SVMs: Non-Separable Data, Convex Surrogate Loss, Multi-Class Classification, Kernels

Karl Stratos

June 21, 2018

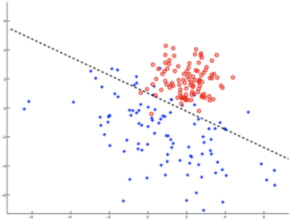
Tangent: Some Loose Ends in Logistic Regression

- Polynomial feature expansion in logistic regression
- Regularization in logistic regression
- Classification metrics

Increasing the Complexity of Logistic Regression

- The predicted probability of a logistic regressor is $p(1|\boldsymbol{x}, \boldsymbol{w}, w_0) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + w_0)$.

- Linear decision boundary



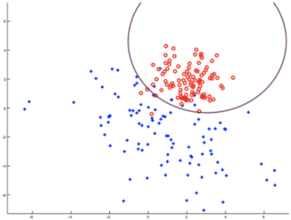
$$\boldsymbol{w} \cdot \boldsymbol{x} + w_0 = 0 \qquad \Leftrightarrow \qquad p(1|\cdot) = \frac{1}{2}$$

Polynomial Feature Expansion

- We can use the same polynomial feature expansion that we used in linear regression.
- For instance, with $d = 2$ dimensions

$$p(1|\boldsymbol{x}, \boldsymbol{w}, w_0) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$$

- Nonlinear decision boundary



$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 = 0 \quad \Leftrightarrow \quad p(1|\cdot) = \frac{1}{2}$$

Tangent: Some Loose Ends in Logistic Regression

- Polynomial feature expansion in logistic regression
- Regularization in logistic regression
- Classification metrics

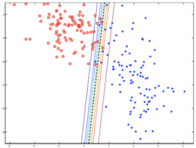
Regularization in Logistic Regression

- Same idea as in linear regression: penalize the squared l_2 or l_1 norm of the model parameter to prevent the model from becoming too "confident" about the training data.

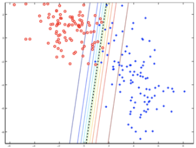
- Squared l_2 regularization with hyperparameter σ^2

$$-\sum_i \log p(y^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{w}) + \frac{1}{2\sigma^2}\,||\boldsymbol{w}||^2$$

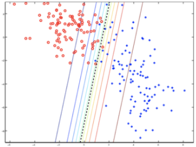
ML



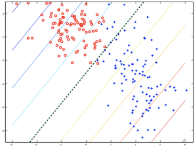
$\sigma^2 = 1$



$\sigma^2 = 0.1$



$\sigma^2 = 0.01$



Tangent: Some Loose Ends in Logistic Regression

- Polynomial feature expansion in logistic regression
- Regularization in logistic regression
- Classification metrics

Label Imbalance Problem

- Suppose most of the labels are $y = 0$, say 99.9% of the time.
- In this scenario, classification accuracy is not a useful metric.

$$\frac{tp + tn}{tp + fp + tn + fn}$$

  - Just by guessing 0 all the time, we get accuracy 99.9%!

Label Imbalance Problem

- Suppose most of the labels are $y = 0$, say 99.9% of the time.

- In this scenario, classification accuracy is not a useful metric.

$$\frac{tp + tn}{tp + fp + tn + fn}$$

  - Just by guessing 0 all the time, we get accuracy 99.9%!

- Consider other metrics, such as
  - **Precision**: Out of your $y = 1$ predictions, how many were actually 1?

$$\frac{tp}{tp + fp}$$

  - **Recall**: Out of points that are labeled 1, how many did you label as $y = 1$?

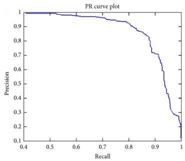
$$\frac{tp}{tp + fn}$$

More Metrics

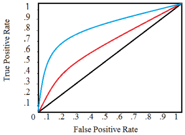
- **F1**: Harmonic mean of precision and recall

$$2\frac{precision \cdot recall}{precision + recall}$$

- **PR curve**: change threshold and plot precision/recall



- **ROC curve**: change threshold and plot false/true positive rates



Transition Slide

Back to SVMs

Review: Basic Support Vector Machines (SVMs)

- Training data: $S = \left\{(\boldsymbol{x}^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where $y^{(i)} \in \{\pm 1\}$ is a *binary* label of $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$.

