ICS273A: Machine Learning **Winter 2008**

Lecture 14 — Feb 27

Scribe: Jing Zhang, Dan Pan *Lecturer: Deva Ramanan*

Note: These lecture notes are still rough, and have only have been mildly proofread.

14.1 Loss functions; a unifying view

We can generalize the loss function as (14.1). In (14.1), we notice that loss function consists of two parts ,loss term and regularization term. We will introduce these two terms in the following sections.

$$J(w) = \sum_i L(m_i(w)) + \lambda R(w) \tag{14.1}$$

$$m_i = y^{(i)} f_w(x^{(i)}) \tag{14.2}$$

$$y^{(i)} \in \{-1, 1\} \tag{14.3}$$

$$f_w(x^{(i)}) = w^T x^{(i)} \tag{14.4}$$

14.1.1 Loss Term

In this section, we use 5 examples to illustrate the loss term. The five example are the Gold Standard(ideal case),Hinge(for soft margin SVM), log(for logistic regression,cross entropy error), squared loss(for linear regression) and Boosting.
First, let's see the "gold standard" loss function. We've implicitly been using it to evaluate our classifiers - we count the number of mixtakes that are made. This is often called "0-1" loss, or L_{01}

$$L_{01}(m) = \begin{cases} 0 & \text{if } m \geq 0 \\ 1 & \text{if } m < 0 \end{cases}$$

Second, let's look at loss term for soft margin SVM. We use L_{hinge} to represent the hinge loss.

$$J(w) = \frac{1}{2} \parallel w \parallel^2 + \sum_i \max(0, 1 - y^i w^T x^i) \tag{14.5}$$

$$= \frac{1}{2} \parallel w \parallel^2 + \sum_i \max(0, 1 - m_i(w)) \tag{14.6}$$

$$= R_2(w) + \sum_i L_{hinge}(m_i) \tag{14.7}$$

Third, let's see log loss, which is equivalent to the cross entropy loss function used to train a logistic regression model

$$J(w) = \lambda \parallel w \parallel^2 + \sum_i y^i \log g_w(x^{(i)}) + (1 - y^i)(\log 1 - g(x^{(i)})), y^i \in (0, 1) \tag{14.8}$$

$$g_w(x^{(i)}) = \frac{1}{1 + e^{-f_w(x^{(i)})}}$$

$$f_w(x^{(i)}) = w^T x^{(i)}$$

Because

$$\begin{aligned} 1 - g(x^{(i)})) &= 1 - \frac{1}{1 + e^{-f_w(x^{(i)})}} \\ &= \frac{e^{-f_w(x^{(i)})}}{1 + e^{-f_w(x^{(i)})}} \\ &= \frac{1}{1 + e^{f_w(x^{(i)})}} \end{aligned}$$

So, we can transform the equation into the following form,

$$J(w) = \lambda \parallel w \parallel^2 + \sum_i \log 1 + e^{-\bar{y^{(i)}} f_w(x^{(i)})} \tag{14.9}$$

$$L(m) = \log 1 + e^{-m} \tag{14.10}$$

$$m^i = \bar{y^{(i)}} f_w(x^{(i)})$$

$$\bar{y^{(i)}} = \begin{cases} -1 & \text{if } y^{(i)} = 0 \\ 1 & \text{if } y^{(i)} = 1 \end{cases}$$

Fourth, let's see Linear Regression, which have a squared loss term. We will use L_2 to scribe the squared loss term. We can see from fig(1) the squared term is much higher than the Gold Stand,so it is a bad function.

$$L_2(m) = (f_w(x) - y)^2 = (m-1)^2 \tag{14.11}$$

Later on, we'll look at a learning algorithm called boosting. It can be seen as a greedy optimization of the exponential loss term,

$$J(w) = \lambda R(w) + \sum_i \exp(-y^{(i)} f_w(x^{(i)})) \tag{14.12}$$

$$L_{exp}(m_i) = \exp(-m_i(w)) \tag{14.13}$$

In fig(1)we show the curves of these five functions, the blue line is for 0-1 loss function,

