Supervised learning

Loss functions
Optimization framework: ERM principle
Constraints for optimization
Gradient descent formal algo
Generalization
Regularization

=> all done in course
Basic Classification pipeline

Training
- Training Images
- Image Features
- Classifier Training
- Trained Labels
- Deep CNN End-to-end learning

Testing
- Test Image
- Image Features
- Trained Classifier
- Prediction Monkey
Classification pipeline

To summarize:

• Theory: Risk minimization, Regularization, Generalization
• Supervised Learning, Learning from examples: ERM
  – To be explained: training/validation/test sets
• Algos:
  – k Nearest Neighbors
  – (linear/kernel-based) SVM classifiers
  – Learning binary / Multiclass classifiers
  – Neural Nets, Deep architectures
Recognition/classification

1. Introduction
2. Supervised learning
3. SVM classifiers
4. Datasets and evaluation
SVM

Notations:

• Image/Patterns $\mathbf{x} \in \mathbf{X}$

• $\Phi$: function transforming the patterns into feature vectors $\Phi(x)$

• $\langle \cdot, \cdot \rangle$ dot product in the feature space endowed by $\Phi(\cdot)$

• Classes $y = \pm 1$

Early kernel classifiers derived from the perceptron [Rosenblatt58]:

• taking the sign of a linear discriminant function:

$$f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b$$

• Classifiers called $\Phi$-machines
SVM

- Question: how to find/estimate $f$?
  - Feature function $\Phi$ usually hand-chosen for each problem
  - Several $\Phi$ for image processing like BoW
  - $w$ and $b$: parameters to be determined

$$f(x) = \langle w, \Phi(x) \rangle + b$$

- Learning algorithm on a set of training examples:
  $\mathcal{A} = (x_1, y_1) \cdots (x_n, y_n)$
Which hyperplane? w? b?

\[ w \cdot x + b > 0 \]

\[ w \cdot x + b = 0 \]

\[ w \cdot x + b < 0 \]
SVM optimization: maximizing the margin between + and -
Def.: Margin = distance between the hyperplanes $f(x) = 1$ and $f(x) = -1$ (dashed lines in Figure).
Intuitively, a classifier with a larger margin is more robust to fluctuations
Hard Margin, details in course . . .
Final expression for the Hard Margin SVM optimization:

$$\min_{w,b} P(w, b) = \frac{1}{2} \|w\|^2 \quad \text{with} \quad \forall i \quad y_i f(x_i) \geq 1$$
SVM

• Hard Margin: OK if data are linearly separated

• Otherwise: noisy data (in red) disrupt the optim.

• Solution: Soft SVM
SVM: Soft Margin

Introducing the slack variables $\xi_i$, one usually gets rid of the inconvenient max of the loss and rewrite the problem as

$$\min_{w,b} P(w, b) = \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} \xi_i$$

with

$$\begin{cases}
\forall i \quad y_i f(x_i) \geq 1 - \xi_i \\
\forall i \quad \xi_i \geq 0
\end{cases}$$

For very large values of the hyper-parameter $C$, **Hard Margin** case:
- Minimization of $||w||$ (ie margin maximization) under the constraint that all training examples are correctly classified with a loss equal to zero.

Smaller values of $C$ relax this constraint: **Soft Margin** case
- SVMs that produces markedly better results on noisy problems.
SVM learning scheme

Equivalently, minimizing the following objective function in feature space with the hinge loss function:

$$\ell(y_i f(x_i)) = \max(0, 1 - y_i f(x_i))$$

$$\min_{w, b} P(w, b) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \ell(y_i f(x_i))$$

- Regularization
- Margin Maximization
- Data fitting
- Constraint satisfaction
Learning SVMs: Primal/Dual

- In practice: Convex optimization problem
  - Primal optimization: \( f(x) = \langle w, \Phi(x) \rangle + b \)
  - Dual optimization: learning SVMs can be achieved by solving the dual of this convex optimization problem