- S is assumed to be *linearly separable*: there exists $\boldsymbol{w}$ such that $y^{(i)}\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} > 0$ for all $i = 1 \dots n$.

- Find a separator that maximizes the **margin** on S:

Review: Basic Support Vector Machines (SVMs)

- Training data: $S = \left\{(\boldsymbol{x}^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where $y^{(i)} \in \{\pm 1\}$ is a *binary* label of $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$.

- S is assumed to be *linearly separable*: there exists $\boldsymbol{w}$ such that $y^{(i)}\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} > 0$ for all $i = 1 \dots n$.

- Find a separator that maximizes the **margin** on S:

$$
\begin{aligned}
\boldsymbol{w}_S^{\text{svm}} &:= \operatorname*{arg\,max}_{\boldsymbol{w} \in \mathbb{R}^d} \; \min_{i=1}^{n} \; y^{(i)} \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \cdot \boldsymbol{x}^{(i)} \\
&\overset{\text{wlog}}{\equiv} \operatorname*{arg\,max}_{\boldsymbol{w} \in \mathbb{R}^d:\, ||\boldsymbol{w}||=1} \; \min_{i=1}^{n} \; y^{(i)} \boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \\
&\propto \operatorname*{arg\,min}_{\substack{\boldsymbol{w} \in \mathbb{R}^d: \\ y^{(i)}\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \geq 1 \; \forall i}} \; ||\boldsymbol{w}||^2
\end{aligned}
$$

... Equivalently, find a separator with the minimum l_2 norm.

Review: Inner Product Formulation

Using the **representer theorem**

$$\exists \beta_1 \dots \beta_n \in \mathbb{R} : \quad \boldsymbol{w}_S^{\text{svm}} = \sum_{i=1}^{n} \beta_i \boldsymbol{x}^{(i)}$$

we can convert the original problem into an equivalent problem

$$\min_{\beta_1 \dots \beta_n \in \mathbb{R}} \sum_{i,j=1}^{n} \beta_i \beta_j \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$$

$$\text{subject to } y^{(i)} \sum_{j=1}^{n} \beta_j \boldsymbol{x}^{(j)} \cdot \boldsymbol{x}^{(i)} \geq 1 \qquad \forall i = 1 \dots n$$

where the only information from data we need for training is the inner product between input points.

- Likewise at test time: $\boldsymbol{w}_S^{\text{svm}} \cdot \boldsymbol{x} = \sum_{i=1:\ \beta_i \neq 0}^{n} \beta_i \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}$
- Allows for the use of **kernels** (later).

Today

- How to handle **non-separable data**
  - Connection to a convex surrogate loss on the 0-1 loss
- How to handle **multi-class classification**
- The **kernel trick**

Overview

Non-Separable Data

Multi-Class Classification

Kernel Trick

Introduce Slack Variables

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} ||\boldsymbol{w}||^2$$

$$\text{subject to } y^{(i)} \boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \geq 1 \qquad \forall i = 1 \dots n$$

$$\Downarrow$$

$$\min_{\boldsymbol{w} \in \mathbb{R}^d,\, \xi_1 \dots \xi_n \in \mathbb{R}} ||\boldsymbol{w}||^2 + \sum_{i=1}^{n} \xi_i$$

$$\text{subject to } y^{(i)} \boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \geq 1 - \xi_i \qquad \forall i = 1 \dots n$$

$$\xi_i \geq 0 \qquad \forall i = 1 \dots n$$

Unconstrained Formulation

"Soft" SVM solution

$$\boldsymbol{w}_S^{\text{soft}}, \xi_1^* \dots \xi_n^* := \underset{\boldsymbol{w} \in \mathbb{R}^d,\ \xi_1 \dots \xi_n \in \mathbb{R}}{\arg\min} ||\boldsymbol{w}||^2 + \sum_{i=1}^{n} \xi_i$$

with constraints $\xi_i \geq \max\left(0, 1 - y^{(i)} \boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)$ for all $i = 1 \dots n$.

