Review 2: Loss minimization, SVM and Logistic Regression

Dan Roth
Department of Computer Science
University of Illinois at Urbana-Champaign

Augmented and modified by Vivek Srikumar
Recall: Hypothesis space

- A **hypothesis space** is the set of possible functions we consider.

- Instead choose a hypothesis space that is smaller than the space of all functions.
  - Only simple conjunctions (with four variables, there are only 16 conjunctions without negations).
  - m-of-n rules: Pick a set of n variables. At least m of them must be true.
  - Linear functions.

- How do we pick a hypothesis space?
  - Using some background knowledge (or by guessing).
Perceptron algorithm

Given a training set \( D = \{(x,y)\}, \ x \in \mathbb{R}^n, \ y \in \{-1,1\} \)

1. Initialize \( w = 0 \in \mathbb{R}^n \)

2. For epoch = 1 … T:
   1. For each training example \((x,y)\) \(\in D\):
      1. Predict \( y' = \text{sgn}(w^T x) \)
      2. If \( y \neq y' \), update \( w \leftarrow w + y x \)

3. Return \( w \)

**Prediction:** \( \text{sgn}(w^T x) \)
Where are we?

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Support vector machines
5. Learning as optimization
6. Logistic Regression
What is the Perceptron algorithm doing?

- Mistake-bound on the training set

- What about future examples? Can we say something about them?

- Can we say anything about the future?
Recall: Margin

- The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
Which line is a better choice? Why?
Which line is a better choice? Why?

A new example, not from the training set might be misclassified if the margin is smaller.
Maximal margin and generalization

• Larger margin gives better generalization

• What we have seen so far: Learning = make as few errors as possible on the training set

• Maximizing margin => fewer errors on future examples
  – This idea forms the basis of many learning algorithms
    • SVM, voted perceptron, AdaBoost,...
Maximizing margin

• Margin = distance of the closest point from the hyperplane

\[ \gamma = \min_{\langle x, y \rangle} \frac{yw^T x}{\|w\|} \]

• We want \( \max_w \gamma \)
Recall: The geometry of a linear classifier

\[ \text{sgn}(b + w_1 x_1 + w_2 x_2) \]

We only care about the sign, not the magnitude.
Maximizing margin

- Margin = distance of the closest point from the hyperplane
  \[ \gamma = \min_{\langle x, y \rangle} \frac{y w^T x}{\|w\|} \]

- We want \( \max_w \gamma \)

- We only care about the sign of \( w \) in the end and not the magnitude
  - Set the activation of the closest point to be 1 and allow \( w \) to adjust itself
  - Sometimes called the *functional margin*

\( \max_w \gamma \) is equivalent to \( \min_w \|w\| \) in this setting
Max-margin classifiers

- Learning a classifier:
  \[ \min \|w\| \text{ such that the activation of the closest point is } 1 \]

- Learning problem:
  \[
  \begin{align*}
    \min_{w} & \quad \frac{1}{2} w^T w \\
    \text{s.t.} & \quad \forall i, \quad y_i w^T x_i \geq 1
  \end{align*}
  \]

- This is called the "hard" Support Vector Machine

- We will look at solving this optimization problem later
What if the data is not separable?

Hard SVM

\[
\min_w \quad \frac{1}{2} w^T w \\
\text{s.t. } \forall i, \quad y_i w^T x_i \geq 1
\]
What if the data is not separable?

