Probabilistic Models and Machine Learning

No.5

Discriminative Models

Hui Jiang
Department of Electrical Engineering and Computer Science
Lassonde School of Engineering
York University, Toronto, Canada

Outline

- Supervised Machine Learning:
  - Generative vs. Discriminative models
- Statistical Learning Theory
- Linear Models:
  - Perceptron
  - Linear Regression
  - Minimum Classfication Error
- Support Vector Machines
- Rigde Regression and LASSO
- Compressed Sensing
- Neural Networks

(Supervised) Machine Learning

- (optional) feature extraction
- LIST_A: choose a model from
- LIST_B: choose a learning criterion from
  - Leads to an objective function of model parameters
- LIST_C: choose an optimization algorithm from
- (optional) theoretical guarantees:
  - whether learning converges?
  - how learning generalizes?

(Supervised) Machine Learning

- **LIST_A**: cho
- LIST_B: c
  - Leads to
- LIST_C: cho
- (optional) Th
  - Whether l
  - How learn

- linear model
- quadratic model (polynomial models)
- logistic sigmoid
- probit model
- nonlinear kernels
- neural networks
  - FFNN, CNN, transformer
  - RNN, LSTM

discriminative models

- naïve Bayes classifier
- multinomial model
- Gaussian model
- Markov chain model
- mixture model
- hidden Markov model
- latent Dirichlet allocation
- conditional random fields
- graphical models
  - Bayes nets, Markov random fields
- Gaussian process

generative models

(Supervised) Machine Learning

- LIST_A: choose a model from
- **LIST_B**: cho
  - Leads t
- LIST_C: choo
- (optional) The
  - Whether l
  - How learning generalizes?

- least square error
- minimum classification error
- minimum cross-entropy
- maximum mutual information
- maximum margin

} discriminative models

- maximum likelihood
- maximum conditional likelihood
- maximum *a posterior*
- maximum marginal likelihood

} generative models

(Supervised) Machine Learning

- LIST_A: choose a model from
- LIST_B: choose a learning criterion from
  - Leads to an objective function of model parameters
- LIST_C:
  - gradient descent
    - stochastic gradient descent (SGD)
  - Newton's method
  - quasi-Newton method
    - quickprop, R-prop
    - BFGS, L-BFGS
  - expectation-maximization (EM)
  - sequential line search
  - alternating direction method of multipliers (ADMM)
- (optional) The
  - Whether le
  - How learnir

(Supervised) Machine Learning

- Not all combinations make senses ...

- Some typical examples:
  - Linear regression: linear model + least square error
  - Logistic regression: logistic sigmoid + maximum likelihood
  - Linear SVM: linear model + maximum margin
  - Nonlinear SVM: nonlinear kernels + maximum margin
  - Deep learning: neural networks + cross-entropy + SGD

Pattern classification based on Discriminant models

- We can build an classifier based on some discriminant functions to model class boundary info directly.
- Classifiers are based on discriminant functions:
  - For N classes, we define a set of discriminant functions $h_i(X)$ (i=1,2,…,N), one for each class.
  - For an unknown pattern with feature vector Y, the classifier makes the decision as

$$\omega_Y = \arg\max_i h_i(Y)$$

  - Each discriminant function $h_i(X)$ has a pre-defined function form and a set of unknown parameters θ_i, rewrite it as $h_i(X ; \theta_i)$.
  - Parameters θ_i (i=1,2,…,N) need to be estimated from some training data.

Statistical learning theory (1)

- training samples: $\mathbf{X}_N = \Big\{(\mathbf{x}_i, y_i) \mid i = 1, \cdots, N\Big\}$
- random variables $\mathbf{x}$ and y: joint distribution $p(\mathbf{x}, y)$
- input space $\mathbb{X}$: $\mathbf{x} \in \mathbb{X}$
- output space $\mathbb{Y}$: $y \in \mathbb{Y}$
  - $\mathbb{Y}$ is discrete or categorical for classification
  - $\mathbb{Y}$ is continuous for regression, e.g. R.
- machine learning tries to learn a model: $y = h(\mathbf{x})$.
- hypothesis space $\mathbb{H}$: $h(\cdot)$ is learned from, $h(\cdot) \in \mathbb{H}$
- loss function $l(y, y')$:
  - zero-one loss, squared error, cross-entropy, ...

Statistical learning theory (2)

- empirical loss (a.k.a., empirical risk, in-sample error):

$$R_{\text{emp}}(h|\mathbf{X}_N) = \frac{1}{N}\sum_{i=1}^{N} l\Big(y_i, h(\mathbf{x}_i)\Big)$$

- expected loss (a.k.a., expected risk, generalization error):

$$R(h) = \mathbb{E}_{p(\mathbf{x},y)}\Big[l\big(y, h(\mathbf{x})\big)\Big] = \int\int_{\mathbf{x},y} l\big(y, h(\mathbf{x})\big)p(\mathbf{x}, y)d\mathbf{x}dy$$

- $R_{\text{emp}}(h|\mathbf{X}_N) \neq R(h)$ but $\lim_{N\to\infty} R_{\text{emp}}(h|\mathbf{X}_N) = R(h)$
- supervised machine learning:

$$h^* = \arg\min_{h\in\mathbb{H}} \; R_{\text{emp}}(h|\mathbf{X}_N)$$

Statistical learning theory (3)

- learnable or not: empirical risk minimization (ERM) leads to small generalization error, i.e., $R(h^*)$ is sufficiently small.
- learnability depends on whether the maximum gap

$$\Pr\left[\sup_{h\in\mathbb{H}}\left|R(h) - R_{\text{emp}}(h|\mathbf{X}_N)\right| > \epsilon\right]$$

  is sufficiently small for $\forall\epsilon > 0$.
- the key to learnability: $\mathbb{H}$ must be chosen properly.
- VC generalization bounds (Vapnik-Chervonenkis theory):

$$R(h) \leq R_{\text{emp}}(h|\mathbf{X}_N) + \sqrt{\frac{8d_{vc}(\ln\frac{2N}{d_{vc}} + 1) + 8\ln\frac{4}{\delta}}{N}}$$

  where d_{vc} is called called VC-dimension, only depending on $\mathbb{H}$.

Generalization bound (1)

Given $\{x_1, x_2, \cdots, x_N\}$ are N i.i.d. samples of a random variable $\mathbf{x}$ distributed by $p(\mathbf{x})$, and $a \leq x_i \leq b$ for every i, $\forall \epsilon > 0$, we have

- the weak law of large numbers:

$$\lim_{N \to \infty} \Pr\left[\left|\mathbb{E}_{p(\mathbf{x})}[\mathbf{x}] - \frac{1}{N}\sum_{i=1}^{m} x_i\right| > \epsilon\right] = 0$$

- Heoffding's ineqality (one of concentration inequalities):

$$\Pr\left[\left|\mathbb{E}_{p(\mathbf{x})}[\mathbf{x}] - \frac{1}{N}\sum_{i=1}^{m} x_i\right| > \epsilon\right] \leq 2e^{-\frac{2N\epsilon^2}{(b-a)^2}}$$

Generalization bound (2)

- for a single model $h(\cdot)$:

$$\Pr\left[\left|R(h) - R_{\text{emp}}(h|\mathbf{X}_N)\right| > \epsilon\right] \leq 2e^{-2N\epsilon^2}$$

- extend for a finite hypothesis space $\mathbb{H}$:

$$\Pr\left[\sup_{h\in\mathbb{H}}\left|R(h) - R_{\text{emp}}(h|\mathbf{X}_N)\right| > \epsilon\right] \leq 2|\mathbb{H}|e^{-2N\epsilon^2}$$