Note that $\xi_i^* = \max\left(0, 1 - y^{(i)} \boldsymbol{w}_S^{\text{soft}} \cdot \boldsymbol{x}^{(i)}\right)$, so

$$\boldsymbol{w}_S^{\text{soft}} = \underset{\boldsymbol{w} \in \mathbb{R}^d}{\arg\min}\ ||\boldsymbol{w}||^2 + \sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)$$

No constraints :) Convex but not differentiable, can still be optimized by subgradient descent

Soft SVMs as Empirical Risk Minimization

- What we really want: minimize the 0-1 loss

$$\boldsymbol{w}_S^* = \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \sum_{i=1}^{n} \left[\left[y^{(i)} \boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \leq 0 \right]\right]$$

where $[[\tau]]$ is 1 if τ is true and 0 otherwise

- Difficult to optimize (neither convex nor differentiable)

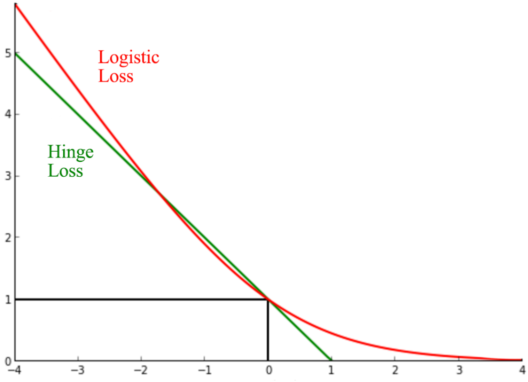
- Instead minimize the **hinge loss**

$$\boldsymbol{w}_S^* = \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \sum_{i=1}^{n} \underbrace{\max\left(0, 1 - y^{(i)} \boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)}_{\text{hinge}(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)})}$$

which is a convex upper bound on the 0-1 loss

A Big Picture of Binary Classification

Both logistic regression and soft SVMs are l_2-regularized minimization of the 0-1 loss by convex surrogates.



Generalized Representer Theorem

Claim. Let $l : \mathbb{R}^n \to [0, \infty)$ be any function and define

$$\boldsymbol{w}^* = \operatorname*{arg\,min}_{\boldsymbol{w} \in \mathbb{R}^d} ||\boldsymbol{w}||^2 + l\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{w} \cdot \boldsymbol{x}^{(n)}\right)$$

Then $\boldsymbol{w}^* = \sum_{i=1}^{n} \beta_i \boldsymbol{x}^{(i)}$ for some $\beta_1 \ldots \beta_n \in \mathbb{R}$.

Proof. Same as in the hard SVM case

Thus we can simiarly derive an inner product formulation of the soft SVM solution (will come back to this later):

$$\boldsymbol{w}_S^{\text{soft}} = \operatorname*{arg\,min}_{\boldsymbol{w} \in \mathbb{R}^d} \; ||\boldsymbol{w}||^2 + \underbrace{\sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)}_{l\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{w} \cdot \boldsymbol{x}^{(n)}\right)}$$

Overview

Non-Separable Data

Multi-Class Classification

Kernel Trick

One-Vs-All

- Training data: $S = \left\{(\boldsymbol{x}^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where $y^{(i)} \in \{1 \ldots m\}$ is the label of $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$.
- Parameters: $\boldsymbol{w}^y \in \mathbb{R}^d$ for each $y \in \{1 \ldots m\}$
- "One-vs-all" soft SVM objective: optimize

$$\min_{\boldsymbol{w} \in \mathbb{R}^d,\, \xi_1 \ldots \xi_n \in \mathbb{R}} ||\boldsymbol{w}||^2 + \sum_{i=1}^{n} \xi_i$$

such that $\xi_i \geq 0$ for all $i = 1 \ldots n$ and

$$\boldsymbol{w}^{y^{(i)}} \cdot \boldsymbol{x}^{(i)} - \boldsymbol{w}^{y} \cdot \boldsymbol{x}^{(i)} \geq 1 - \xi_i$$

for all $i = 1 \ldots n$ and $y \in \{1 \ldots m\}$ such that $y \neq y^{(i)}$

Unconstrained Formulation

$$\boldsymbol{w}_S^{\text{one}-\text{vs}-\text{all}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\arg\min}\ ||\boldsymbol{w}||^2 + \sum_{i=1}^{n} \sum_{y\in\{1\ldots m\}:\ y\neq y^{(i)}} \max\left(0, 1 + \boldsymbol{w}^{y} \cdot \boldsymbol{x}^{(i)} - \boldsymbol{w}^{y^{(i)}} \cdot \boldsymbol{x}^{(i)}\right)$$

