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

- 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

Generative vs. discriminative models

- **Posterior probability $p(\omega_i|X)$ plays the key role in pattern classification, also machine learning.**
  - **Generative Models: focus on probability distribution of data**

    $$p(\omega_i|X) \sim p(\omega_i) \cdot p(X|\omega_i)$$

    $$\approx p'(\omega_i) \cdot p'(X|\omega_i) \qquad \textit{(the plug-in MAP rule)}$$

  - **Discriminative Models: directly model discriminant function:**

    $$p(\omega_i|X) \sim g_i(X)$$

Pattern classification based on Discriminant Functions (I)

- Instead of designing a classifier based on probability distribution of data, we can build an ad-hoc classifier based on some discriminant functions to model class boundary info directly.
- Classifier based on discriminant functions:
  - For N classes, we define a set of discriminant functions $h_i(X)$ $(i=1,2,\ldots,N)$, 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 $h_i(X ; \theta_i)$.
  - Similarly θ_i $(i=1,2,\ldots,N)$ need to be estimated from some training data.

Statistical Learning Theory

- Training samples (x_i,y_i) $(i=1,2,\ldots,m)$
- Random sariables X, Y: joint distribution P(X,Y)
- Input space $\mathbb{X}$: X from $\mathbb{X}$
- Output space $\mathbb{Y}$: Y from $\mathbb{Y}$

- Machine Learning tries to :

$$y = h(X) + \varepsilon$$

- Hypothesis space $\mathbb{H}$: h(.) from $\mathbb{H}$

- Loss Function: L(y, y')

  0/1 loss, squared error , …

Statistical Learning Theory

- Empirical Loss (*a.k.a.* empirical risk, in-sample error):

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

- Generalization error (*a.k.a.* generalization risk)

$$R(h) = \mathbb{E}_{(x,y)\sim P(X,Y)}[L(y, h(x))]$$

- Empirical Loss != Generalization error

- Learnable or not: empirical risk minimization (ERM) ➔ minimizing the generalization error.

Statistical Learning Theory

- **Learnibility depends on:**

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} |R(h) - R_{\mathrm{emp}(h)}| > \epsilon\right]$$

- **VC Generalization bounds (Vapnik-Chervonenkis theory):**

$$R(h) \leq R_{\mathrm{emp}}(h) + \sqrt{\frac{8d_{\mathrm{vc}}(\ln\frac{2m}{d_{\mathrm{vc}}} + 1) + 8\ln\frac{4}{\delta}}{m}}$$

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

Generalization Bounds

- **The weak law of large numbers:**

$$\lim_{m\to\infty} \mathbb{P}\left[\left|\mathop{\mathbb{E}}_{X\sim P}[X] - \frac{1}{m}\sum_{i=1}^{m} x_i\right| > \epsilon\right] = 0$$

- **Concentration inequalities (Heoffding's inequality)**

If $x_1, x_2, \ldots, x_m$ are m i.i.d. samples of a random variable X distributed by P, and $a \leq x_i \leq b$ for every i, then for a small positive non-zero value ϵ:

$$\mathbb{P}\left[\left|\mathop{\mathbb{E}}_{X\sim P}[X] - \frac{1}{m}\sum_{i=0}^{m} x_i\right| > \epsilon\right] \leq 2\exp\left(\frac{-2m\epsilon^2}{(b-a)^2}\right)$$

Generalization Bounds

- **For a single hypothesis h:**

$$\mathbb{P}[|R(h) - R_{\text{emp}}(h)| > \epsilon] \leq 2\exp(-2m\epsilon^2)$$

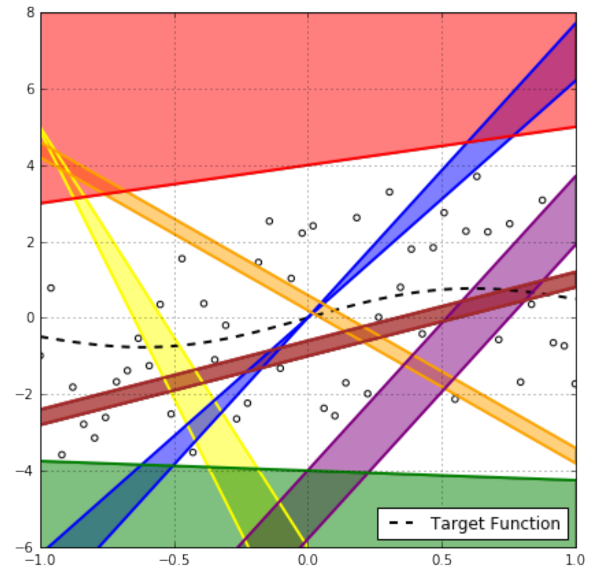
- **Extend for the whole hypothesis space:**

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} |R(h) - R_{\text{emp}}(h)| > \epsilon\right] \leq 2|\mathcal{H}|\exp(-2m\epsilon^2)$$

- **The first bound:**

$$R(h) \leq R_{\text{emp}}(h) + \sqrt{\frac{\ln|\mathcal{H}| + \ln\frac{2}{\delta}}{2m}}$$

VC-Dimension



- How about infinite number of h in ℍ?

