eigenvalues.

Multi-Class LDA: Other Objective Functions

- There are many possible choices of criterion for multi-class LDA, e.g.:

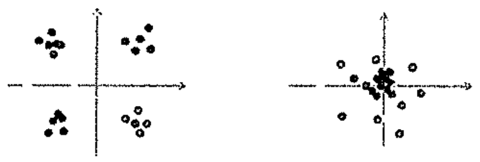
$$J(W) = tr(S_W'^{-1} S_B') = tr\left((W^T S_W W)^{-1} (W^T S_B W)\right)$$

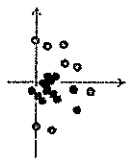
- The solution is given by solving a generalized eigenvalue problem $S_w^{-1} S_b$
  - Solution: eigen vectors corresponding to the largest eigen values constitute the new variables

6.6 Topics about LDA

LDA Criterion Limitation

- When $\mu_1 = \mu_2$, LDA criterion can not lead to a proper projection ($J(w) = 0$)
  - However, discriminatory information in the scatter of the data may be helpful





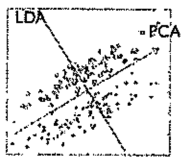
- If classes are non-linearly separable they may have large overlap when projected to any line
- LDA implicitly assumes Gaussian distribution of samples of each class

Issues in LDA

- Singularity or undersampled problem (when $N < d$)
  - Example: gene expression data, images, text documents
- Can reduces dimension only to $d' \leq C - 1$ (unlike PCA)
- Approaches to avoid these problems:
  - PCA+LDA, Regularized LDA, Locally FDA (LFDA), etc.

Summary

- Although LDA often provide more suitable features for classification tasks, PCA might outperform LDA in some situations such as:
  - when the number of samples per class is small (overfitting problem of LDA)
  - when the training data non-uniformly sample the underlying distribution
  - when the number of the desired features is more than $C - 1$
- Advances in the recent decade:
  - Semi-supervised feature extraction
  - Nonlinear dimensionality reduction



Question 4

(a) What are the main purposes of a principal component analysis?

[5 marks]

(b) In the case of using PCA for dimensionality reduction, How does one determine the appropriate number of principal components to retain?

[5 marks]

[Total for Question 4: 10 marks]

ISML 0 - Basic & Bayesian Decision Theory

1 Basic

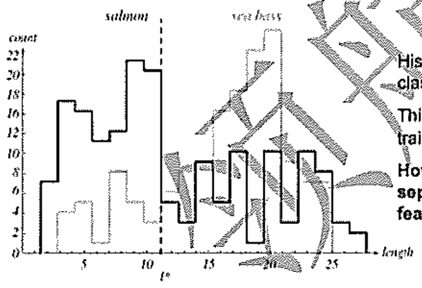
1.1 Notations

- **Pattern** - an example, instance, sample, or specimen
- **Recognition** - the identification of something as having been previously seen, heard, known, etc.
- **Pattern recognition** - the act of taking in raw data and making an action based on the "category" of the pattern.
    - o Classification theory is the most important domain- independent theory of pattern recognition

1.2 Process of ML

A binary classification problem (sea bass vs. salmon)

- Data collection
  set up a camera to collect the sample images from the two classes
- Study the difference between the sample images
  Length, lightness, width, number and shape of fins, position of the mouth, etc
- Feature extraction
  Extract features which are used to represent each sample.
    - o Case one
        - We observe that "*a sea bass is generally longer than a salmon*"
        - An obvious feature: *the length of a fish*
        - Choose a threshold in length l^*, to classify a fish
        - We need a set of **training samples** to choose such a threshold

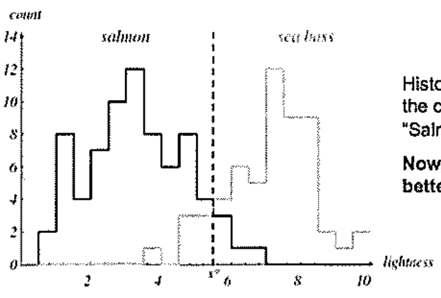


Histogram for the length feature for the classes of "Sea bass" and "Salmon".

This histogram is obtained from the training samples

However, we cannot reliably separate them by using the "length" feature alone.

    - o Case two
        - Another feature: *the lightness of fish scales*
        - Choose a threshold in lightness x^*, , to classify a fish

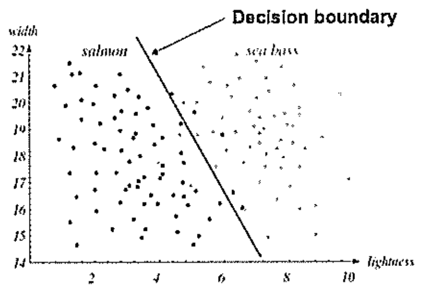


Histogram for the lightness feature for the classes of "Sea bass" and "Salmon".

Now, the two classes are much better separated from each other

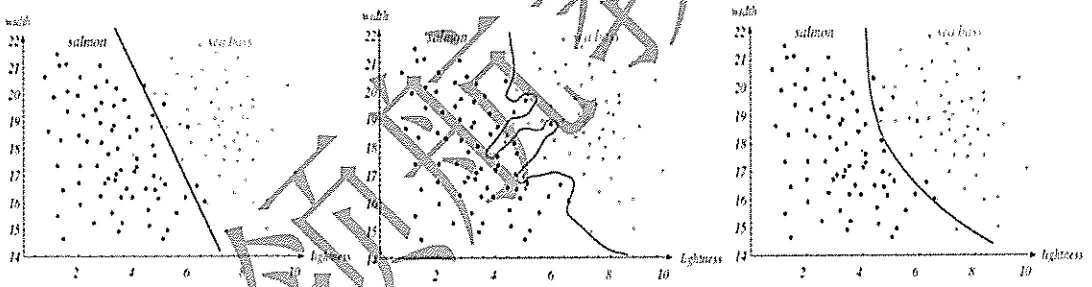
- Case three
  - We are still not satisfied with the **classification performance**
  - But what if no *single* feature is better than the "*lightness*"?
  - We must resort to the use of **more than one feature**
  - Each fish a 2D feature vector a point in a 2D feature space
  - Classification: partition the feature space into two regions