Hard SVM

\[
\min_{\mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w}
\]

s.t. \( \forall i, \ y_i \mathbf{w}^T \mathbf{x}_i \geq 1 \)

• This is a constrained optimization problem

• If the data is not separable, there is no \( \mathbf{w} \) that will classify the data

• Infeasible problem, no solution!
Dealing with non-separable data

Key idea: Allow some examples to “break into the margin”

This separator has a large enough margin that it should generalize well. So, while computing margin, ignore the examples that make the margin smaller or the data inseparable.
Soft SVM

• Hard SVM:

\[
\min_w \frac{1}{2} w^T w \\
\text{s.t. } \forall i, \quad y_i w^T x_i \geq 1
\]

Maximize margin
Every example has an functional margin of at least 1

• Introduce one slack variable \( \xi_i \) per example and require \( y_i w^T x_i \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)

• New optimization problem for learning

\[
\min_{w, \xi} \frac{1}{2} w^T w + C \sum_i \xi_i \\
\text{s.t. } \forall i, \quad y_i w^T x_i \geq 1 - \xi_i \\
\forall i, \quad \xi_i \geq 0.
\]
Soft SVM

• Hard SVM:

\[
\min_w \frac{1}{2} w^T w \\
\text{s.t. } \forall i, \quad y_i w^T x_i \geq 1
\]

Maximize margin
Every example is at least at a distance 1 from the hyperplane

• Introduce one slack variable \( \xi_i \) per example and require \( y_i w^T x_i \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)

• Soft SVM learning:

\[
\min_{w,\xi} \frac{1}{2} w^T w + C \sum_i \xi_i \\
\text{s.t. } \forall i, \quad y_i w^T x_i \geq 1 - \xi_i\\n\forall i, \quad \xi_i \geq 0.
\]

Maximize margin
Tradeoff between the two terms
Minimize total slack (i.e. allow as few examples as possible to violate the margin)
Soft SVM

\[
\min_{w, \xi} \frac{1}{2} w^T w + C \sum_i \xi_i \\
\text{s.t. } \forall i, \ y_i w^T x_i \geq 1 - \xi_i \\
\forall i, \ \xi_i \geq 0.
\]

• Eliminate the slack variables to rewrite this as

\[
\min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i w^T x_i)
\]

• This form is more interpretable
Maximizing margin and minimizing loss

\[
\min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i w^T x_i)
\]

• Three cases
  – An example is correctly classified: penalty = 0
  – An example is incorrectly classified: penalty = 1 – y_i w^T x_i
  – An example is correctly classified but within the margin: penalty = 1 – y_i w^T x_i

• This is the hinge loss function
The Hinge Loss

Loss vs. $yw^T x$
SVM objective function

\[
\min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i w^T x_i)
\]

**Regularization term:**
- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

**Empirical Loss:**
- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

A hyper-parameter that controls the tradeoff between a large margin and a small hinge-loss
1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Support vector machines
5. Learning as optimization
6. Logistic Regression
Computational Learning Theory

• Studies theoretical issues about the importance of representation, sample complexity (how many examples are enough), computational complexity (how fast can I learn)
  – Led to algorithms such as support vector machine and boosting

• No assumptions about the distribution of examples
  – But assume that both training and test examples come from the same distribution

• Provides bounds that depend on the size of the hypothesis class
Learning as loss minimization

- Collect some annotated data. More is generally better
- Pick a hypothesis class (also called model)
  - Eg: binominal, linear classifiers
  - Also, decide on how to impose a preference over hypotheses
- Choose a **loss function**
  - Eg: negative log-likelihood
  - Decide on how to penalize incorrect decisions
- Minimize the loss
  - Eg: Set derivative to zero, more complex algorithm
Learning as loss minimization: The setup

- Examples \(<x, y>\) are created from some unknown distribution \(P\)
- Identify a hypothesis class \(H\)
- Define penalty for incorrect predictions:
  - The loss function \(L\)
- Learning: Pick a function \(f \in H\) to minimize expected loss
  \[
  \min_{H} \mathbb{E}_P[L]
  \]
- Use samples from \(P\) to estimate expectation:
  - The training set \(D = \{<x, y>\}\)
  - “Empirical risk minimization”
  \[
  \min_{f \in H} \sum_{D} L(y, f(x))
  \]
Regularized loss minimization: Logistic regression

• Learning: \( \min_{f \in H} \text{regularizer}(f) + C \sum_i L(y_i, f(x_i)) \)