- the first bound:

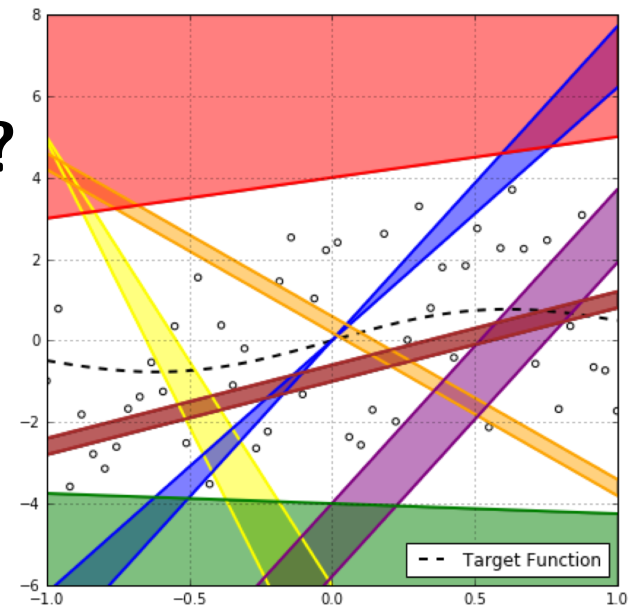
$$R(h) \leq R_{\text{emp}}(h|\mathbf{X}_N) + \sqrt{\frac{\ln|\mathbb{H}| + \ln\frac{2}{\delta}}{2N}}$$

which holds in probability $1 - \delta$.

VC-Dimension

- How about infinite number of h() in $\mathbb{H}$?

  Not all h are different ...



- VC-dimension:
  - Max # of points the hypothesis space $\mathbb{H}$ can shatter
  - Roughly represents model capability
  - VC-dimension of linear classifier: D+1
  - VC-dimension of Neural network $\leq$ num of weights

Examples of generalization bounds

$$R(h) \leq R_{\text{emp}}(h|\mathbf{X}_N) + \sqrt{\frac{8d_{vc}(\ln\frac{2N}{d_{vc}}+1)+8\ln\frac{4}{\delta}}{N}}$$

- Example I: use N=1000 data samples (feature dimension 100) to learn a linear classifier (d_{vc} = 101), training error rate is 1%, set δ=0.01 (99% chance correct)

Pattern classification based on Discriminant Functions

- Some common forms for discriminant functions:
  - Linear discriminant function:

$$h(\mathbf{x}) = \mathbf{w}^t \cdot \mathbf{x} + \mathbf{b}$$

  - Quadratic discrimiant function: (2nd order)
  - Polynomial discriminant function: (N-th order)
  - Neural network: (arbitrary nonlinear functions)

Pattern classification based on Linear Discriminant Functions

- Unknown parameters of discriminant functions are estimated to optimize an objective function by some gradient descent method :

  - Perceptron: a simple learning algorithm.

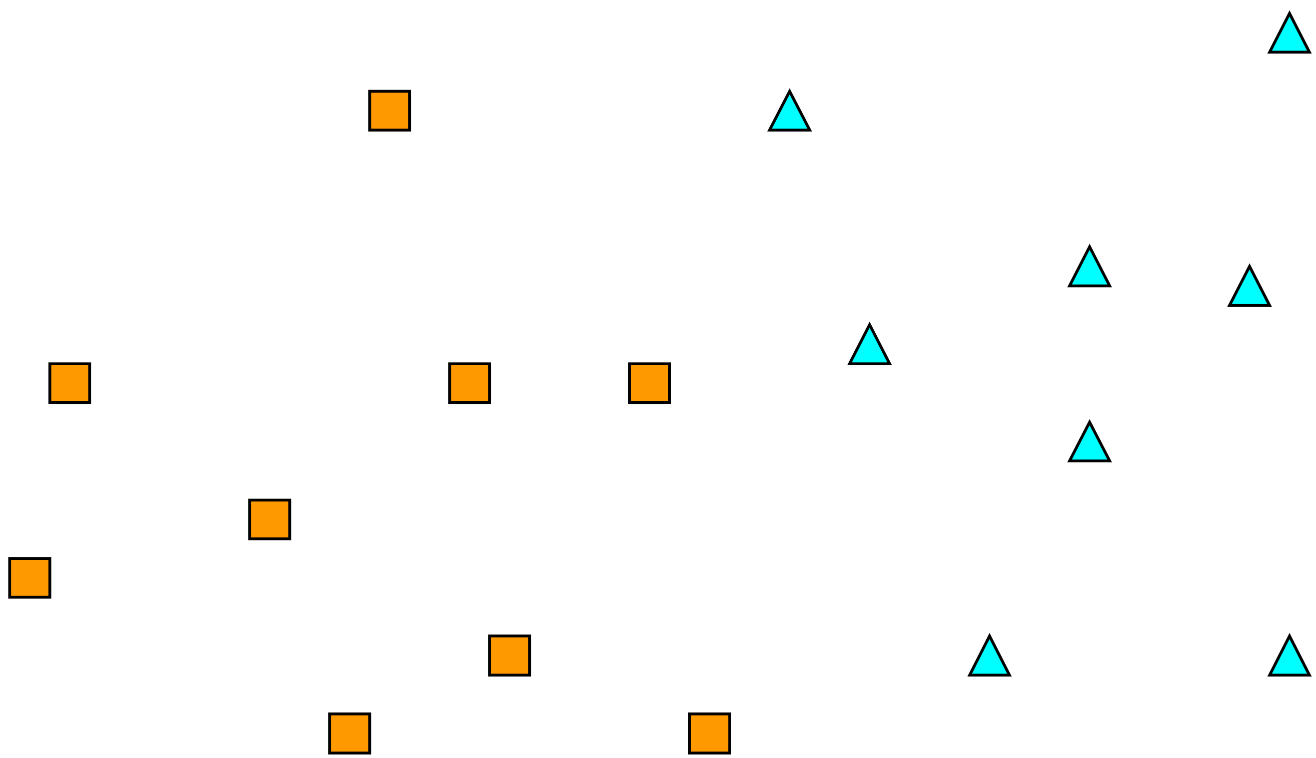
  - Linear Regression: achieving a good mapping.

  - Logistic Regression: minimizing empirical classification errors.

  - Support Vector Machine (SVM): maximizing separation margin.

Binary Classification Task

- Separating two classes using linear models



Label: +1

Label: -1

Perceptron

- Rosenblatt (1960)
- Use a linear model for 2-class problems:
$$f(\mathbf{x}|\mathbf{w}, b) = \begin{cases} +1 & \text{if } \mathbf{w}^\intercal \mathbf{x} + b > 0 \\ -1 & \text{otherwise} \end{cases}$$
- training set: $\{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^D, y_i = \pm 1, i = 1, \cdots, N\}$

Algorithm 1 Perceptron: a simple iterative learning algorithm

randomly initialize $\mathbf{w}^{(0)}$ and $b^{(0)}$, set $n = 0$
for each sample $(\mathbf{x}_i, y_i)$ **do**
  calculate the actual output $h_i = f(\mathbf{x}_i|\mathbf{w}^{(n)}, b^{(n)})$
  if upon a mistake: $h_i \neq y_i$ **then**
    $\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} + y_i \mathbf{x}_i$
    $\mathbf{b}^{(n+1)} = \mathbf{b}^{(n)} + y_i$
  end if
  $n = n + 1$
end for

Convergence of Perceptron

- **If the training data is linearly separable, then the perceptron is guaranteed to converge, and there is an uppper bound on the number of times the perceptron will adjust its weights during the training.**