  Not all h are different …

- VC-dimension:
  - Max # of points the hypothesis space ℍ can shatter
  - Roughly represents model capability
  - VC-dimension of linear classifier: D+1
  - VC-dimension of Neural network ≤ num of weights

Examples of generalization bounds

$$R(h) \leq R_{\text{emp}}(h) + \sqrt{\frac{8d_{vc}(\ln\frac{2m}{d_{vc}}+1)+8\ln\frac{4}{\delta}}{m}}$$

- **Example I**: use m=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 (II)

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

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

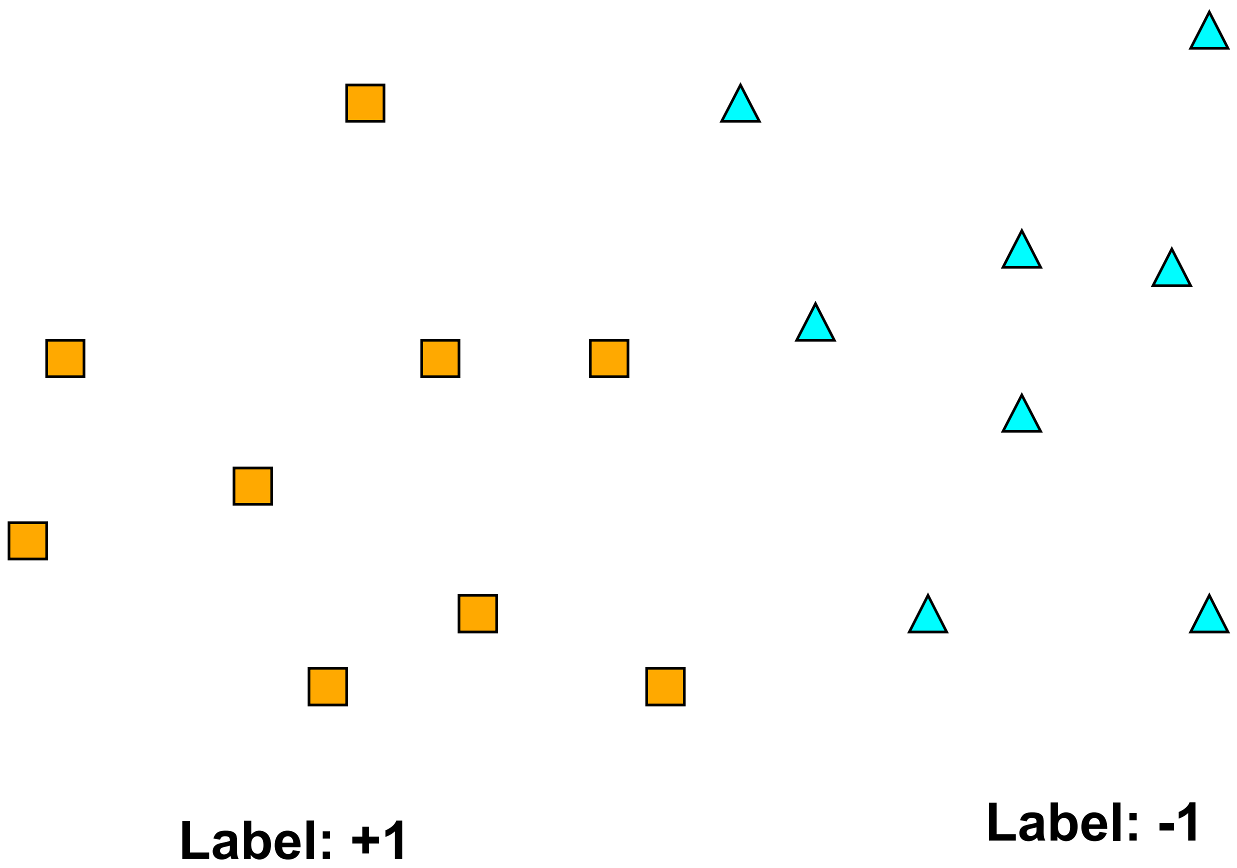
  - Quadratic discrimiant function: (2nd order)
  - Polynomial discriminant function: (N-th order)
  - *Neural network*: (arbitrary nonlinear functions)
  - Optimal discriminant functions: optimal MAP classifier is a special case when choosing discriminant functions as class posterior probabilities.

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.
  - Minimum Classification Error (MCE): minimizing empirical classification errors.
  - Support Vector Machine (SVM): maximizing separation margin.

Binary Classification Task

- **Separating two classes using linear models**



Perceptron

- Rosenblatt (1962)
- Linear models for two-class problems

$$f(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{x} \cdot \mathbf{w} + \mathbf{b} > 0 \\ -1 & \text{otherwise} \end{cases}$$

- Perceptron algorithm: a very simple learning algorithm

  - ***Randomly initialize*** *w(0)* ***and*** *b(0)*, *t=0*
  - ***For each sample*** (x_i, y_i) $(i=1,\ldots,m)$
    - ***Calculate the actual output:***

      $h_i(t) = \text{sign}(f(x_i))$
    - ***On a mistake Update the weights upon mistakes:***

      $w(t+1) = w(t) + y_i x_i$

      $b(t+1) = b(t) + y_i$
    - $t = t + 1$
  - ***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*