• With linear classifiers: \( \min_{w} \frac{1}{2} w^T w + C \sum_i L(y_i, x_i, w) \)

• What is a loss function?
  – Loss functions should penalize mistakes
  – We are minimizing average loss over the training data

• What is the ideal loss function for classification?
The 0-1 loss

- Penalize classification mistakes between true label $y$ and prediction $y'$

$$L_{0-1}(y, y') = \begin{cases} 1 & \text{if } y \neq y', \\ 0 & \text{if } y = y'. \end{cases}$$

- For linear classifiers, the prediction $y' = \text{sgn}(w^T x)$

- Mistake if $y \ w^T x \leq 0$

$$L_{0-1}(y, x, w) = \begin{cases} 1 & \text{if } y \ w^T x \leq 0, \\ 0 & \text{otherwise}. \end{cases}$$

- Minimizing 0-1 loss is intractable. Need surrogates
Loss functions

Typically plotted as a function of $yw^Tx$
Support Vector Machines: Summary

- SVM = linear classifier + regularization
- Recall that perceptron did not have regularization
- Ideally, we would like to minimize 0-1 loss, but cannot
- SVM minimizes hinge loss
  - Variants exist
- Will not cover
  - Dual formulation, support vectors, kernels
Solving the SVM optimization problem

$$\min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i w^T x_i)$$

This function is convex in $w$
Convex functions

• A function $f$ is **convex** if for every $u, v$ in the domain, and for every $a \in [0,1]$ we have
  $$f(a\ u + (1-a)\ v) \leq a\ f(u) + (1-a)\ f(v)$$

• The necessary condition for $w^*$ to be a minimum for a function $f$: $df(w^*)/dw = 0$

• For convex functions, this is both necessary and sufficient
Solving the SVM optimization problem

$$\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$$

This function is convex in $\mathbf{w}$

- This is a quadratic optimization problem because the objective is quadratic
- Earlier methods: Used techniques from Quadratic Programming
  - Very slow
- No constraints, can use *gradient descent*
  - Still very slow!
Gradient descent for SVM

• Gradient descent algorithm to minimize a function $f(w)$:
  – Initialize solution $w$
  – Repeat
    • Find the gradient of $f$ at $w$: $\nabla f$
    • Set $w \leftarrow w - r \nabla f$

• Gradient of the SVM objective requires summing over the entire training set
  – Slow
  – Does not really scale
Stochastic gradient descent for SVM

Given a training set $D = \{(x,y)\}, x \in \mathbb{R}^n, y \in \{-1,1\}$

1. Initialize $w = 0 \in \mathbb{R}^n$

2. For epoch = 1 … T:
   1. For each training example $(x, y) \in D$:
      1. Treat $(x,y)$ as a full dataset and take the derivative of the SVM objective at the current $w$ to be $\nabla$
      2. $w \leftarrow w - r \nabla$

3. Return $w$

What is the gradient of the hinge loss with respect to $w$?
(The hinge loss is not a differentiable function!)
Stochastic sub-gradient descent for SVM

Given a training set \( D = \{(x,y)\}, x \in \mathbb{R}^n, y \in \{-1,1\} \)

1. Initialize \( w = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:
   1. For each training example \((x, y)\) \in D:
      - If \( y \cdot w^T x \leq 1 \),
        - then \( w \leftarrow (1-r)w + r \cdot C \cdot y \cdot x \)
      - else \( w \leftarrow (1-r) \cdot w \)

3. Return \( w \)

**Prediction:** \( \text{sgn}(w^T x) \)

**Compare to the perceptron algorithm!**

Perceptron update:
If \( y \cdot w^T x \leq 0 \), update \( w \leftarrow w + r \cdot y \cdot x \)
SVM summary from optimization perspective

- Minimize regularized hinge loss
- Solve using stochastic gradient descent
  - Very fast, run time does not depend on number of examples
  - Compare with Perceptron algorithm: Perceptron does not maximize margin width
    - Perceptron variants can force a margin
    - Convergence criterion is an issue; can be too aggressive in the beginning and get to a reasonably good solution fast; but convergence is slow for very accurate weight vector
- Another successful optimization algorithm:
  - Dual coordinate descent, implemented in liblinear

Questions?
Where are we?

