COMPSCI 689
Lecture 7: Regularization and SVMs

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

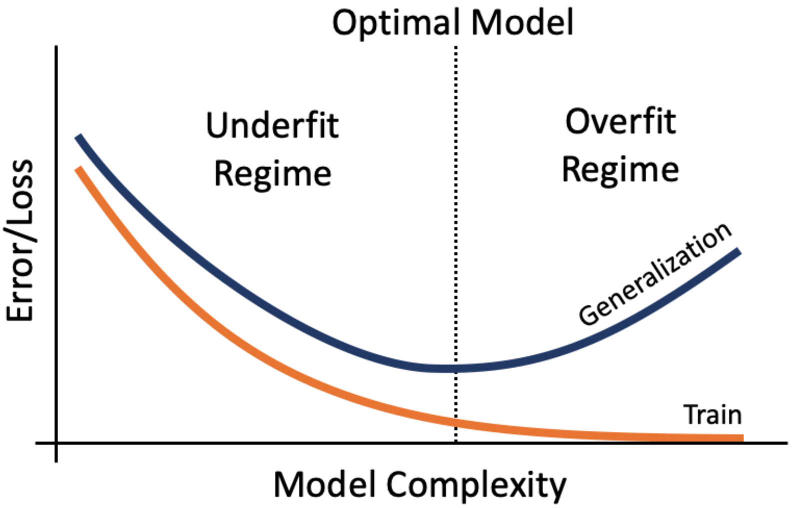
Basis Function Expansion

- A simple solution to the linearity problem is to apply a set of functions $\phi_1,...,\phi_K$ to the raw feature vector $\mathbf{x}$ to map it in to a new feature space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$$

- This is called a *basis function expansion* since $K > D$ in general. This requires that we know the functions $\phi_1,...,\phi_K$ that we want to apply in advance.
- We then define a linear model in this new feature space using a new weight vector $\mathbf{w} \in \mathbb{R}^K$.
- In the regression setting, we obtain the model $\hat{y} = \phi(\mathbf{x})\theta$.
- In the classification setting, we obtain the model $\hat{y} = \text{sign}(\phi(\mathbf{x})\theta)$.

Overfitting and Underfitting



Controlling Complexity

- There are two basic ways to control the complexity of a basis expanded model.
- The first approach is to control the number and/or complexity of the terms used in the basis expansion.
- The second approach is to control the magnitude of the weights.

Controlling Weights

- The second approach to controlling the complexity of models is to control the magnitude of their weights.
- For many models, smaller magnitude weights correspond to smoother functions.
- Controlling model complexity via weight magnitudes gives us a complementary approach to selecting from a discrete set of models of different complexity.

Regularization

- One way to control the magnitude of the weights in a basis expanded linear model (and many other models) is to add a term to the optimization objective function that penalizes solutions where the weights have large magnitudes.

- The two most common ways to measure the magnitude of the weights are via the ℓ_1 norm $\|\mathbf{w}\|_1 = \sum_{d=1}^{D} |\mathbf{w}_d|$ and squared ℓ_2 norm of the weights $\|\mathbf{w}\|_2^2 = \sum_{d=1}^{D} \mathbf{w}_d^2$.

- These regularization functions can equivalently be thought of as penalizing the distance (under the corresponding norm) of the weights from 0 (Note that we typically do not regularize the bias term in the model).

- Note that this is a very general approach. It applies to basis expanded linear models as well as most other parametric models.

Regularized Risk Minimization

- When regularization is applied in the expected risk minimization framework, the resulting framework is referred to as *regularized risk minimization* (RRM).
- Below, we show the RRM objective function for the case where the regularization function is based on the squared two norm:

$$\hat{\theta} = \underset{\theta \in \Theta}{\arg\min} \left(\left(\sum_{n=1}^{N} L(y_n, f_\theta(\mathbf{x}_n)) \right) + \lambda \sum_{k=1}^{K} w_k^2 \right)$$

Regularized Risk Minimization

- We can equivalently scale the loss term instead of the regularizer:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} C \left(\sum_{n=1}^{N} L(y_n, f_\theta(\mathbf{x}_n)) \right) + \|\mathbf{w}\|_2^2$$

Regularization Dynamics

$$\hat{\theta} = \underset{\theta \in \Theta}{\arg\min} \left(\sum_{n=1}^{N} L(y_n, f_\theta(\mathbf{x}_n)) \right) + \lambda \sum_{k=1}^{K} w_k^2$$

- When λ is large, the regularization term encourages the weights to be small and they will eventually be driven to 0 as λ goes to infinity.
- As λ goes to 0, we recover ERM and the weights are free to take arbitrary values.