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

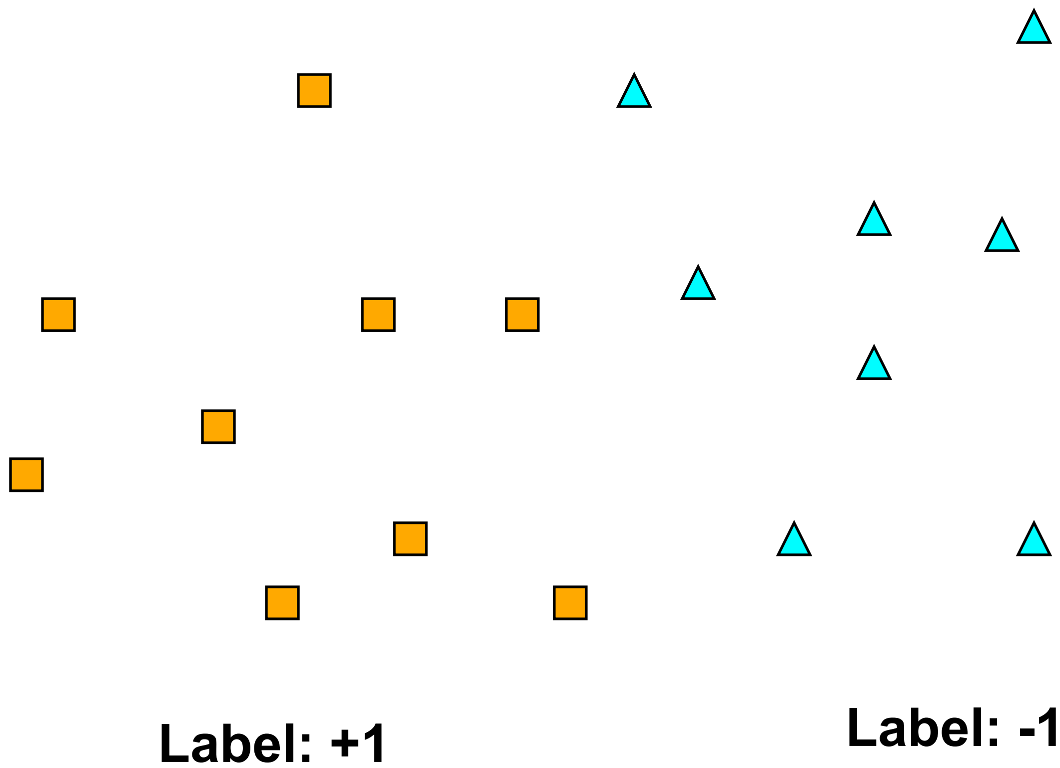
- **Proof can be found:**

[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:**

$$X = \begin{bmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \vdots \\ \mathrm{x}_T \end{bmatrix} \quad \xrightarrow{\mathrm{Y} = \mathrm{X}\mathrm{w}^T} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{bmatrix} = \begin{bmatrix} +1 \\ -1 \\ \vdots \\ +1 \end{bmatrix}$$

$$w^* = \arg\min_w \sum_i \left(\mathrm{x}_i w^T - y_i\right)^2$$

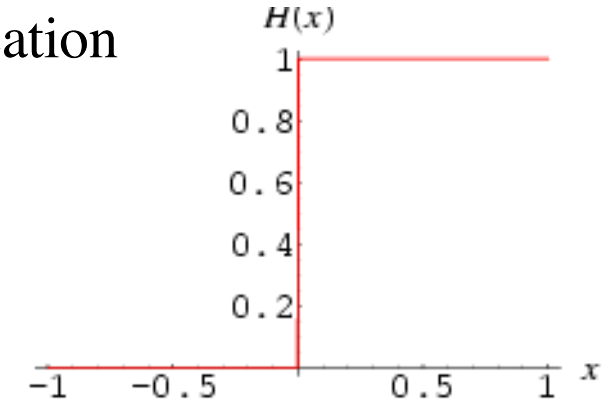
$$\mathrm{w}^* = \left(X^T X\right)^{-1} X^T Y$$

- **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.**

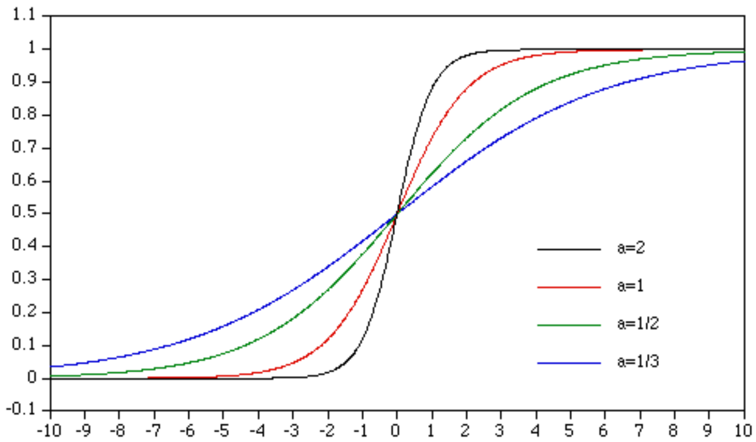
$$(\mathrm{x}_i, y_i) \Rightarrow \begin{cases} g_i = -y_i \mathrm{x}_i \mathrm{w}^T < 0 & \text{correct classification} \\ g_i = -y_i \mathrm{x}_i \mathrm{w}^T > 0 & \text{wrong classification} \end{cases}$$