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$



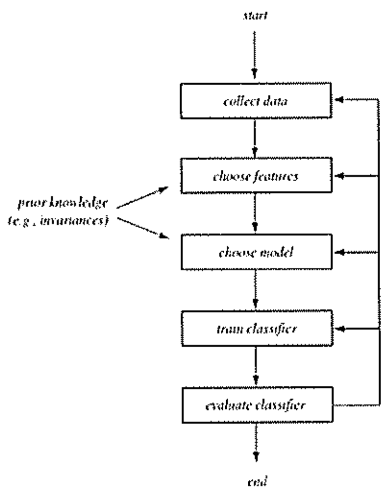
1.3 Generalization

- Does using more features always improve classification result? No
- How to select the most useful features and avoid using redundant features?
- Generalize: to make generally or universally applicable
- A classifier is used to suggest actions on "new" samples
- A complex decision boundary "tuned" to the particular training samples will not provide good generalization performance
- Trade-off between training error and model complexity



Good Poor Better

1.4 Design of a recognition system



- Collect a **sufficient** number of **representative** samples
- Find features
  - **Discriminative**
  - **Simple** to extract
  - **Invariant** to irrelevant changes
  - **robust** to noise
- Select a good model
- Training — estimate the classifier parameters using the training samples
- Evaluation
  - Measure the performance
  - Identify the need for improvements
- Computational complexity

1.5 Type of learning

- Learning
  - A process of incorporating information from training samples
  - Algorithms for reducing the error on training samples
- Supervised learning
  - Each training sample has a label or target
- Unsupervised learning
  - There is no label or target value.
  - Learning algorithms are used to form "natural groupings" of training samples
- Reinforcement learning
  - concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward
  - Correct input/output pairs need not be presented, and sub-optimal actions need not be explicitly corrected
  - The focus is on performance, which involves finding a balance between exploration and exploitation

2 Bayesian Decision Theory

- A fundamental statistical approach to the problem of pattern recognition
- The decision problem is posed in probabilistic terms.

2.1 Priori probability

- e.g., ω=ω1 for "sea bass", ω=ω2 for "salmon"
- ω1∪ω2 =U and ω1 ∩ω2 =∅
- a priori probability P(ω1), P(ω2)
- P(ω1) + P(ω2) = 1
- Reflect our prior knowledge of how likely ω1 or ω2 will happen **before** we see any training samples
- A decision based on the prior probabilities

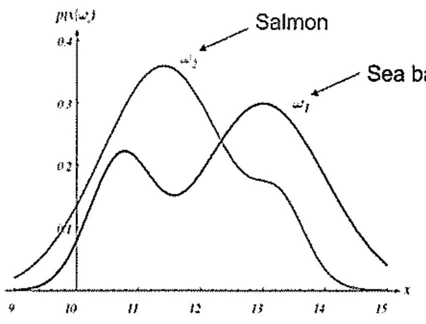
$$\omega = \begin{cases} \omega_1, & if\ P(\omega_1) > P(\omega_2) \\ \omega_2, & otherwise \end{cases}$$

- Probability of error

$$min[1 - P(\omega_1), 1 - P(\omega_2)]$$

2.2 Class-conditional probability

- Feature: the "lightness" of fish scales, x
- x , a continuous variable, whose distribution depends on the state of nature ω is expressed as p(x|ω)
- p(x|ω): class-conditional probability density function

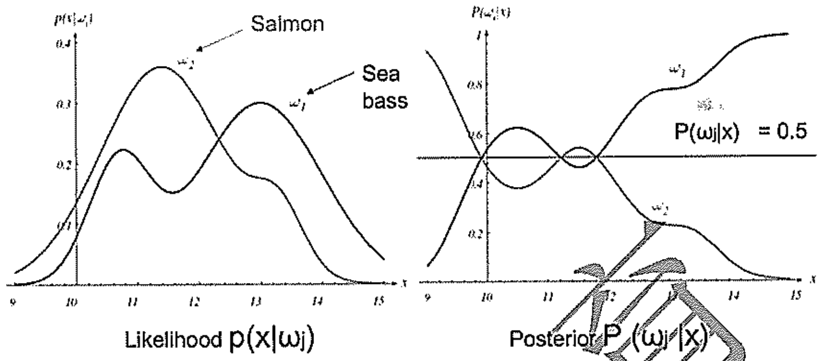


2.3 Bayes formula

- Suppose we have known P(ωi) and p(x|ωi) (i = 1, 2)
- Now, a measure of "lightness", x, become available (we know the p(x))
- Question: what is the category of this fish?
- $P(\omega_j|x) = \frac{p(x|\omega_j)\cdot P(\omega_j)}{p(x)} \Rightarrow posterior = \frac{likelihood \times prior}{evidence}$

2.4 Example

- Let prior probability be $P(\omega_1) = \frac{2}{3}$ and $P(\omega_2) = \frac{1}{3}$



- Make a decision based on posterior probabilities, $P(\omega_j|x)$

$$\omega = \begin{cases} \omega_1, & if\ P(\omega_1|x) > P(\omega_2|x) \\ \omega_2, & otherwise \end{cases}$$

- The probability of error

$$P(error|x) = \begin{cases} 1 - P(\omega_1|x), & if\ we\ decide\ \omega = \omega_1 \\ 1 - P(\omega_2|x), & if\ we\ decide\ \omega = \omega_2 \end{cases}$$

$$P(error|x) = min[1 - P(\omega_1|x), 1 - P(\omega_2|x)]$$

ISML 01 - Basic & SVM

1 Basic

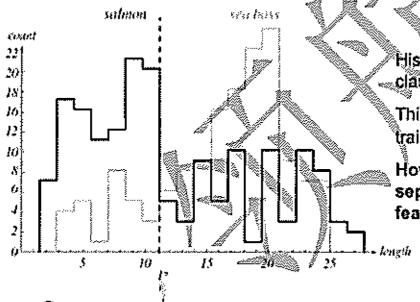
1.1 Notations

- **Pattern** - an example, instance, sample, or specimen
- **Recognition** - the identification of something as having been previously seen, heard, known, etc.
- **Pattern recognition** - the act of taking in raw data and making an action based on the "category" of the pattern.
    - o Classification theory is the most important domain- independent theory of pattern recognition

1.2 Process of ML

A binary classification problem (sea bass vs. salmon)

