Linear Models

Li, Jia

DSAA 5002
The Hong Kong of Science and Technology (Guangzhou)

2025 Fall

Sep 15

Overview

- Linear models
  - Perceptron: model and learning algorithm combined as one
  - Is there a better way to learn linear models?
- We will separate models and learning algorithms
  - Learning as optimization
  - Surrogate loss function
  - Regularization

  model design

  - Gradient descent
  - Batch and online gradients
  - Support vector machines

  optimization

Learning as Optimization

$$\min_{\mathbf{w}} \sum_{n} \mathbf{1}[y_n \mathbf{w}^T \mathbf{x}_n < 0]$$

fewest mistakes

- The perceptron algorithm will find an optimal **w** if the data is separable
  - efficiency depends on the margin and norm of the data
- However, if the data is not separable, optimizing this is NP-hard
  - i.e., there is no efficient way to minimize this unless **P=NP**

Learning as Optimization

hyperparameter

$$\min_{\mathbf{w}} \sum_n \mathbf{1}[y_n \mathbf{w}^T \mathbf{x}_n < 0] + \lambda R(\mathbf{w})$$

fewest mistakes

simpler model

- In addition to minimizing training error, we want a simpler model
  - Remember our goal is to minimize generalization error
- We can add a regularization term $R(\mathbf{w})$ that prefers simpler models
  - For example we may prefer decision trees of shallow depth
- Here λ is a hyperparameter of optimization problem

Learning as Optimization

hyperparameter

$$\min_{\mathbf{w}} \sum_n \mathbf{1}[y_n \mathbf{w}^T \mathbf{x}_n < 0] + \lambda R(\mathbf{w})$$

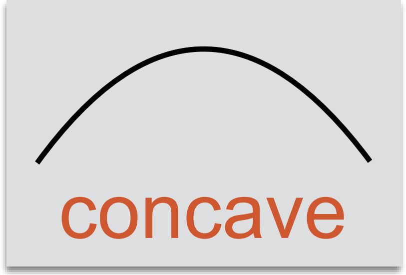
fewest mistakes

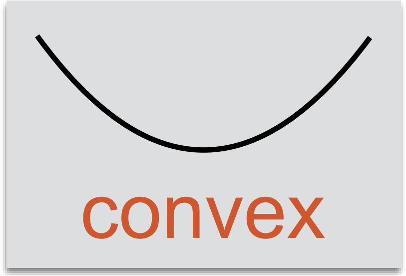
simpler model

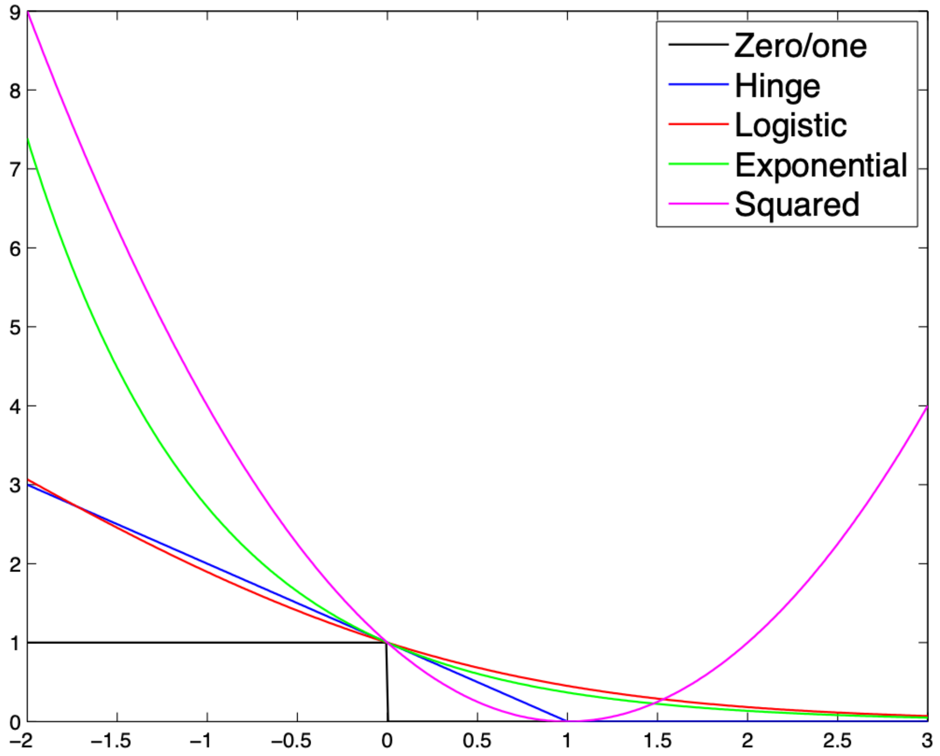
- The questions that remain are:
  - What are good ways to adjust the optimization problem so that there are efficient algorithms for solving it?
  - What are good regularizations ***R(w)*** for hyperplanes?
  - Assuming that the optimization problem can be adjusted appropriately, what algorithms exist for solving the regularized optimization problem?