Theorem 1 *Let $\mathcal{S}$ be a sequence of labeled examples consistent with a linear threshold function $\mathbf{w}^* \cdot \mathbf{x} > 0$, where $\mathbf{w}^*$ is a unit-length vector. Then the number of mistakes M on $\mathcal{S}$ made by the online Perceptron algorithm is at most $(1/\gamma)^2$, where*

$$\gamma = \min_{\mathbf{x} \in \mathcal{S}} \frac{|\mathbf{w}^* \cdot \mathbf{x}|}{||\mathbf{x}||}.$$

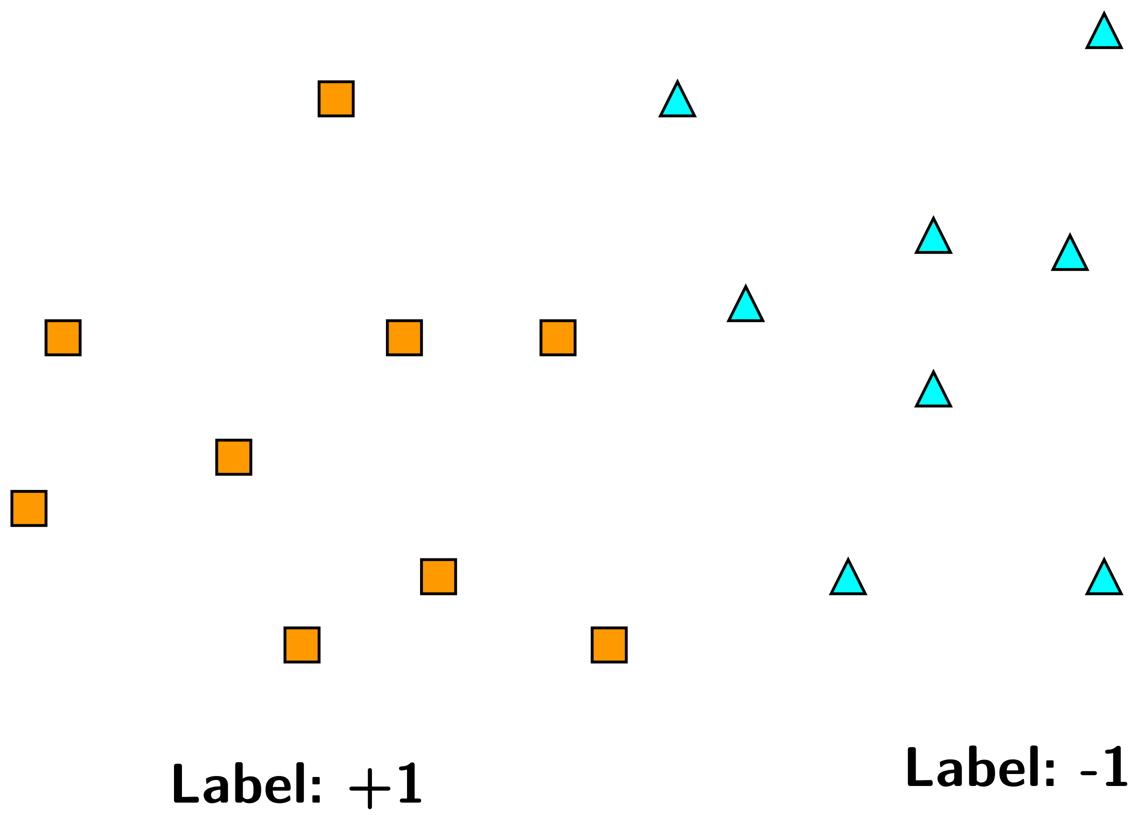
- **Proof can be found:**

[Nov62] A.B.J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata, Vol. XII*, pages 615–622, 1962.

$$M\gamma \leq \frac{\mathbf{w}^* \cdot \sum_{t \in I} y_t \mathbf{x}_t}{||\mathbf{w}^*||} \leq ||\sum_{t \in I} y_t \mathbf{x}_t|| \leq \sqrt{\sum_{t \in I} ||\mathbf{x}_t||^2} \leq \sqrt{M}$$

Linear Regression

- Find a good mapping from x to y (+1 or -1)



Linear Regression

- **Find a good mapping from X to y:**

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{bmatrix} \xrightarrow{Y = Xw^T} Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{bmatrix} = \begin{bmatrix} +1 \\ -1 \\ \vdots \\ +1 \end{bmatrix}$$

$$w^* = \arg\min_w \sum_i \left(x_i w^T - y_i \right)^2$$

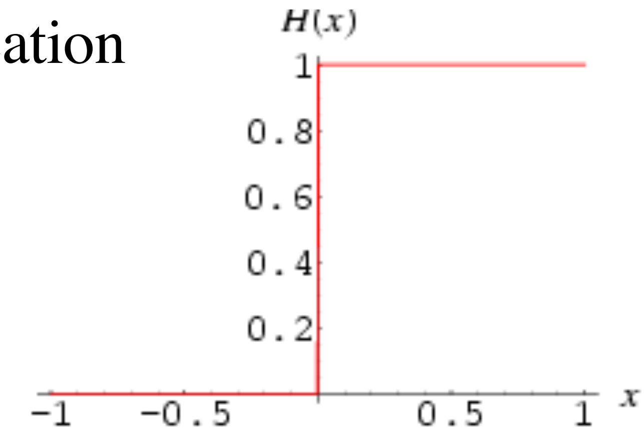
$$w^* = \left(X^T X \right)^{-1} X^T Y$$

- **Matrix inversion is expensive when x is high-dimension**
- **Linear regression does NOT work well for classification**

Minimum Classification Error (MCE)

- **Counting errors in training samples.**

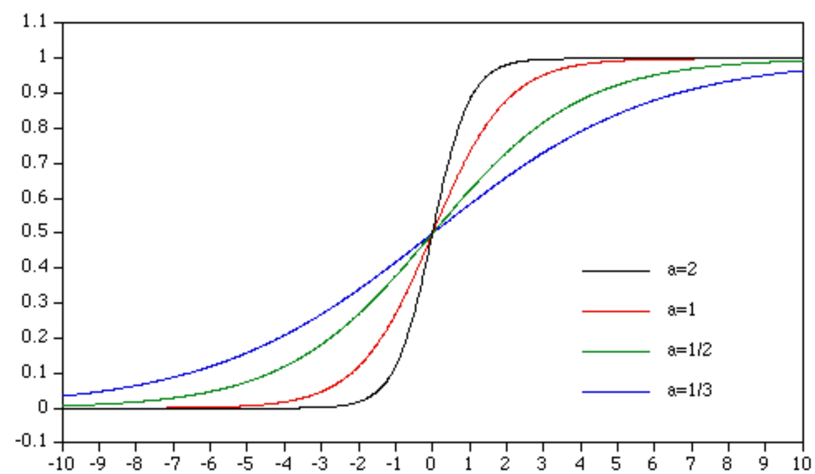
$$(\mathrm{x}_i, y_i) \Rightarrow \begin{cases} g_i = -y_i \mathrm{x}_i \mathrm{w}^T < 0 & \text{correct classification} \\ g_i = -y_i \mathrm{x}_i \mathrm{w}^T > 0 & \text{wrong classification} \end{cases}$$



$$w^* = \arg\min_w \sum_i H(g_i) = \arg\min_w \sum_i H(-y_i \mathrm{x}_i \mathrm{w}^T)$$

$$w^* = \arg\min_w \sum_i l(g_i) = \arg\min_w \sum_i l(-y_i \mathrm{x}_i \mathrm{w}^T)$$

$$l(x) = \frac{1}{1 + e^{-\sigma x}}$$ logistic sigmoid function



Minimum Classification Error (MCE)