- Data collection
  set up a camera to collect the sample images from the two classes
- Study the difference between the sample images
  Length, lightness, width, number and shape of fins, position of the mouth, etc
- Feature extraction
  Extract features which are used to represent each sample.
    - o Case one
      - We observe that "*a sea bass is generally longer than a salmon*"
      - An obvious feature: *the length of a fish*
      - Choose a threshold in length l^*, to classify a fish
      – We need a set of **training samples** to choose such a threshold

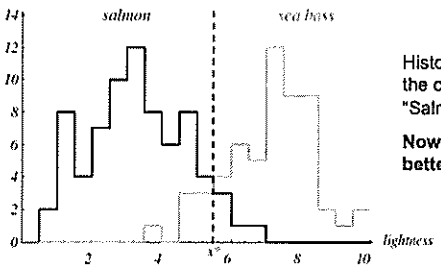


Histogram for the length feature for the classes of "Sea bass" and "Salmon".

This histogram is obtained from the training samples

However, we cannot reliably separate them by using the "length" feature alone.

- o Case two
  - Another feature: *the lightness of fish scales*
  - Choose a threshold in lightness x^*, , to classify a fish

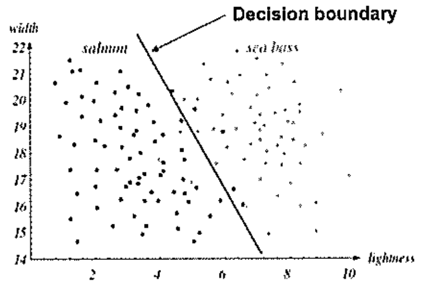


Histogram for the lightness feature for the classes of "Sea bass" and "Salmon".

Now, the two classes are much better separated from each other

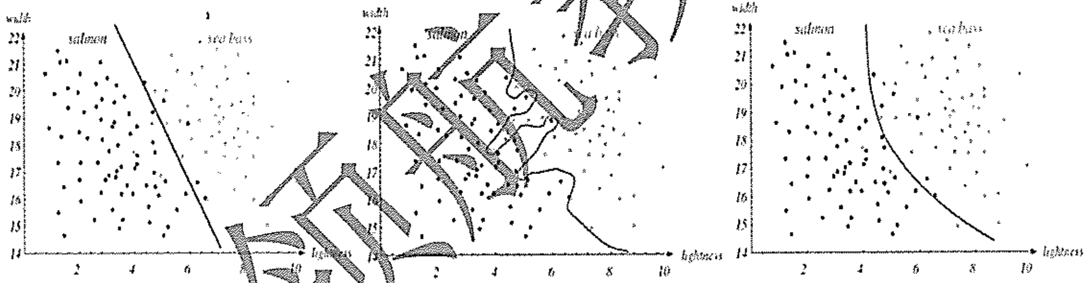
- Case three
  - We are still not satisfied with the **classification performance**
  - But what if no *single* feature is better than the "*lightness*"?
  - We must resort to the use of **more than one feature**
  - Each fish   a 2D feature vector   a point in a 2D feature space
  - Classification: partition the feature space into two regions

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$



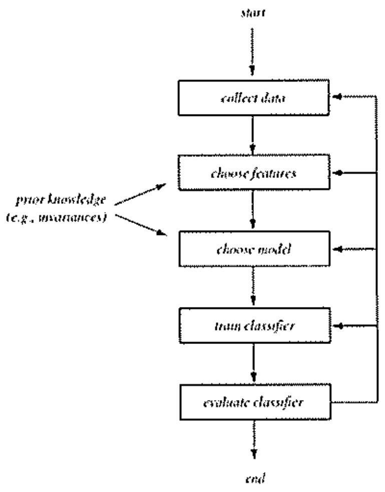
1.3 Generalization

- Does using more features always improve classification result? No
- How to select the most useful features and avoid using redundant features?
- Generalize: to make generally or universally applicable
- A classifier is used to suggest actions on "new" samples
- A complex decision boundary "tuned" to the particular training samples will not provide good generalization performance
- Trade-off between training error and model complexity



Good Poor Better

1.4 Design of a recognition system



- Collect a **sufficient** number of **representative** samples
- Find features
  - **Discriminative**
  - **Simple** to extract
  - **Invariant** to irrelevant changes
  - **robust** to noise
- Select a good model
- Training — estimate the classifier parameters using the training samples
- Evaluation
  - Measure the performance
  - Identify the need for improvements
- Computational complexity

1.5 Type of learning

- Learning
  - A process of incorporating information from training samples
  - Algorithms for reducing the error on training samples
- Supervised learning
  - Each training sample has a label or target
- Unsupervised learning
  - There is no label or target value.
  - Learning algorithms are used to form "natural groupings" of training samples
- Reinforcement learning
  - concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward
  - Correct input/output pairs need not be presented, and sub-optimal actions need not be explicitly corrected
  - The focus is on performance, which involves finding a balance between exploration and exploitation

2 Bayesian Decision Theory

- A fundamental statistical approach to the problem of pattern recognition
- The decision problem is posed in probabilistic terms.

2.1 Priori probability

- e.g., $\omega=\omega1$ for "sea bass", $\omega=\omega2$ for "salmon"
- $\omega1 \cup \omega2 = U$ and $\omega1 \cap \omega2 = \emptyset$
- a priori probability $P(\omega1)$, $P(\omega2)$
- $P(\omega1) + P(\omega2) = 1$
- Reflect our prior knowledge of how likely $\omega1$ or $\omega2$ will happen **before** we see any training samples
- A decision based on the prior probabilities

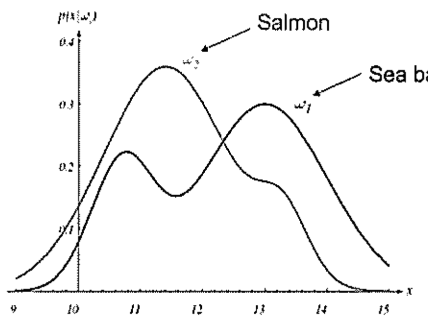
$$\omega = \begin{cases} \omega_1, & if\ P(\omega_1) > P(\omega_2) \\ \omega_2, & otherwise \end{cases}$$