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Support vector machine
5. Learning as optimization
6. Logistic Regression
Regularized loss minimization: Logistic regression

- Learning: \( \min_{f \in H} \text{regularizer}(f) + C \sum_i L(y_i, f(x_i)) \)

- With linear classifiers: \( \min_w \frac{1}{2} w^T w + C \sum_i L(y_i, x_i, w) \)

- SVM uses the hinge loss

- Another loss function: The logistic loss

\[
L_{\text{logistic}}(y, x, w) = \log(1 + e^{-y w^T x})
\]
Loss functions

Typically plotted as a function of $yw^Tx$

- $0-1$ loss
- Hinge loss
- Logistic

SVM
Logistic regression

Smooth, differentiable
The probabilistic interpretation

Suppose we believe that the labels are generated using the following probability distribution:

\[
P(y = 1 | x, w) = \frac{e^{w^T x}}{1 + e^{w^T x}} = \frac{1}{1 + e^{-w^T x}}
\]

\[
P(y = -1 | x, w) = \frac{1}{1 + e^{w^T x}}
\]

\[
P(y | x, w) = \frac{1}{1 + \exp(-y w^T x)}
\]

Predict label = 1 if \( P(1 | x, w) > P(-1 | x, w) \)

– Equivalent to predicting 1 if \( w^T x \geq 0 \)

– Why?
The probabilistic interpretation

Suppose we believe that the labels are generated using the following probability distribution:

\[
P(y = 1|x, w) = \frac{e^{w^T x}}{1 + e^{w^T x}} = \frac{1}{1 + e^{-w^T x}}
\]

\[
P(y = -1|x, w) = \frac{1}{1 + e^{w^T x}}
\]

\[P(y|x, w) = \frac{1}{1 + \exp(-yw^T x)}\]

What is the log-likelihood of seeing a dataset \(D = \{<x_i, y_i>\}\) given a weight vector \(w\)?

\[
\log P(D|w) = - \sum_i \log (1 + \exp(-yw^T x))
\]
Prior distribution over the weight vectors

A prior balances the tradeoff between the likelihood of the data and existing belief about the parameters

- Suppose each weight $w_i$ is drawn independently from the normal distribution centered at zero with variance $\sigma^2$
  - Bias towards smaller weights
  \[
P(w_i) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{w_i^2}{2\sigma^2}\right)
  \]
- Probability of the entire weight vector:
  \[
  \log P(w) = -\frac{1}{2\sigma^2} \mathbf{w}^T \mathbf{w} + \text{constant terms}
  \]
Regularized logistic regression

What is the probability of seeing a dataset $D = \{<x_i, y_i>\}$ and a weight vector $w$?

$$P(w \mid D) \propto P(D, w) = P(D \mid w) P(w)$$

Learning: Find weight vector by maximizing the posterior distribution $P(w \mid D)$

$$\log P(D, w) = -\frac{1}{2\sigma^2} w^T w - \sum_i \log (1 + \exp(-y_i w^T x))$$

Exercise: Derive the stochastic gradient descent algorithm for logistic regression.

Once again, regularized loss minimization! This is the Bayesian interpretation of regularization.
Regularized loss minimization

- Learning objective for both SVM, logistic regression:
  
  \[
  \text{loss over training data + regularizer}
  \]

- Different loss functions
  - Hinge loss vs. logistic loss
- Same regularizer, but different interpretation
  - Margin vs prior
- Hyper-parameter controls tradeoff between the loss and regularizer
- Other regularizers/loss functions also possible

Questions?
Review of binary classification

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Support vector machine
5. Learning as optimization
6. Logistic Regression

Questions?