One-Vs-One

- "One-vs-one" soft SVM objective: optimize

$$\min_{\boldsymbol{w}\in\mathbb{R}^d,\ \xi_1\ldots\xi_n\in\mathbb{R}} ||\boldsymbol{w}||^2 + \sum_{i=1}^{n} \xi_i$$

such that $\xi_i \geq 0$ for all $i = 1 \ldots n$ and

$$\boldsymbol{w}^{y^{(i)}} \cdot \boldsymbol{x}^{(i)} - \max_{y\in\{1\ldots m\}: y\neq y^{(i)}} \boldsymbol{w}^{y} \cdot \boldsymbol{x}^{(i)} \geq 1 - \xi_i$$

for all $i = 1 \ldots n$

- Unconstrained Formulation

$$\boldsymbol{w}_S^{\text{one}-\text{vs}-\text{all}} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} ||\boldsymbol{w}||^2 + \sum_{i=1}^{n} \max\left(0, 1 + \max_{y\in\{1\ldots m\}:\ y\neq y^{(i)}} \boldsymbol{w}^{y} \cdot \boldsymbol{x}^{(i)} - \boldsymbol{w}^{y^{(i)}} \cdot \boldsymbol{x}^{(i)}\right)$$

Overview

Non-Separable Data

Multi-Class Classification

Kernel Trick

Inner Product Formulation of Soft SVM (Binary)

- $G \in \mathbb{R}^{n \times n}$: a symmetric matrix with $G_{i,j} = \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$ (i.e., the Gram matrix).

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^n} \boldsymbol{\beta}^\top G \boldsymbol{\beta} + \sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \boldsymbol{\beta}^\top G_i\right)$$

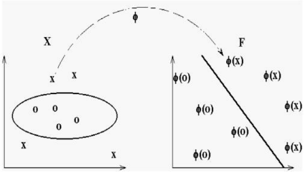
$$\boldsymbol{w}_S^{\text{soft}} = \sum_{i=1}^{n} \boldsymbol{\beta}_i^* \boldsymbol{x}^{(i)}$$

- User manual
  1. Calculate $G_{i,j} = \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$ for every $i, j = 1 \ldots n$.
  2. Find $\boldsymbol{\beta}^*$ using G (e.g., by subgradient descent).
  3. Test time: given a new point $\boldsymbol{x}$ to classify, return

$$\text{sign}\left(\sum_{i=1:\boldsymbol{\beta}_i^* \neq 0}^{n} \boldsymbol{\beta}_i^* \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}\right)$$

Recall: Polynomial Feature Expansion

- Idea: transform input by $\phi : \mathbb{R}^d \to \mathbb{R}^D$ to allow the linear model to better fit the data.



- Example: expansion by a degree $p = 2$ polynomial with bias $c = 1$

$$\phi^{\text{poly}(2,1)}(x_1 \dots x_d) = \left(\left(x_i^2\right)_i, \left(\sqrt{2} x_i x_j\right)_{i<j}, \left(\sqrt{2} x_i\right)_i, 1 \right)$$

- **Computationally expensive**: time to calculate feature expansion $O(d^p)$ exponential in p

But Computing Inner Product is Easy!

- Inner product between two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in the feature space

$$\phi^{\text{poly}(2,1)}(\boldsymbol{x}) \cdot \phi^{\text{poly}(2,1)}(\boldsymbol{y}) = \sum_{i,j} x_i x_j y_i y_j + 2 \sum_i x_i y_i + 1$$
$$= \left(\boldsymbol{x}^\top \boldsymbol{y} + 1\right)^2$$

- Instead of computing $O(d^2)$ terms in each $\phi^{\text{poly}(2,1)}(\boldsymbol{x})$ and $\phi^{\text{poly}(2,1)}(\boldsymbol{y})$ and then taking a dot product, we can just
  1. Compute $z = \boldsymbol{x}^\top \boldsymbol{y}$ ($O(d)$-time operation)
  2. Square $z + 1$ ($O(1)$-time operation)