Convex Surrogate Loss Functions

- Zero/one loss is hard to optimize
  - Small changes in **w** can cause large changes in the loss
- Surrogate loss: replace Zero/one loss by a smooth function
  - Easier to optimize if the surrogate loss is convex







$$y = +1 \qquad \hat{y} \leftarrow \mathbf{w}^T \mathbf{x}$$

$$\begin{aligned}
\text{Zero/one:} \quad & \ell^{(0/1)}(y, \hat{y}) = \mathbf{1}[y\hat{y} \leq 0] \\
\text{Hinge:} \quad & \ell^{(\text{hin})}(y, \hat{y}) = \max\{0, 1 - y\hat{y}\} \\
\text{Logistic:} \quad & \ell^{(\log)}(y, \hat{y}) = \frac{1}{\log 2} \log\left(1 + \exp[-y\hat{y}]\right) \\
\text{Exponential:} \quad & \ell^{(\exp)}(y, \hat{y}) = \exp[-y\hat{y}] \\
\text{Squared:} \quad & \ell^{(\text{sqr})}(y, \hat{y}) = (y - \hat{y})^2
\end{aligned}$$

Weight Regularization

- What are good regularization functions $R(\boldsymbol{w})$ for hyperplanes?
- We would like the weights —
  - To be small —
    - Change in the features cause small change to the score
    - Robustness to noise
  - To be sparse —
    - Use as few features as possible
    - Similar to controlling the depth of a decision tree
- This is a form of inductive bias

Weight Regularization

- Just like the surrogate loss function, we would like *R(**w**)* to be convex
- Small weights regularization

$$R^{(\text{norm})}(\mathbf{w}) = \sqrt{\sum_d w_d^2} \qquad R^{(\text{sqrd})}(\mathbf{w}) = \sum_d w_d^2$$

- Sparsity regularization

$$R^{(\text{count})}(\mathbf{w}) = \sum_d \mathbf{1}[|w_d| > 0]$$

not convex

- Family of "p-norm" regularization

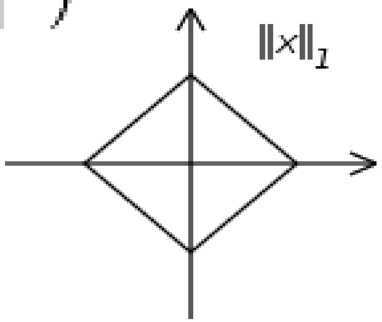
$$R^{(\text{p-norm})}(\mathbf{w}) = \left(\sum_d |w_d|^p\right)^{1/p}$$

Contours of p-norms

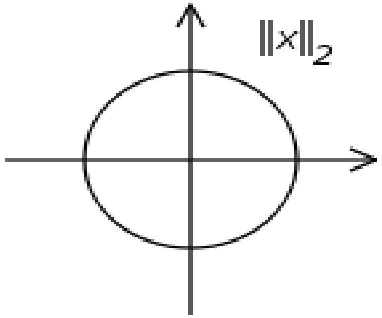
$$\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{\frac{1}{p}}$$

convex for $p \geq 1$

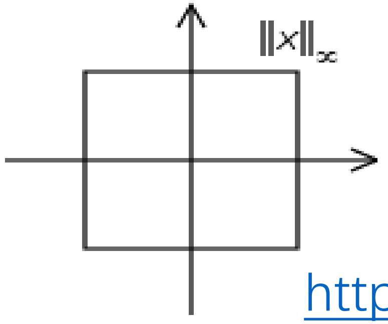
$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$