- Optimization using gradient decent or SGD
- The objective function (the smoothed training errors):

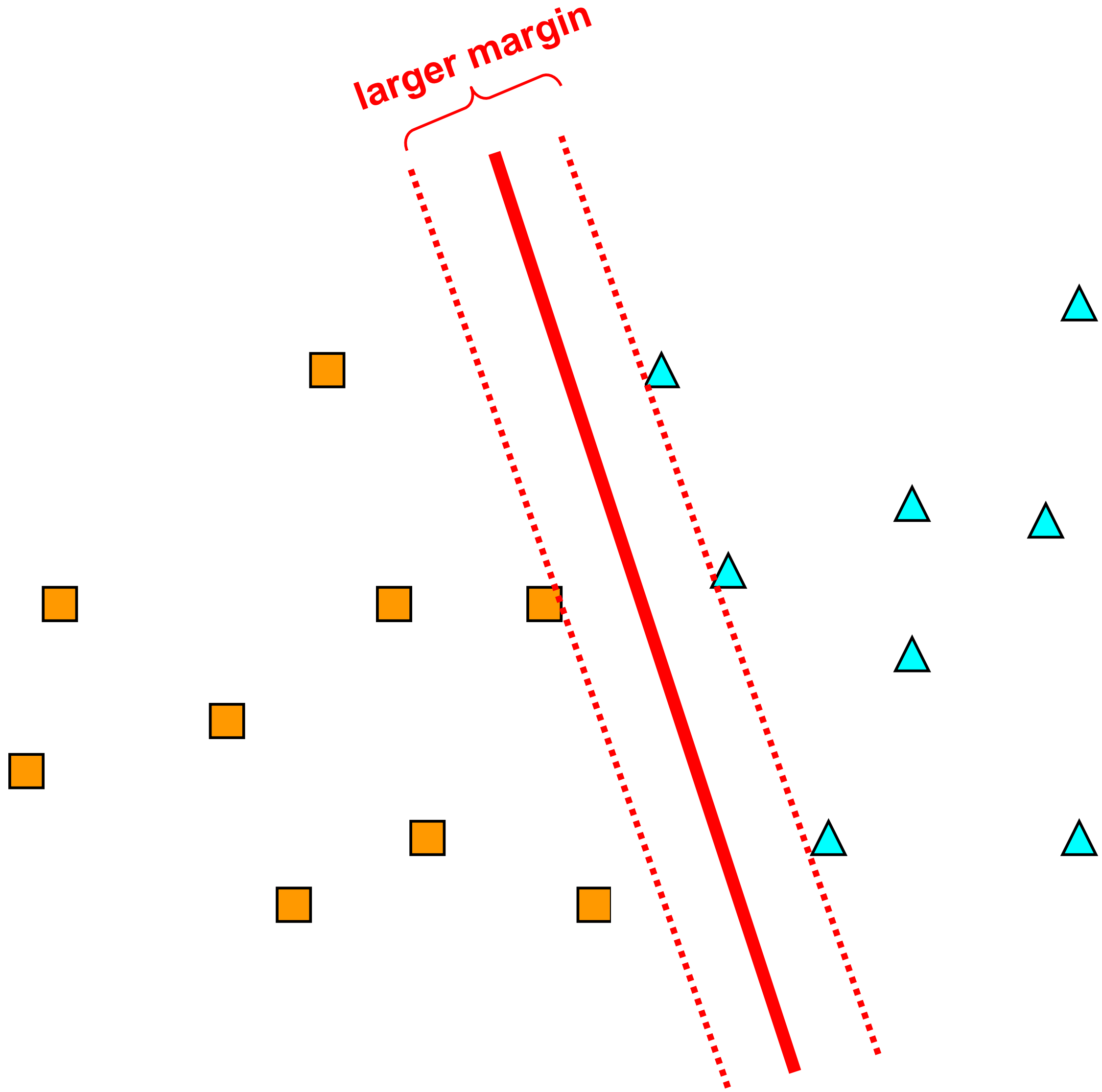
$$E(\mathbf{w}) = \sum_i l(y_i \mathbf{x}_i \mathbf{w}^T)$$

- The gradient is computed as:

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \sum_i l(y_i \mathbf{x}_i \mathbf{w}^T)\Big(1 - l(y_i \mathbf{x}_i \mathbf{w}^T)\Big) y_i \mathbf{x}_i$$

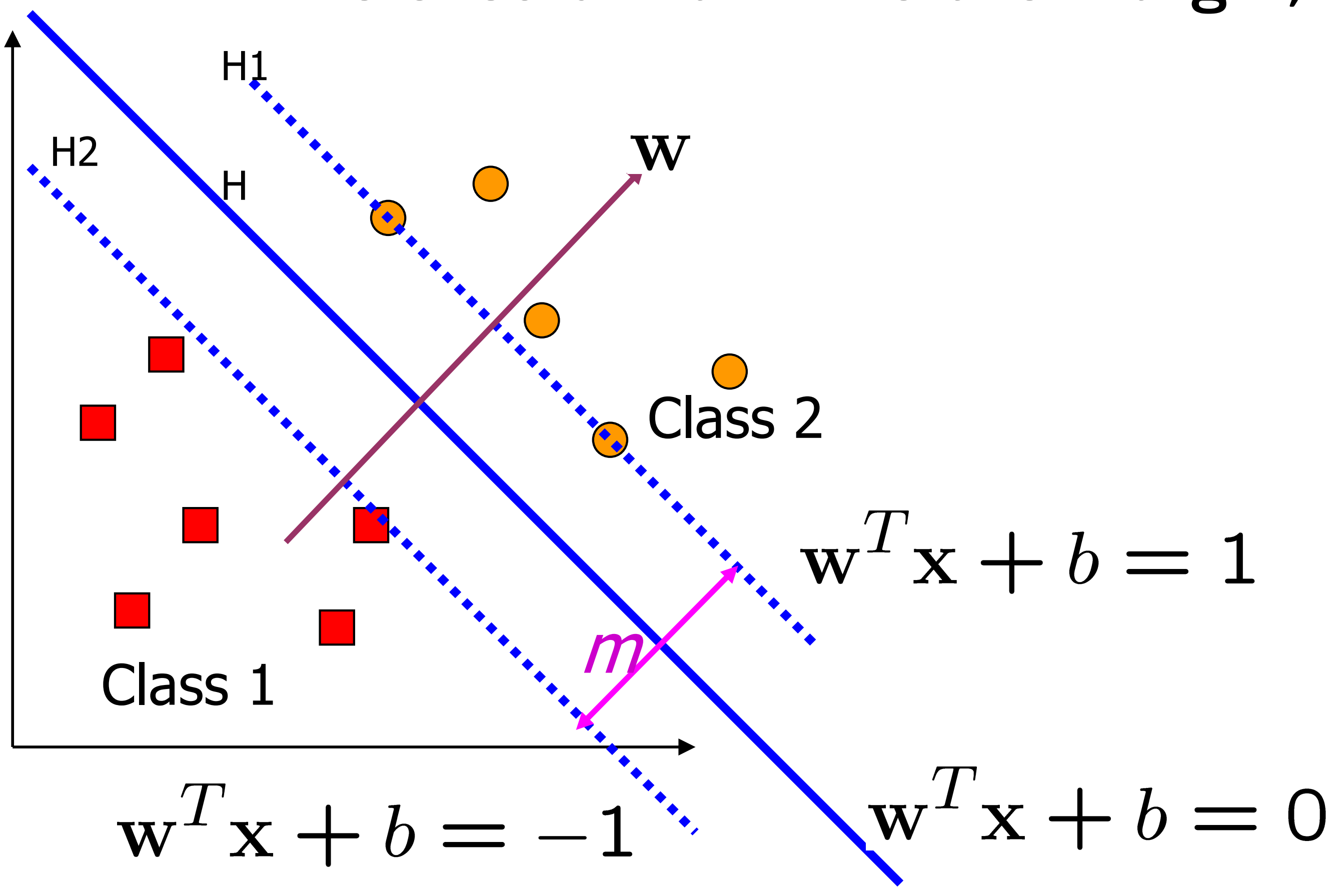
- This method is similar to logistic regression.