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

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

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



Minimum Classification Error (MCE)

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

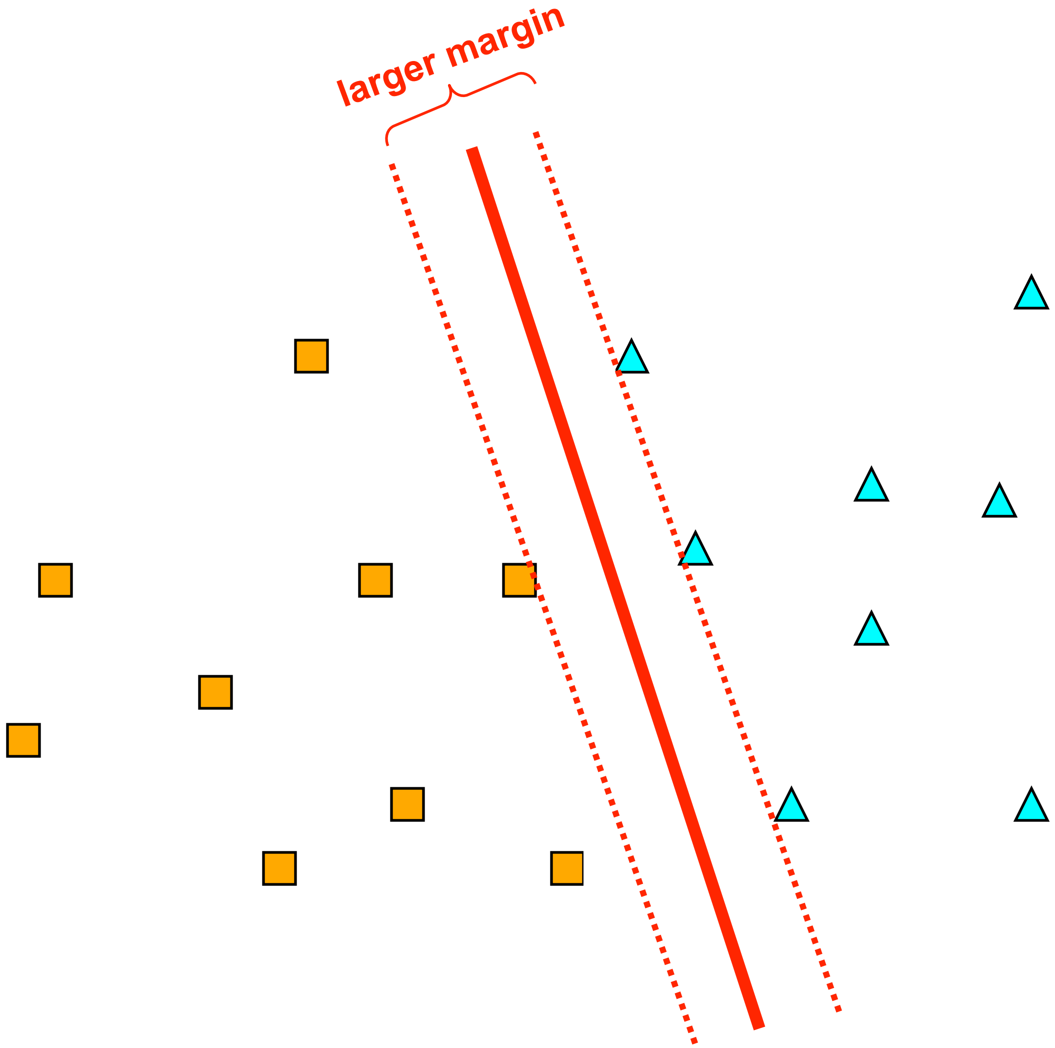
$$E(\mathbf{w}) = \sum_i l(y_i \mathbf{x}_i \mathbf{w}^T)$$

- The gradient is computed as:

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

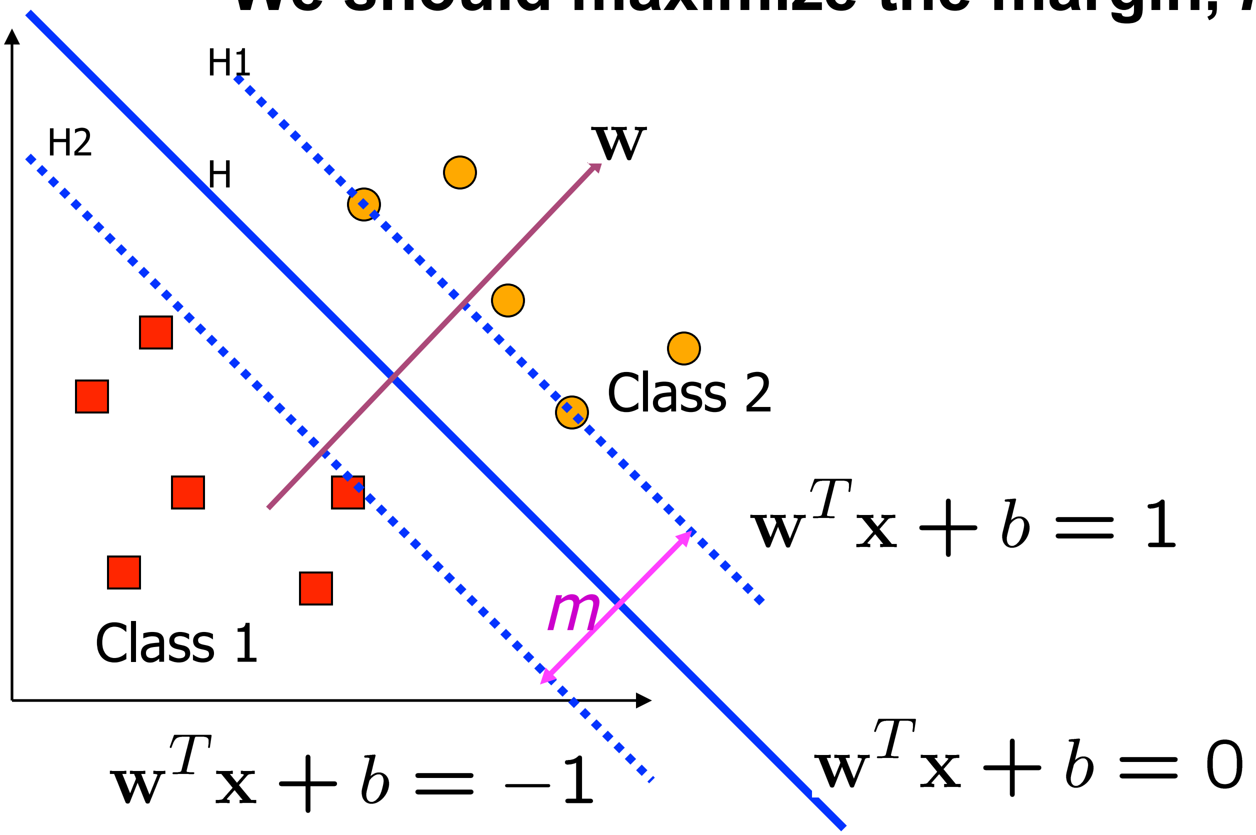
- May also use SGD ….