$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$



$$\|x\|_\infty = \max_{i=1,\ldots,n} |x_i|$$

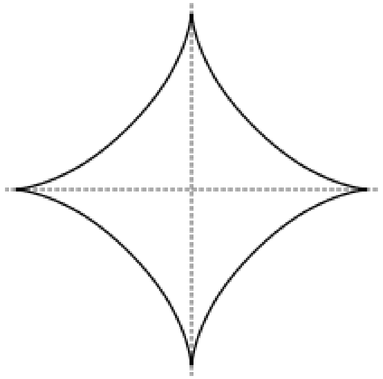


http://en.wikipedia.org/wiki/Lp_space

Contours of p-norms

$$\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{\frac{1}{p}}$$ not convex for $0 \leq p < 1$

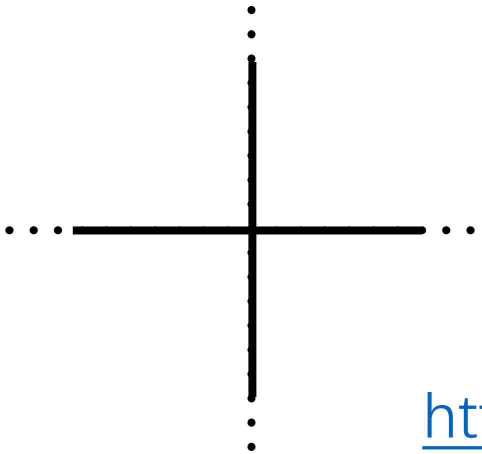
$$p = \frac{2}{3}$$



Counting non-zeros:

$$R^{(\text{count})}(\mathbf{w}) = \sum_d \mathbf{1}[|w_d| > 0]$$

$$p = 0$$



http://en.wikipedia.org/wiki/Lp_space

General Optimization Framework

hyperparameter

$$\min_{\mathbf{w}} \sum_n \ell\left(y_n, \mathbf{w}^T \mathbf{x}_n\right) + \lambda R(\mathbf{w})$$