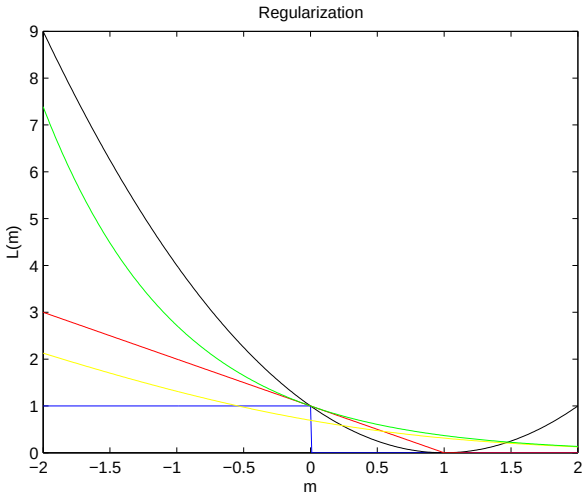

Figure 14.1. loss functions

the black line is for squared loss function,the red line is for Hinge loss, the yellow line is for logistic loss and the green line is for boosting loss. We can learn the following four points from analysis of these five loss terms.

1. L_{hinge},L_{log} are forgiving of errors/noise in training data;
2. All these loss functions are convex surrogates of L_{01}. This implies we can still use our generalization bounds

$$\begin{aligned} \xi_{01} &\leq \hat{\xi}_{01}(h) + fudge \\ &\leq \hat{\xi}_{hinge}(h) + fudge \end{aligned}$$

3. If our goal was not to minimize zero-one loss L_{01} onx^{new}, y^{new}, but rather to maximize the probability $P(y^{new}|x^{new})$, then L_{log} is the correct function to use. But note that maximizing the probability $P(y^{new}|x^{new})$ implicitly means we are scoring our results with unbounded loss. In other words, it is possible that we see a particular training example that is infinitely bad.

4. Bounded loss versus convexity. In some sense, loss should be intuitively bounded - we should not pay an infinite cost for missclassifying any one example. Note that bounded loss (L_{01}) is often used to score classifiers on test data. But bounded loss also implies non-convexity - e.g., $L_{01}(w)$ is not a convex function of w. This suggests that bounded loss functions are difficult to optimize. Convex surrogates are easier to work with.

14.1.2 Regularization Term

In this section, we will look into several kinds of useful regularization terms$\lambda R(w)$ from Eq. 14.1

Here are some popular regularization terms

$$R_2 = \frac{1}{2}\|w\|^2 \tag{14.14}$$

$$R_1 = \sum_i |w_i| \tag{14.15}$$

$$R_0 = |\{i : w_i \neq 0\}| \tag{14.16}$$

In general, one can interpret the above loss functions as members of the general family

$$R_p = (\sum_i |w_i|^p)^{\frac{1}{p}} \tag{14.17}$$

For $p = 2$, we obtain the square root of R_2. Again for computational convenience, it is often easier to work with the norm squared. One obtains R_0 as p approaches 0. R_2 is most common, as it is easy to apply to the gradient descent optimization of $J(w)$. One can show it simply adds a " $+ \lambda w''$ term to the gradient due to the loss summation. R_0 arguably is the most intuitive regularization term, since it also performs *feature selection*. This regularization term will tend to produce weight vectors with many zeros. Hence there is a computational benefit at run-time, since we only need to compute the *sparse* set of features associated with non-zero weights. Note that this notion of sparseness is different from the notion we encountered for SVMs. For a $m \times n$ design matrix X with each row is a feature vector, one can loosely think of "SVM-sparseness" as trying to reduce m, while "R_0-sparseness" as trying to reduce n.

Note that R_2 and R_1 are convex, while R_p for $p < 1$ is nonconvex, including R_0. One can interpret R_1 as a convex approximation to R_0. In practice, one observes that R_1 regularization tends to produce sparse weight vectors. We show examples of these regularization terms for w in one and two dimensions.

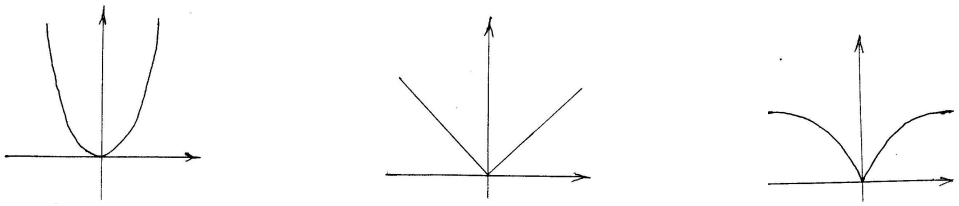

Figure 14.2. Plots of the function $R_p(w)$ corresponding to $p = \{2, 1, .3\}$ for a one-dimensional w.