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:**

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

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

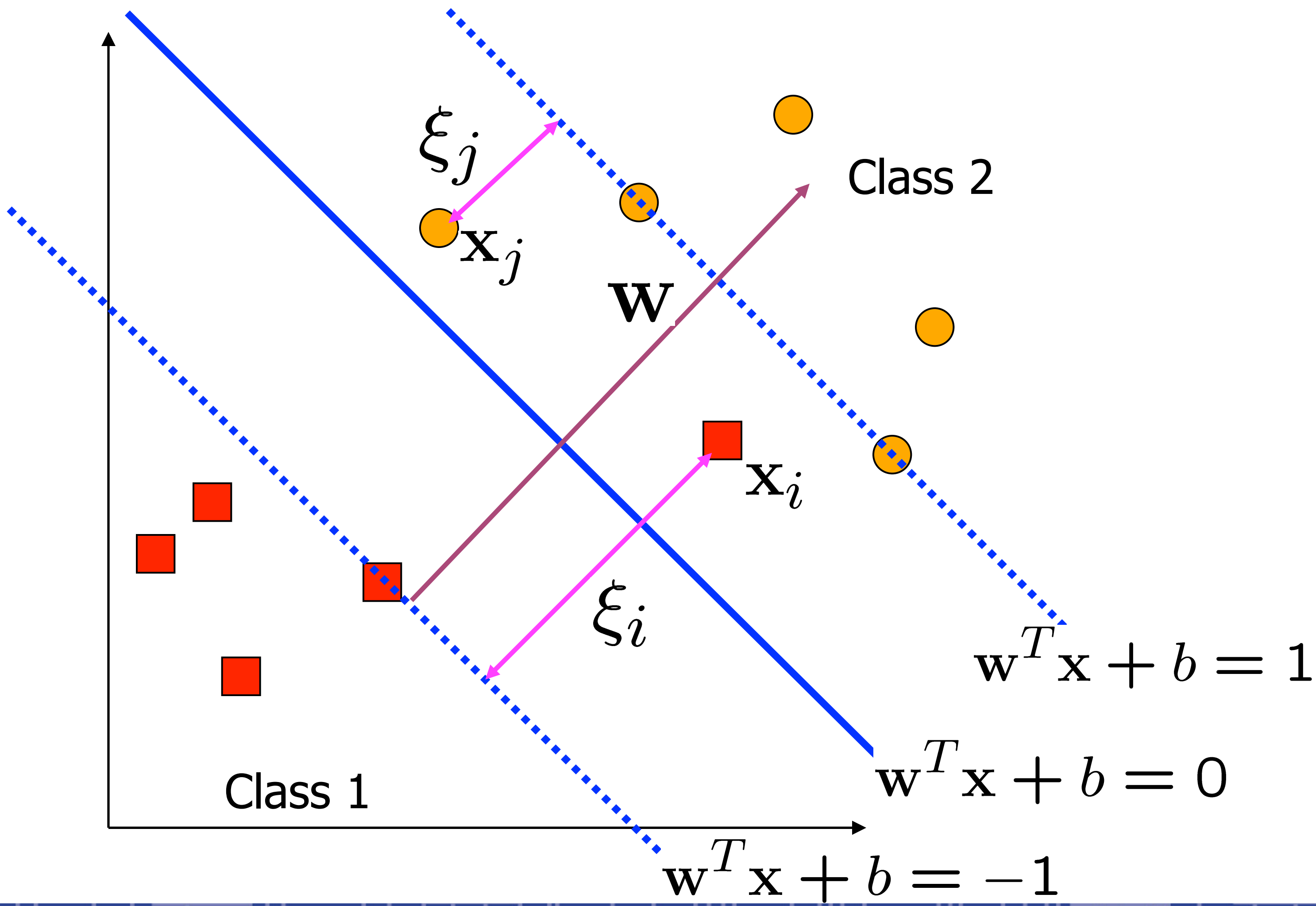
- **Convert to its dual problem:**

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

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

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:**

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

subject to

$$\mathrm{y}_i(\mathrm{x}_i \mathrm{w}^T + b) > 1 - \xi_i \quad \xi_i > 0 \quad (\forall i)$$

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

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

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

Support Vector Machine (IV)

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

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

subject to

$$\mathrm{y}_i(\mathrm{x}_i \mathrm{w}^T + b) > 1 - \xi_i \quad \xi_i > 0 \quad (\forall i)$$

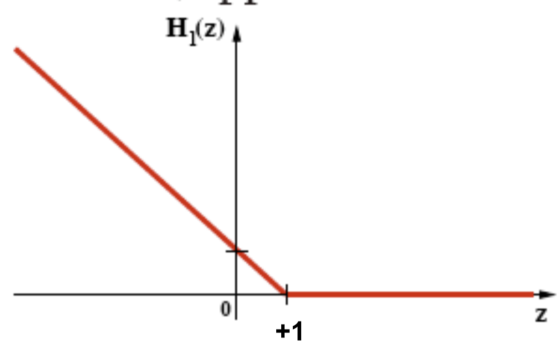
- **Soft-margin SVM is equivalent to the following cost function:**

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

$$f(\mathrm{x}_i) = y_i(\mathrm{x}_i \mathrm{w}^T + b)$$

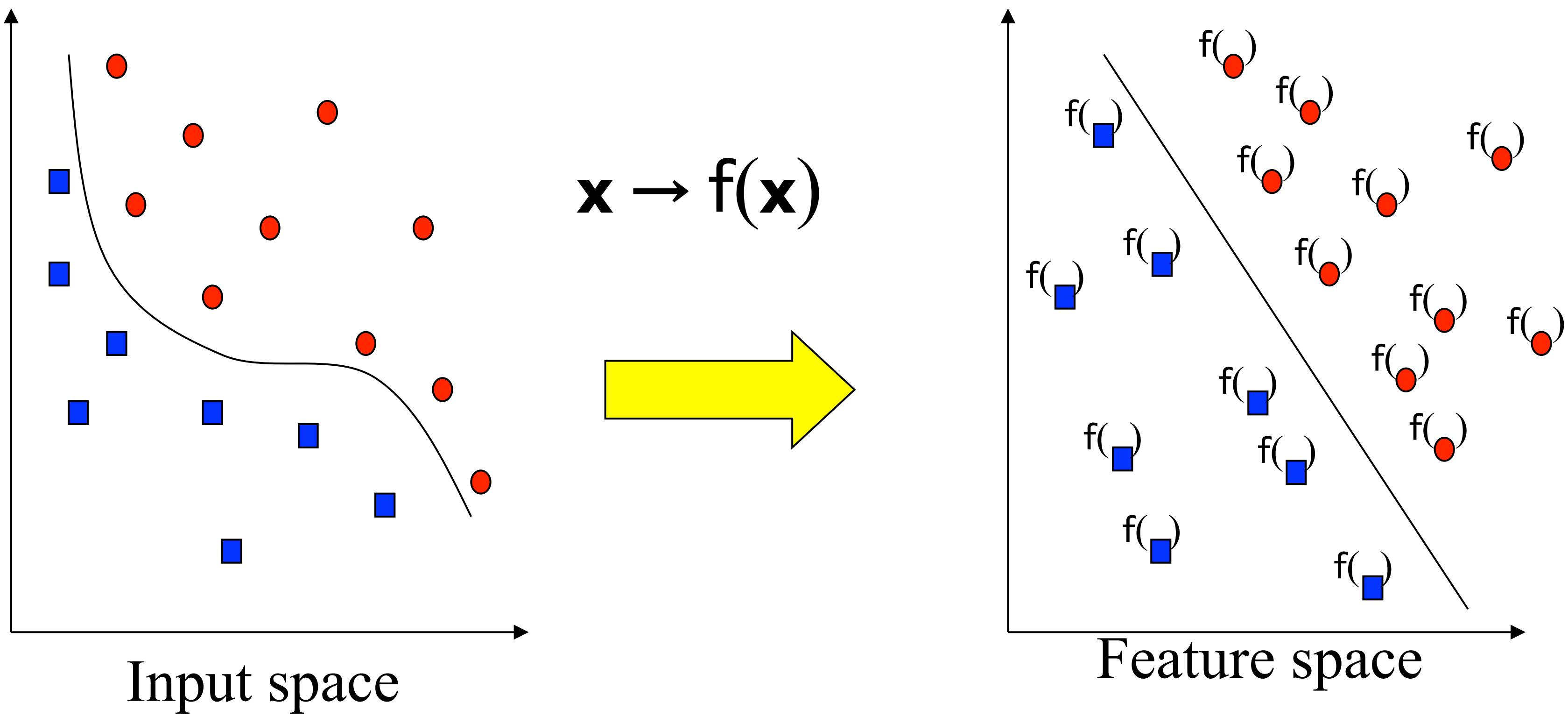
Ideally H_1 would count the number of errors, approximate with:

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



Support Vector Machine (IV)

- For nonlinear separation boundary:
  - use a Kernel function



Support Vector Machine (VI)

- **Nonlinear SVM based on a nonlinear mapping:**

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

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

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

- **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()**

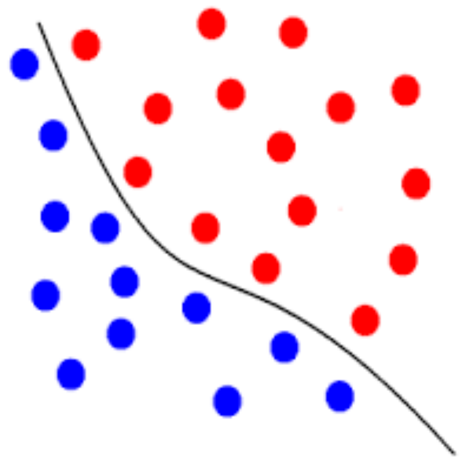
Support Vector Machine (VII)

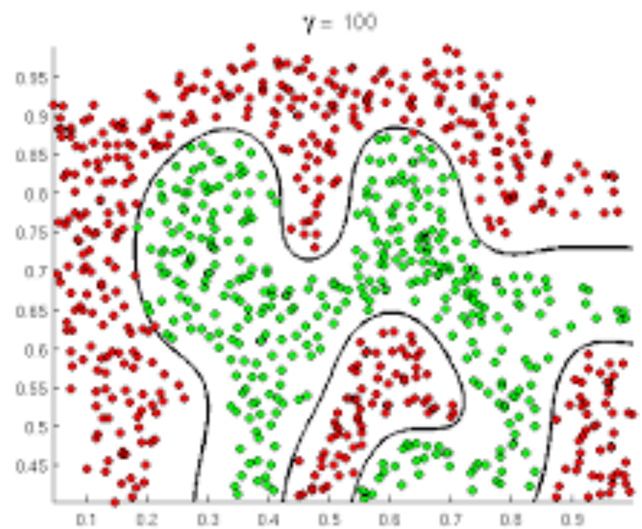
- **Popular Kernel functions:**
  - **Polynomial kernels**

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

  - **Gaussian (RBF) kernels**

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





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

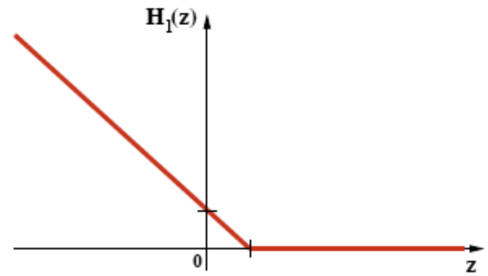
Learning Dicriminative Models in general

- **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:**

Q = error function + regularization term

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**

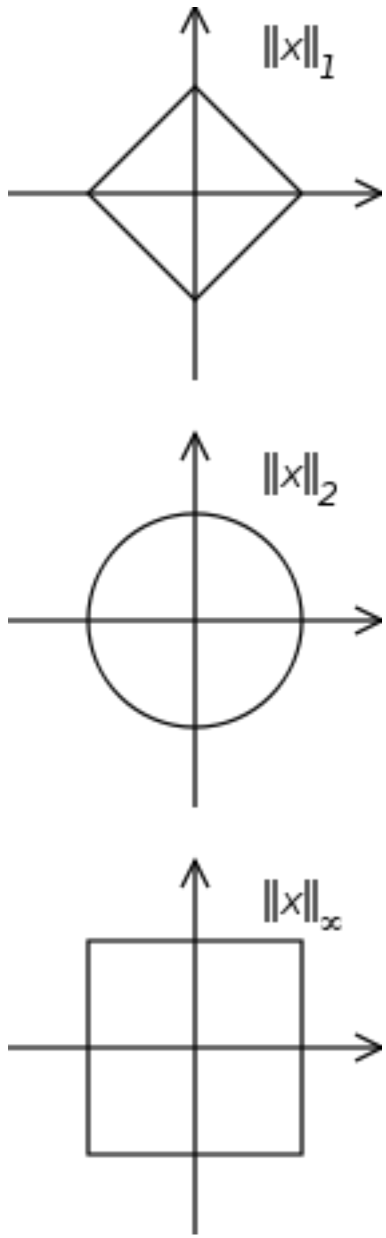
$$|x_1|^0 + |x_2|^0 + \cdots + |x_n|^0$$