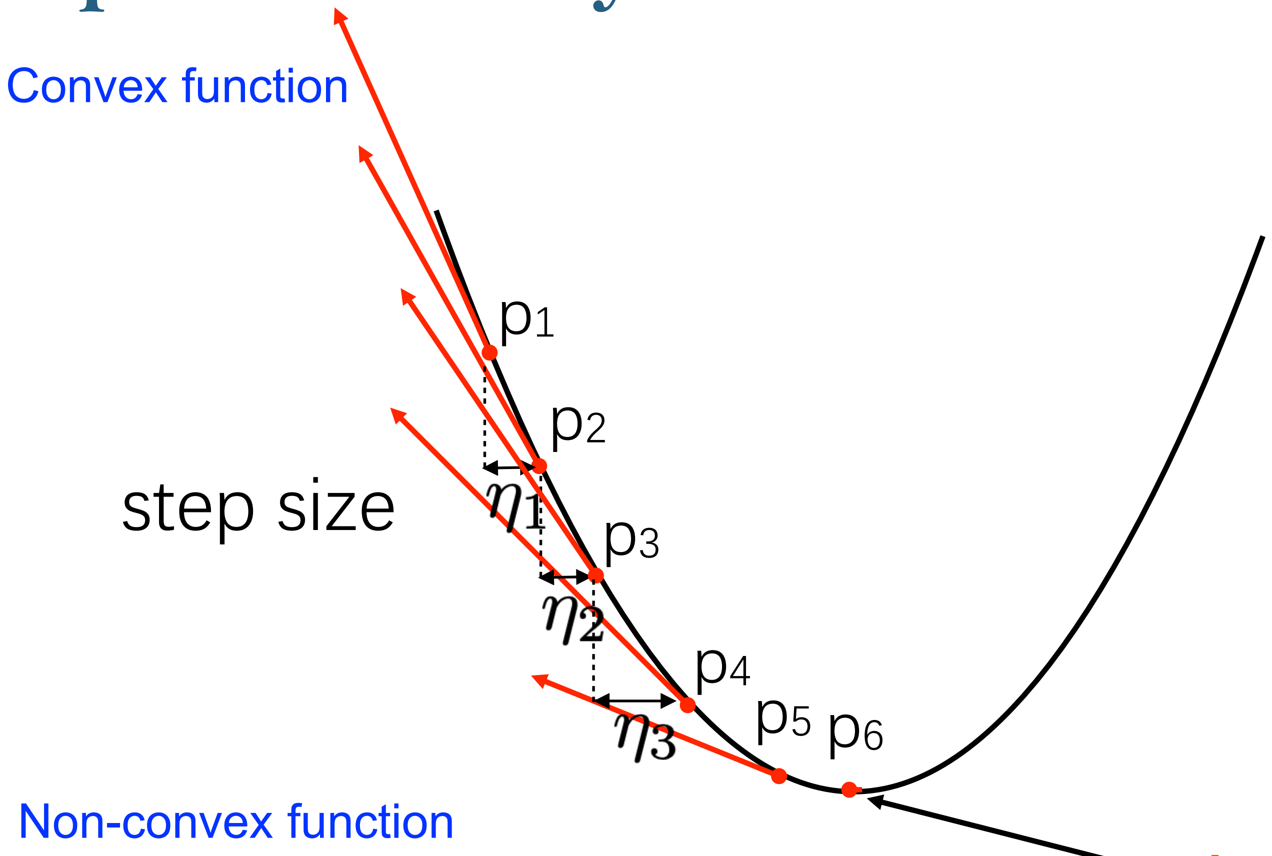
surrogate loss

regularization

- Select a suitable:
  - convex surrogate loss
  - convex regularization
- Select the hyperparameter λ
- Minimize the regularized objective with respect to **w**
- This framework for optimization is called Tikhonov regularization or
- generally Structural Risk Minimization (SRM)