Large-Margin Classifier: Support Vector Machine (SVM)



Support Vector Machine (I)

- **The decision boundary H should be as far away from the data of both classes as possible**
  - **We should maximize the margin, m**



$$m = \frac{2}{||\mathbf{w}||}$$

Support Vector Machine (II)

- **The decision boundary can be found by solving the following constrained optimization problem:**

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 \qquad ||w||^2 = w^T w$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

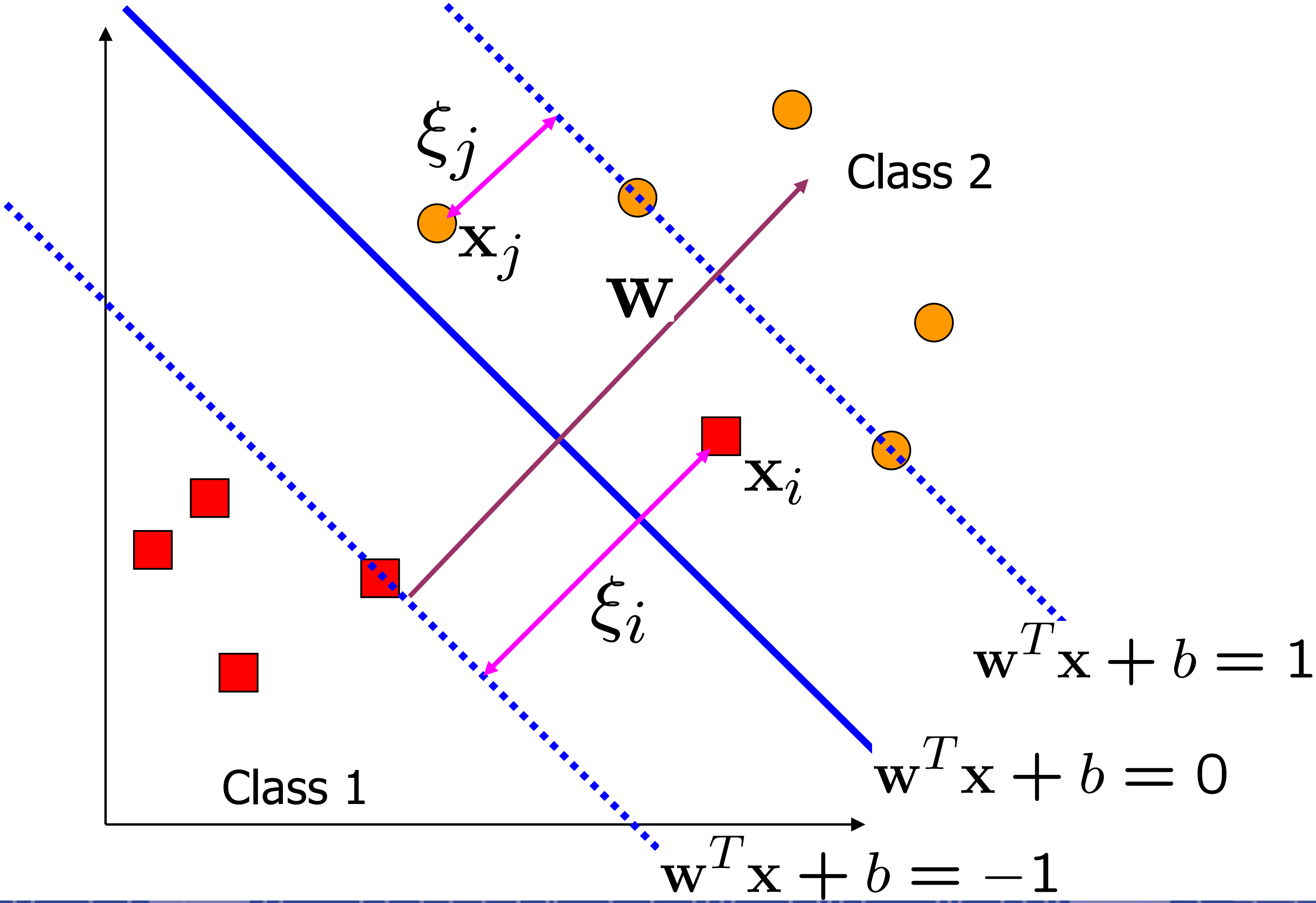
- **Convert to its dual problem:**

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Linearly Non-Separable cases

- We allow "error" x_i in classification → soft-margin SVM



Support Vector Machine (III)

- **Soft-margin SVM can be formulated as:**

$$w^* = \min_{w,\xi_i} \left[\frac{1}{2} \| w \|^2 + C \cdot \sum_i \xi_i \right]$$

subject to

$$y_i(x_i w^T + b) > 1 - \xi_i \quad \xi_i > 0 \quad (\forall i)$$

- **It can be converted to the dual form:**

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to} \quad 0 \le \alpha_i \le \boxed{C} \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

Support Vector Machine (IV)

- **Soft-margin SVM can be formulated as:**

$$w^* = \min_{w,\xi_i} \left[\frac{1}{2} \| w \|^2 + C \cdot \sum_i \xi_i \right]$$

subject to

$$y_i(x_i w^T + b) > 1 - \xi_i \quad \xi_i > 0 \quad (\forall i)$$

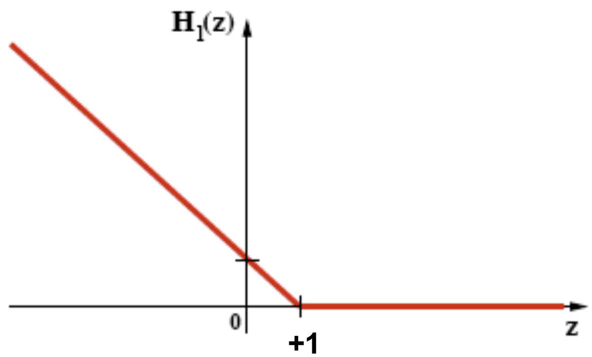
- **Soft-margin SVM is equivalent to the following cost function:**

$$\min P(\boldsymbol{w}, b) = \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{maximize margin}} + \underbrace{C \sum_i H_1[\, y_i \, f(\boldsymbol{x}_i)\,]}_{\text{minimize training error}}$$

