Definitions and notations

Zhehao Wang

December 9, 2018

1 Calculus and linear algebra

1.1 Limit

Let $f(x)$ be a function defined on an interval that contains $x = a$, except possibly at $x = a$, then we say that

$$\lim_{x \to a} f(x) = L$$

if for every $\epsilon > 0$ there is some number $\delta > 0$ such that

$$|f(x) - L| < \epsilon \text{ whenever } 0 < |x - a| < \delta$$

1.2 Gradient

Given $f(\vec{x})$ where $\vec{x} = (x_1, \ldots, x_n)$ on $\mathbb{R}^n$

$$\bigtriangledown f(a_1, \ldots, a_n) = (\frac{\partial f}{\partial x_1}(a_1, \ldots, a_n), \ldots, \frac{\partial f}{\partial x_n}(a_1, \ldots, a_n))$$

Intuition: gradient is a vector (the rate of change of your function, when you move in a certain direction), which in a two-dimensional space, tangents the curve at a given point.

1.3 Directional derivative

Homework notation

$$f'(x; u) = \lim_{h \to 0} \frac{f(x + hu) - f(x)}{h}$$

Wikipedias notation

$$\bigtriangledown_v f(\vec{x}) = \lim_{h \to 0} \frac{f(\vec{x} + h\vec{v}) - f(\vec{x})}{h}$$

Intuition: the rate-of-change of a function $f(\vec{x})$ on direction $\vec{v}$. Remember that this is a scalar (since we are given the direction).

$$f'(x;u) = \bigtriangledown f(x)^T u$$

Meaning gradient on a certain direction vector u is $f(x)$'s directional derivative on u.

1.4 Vector norm

A norm is a function that assigns a stricly positive length or size to each vector in a vector space (except the zero vector which is assigned a length of 0).
Absolute value norm is a norm on the one-dimensional vector spaces formed by real or complex numbers.

$$\|x\| = |x|$$

Euclidean norm on a Euclidean space $\mathbb{R}^n$ is such

$$\|\vec{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$$

Manhattan or taxicab norm

$$\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

p-norm

$$\|\vec{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$$

Note that when $p = 1$, we get Manhattan norm, and when $p = 2$, we get Euclidean norm.
When $p = \infty$

$$\|\vec{x}\|_\infty = \max_i |x_i|$$

1.5 argmax

Points of the domain of some function at which the function values are maximized.
Given an arbitrary set X, a totally ordered set Y and a function $f : X \to Y$, the arg max over some subset S of X is defined by

$$\arg\max_{x \in S \subseteq X} f(x) = \{x \mid x \in S \wedge \forall y \in S : f(y) \leq f(x)\}$$

1.6 Dot product, inner product

Maps two equal-size vectors to a real value.

$$\cdot : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

Defined geometrically, given two vectors a and b,

$$a \cdot b = \|a\| \, \|b\| \, cos(\theta)$$

where θ is the angle between a and b.
Defined algebraically,

$$a \cdot b = a^T b$$

1.7 Cauchy-Schwarz inequality

Given two vectors u and v

$$|\langle u, v \rangle|^2 \leq |\langle u, u \rangle| \cdot |\langle v, v \rangle|$$

Where $|\langle u, u \rangle|$ is the inner product. The equality holds only when u and v are linearly independent (parallel).

1.8 Jacobian matrix

Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function.
Suppose $\vec{f} : \mathbb{R}^n \to \mathbb{R}^m$, the Jacobian matrix $\vec{J}$ of $\vec{f}$ is defined as follows

$$\vec{J} = \begin{bmatrix} \frac{\partial \vec{f}}{\partial x_1} & \cdots & \frac{\partial \vec{f}}{\partial x_n} \end{bmatrix}$$

or component-wise $\vec{J}_{ij} = \frac{\partial \vec{f}_i}{\partial x_j}$, meaning

$$\vec{J} = \begin{bmatrix} \frac{\partial \vec{f}_1}{\partial x_1} & \cdots & \frac{\partial \vec{f}_1}{\partial x_n} \\ & \cdots & \\ \frac{\partial \vec{f_m}}{\partial x_1} & \cdots & \frac{\partial \vec{f_m}}{\partial x_n} \end{bmatrix}$$