- Probability of error

$$min[1 - P(\omega_1), 1 - P(\omega_2)]$$

2.2 Class-conditional probability

- Feature: the "lightness" of fish scales, x
- x , a continuous variable, whose distribution depends on the state of nature ω is expressed as p(x|ω)
- p(x|ω): class-conditional probability density function

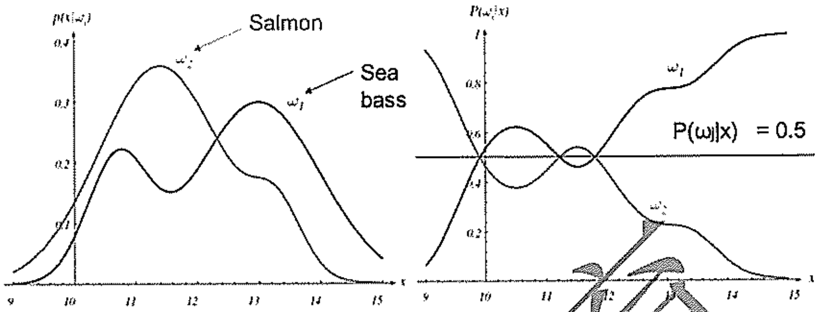


2.3 Bayes formula

- Suppose we have known P(ωi) and p(x|ωi) (i = 1, 2)
- Now, a measure of "lightness", x, become available (we know the p(x))
- Question: what is the category of this fish?
- $P(\omega_j|x) = \frac{p(x|\omega_j)\cdot P(\omega_j)}{p(x)} \Rightarrow posterior = \frac{likelihood \times prior}{evidence}$

2.4 Example

- Let prior probability be $P(\omega_1) = \frac{2}{3}$ and $P(\omega_2) = \frac{1}{3}$



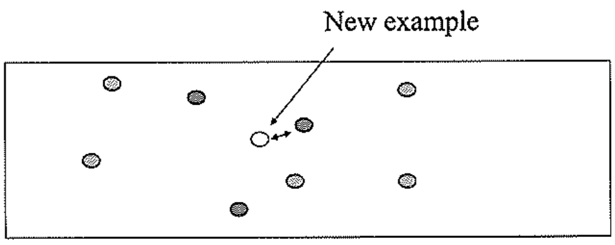
Likelihood p(x|ωj) Posterior P (ωj |x)

- Make a decision based on posterior probabilities, $P(\omega_j|x)$
$$\omega = \begin{cases} \omega_1, & if\ P(\omega_1|x) > P(\omega_2|x) \\ \omega_2, & otherwise \end{cases}$$
- The probability of error
$$P(error|x) = \begin{cases} 1 - P(\omega_1|x), & if\ we\ decide\ \omega = \omega_1 \\ 1 - P(\omega_2|x), & if\ we\ decide\ \omega = \omega_2 \end{cases}$$
$$P(error|x) = min[1 - P(\omega_1|x), 1 - P(\omega_2|x)]$$

3 Nearest Neighbor Classifiers (Nearest neighbor and KNN)

3.1 Nearest Neighbor

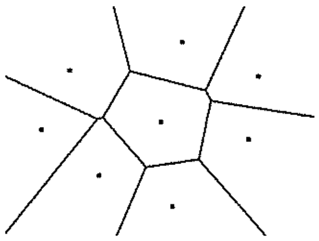
- Given a new example x to be classified (a testing sample), search for the training example (x_i, y_i) whose x_i is most similar (or closest in distance) to x, and predict y_i.



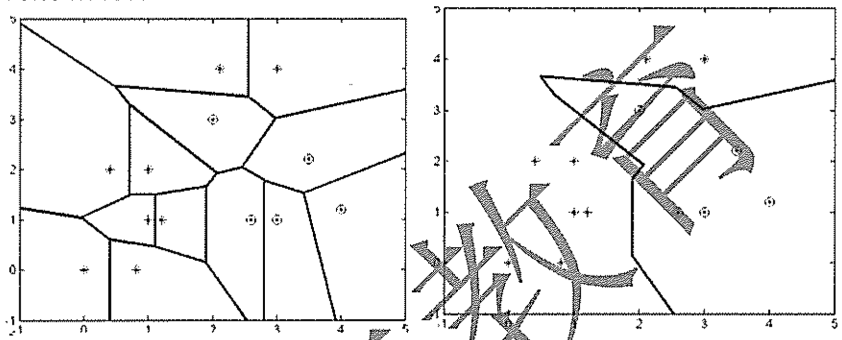
- Measure the similarity or distance - Euclidean distance
$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}, x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}, D(x, x_i) = \|(x - x_i)\| = \sqrt{\sum_{j=1}^{m} (x_j - x_{ij})^2}$$

3.2 The Voronoi Diagram

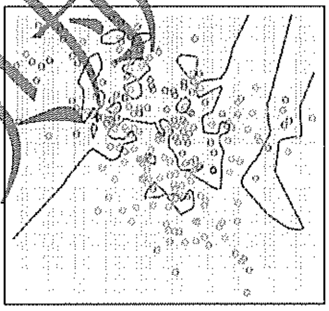
- Describes the areas that are nearest to any given point
- These areas can be viewed as zones of control - each zone is controlled by one of the points



- Decision Boundaries: subset of the Voronoi diagram
- Decision boundary are formed by only retaining these line segments separating different classes

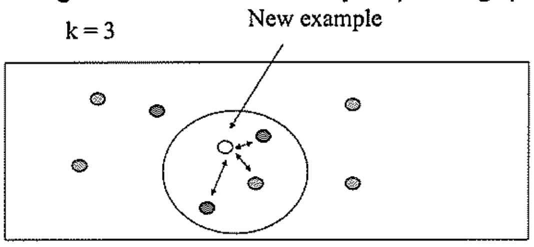


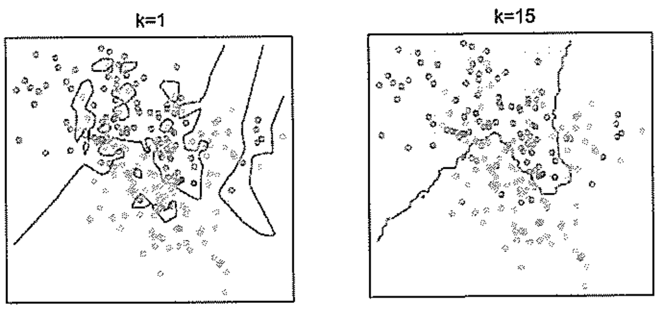
- With large number of examples and noise in the labels, the decision boundary can become nasty - if the nearest neighbour happens to be a noisy point (the islands in figure), the prediction will be incorrect.