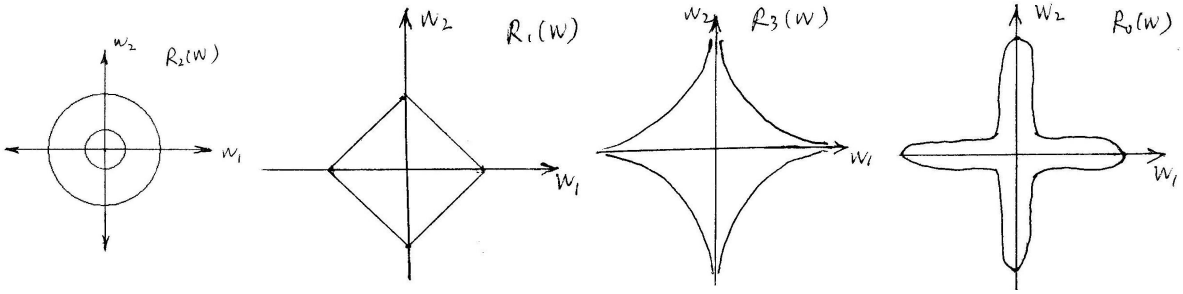

Figure 14.3. Iso-contour lines for the function $R_p(w)$ corresponding to $p = \{2, 1, .3, 0\}$ for a two-dimensional w.

14.2 Principal Component Analysis (PCA)

Because sometimes our input data might have thousands or millions of dimensions, we need to do dimension reduction before beginning training process.

14.2.1 Dimensionality reduction

Say $x^i \in R^{5000}$, we want to reduce dimensionality to $x^{(i)} \in R^{50}$. We train 50 weights rather than 5000. One can loosely see this an approximation to R_0 regularization. The dimension reduction process is as follow,
[h!]

1. center them $\hat{x}^{(i)} = x^{(i)} - \mu$, where $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$;
2. Let's find a direction vector v, where $\|v\| = 1$, s.t. the projection $v^T x^{(i)}$ captures most of the variation in $x^{(i)}$.

$$\max_v (\frac{1}{m}\sum_i (v^T \hat{x}^{(i)} - mean)^2), \|v\|^2 = 1 \tag{14.18}$$

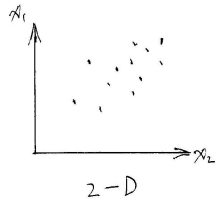

Figure 14.4. 2-D dimension reduction

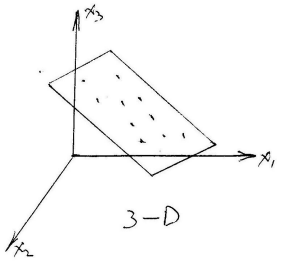

Figure 14.5. 3-D dimension reduction

Because already centered, so $mean = 0$, so we can change the function into the following form,

$$\max_v(\frac{1}{m}\sum_i (v^T\hat{x}^{(i)})(v^T\hat{x}^{(i)})^T), \|v\|^2 = 1$$

$$\max_v(\frac{1}{m}v^T(\sum_i \hat{x}^{(i)}(\hat{x}^{(i)})^T)v), s.t. v^Tv = 1$$

we use Σ to represent the sample covariance of $\hat{x}^{(i)}$, where $\Sigma = \sum_i \hat{x}^{(i)}(\hat{x}^{(i)})^T$. We can write Lagrange Duality as follow,

$$L(v, \lambda) = -\frac{1}{2}v^T\Sigma v + \lambda(v^Tv - 1) \tag{14.19}$$

$$\frac{\delta L}{\delta v} = 0 \Longrightarrow -\Sigma v + \lambda v = 0 \Longrightarrow \Sigma v = \lambda v$$

We see that v is an eigenvector of Σ.
What about projecting into 2 dimensions?

$$\max_{v_2}(v_2^T\Sigma v_2), s.t. v_2^Tv_2 = 1, v_2^Tv_1 = 0$$

3. PCA: We would like to project $x^{(i)}$ into a $k << n$ dimensional subspace spanned by the vectors $v_1, v_2, \ldots, v_k$. We want to choose the vectors so as to maximize the variation of the projected data. We do this by choosing $v_1, v_2, \ldots, v_k$ to be the top k eigenvectors (associated with the k largest eigvenvalues) of the covariance matrix $\Sigma \frac{1}{n} \sum x^{(i)} x^{(i)T}$