Kernel Trick

- Idea: when all we need is inner product, we can do "implicit" feature expansion by a kernel function without ever computing the explicit feature expansion

- **Kernel function** $K(\boldsymbol{x}, \boldsymbol{y})$ is any function that defines pairwise similarity between two data points such that

$$K(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{y})$$

  for *some* input mapping $\phi : \mathbb{R}^d \rightarrow ?$

- Applicable beyond SVMs (e.g., kernel PCA)

Degree-p Polynomial Kernel

Parameters: degree p, bias c

$$K^{\mathrm{poly}(p,c)}(\boldsymbol{x}, \boldsymbol{y}) = \left(\boldsymbol{x}^\top \boldsymbol{y} + c\right)^p$$

We saw that the underlying feature expansion is some degree p polynomial

Radial Basis Function (RBF) Kernel

Parameter: $\sigma^2 > 0$

$$K^{\mathrm{RBF}(\sigma^2)}(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\left\|\boldsymbol{x} - \boldsymbol{y}\right\|^2}{2\sigma^2}\right)$$

What is the underlying feature expansion? For $\sigma^2 = 1$,

$$\begin{aligned} K^{\mathrm{RBF}(\sigma^2)}(\boldsymbol{x}, \boldsymbol{y}) &= C \sum_{p=0}^{\infty} \frac{1}{p!} \left(\boldsymbol{x}^\top \boldsymbol{y}\right)^p \\ &= C \sum_{p=0}^{\infty} \frac{1}{p!} \phi^{\mathrm{poly}(p,0)}(\boldsymbol{x}) \cdot \phi^{\mathrm{poly}(p,0)}(\boldsymbol{y}) \end{aligned}$$

The underlying feature space is *infinite-dimensional*.

Summary of Kernel Trick for Soft SVM

- Before ("linear kernel")
  1. Calculate $G_{i,j} = \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$ for every $i, j = 1 \ldots n$.
  2. Find $\boldsymbol{\beta}^*$ using G (e.g., by subgradient descent).
  3. Test time: given a new point $\boldsymbol{x}$ to classify, return

$$\text{sign}\left(\sum_{i=1:\beta_i^* \neq 0}^{n} \beta_i^* \boldsymbol{x}^{(i)} \cdot \boldsymbol{x} \right)$$

- Choose some kernel $K(\boldsymbol{x}, \boldsymbol{y})$.
  1. Calculate $G_{i,j} = K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right)$ for every $i, j = 1 \ldots n$.
  2. Find $\boldsymbol{\beta}^*$ using G (e.g., by subgradient descent).
  3. Test time: given a new point $\boldsymbol{x}$ to classify, return

$$\text{sign}\left(\sum_{i=1:\beta_i^* \neq 0}^{n} \beta_i^* K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}\right) \right)$$

Aside: Kernel Approximation

- Kernel trick is clever but requires the inner product formulation. This requires storing the $n \times n$ Gram matrix: not scalable.

- Kernel approximation: approximate the implicit feature expansion defined under a kernel, and use that expansion directly

- Rahimi and Recht (2007): $z(\boldsymbol{x}) \in \mathbb{R}^N$ where $z_i(\boldsymbol{x}) := \sqrt{2/N} \cos(\mu_i \cdot \boldsymbol{x} + b_i)$ given by $\mu_i \sim \mathcal{N}(0, I_d)$ and $b_i \sim \mathcal{U}(0, 2\pi)$

$$\mathbf{E}[z(\boldsymbol{x}) \cdot z(\boldsymbol{y})] = K^{\mathrm{RBF}(1)}(\boldsymbol{x}, \boldsymbol{y})$$

  Use $z(\boldsymbol{x}) \in \mathbb{R}^N$ directly (no kernel trick)

Summary

- Soft SVM = hard SVM + slack variables to handle non-separable data
  - A unifying framework for SVMs and logistic regression: convex surrogate loss on the 0-1 loss
- SVMs can be naturally extended to handle multi-class classification
- Kernel trick: when training and inference only depend on inner product between data points, we can replace the inner product with a kernel function and perform implicit feature expansion