$f(x_i) = y_i(x_i w^T + b)$

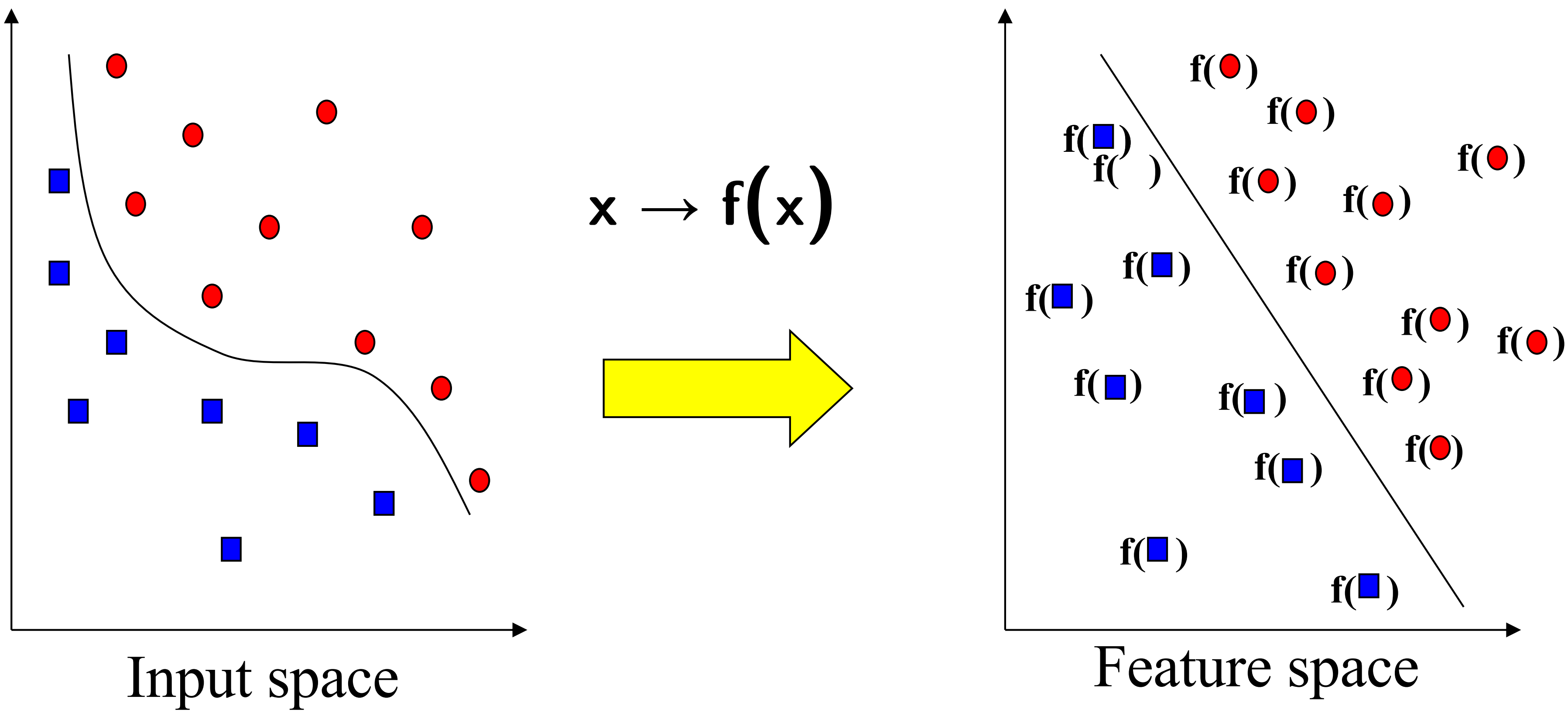
Ideally H_1 would count the number of errors, approximate with:

Hinge Loss $H_1(z) = \max(0, 1 - z)$



Support Vector Machine (IV)

- For nonlinear separation boundary:
  - use a feature mapping function



Support Vector Machine (VI)

- Nonlinear SVM based on a nonlinear mapping:

$$\mathbf{x}_i \Longrightarrow f(\mathbf{x}_j)$$

$$\max \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j f(\mathbf{x}_i)^T f(\mathbf{x}_j)$$

$$\text{subject to} \quad 0 \le \alpha_i \le C \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- Replace it by a Kernel function

$$\Phi(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_i)^T f(\mathbf{x}_j)$$

- Kernel trick: no need to know the original mapping function f()

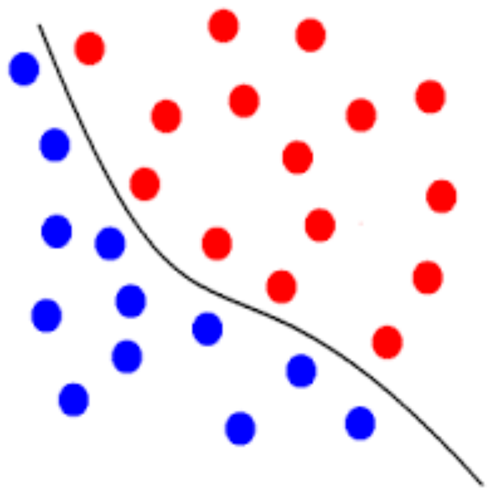
Support Vector Machine (VII)

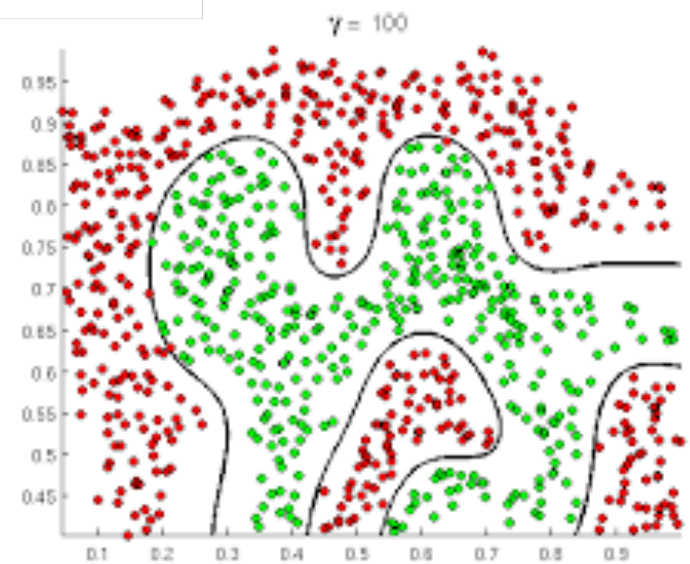
- Popular kernel functions:
  - Polynomial kernels

$$\Phi(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^p \quad \text{or} \quad \Phi(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^p$$

  - Gaussian (RBF) kernels

$$\Phi(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2)$$





Projected Gradient Descent for SVMs

- Dual problem of SVM is a dense quadratic programming:

$$\min_{\boldsymbol{\alpha}} \quad \overbrace{\frac{1}{2}\boldsymbol{\alpha}^\intercal \mathbf{Q}\boldsymbol{\alpha} - \mathbf{e}^\intercal \boldsymbol{\alpha}}^{L(\boldsymbol{\alpha})}$$

subject to $\mathbf{y}^\intercal \boldsymbol{\alpha} = 0$, $0 \leq \alpha_i \leq C$, where $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_T \end{bmatrix}_{T \times 1}$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_T \end{bmatrix}_{T \times 1} \qquad \mathbf{e} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{T \times 1}$$

$$\mathbf{Q} = \begin{bmatrix} Q_{ij} \end{bmatrix}_{T \times T} = \begin{bmatrix} \mathbf{y}\mathbf{y}^\intercal \end{bmatrix}_{T \times T} \odot \begin{bmatrix} \Phi(\mathbf{x}_i, \mathbf{x}_j) \end{bmatrix}_{T \times T}$$

Projected Gradient Descent for SVMs

- Set $n = 0$ and $\boldsymbol{\alpha}^{(0)} = 0$.
- Do until converge:
  1. compute the gradient: $\nabla L(\boldsymbol{\alpha}^{(n)}) = \mathbf{Q}\boldsymbol{\alpha}^{(n)} - \mathbf{e}$.
  2. project the gradient to the hyperplane $\mathbf{y}^\intercal \boldsymbol{\alpha} = 0$:

$$\tilde{\nabla} L(\boldsymbol{\alpha}^{(n)}) = \nabla L(\boldsymbol{\alpha}^{(n)}) - \frac{\mathbf{y}^\intercal \nabla L(\boldsymbol{\alpha}^{(n)})}{||\mathbf{y}||^{\mathbf{2}}} \mathbf{y}.$$

  3. projected gradient descent: $\boldsymbol{\alpha}^{(n+1)} = \boldsymbol{\alpha}^{(n)} - \epsilon_n \cdot \tilde{\nabla} L(\boldsymbol{\alpha}^{(n)})$.
  4. n = n + 1.