Regularized models

- When we learn the linear regression model with squared loss and the $\|\mathbf{w}\|_2^2$ regularizer, the resulting model is often referred to as *ridge regression*.
- In logistic regression and neural network models, the application of the $\|\mathbf{w}\|_2^2$ regularizer is often referred to as *weight decay* because it contributes a term of $-2\lambda\mathbf{w}$ to a gradient descent-based optimizer.

Selecting the Regularization Hyper-Parameter

- To select a regularization hyper-parameter, we can again apply a Train-Validation procedure.
- It is typical to use an exponential grid of values for λ such as $\lambda \in \{10^{-4}, 10^{-3}, ..., 10^{1}, 10^{2}\}$.
- When running such an experiment with a given range of values for λ, it is important to check that the optimal value over the range is not achieved at the minimum or maximum value tested (if it is, the range needs to be extended and the experiment re-run).
- As noted previously, to get a valid estimate of the generalization performance of the model with the optimal hyper-parameters, it is necessary to run a Train-Validation-Test experiment.

Support Vector Machines

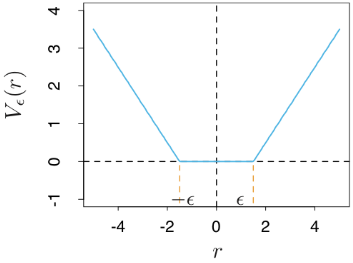
- Support vector machines are a class of supervised learning approaches based on linear models that use non-differentiable loss functions.
- They are learned using regularized risk minimization and can be used with basis expansions.
- The loss functions used have some interesting properties relative to the squared loss and logistic loss.
- There are specific approaches for both regression (SVR) and classification (SVC).

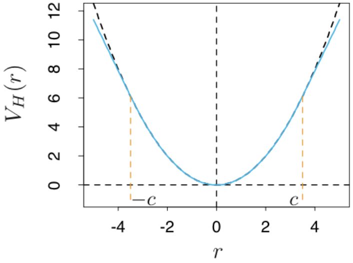
Problems with OLS Linear Regression

- One of the drawbacks of squared loss is that it can be very sensitive to the presence of *outliers*.
- An outlier is a data point with an output value that is much larger or smaller than the output values of other nearby data points.
- Outliers can be caused by errors in the data collection, or by a data generating process that has heavy tails.
- Squared loss prefers to have many small errors instead of a few large errors, and will thus deflect the regression surface toward outliers in order to minimize the MSE.

Regression with Other Losses

Common alternative losses for regression include the epsilon insensitive loss and the Huber loss.





Both of these losses are specifically designed to limit the influence of *outliers* on the model fit.

The specific combination of the epsilon insensitive loss with a squared ℓ_2 norm regularizer is referred to as *support vector regression* (SVR).

Support Vector Regression

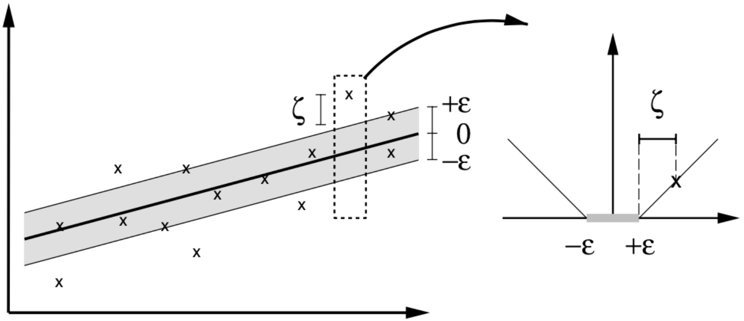
$$f_\theta(\mathbf{x}) = \mathbf{x}\theta$$

$$\hat{\theta} = \arg\min_{\theta} C \sum_{n=1}^{N} L_\epsilon(y_n, f_\theta(\mathbf{x}_n)) + ||\mathbf{w}||_2^2$$