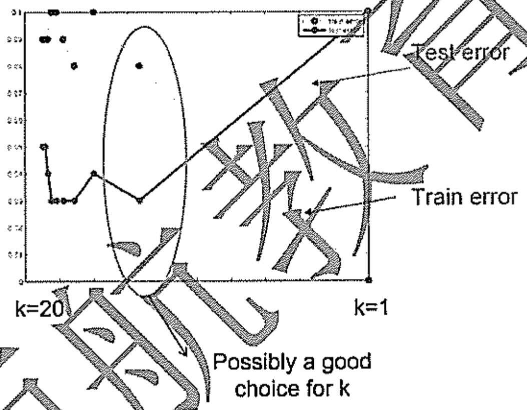
3.3 K-Nearest Neighbour (KNN)

- Find the k-nearest neighbours and have a majority voting. (K should be odd)



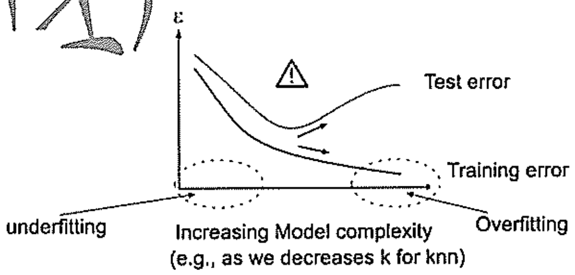


- Different k values give different results
- Why larger k produces smoother boundaries?
  - The impact of class label cancelled out by one another
- When k is too large, what will happen? if k=8, always green - overly simplified decision boundary
- How to choose k?
  - by training error? No, k=1 always has training error = 0, because for any training sample, its nearest neighbour is always itself
  - by testing error? Yes



3.4 Model selection

- if we use training error to select models, we will always choose more complex ones


ε
⚠
Test error
Training error
underfitting
Increasing Model complexity
(e.g., as we decreases k for knn)
Overfitting

3.4.1 Overfitting

- Interpretation
  - Fitting to the particularities of the data
  - Fitting too many parameters with too few data points
- Overfitting can be worsened with
  - Too many parameters (or over-complex model)
  - Too few training examples

3.4.2 Under-fitting

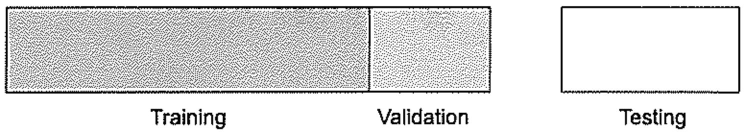
- When the model is not complex enough to capture the variability in the data

3.4.3 The goal of model selection

- to find a middle point to avoid both over-fitting and under-fitting

3.4.4 Validation Set

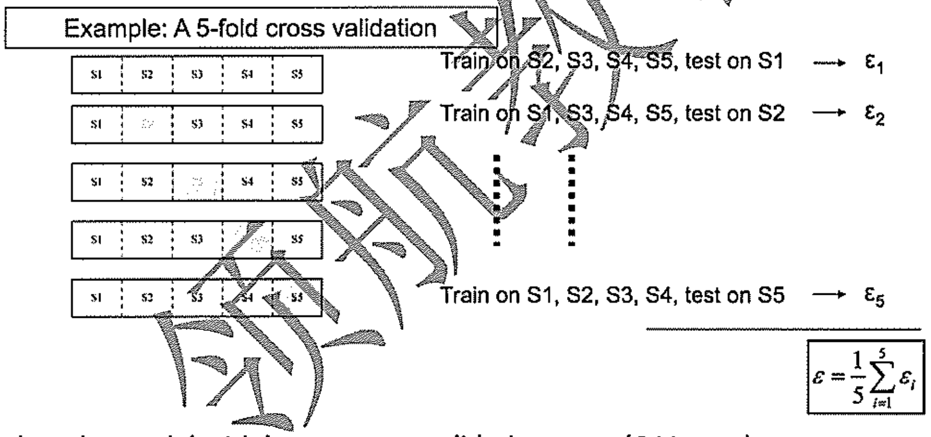
- Keep part of the labelled data (training set) apart as the validation data
- Evaluate different k values based on the prediction accuracy on the validation data
- Choose k that minimize validation error



- Validation set size
  - The larger the validation set, the more reliable model selection choices are
  - If the total labelled set is small, might not be able to get a big enough validation set - due to unreliable model selection

3.4.5 K-fold cross validation

- Randomly split the training set into K equal-sized subsets
- Perform learning/testing K times, each time reserve one subset for validation set, train on the rest


Example: A 5-fold cross validation
S1 S2 S3 S4 S5
Train on S2, S3, S4, S5, test on S1 → ε_1
Train on S1, S3, S4, S5, test on S2 → ε_2
Train on S1, S2, S3, S4, test on S5 → ε_5
$\varepsilon = \frac{1}{5}\sum_{i=1}^{5}\varepsilon_i$

- Select the model with lowest cross-validation error (C-V error)

3.5 Practical issues with KNN

- Why normalise data?
  - If not normalised, features with larger ranges will have higher impact on the distance
- Curse of dimensionality
  - In high dimensional space, data becomes so sparse that the nearest neighbour is still very far, not informative at all
  - Often need to be used with dimension reduction
- Computationally expensive for large data

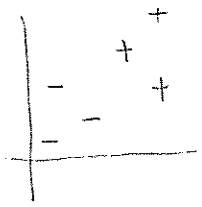
Final words on KNN

- KNN is what we call *lazy learning (vs. eager learning)*
  - Lazy: learning only occur when you see the test example
  - Eager: learn a model before you see the test example, training examples can be thrown away after learning
- Advantage:
  - Conceptually simple, easy to understand and explain
  - Very flexible decision boundaries
  - Not much learning at all!
- Disadvantage
  - It can be hard to find a good distance measure
  - Irrelevant features and noise can be very detrimental
  - Typically can not handle more than a few dozen attributes unless a good distance measure can be learned
  - Computational cost: requires a lot computation and memory