From 2-class to Multi-class

- Use multiple 2-class classifiers
  - One vs. One
  - One vs. all

- Direct Multi-class formulation
  - Multiple linear discriminants
  - MCE classifiers for N-class
  - Multi-class SVMs

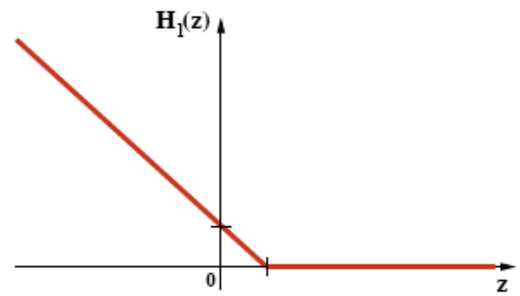
Learning Dicriminative Models in general

- The objective function for learning SVMs:

$$\min P(\boldsymbol{w}, b) = \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{maximize margin}} + \underbrace{C\sum_i H_1[\,y_i\, f(\boldsymbol{x}_i)\,]}_{\text{minimize training error}}$$

Ideally H_1 would count the number of errors, approximate with:

Hinge Loss $H_1(z) = \max(0, 1 - z)$

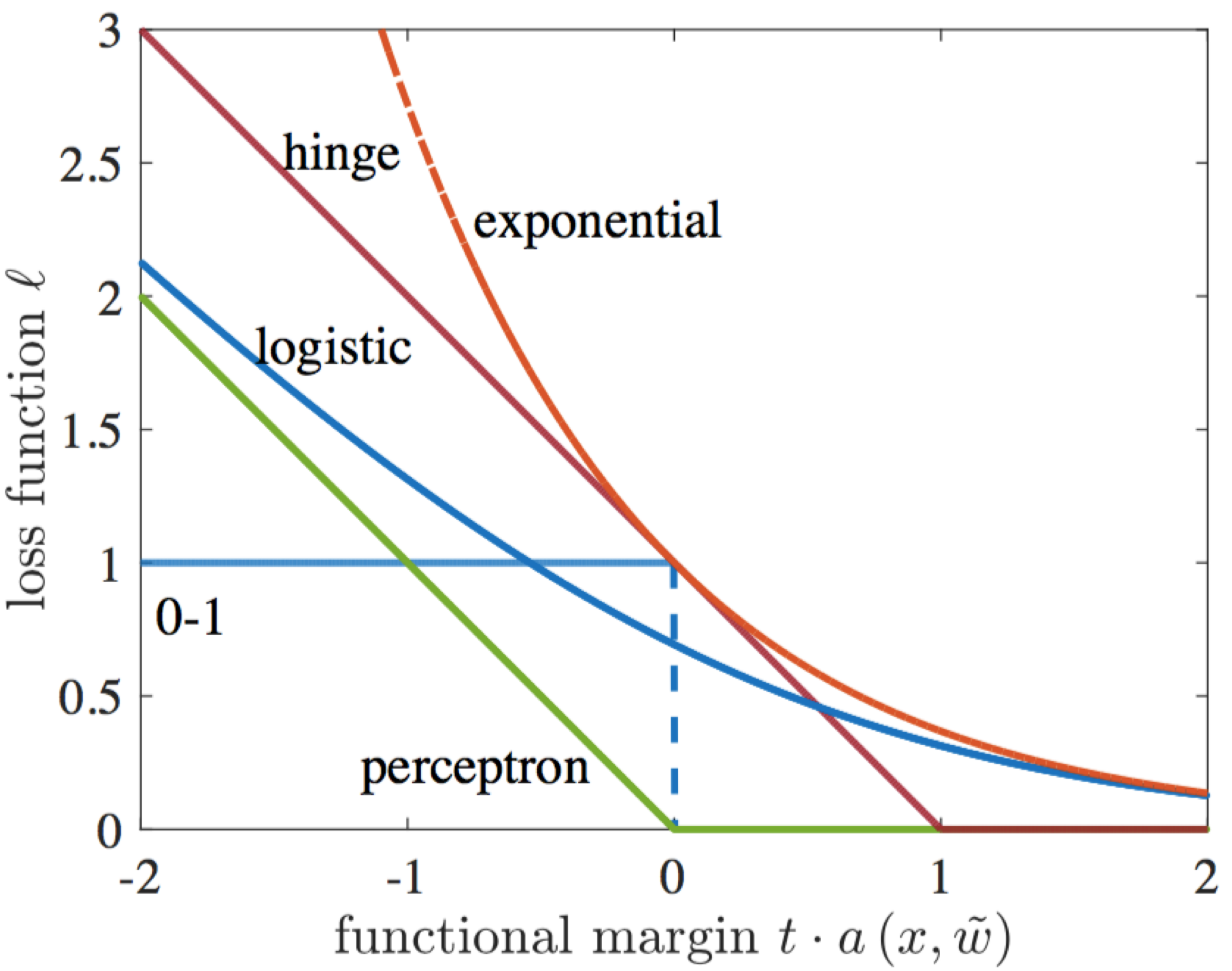


- The objective fucntion for learning discriminative models in general:

$$\text{Q} = \text{error function} + \text{regularization term}$$

Error Functions in ML

- **Some popular error functions used in machine learning:**



L_p norm

- L_p norm is defined as:

$$\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{\frac{1}{p}}$$

- L_2 norm (Eucleadian norm):

$$\|x\|_2 = \left(x_1^2 + x_2^2 + \cdots + x_n^2\right)^{\frac{1}{2}}$$

- L_0 norm: num of non-zero entries

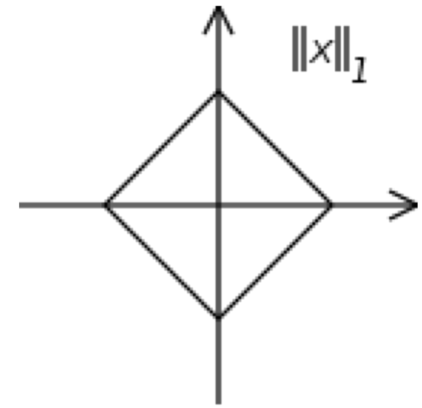
$$|x_1|^0 + |x_2|^0 + \cdots + |x_n|^0$$

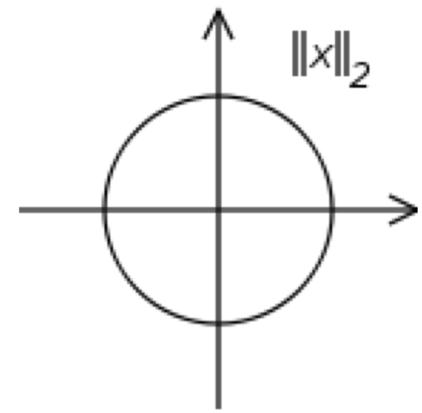
- L_1 norm:

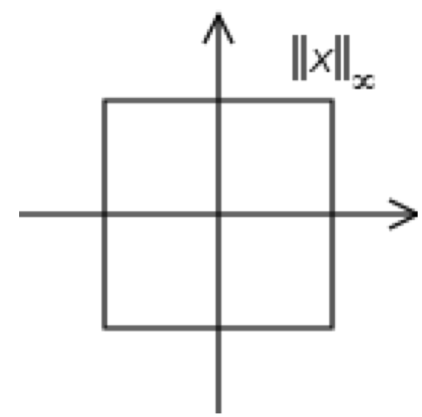
$$|x_1| + |x_2| + \cdots + |x_n|$$

- L_∞ norm (maximum norm):

$$\|x\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$$







L_p norm in 3-D

- L_p norm constraints in 3-D:

$$\|x\|_p \le 1$$

Ridge Regression

- Ridge Regression = Linear Regression + L_2 norm

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2 \right\}$$

- A closed-form solution:

$$\hat{\beta}_j = (1 + N\lambda)^{-1} \hat{\beta}_j^{\text{OLS}}$$

$$\hat{\beta}^{\text{OLS}} = (X^T X)^{-1} X^T y$$

LASSO

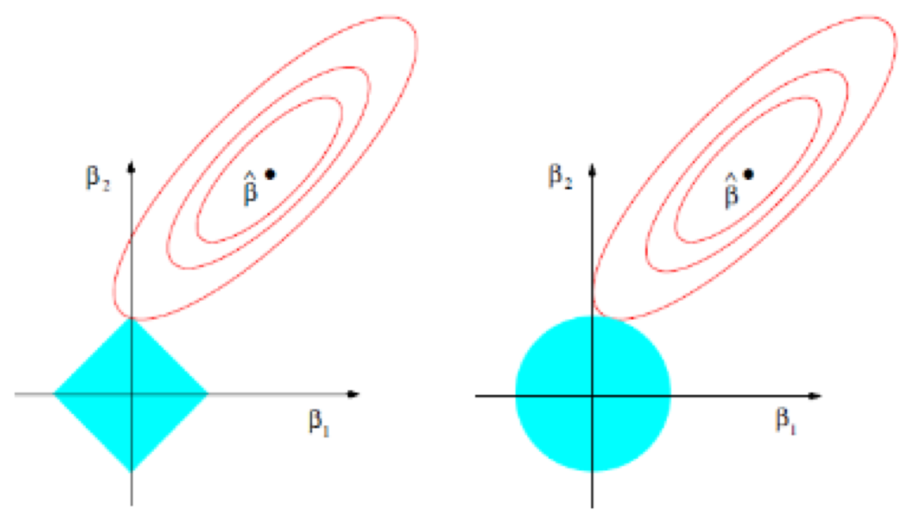
- **LASSO: least absolute shrinkage and selection operator**
- **LASSO = Linear Regression + L_1 norm regularization**

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

- **Equivallent to**

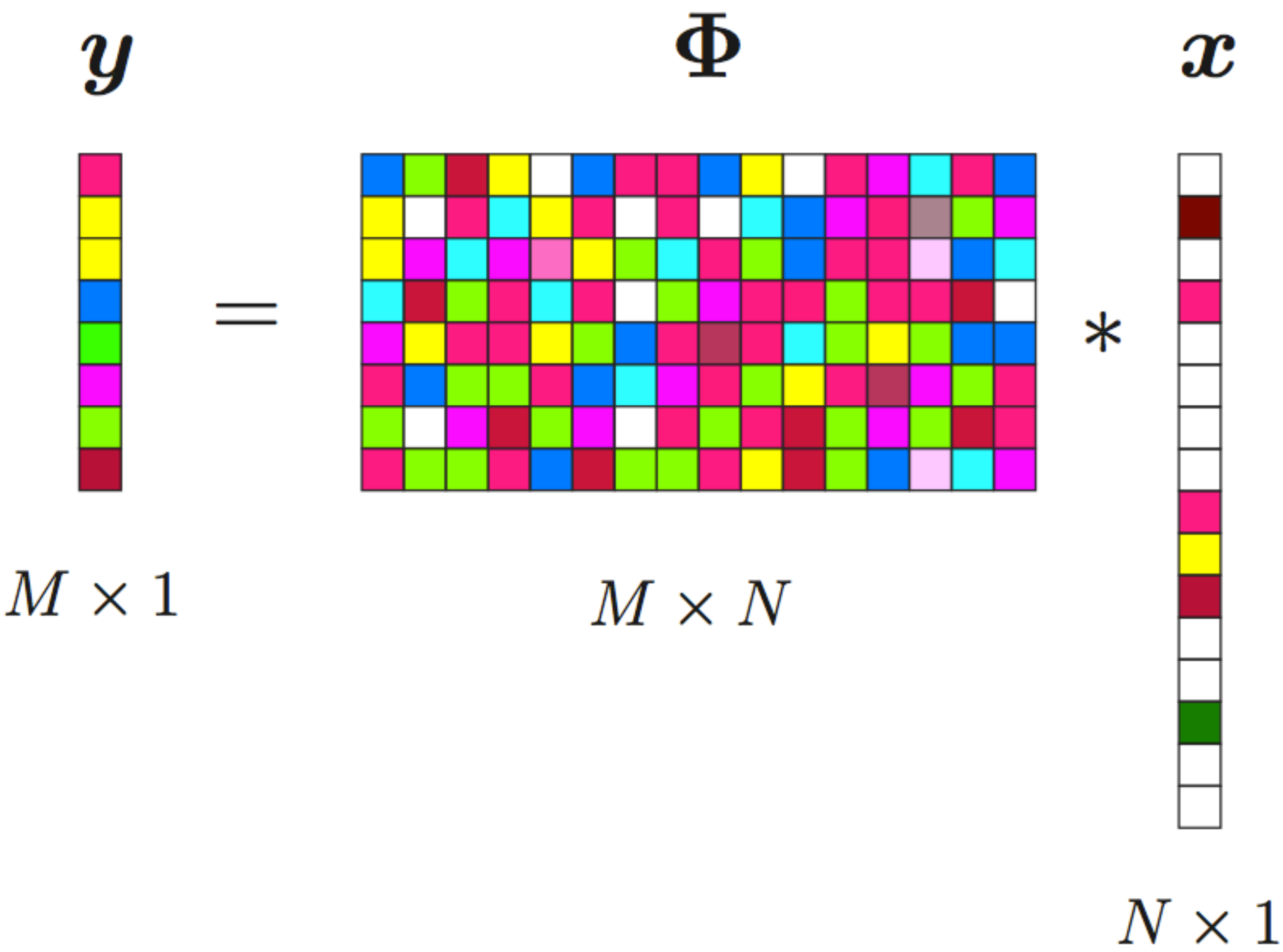
$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 \right\} \text{ subject to } \|\beta\|_1 \le t.$$

- **Leading to sparse solution.**
- **Need subgradient methods.**



Compressed Sensing

- a.k.a. Compressive Sensing; Sparse Coding
- A real object = sparse coding from a large dictionary



Compressed Sensing

- **Math formulation:**

$$\min \quad ||\mathbf{x}||_0 \quad \text{subject to} \quad \Phi\mathbf{x} = \mathbf{y}$$

- **Or some simpler ones:**

$$\min \quad ||\mathbf{x}||_1 \quad \text{subject to} \quad \Phi\mathbf{x} = \mathbf{y}$$

$$\min \quad ||\Phi\mathbf{x} - \mathbf{y}||_2 + \lambda||\mathbf{x}||_1$$

Advanced Topics

- Mutli-class SVMs
- Max-margin Markov Networks
- Compressed Sensing (or Sparse Coding)
- Relevance Vector Machine
- Transductive SVMs