- **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:

$$\| x \|_p \leq 1$$

Ridge Regression

- **Ridge Regression = Linear Regression + L_2 norm**

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

- **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

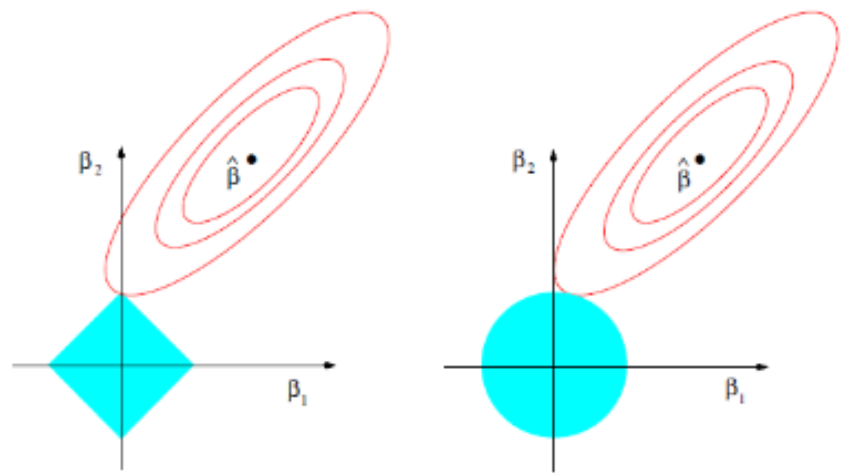
- **LASSO: least absolute shrinkage and selection operator**
- **LASSO = Linear Regression + L_1 norm**

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

- **Equivallent to**

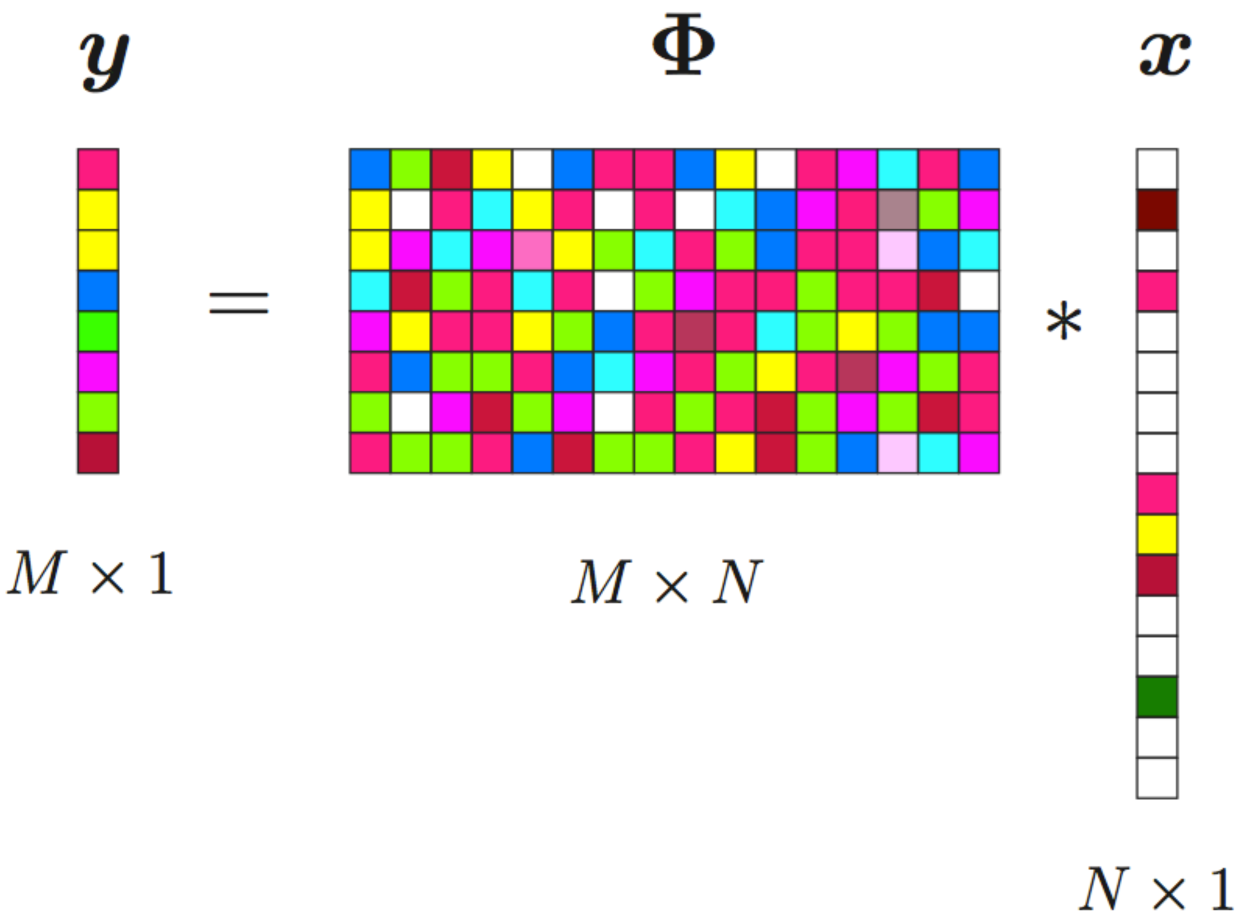
$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 \right\} \text{ subject to } \|\beta\|_1 \leq t.$$

- **Leading to sparse solution.**
- **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