ISML SVM

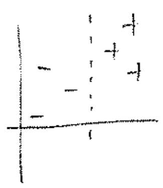
● The goal of SVM

A set of train samples with two classes, "+" and "−" (e.g. in 2D dimention)

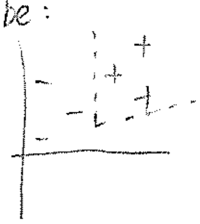


There are many ways to seperate them using a linear classifier

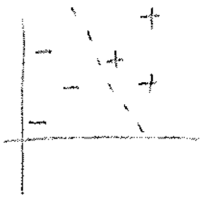
The classifiers could be:



or



or



All these classifier makes training accurate to be 100%

Which one will perform best for unknown testing samples? $\boxed{\text{Goal}}$

● Notations

▲ samples: For one sample $X_i = \begin{bmatrix} X_{i1} \\ X_{i2} \\ \vdots \\ X_{im} \end{bmatrix}$ $y_i = 1$ or -1

(a sample has m attributes, and a label)

For all samples: $X = [X_1, X_2, \dots, X_n]$ $Y = \underbrace{[1, -1, 1, 1, \dots -1]}_{n \times (1 \text{ or } -1)}$

▲ line in m-D space

$W^T X + b = 0$, where W is a metrix of weights. b is the intercept

e.g. in a 2-D space. a line could be presented as:

$W^T x + b = 0 \Leftrightarrow [w_1, w_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b = w_1 x_1 + w_2 x_2 + b = 0$

where $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ are weights. $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ is a point on this line

$\Rightarrow$ a linear classifier: $\boxed{W^T x + b = 0}$

for x on the classifier $w^T x + b = 0$

for x above the classifier $w^T x + b > 0$ ("+" samples)

x below $w^T x + b < 0$ ("−" samples)

$\boxed{\text{Goal}}$ A better classifier will make $\underline{\underline{\text{all}}}$ samples as far as possible

that is, to make

$$\begin{cases} w^T x_+ + b & \text{as large as possible, for all "+" samples} \\ w^T x_- + b & \text{as small as possible. for all "−" samples} \end{cases}$$

▲ Margin (Hard Margin)

we can set a constraint to the classifier

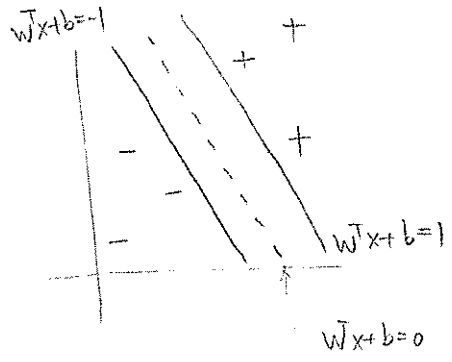
$\begin{cases} \text{for "+" samples, } w^Tx_+ + b \text{ should be at least large than 1} \\ \text{for "-" samples, } w^Tx_- + b \text{ should be at least smaller than } -1 \end{cases}$

For mathematically convienient, we can introduce a new variable

$y_i = \begin{cases} 1, & \text{for "+"} \\ -1, & \text{for "-"} \end{cases}$ (y_i is actually the label of sample)

which will make $\begin{cases} w^Tx_+ + b \geq 1, \text{ for "+"} \\ w^Tx_- + b \leq -1, \text{ for "-"} \end{cases}$ became $\Rightarrow y_i(w^Tx+b) \geq 1$

$\Rightarrow \boxed{y_i(w^Tx+b) - 1 \geq 0}$



as shown on left

$w^Tx+b-1=0$ and $w^Tx+b+1=0$

are the two margins

☆ To make it clear:

- During training, we use training data to get a "best" classifier

  The classifier should have $y_i(w^Tx+b) - 1 \geq 0$ for all training samples

- During testing, we use the classifier and x to predict the label

  All samples above classifier ($w^Tx+b \geq 0$) will be predicted as "+"

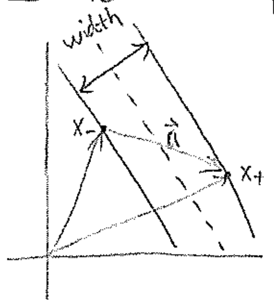
  below ($w^Tx+b \leq 0$) "-"

※ ◎ · Why we need a max margin?

If the classifier is very close to a training sample, a new sample (testing sample) close to this sample is likely to be on the other side of classifier and identified in a wrong class.

$\boxed{\text{Goal}}$ Max the width of margin

▲ The width of margin



if a "+" sample, x_+ and a "-" Sample, x_- on the margins

the vector $\vec{a} = \vec{x}_+ - \vec{x}_-$

then, $\vec{a}$'s projection on the direction $\vec{w}$ is the width

$\vec{w}$ is the vector virtical to $w^Tx+b=0$

$$\text{width} = \vec{a}\cdot\frac{\vec{w}}{\|\vec{w}\|} = \frac{(\vec{x}_+ - \vec{x}_-)\vec{w}}{\|\vec{w}\|} = \frac{w^Tx_+ - w^Tx_-}{\|w\|} = \frac{(1-b)-(-1-b)}{\|w\|} = \frac{2}{\|w\|}$$

as x_+ and x_- on the margins $\begin{cases} w^Tx_+ + b = 1 \\ w^Tx_- + b = -1 \end{cases}$

$$\Rightarrow \boxed{\text{width} = \frac{2}{\|\vec{w}\|}}$$

Goal Maxium the width $= \max\left(\frac{2}{\|w\|}\right)$

$\Leftrightarrow \min(\|w\|)$

$\Leftrightarrow \boxed{\min\left(\frac{1}{2}\|w\|^2\right)}$ ← for mathmatically convienient

● Primal Problem (Hard Margin)

$$\boxed{\min \frac{1}{2}\|w\|^2,\quad \text{s.t.}\quad y_i(w^Tx+b)-1\geq 0}$$ Primal

▲ Solve this optimize problem we will get the w and b then we have the classifier

● Dual Problem (Hard Margin)