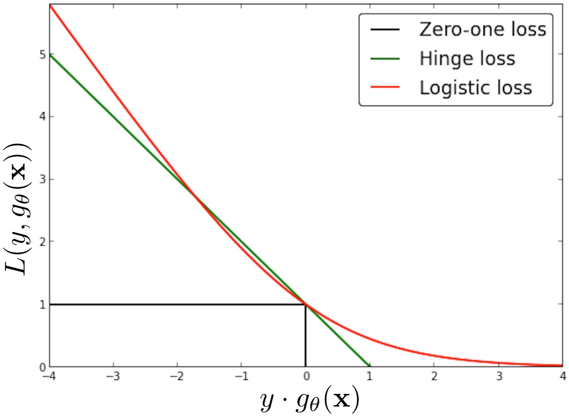
$$L_\epsilon(y, y') = \begin{cases} 0 & \text{... if } |y - y'| < \epsilon \\ |y - y'| - \epsilon & \text{... otherwise} \end{cases}$$

Note that for SVR, the objective function is continuous and convex, but not differentiable, thus our optimization toolkit will need updating to fit this model class.

Support Vector Regression Example



Classification Losses



The specific combination of the hinge loss with a squared ℓ_2 norm regularizer is referred to as a *support vector classifier*.

Hinge Loss

- Like the logistic loss, the hinge loss provides an upper bound on the classification error. The basic structure of the function is also similar to that of the logistic loss:

$$L_h(y, g_\theta(\mathbf{x})) = \max(0, 1 - y g_\theta(\mathbf{x}))$$

- However, while the hinge loss is continuous and convex, it is not differentiable.

Support Vector Classifier

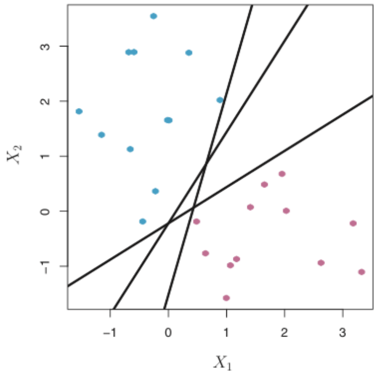
$$f_\theta(\mathbf{x}) = \operatorname{sign}(g_\theta(\mathbf{x}))$$

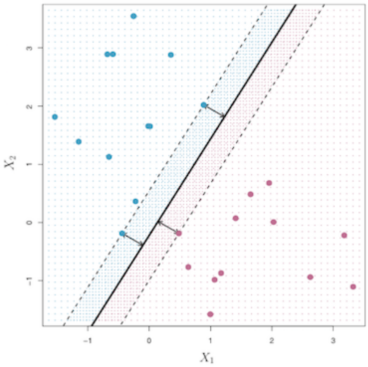
$$g_\theta(\mathbf{x}) = \mathbf{x}\theta$$

$$\hat{\theta} = \arg\min_\theta C \sum_{n=1}^{N} \max(0, 1 - y_n g_\theta(\mathbf{x}_n)) + ||\mathbf{w}||_2^2$$

Maximum Margin Property

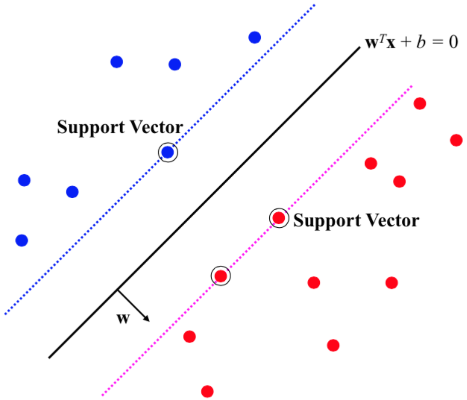
Part of relevance of SVMs stems from the fact that the hinge loss results in the *maximum margin* decision boundary when the training cases are linearly separable.





Support Vector Property

In the linearly separable case, some data points will always fall exactly on the margins. These points are called *support vectors* and they *uniquely determine* the optimal model parameters. SVR has a related support vector property.


$\mathbf{w}^T\mathbf{x} + b = 0$
Support Vector
Support Vector
$\mathbf{w}$