4. Computing the projection. Let $V_{n\times k} = \left[v_1, v_2, \ldots, v_k\right]$ be the matrix of the top k eigenvectors. Let $X_{m\times n} = \begin{bmatrix} x^{(1)} \\ \ldots \\ x^{(m)} \end{bmatrix}$ be the design matrix. Note that $\Sigma = \frac{1}{m} X^T X$. Let $c^{(i)}$ be the k coefficients specifying the coorindates of $x^{(i)}$ when projected onto V. We can write the matrix of projections $C_{m\times k} = \begin{bmatrix} c^{(1)} \\ \ldots \\ c^{(m)} \end{bmatrix}$ as $C = XV$.

5. Computuing the reconstruction. We can reconstruct the point $x^{(i)}$ by computing the linear combination $\hat{x}^{(i)} = c^{(1)} v_1 + \ldots + c^{(k)} v_k$. We can write the $m \times n$ matrix of reconstructions as $\hat{X} = CV^T = XVV^T$.

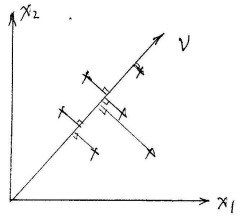

Figure 14.6. projection

14.2.2 Geometry

Because Σ is symmetric, we can write $\Sigma = V\Lambda V^T, V = \left[v_1, v_2, \ldots, v_k\right] .\Omega = diag(\lambda_1, \lambda_2, \ldots, \lambda_k)$. Consider Σ as a linear transformation applied to some vector z, or $\Sigma z = V\Lambda V^T z$.

1. $c = V^T z$ are the coordinates of the vector z when projected onto the orthonormal basis V.

2. $\tilde{c} = \Lambda c$ scales each coordinate c_i by λ_i.

3. $V\tilde{c}$ is the "reconstruction" of z with the linear combination $\sum \tilde{c}_i v_i$

Theorem 14.1. *SVD: Any (possibly non-square matrix) $A_{m\times n}$ can be factored $A = U\Sigma V^T$, where $U_{m\times m}$ an orthornomal matrix ($UU^T = I$) and $V_{n\times n}$ is also an orthonormal matrix*

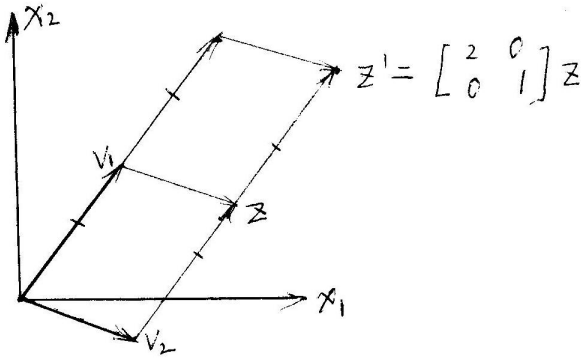

Figure 14.7. Geometry meaning

($VV^T = I$). $\Sigma_{m\times n}$ is a diagonal matrix with entries $\sigma_i \geq 0$. These σ_i are called singular values of A.

Lets similarly factor the transformation Az into $U\Sigma V^T z$.

1. $c = V^T z$ computes the coordinates of the vector z when projected onto the orthonormal basis V.
2. $\tilde{c} = \Sigma c$ scales each coordinate c_i by σ_i.
3. $U\tilde{c}$ is the "reconstruction" of Z in R^m formed with the linear combination $\sum \tilde{c}_i u_i$.

Now let's consider the design matrix $X_{m\times n}$, where each row is an input feature in R^n. We know X can be represented by $X = U\Sigma V^T$ due to the SVD theorm. We can write the sample covariance of the m data points as

$$\Sigma_{cov} = \frac{1}{m}\sum x^{(i)}x^{(i)T} = \frac{1}{m}X^T X \tag{14.20}$$

$$= \frac{1}{m}(V\Sigma U^T)(U\Sigma V^T) \tag{14.21}$$

$$= \frac{1}{m}(V\Sigma(U^T U)\Sigma V^T) \tag{14.22}$$

$$= \frac{1}{m}(V\Sigma^2 V^T) \tag{14.23}$$

Hence the orthogonal basis V produced from the SVD of X is equivalent to the eigenvectors of the covariance matrix Σ_{cov}. This is a useful observation since computationally it may be easier to work with X without computing Σ_{cov}. For example if our features are very high dimensional, Σ_{cov} is expensive to compute.