Apply Lagrange Multiplier on Primal

$$\max\ L(\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i\left[y_i(w^Tx_i+b)-1\right]$$

$$\text{s.t.}\quad \alpha_i \geq 0,\quad \sum\alpha_i y_i = 0,\ \forall i$$

$\frac{\partial L}{\partial w} = w - \sum\alpha_i x_i y_i = 0 \Rightarrow w = \sum\alpha_i x_i y_i$ (we will use this to get w)

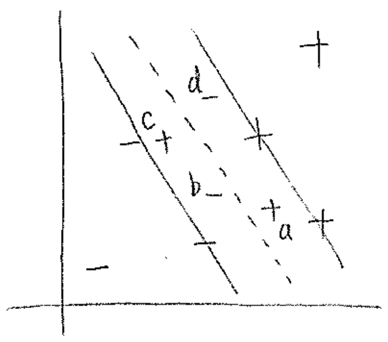
$\frac{\partial L}{\partial b} = -\sum\alpha_i y_i = 0 \Rightarrow \sum\alpha_i y_i = 0$ (the constraint)

Substituting $\begin{cases} w = \sum\alpha_i x_i y_i \\ \sum\alpha_i y_i = 0 \end{cases}$, $L(\alpha) = \frac{1}{2}\sum\alpha_i x_i y_i\cdot\sum\alpha_j x_j y_j - \sum\alpha_i x_i y_i\cdot\sum\alpha_j x_j y_j - \sum\alpha_i y_i b + \sum\alpha_i$

$$\Rightarrow \boxed{\begin{aligned}&\max\ L(\alpha) = -\frac{1}{2}\sum\sum\alpha_i\alpha_j x_i x_j y_i y_j + \sum\alpha_i \\ &\quad\text{s.t.}\quad \alpha_i\geq 0,\ \sum\alpha_i y_i = 0\end{aligned}}$$ Dual

Soft Margin (for non-seperable cases)

Some cases cannot be seperated using a linear classifier, such as:



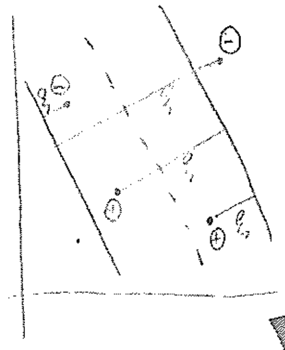
- There are some samples on the wrong side of margin
- Or even worse, there could be some samples on the wrong side of classifier

e.g. point a is a "+" sample. below margin } wrong side
b is a "-" sample above margin. } .of margin

point "c" is a "+" sample. identified as "-" } wrong side of
"d" is a "-" sample. identified as "+" } classifier
↑ this will cause training error

slack penalty ξ

We can still optimise the primal problem with a new variable, slack penalty ξ



ξ_i is the distance to margin for wrong sided points

$\xi_i = 0$. for points corred sided

$\Rightarrow \xi_i \geq 0$ for all points

$\Rightarrow \boxed{\text{soft margin} = 1 - \xi_i}$ (now the margin will be different for each sample)

Primal (Soft Margin)

$$\boxed{\min \ \frac{1}{2}\|w\|^2 + C\cdot\sum \xi_i \ , \ \text{s.t} \ \ y_i(w^T x_i + b) \geq 1 - \xi_i \ . \ \xi_i \geq 0}$$

C: to control the relative weight

$\boxed{\text{Goal}}$ min Primal $\Rightarrow$ min width & $\underbrace{\sum \xi_i}_{\text{penalty}}$ $\Rightarrow$ max number of points with margin at least 1

Also known as l_1 norm soft margin SVM (as $\sum \xi_i$ is l_1 norm)

Dual (Soft Margin)

Apply Lagrange Multiplier

$$\max L(w, b, \xi, \alpha, r) = \underbrace{\frac{1}{2}\|w\|^2 + C\cdot\sum \xi_i}_{\text{the primal}} - \overset{\text{multiplier 1}}{\sum \alpha_i}\underbrace{[y_i(w^T x_i + b) - 1 + \xi_i]}_{\text{constraint 1}} - \overset{\text{multiplier 2}}{\sum r_i}\underbrace{\xi_i}_{\text{constraint 2}}$$

$$= \frac{1}{2}w^T w - \sum \alpha_i x_i y_i w^T - \sum \alpha_i y_i b - \sum \alpha_i + \sum (C - \alpha_i - r_i)\xi_i$$

$$\begin{cases} \frac{\partial L}{\partial w} = w - \sum \alpha_i x_i y_i = 0 \quad \Rightarrow \quad w = \sum \alpha_i x_i y_i \\ \frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \quad \Rightarrow \quad \sum \alpha_i y_i = 0 \\ \frac{\partial L}{\partial \xi} = C - \alpha_i - r_i = 0 \quad \Rightarrow \quad \alpha_i = C - r_i \end{cases}$$

$$\left.\begin{array}{l} \alpha_i = C - r_i \\ r_i \geq 0 \\ \alpha_i \geq 0 \end{array}\right\} \Rightarrow 0 \leq \alpha_i \leq C$$

Substituting:

$$\boxed{\begin{array}{l} \max \ L(\alpha) = \sum \alpha_i - \frac{1}{2}\sum\sum \alpha_i \alpha_j x_i x_j y_i y_j \\ \quad \text{s.t.} \ 0 \leq \alpha_i \leq C, \ \sum \alpha_i y_i = 0 \end{array}}$$ Dual

by solving the Dual Problem, we get α_i

using $w = \sum \alpha_i x_i y_i$, we get w

for point $0 < \alpha_i < C$, $y_i(w^T x + b) - 1 = 0$. we get b

▲ Two type of support vector in soft margin

① $0 < \alpha_i < C$. support vectors that on the margin

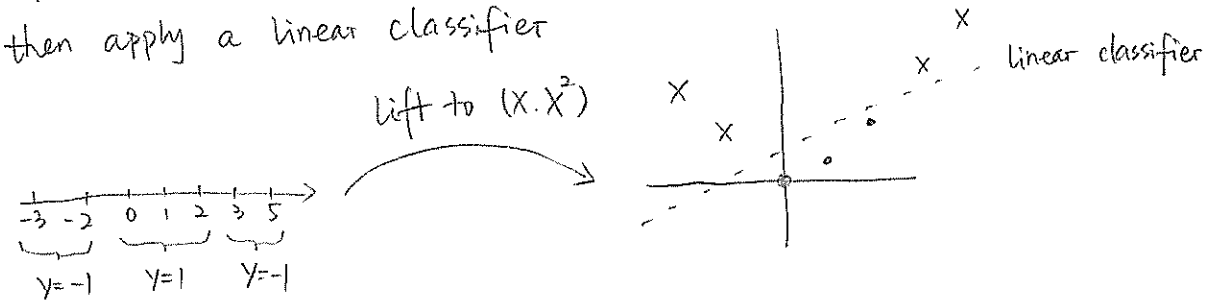
② $\alpha_i = C$ support vectors that on the wrong side

$$\xi_i > 0. \begin{cases} 1 > \xi_i > 0. & \text{correctly classified} \\ \xi_i = 1 & \text{on decision boundary (the classifier)} \\ \xi_i > 1 & \text{mis-classified} \end{cases}$$