- Dual (using Lagragian and \( k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \)):
  - For Hard Margin:
    \[
    \max_{\alpha} \mathcal{L}(\alpha) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \quad \text{with} \quad \begin{cases} 
    \sum_i \alpha_i y_i = 0 \\
    0 \leq \alpha_i
  \end{cases}
    \]
  - For Soft Margin:
    \[
    \max_{\alpha} \mathcal{L}(\alpha) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \quad \text{with} \quad \begin{cases} 
    \sum_i \alpha_i y_i = 0 \\
    0 \leq \alpha_i \leq C
  \end{cases}
    \]
SVM optimization

Standard equivalent formulation without enforcing $\alpha_i$ to be positive:

- Optimization on coefficients $\alpha_i$ of the SVM kernel expansion $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) + b$ by defining the dual objective function:

$$D(\alpha) = \sum_i \alpha_i y_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)$$

- and solving the SVM dual Quadratic Programming (QP) problem.

$$\max_{\alpha} D(\alpha) \quad \text{with} \quad \begin{cases} \sum_i \alpha_i = 0 \\ A_i \leq \alpha_i \leq B_i \\ A_i = \min(0, C y_i) \\ B_i = \max(0, C y_i) \end{cases}$$
Classification pipeline

To summarize on SVM:
Support Vector Machines (SVM) defined by three incremental steps:

1. [Vapnik63]: linear classifier / separates the training examples with the **widest** margin => Optimal Hyperplane
Support Vector Machines (SVM) defined by three incremental steps:

1. [Vapnik63]: linear classifier / separates the training examples with the widest margin => Optimal Hyperplane
2. [Guyon93] Optimal Hyperplane built in the feature space induced by a kernel function
Support Vector Machines (SVM) defined by three incremental steps:

1. [Vapnik63]: linear classifier / separates the training examples with the widest margin => Optimal Hyperplane
2. [Guyon93] Optimal Hyperplane built in the feature space induced by a kernel function
3. [Cortes95] soft version: noisy problems addressed by allowing some examples to violate the margin constraint
Appendix: Solving SVM

• Min P or Max D
  => QP (Quadratic programing) family optimization

• Good news: efficient batch numerical algorithms have been developed
to solve the specific SVM QP problem (hinge loss, convex objective,...)

• Some strategies (exploiting specif.):
  – Conjugate Gradient method [Vapnik]
  – Sequential Minimal Optimization (SMO) [platt].

• In both methods successive searches along well chosen directions

• Some famous SVM solvers like SVMLight [Joachims] or SVMTorch
  propose to use decomposition algorithms to define such directions
• SVMstruct (for structured outputs)
• State-of-the-art implementation of SMO: [libsvm] => used in tutorials
• LibLinear bib for primal optim (with MATLAB)
SMO algo for SVM optimization

1. Set $\alpha \leftarrow 0$ and compute the initial gradient $g$ of $D(\alpha)$

2. Choose a $\tau$-violating pair(*) $(i, j)$ Stop if no such pair exists

3. $\lambda \leftarrow \min \left\{ \frac{g_i - g_j}{k_{ii} + k_{jj} - 2k_{ij}}, B_i - \alpha_i, \alpha_j - A_j \right\}$

4. $\alpha_i \leftarrow \alpha_i + \lambda$, $\alpha_j \leftarrow \alpha_j - \lambda$

5. $g_s \leftarrow g_s - \lambda(k_{is} - k_{js}) \quad \forall s \in \{1 \ldots n\}$

6. Return to step 2

(*) pairs in +1/-1 with significant diff of gradients
A ways to easily satisfy the null sum coeff constraint
Classification pipeline

- **Image**
- **Feature extraction**
- **Local descriptors**
- **Feature coding**
- **Visual codes**
- **Pooling**
- **Image signature**
- **SVM**
- **Class label**