http://en.wikipedia.org/wiki/Tikhonov_regularization

Optimization by Gradient Descent



$$g^{(k)} \leftarrow \nabla_p F(p)|_{p_k}$$

compute gradient at the current location

$$p_{k+1} \leftarrow p_k - \eta_k g^{(k)}$$

take a step down the gradient

local optima = global optima

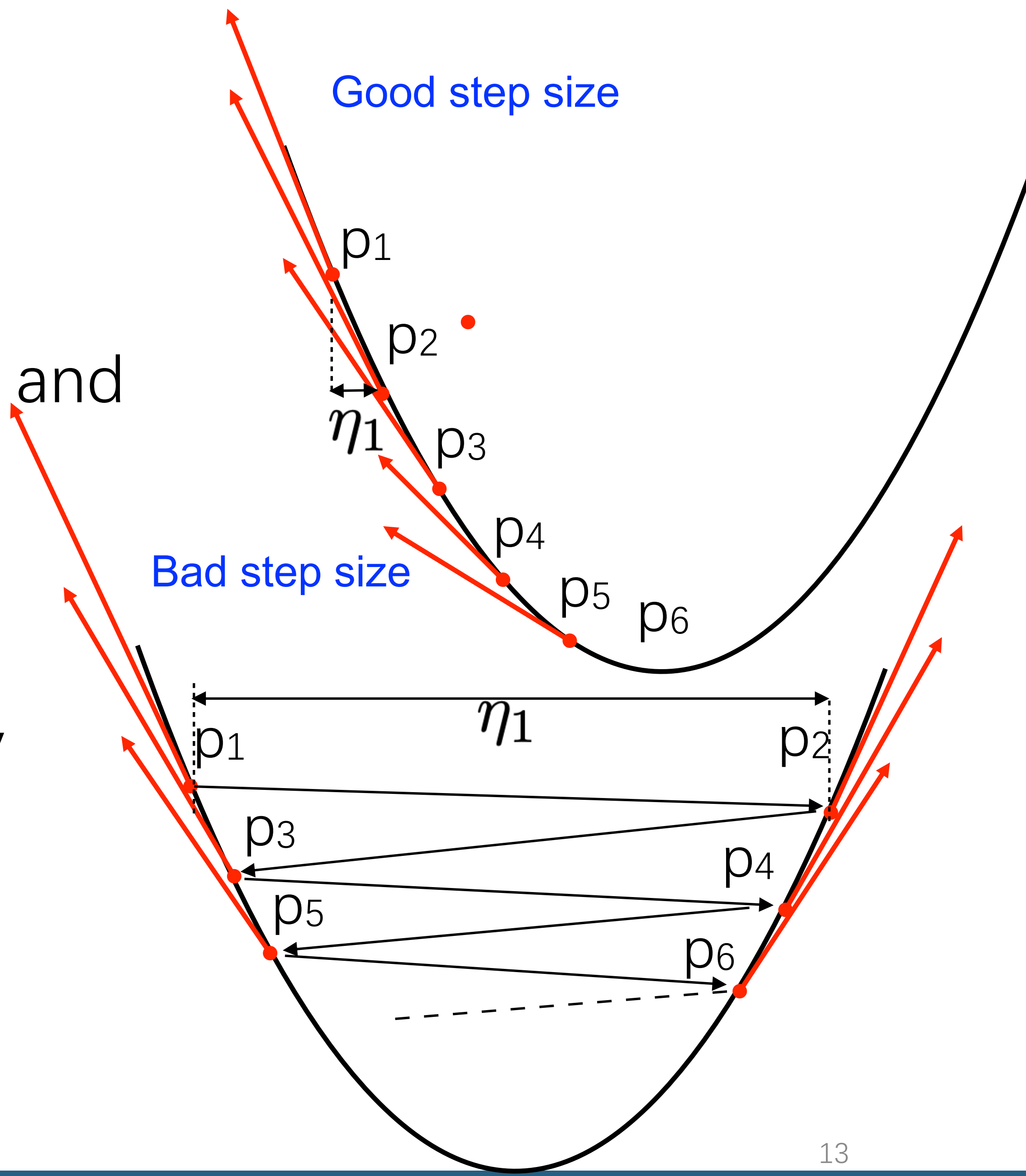
Choice of Step Size

- The step size is important —
  - too small: slow convergence
  - too large: no convergence
- A strategy is to use large step sizes initially and small step sizes later:

$$\eta_t \leftarrow \eta_0/(t_0 + t)$$

- There are methods that converge faster by adapting step size to the curvature of the function
  - Field of convex optimization

http://stanford.edu/~boyd/cvxbook/



Exponential Loss

$$\mathcal{L}(\mathbf{w}) = \sum_n \exp(-y_n \mathbf{w}^T \mathbf{x}_n) + \frac{\lambda}{2}||\mathbf{w}||^2$$ objective

$$\frac{d\mathcal{L}}{d\mathbf{w}} = \sum_n -y_n \mathbf{x}_n \exp(-y_n \mathbf{w}^T \mathbf{x}_n) + \lambda \mathbf{w}$$ gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \left(\sum_n -y_n \mathbf{x}_n \exp(-y_n \mathbf{w}^T \mathbf{x}_n) + \lambda \mathbf{w} \right)$$ update

loss term

$$\mathbf{w} \leftarrow \mathbf{w} + c y_n \mathbf{x}_n$$

high for misclassified points

regularization term

$$\mathbf{w} \leftarrow (1 - \eta\lambda)\mathbf{w}$$

shrinks weights towards zero

Batch and Online Gradients

$$\mathcal{L}(\mathbf{w}) = \sum_n \mathcal{L}_n(\mathbf{w})$$ objective

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{d\mathcal{L}}{d\mathbf{w}}$$ gradient descent