Kernels (for non-linear case)

- Method: lifting data into a higher dimentional space then apply a linear classifier

e.g.



- Curse of dimensionality
  - poor generalization to test data
  - computationally expensive

- SVM avoids "curse of dimensionality" by:
  - enforcing largest margin → permits ~~good~~ generalization
    (generalization in SVM is a function of the margin independent of the dimensionality)
  - Computation is performed only through kernel functions
    (no need to perform explicitly)

    "Kernel trick" → compute $K(X_i, X_j)$, instead of $\varphi(X_i)^T \varphi(X_j)$

- Apply:

Dual: $\max L(\alpha) = \sum \alpha_i - \frac{1}{2} \sum\sum \alpha_i \alpha_j x_i x_j y_i y_j$, depends on sample x only through the dot product $x_i x_j$

we can lift x to high dimension using $\varphi(x)$

Dual change to $\Rightarrow \max L(\alpha) = \sum \alpha_i - \frac{1}{2} \sum\sum \alpha_i \alpha_j y_i y_j \varphi(x_i) \varphi(x_j)$

Kernel function: $K(x_i, x_j) = \varphi(x_i) \varphi(x_j)$

L2-norm soft SVM

primal $\min \frac{1}{2}\|w\|^2 + \frac{c}{2}\sum \xi_i^2 \quad s.t. \; y_i(w^T x_i + b) \geq 1 - \xi_i$

$$\max \; L(w, b, \xi, \alpha) = \frac{1}{2}\|w\|^2 + \frac{c}{2}\sum \xi_i^2 - \sum \alpha_i [y_i(w x_i + b) - 1 + \xi_i]$$

$$\frac{\partial L}{\partial w} = w - \sum \alpha_i y_i x_i = 0 \Rightarrow w = \sum \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = \sum \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi} = c\xi_i - \alpha_i \Rightarrow \xi_i = \frac{\alpha_i}{c}$$

substituting

$$\max \; L(\alpha) = \frac{1}{2}\left(\sum \alpha_i y_i x_i\right)^2 + \frac{c}{2}\sum \frac{\alpha_i^2}{c^2} - \sum \alpha_i y_i \left(\sum \alpha_j y_j x_j\right) x_i$$

$$+ \sum \alpha_i - \frac{1}{c}\sum \alpha_i$$

$$\max \; L(\alpha) = -\frac{1}{2}\sum\sum \alpha_i \alpha_j y_i y_j x_i x_j + \sum \alpha_i - \frac{1}{2c}\sum \alpha_i^2 \qquad \text{(dual)}$$

$$s.t. \; \sum \alpha_i y_i = 0, \; \alpha_i \geq 0$$

ISML Support Vector Machine (SVM)

Key Points - 1

- Primal (Hard Margin)

$$minimize\ \frac{1}{2}\|w\|^2$$

$$s.t.\ y_i(w^T X_i + b) - 1 \geq 0, \forall i$$

- Dual (Hard Margin)

$$maximize\ L(\alpha) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j X_i X_j y_i y_j + \sum_{i=1}^{n} \alpha_i$$

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

- Primal (Soft Margin)

$$minimize\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$s.t. y_i(w^T X_i + b) - 1 + \xi_i \geq 0\ and\ \xi_i \geq 0, \forall i$$

- Dual (Soft Margin)

$$maximize\ L(\alpha) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j X_i X_j y_i y_j + \sum_{i=1}^{n} \alpha_i$$

$$s.t.\ 0 \leq \alpha_i \leq C, and \sum_{i=1}^{n} \alpha_i y_i = 0, \forall i$$

- How to derive the Dual from Primal
- How to apply Lagrange Multiplier

ISML Support Vector Machine (SVM)

Key Points - 2

- Margin
  - Width of hard margin: $\frac{2}{\|w\|}$
  - Hard margin: $y_i(w^T X_i + b) - 1 = 0$
  - Soft margin: $y_i(w^T X_i + b) - 1 + \xi_i = 0$
- Why max the width of margin?
  - If the classifier is very close to a sample, a new sample (within same class) close to this sample is likely to be on the other side of classifier and identified as in the wrong class, which lead to a poor generalization.
  - After the introducing of margin, the width of margin is the smallest distance, or the lower bound, to hyperplane from all samples. To maximize the distances is actually to maximize the width of margin.
- Why Dual?
  - allowing us to derive an efficient algorithm for solving the primal optimization problem
  - allowing us to use kernels to get optimal margin classifiers to work efficiently in very high dimensional spaces
- Why we need soft-margin SVMs
  - we can't guarantee that the data can always be separated by a linear classifier
  - the hard margin SVMs might be susceptible to outliers
- Two types of support vectors for soft margin SVMs
  - support vectors with $0 < \alpha_i < C$, on the margin, $y_i(w^T x + b) - 1 = 0$
  - support vectors with $\alpha_i = C$, on the wrong side of margin, $y_i(w^T x + b) - 1 < 0$
- Why Kernel?
  - To solve a non-linear classification problem with linear classifier
- How to solve non-linear case?
  - lifting data into a higher dimensional space
  - then apply a linear classifier
- Curse of dimensionality
  - poor generalization to test data
  - computationally expensive
- How SVM avoids "curse of dimensionality"?
  - enforcing largest margin permits good generalization (generalization in SVM is a function of the margin, which is independent of the dimensionality)
  - computation is performed only through kernel functions ("kernel trick")