batch gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \left(\sum_n \frac{d\mathcal{L}_n}{d\mathbf{w}} \right)$$

sum of n gradients

update weight after you see all points

online gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \left(\frac{d\mathcal{L}_n}{d\mathbf{w}} \right)$$

gradient at n^{th} point

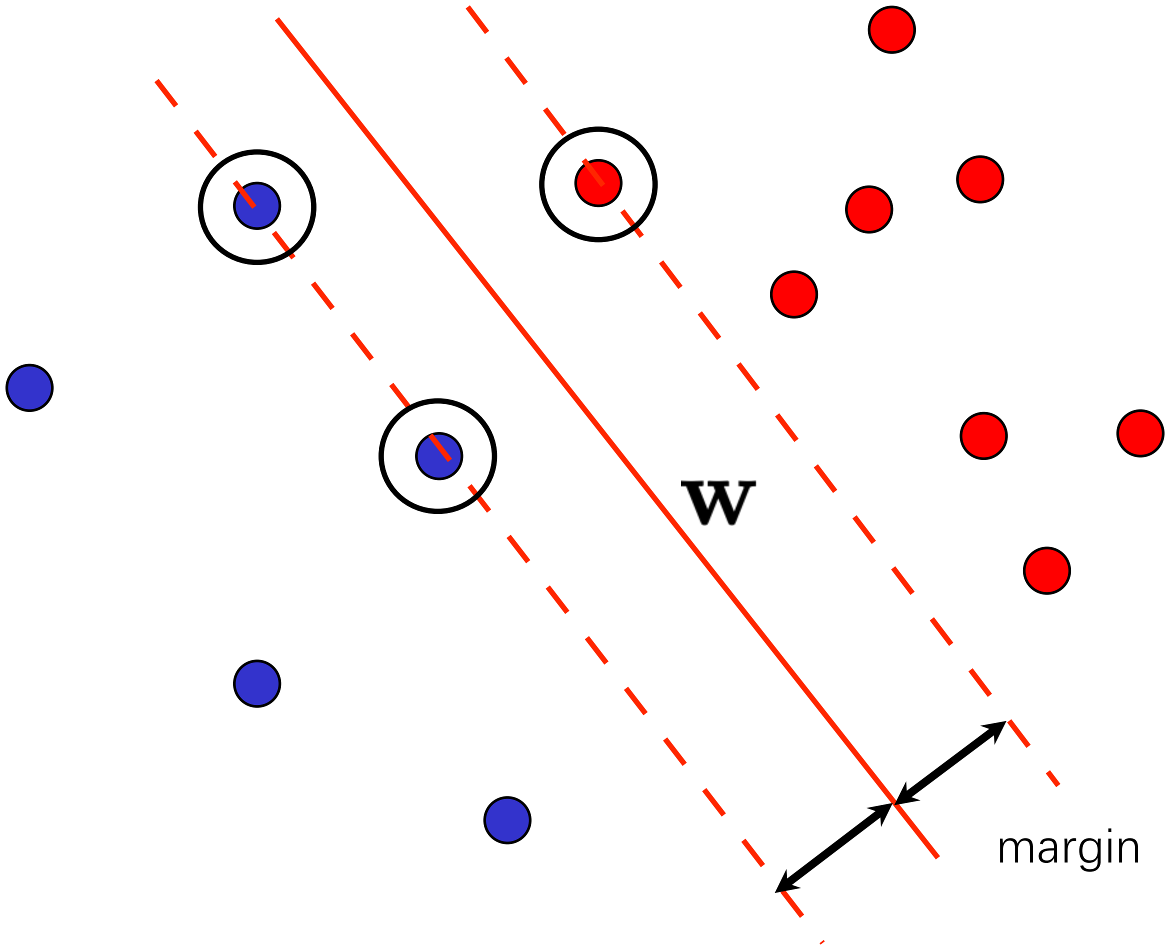
update weights after you see each point

Online gradients are the default method for multi-layer perceptrons

SVM Target

- Let $y_i \in \{+1, -1\}$, $f_{w,b}(x) = w^T x + b$. Margin:

$$\gamma = \min_i \frac{y_i f_{w,b}(x_i)}{||w||}$$



SVM Target

- Support Vector Machine:

$$\max_{w,b} \gamma = \max_{w,b} \min_{i} \frac{y_i f_{w,b}(x_i)}{||w||}$$

SVM Target

- Optimization (Quadratic Programming):

$$\min_{w,b} \frac{1}{2}||w||^2$$

$$y_i(w^T x_i + b) \geq 1, \forall i$$

- Solved by Lagrange multiplier method:

$$\mathcal{L}(w, b, \boldsymbol{\alpha}) = \frac{1}{2}||w||^2 - \sum_i \alpha_i [y_i(w^T x_i + b) - 1]$$

where $\boldsymbol{\alpha}$ is the Lagrange multiplier

Lagrangian

- Consider optimization problem:

$$\min_{w} f(w)$$

$$h_i(w) = 0, \forall 1 \leq i \leq l$$

- Lagrangian:

$$\mathcal{L}(w, \boldsymbol{\beta}) = f(w) + \sum_i \beta_i h_i(w)$$

where β_i's are called Lagrange multipliers

Lagrangian

- Consider optimization problem:

$$\min_{w} f(w)$$

$$h_i(w) = 0, \forall 1 \leq i \leq l$$

- Solved by setting derivatives of Lagrangian to 0

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0; \quad \frac{\partial \mathcal{L}}{\partial \beta_i} = 0$$

Generalized Lagrangian

- Consider optimization problem:

$$\min_{w}\ f(w)$$

$$g_i(w) \leq 0, \forall 1 \leq i \leq k$$

$$h_j(w) = 0, \forall 1 \leq j \leq l$$

- Generalized Lagrangian:

$$\mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(w) + \sum_i \alpha_i g_i(w) + \sum_j \beta_j h_j(w)$$

where α_i, β_j's are called Lagrange multipliers

Lagrange Duality

- The primal problem

$$p^* := \min_{w} f(w) = \min_{w} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \geq 0} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

- The dual problem

$$d^* := \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \geq 0} \min_{w} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

- Always true:

$$d^* \leq p^*$$

Lagrange Duality

- Theorem: under proper conditions, there exists $(w^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ such that

$$d^* = \mathcal{L}(w^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) = p^*$$

Moreover, $(w^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ satisfy Karush-Kuhn-Tucker (KKT) conditions:

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0, \qquad \alpha_i g_i(w) = 0$$

$$g_i(w) \leq 0, \; h_j(w) = 0, \qquad \alpha_i \geq 0$$

SVM Optimization

- Optimization (Quadratic Programming):

$$\min_{w,b} \frac{1}{2}||w||^2$$

$$y_i(w^T x_i + b) \geq 1, \forall i$$

- Generalized Lagrangian:

$$\mathcal{L}(w, b, \boldsymbol{\alpha}) = \frac{1}{2}||w||^2 - \sum_i \alpha_i [y_i(w^T x_i + b) - 1]$$

where $\boldsymbol{\alpha}$ is the Lagrange multiplier

SVM Optimization

- KKT conditions:

$$\frac{\partial \mathcal{L}}{\partial w} = 0, \rightarrow w = \sum_i \alpha_i y_i x_i \text{ (1)}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0, \rightarrow 0 = \sum_i \alpha_i y_i \qquad \text{(2)}$$

- Plug into $\mathcal{L}$:

$$\mathcal{L}(w, b, \boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j \quad \text{(3)}$$

combined with $0 = \sum_i \alpha_i y_i \, , \alpha_i \geq 0$

SVM Optimization

- Reduces to dual problem:

$$\mathcal{L}(w, b, \boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\sum_i \alpha_i y_i = 0, \alpha_i \geq 0$$

- Since $w = \sum_i \alpha_i y_i x_i$, we have $w^T x + b = \sum_i \alpha_i y_i x_i^T x + b$

Slides Credit

[1] Subhransu Maji. Linear model in CMPSCI689.
[2] Yingyu Liang. SVM II in COS495