1.9 Order of a matrix

1.10 Inversibility of a matrix

1.11 Eigenvalue, eigenvector

An **eigenvalue** of a **linear transformation** * is a non-zero vector that changes by only a scalar factor when that linear transformation is applied to it.
Meaning the set of $\vec{v}$ that satisfies

$$T(\vec{v}) = \lambda \vec{v}$$

Where T is the linear transformation, λ is a scalar and called **eigenvalue**.
We can compute the eigenvalues λ of a square matrix A of size n by solving this **determinant**

*Think square transformation matrix

$$|A - \lambda I| = 0$$

Where I is the **identity matrix (unit matrix)** of size n.
For each eigenvalue λ, we can then solve its (corresponding set of) eigenvectors $\vec{v}$ using

$$(A - \lambda I) \cdot \vec{v} = 0$$

Intuition: applying a linear transformation on an eigenvector of that linear transformation yields a vector that is parallel to this eigenvector (multiplier: eigenvalue).

1.12 Positive-definite

A **symmetric** $n \times n$ real matrix is positive definite if the scalar $z^T M z$ is strictly positive for every non-zero column vector $z \in \mathbb{R}^n$.
The identity matrix I is positive definite.
For any real invertible matrix A, the product $A^T A$ is a positive definite matrix.

1.13 Convexity

A **set** C is **convex** if for any $x_1, x_2 \in C$ and any θ with $0 \leq \theta \leq 1$ we have

$$\theta x_1 + (1 - \theta) x_2 \in C$$

Intuition: for all x_1, x_2 in C, all the points on the line segment connecting points x_1, x_2 are in C.

A **function** $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** if dom f is a convex set and if for all $x, y \in$ dom f, and $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1 - \theta) y) \leq \theta f(x) + (1 - \theta) f(y)$$

Intuition: line segment connecting points x, y on the graph of f does not cross the graph of f.
Examples:

- $f(x) = ax + b$ is both convex and concave on $\mathbb{R}$ for all $a, b \in \mathbb{R}$.
- $f(x) = |x|^p$ for $p \geq 1$ is convex on $\mathbb{R}$.
- $f(x) = e^{ax}$ for all a is convex on $\mathbb{R}$.
- Every norm on $\mathbb{R}^n$ is convex.
- $f(x) = \max\{x\}$ is convex on $\mathbb{R}^n$.

A function f is **strictly convex** if the line segment connecting any two points on the graph of f lies strictly above the graph.

- When a function is convex, if there is a local minimum, then it is a global minimum.
- When a function is strictly convex, if there is a local minimum, then it is the unique global minimum.

1.14 The general optimization problem

Standard form: minimize $f_0(x)$ subject to $f_i(x) \leq 0,\ i = 1, \dots m,\ h_i(x) = 0,\ i = 1, \dots m.$* Where f_0 is the objective function and $x \in \mathbb{R}^n$ are the optimization variables.
We can replace $h(x) = 0$ with $h(x) \leq 0$ and $-h(x) \leq 0$, so we don't need the equality constraints.
The set of points satisfying the constraints is called the **feasible set**.
A point in the feasible set is called a **feasible point**.
If x is feasible and $f_i(x) = 0$, then we say inequality constraint $f_i(x) \leq 0$ is **active** at x.
The **optimal value** p^* of the problem is defined as

$$p^* = \inf \{f_0(x) | x \in \text{feasible set}\}$$

x^* is an **optimal point** (a solution to the problem) if x^* is feasible and $f(x^*) = p^*$.

1.15 Lagrangian duality

Given a general optimization problem:

$$\begin{aligned} &\text{minimize} f_0(x) \\ &\text{subject to} \quad f_i(x) \leq 0, i = 1, \dots m. \end{aligned}$$

The **Lagrangian** for it is defined as

$$L(x, \lambda) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$$

Where λ_is are called **Lagrangian multipliers** (**dual variables**).†
Supremum over Lagrangian gives back encoding of objective and constraints:

$$\begin{aligned} \sup_{\lambda \succeq 0} L(x, \lambda) &= \sup_{\lambda \succeq 0} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x)\right) \\ &= \begin{cases} f_0(x), & \text{when } f_i(x) \leq 0\ \forall i \\ \infty, & \text{otherwise} \end{cases} \end{aligned}$$

*Assuming the intersections of domains of all f_i and h_i is not empty.
†Irrelevant with convexity.

And equivalent of the **primal form** of the optimization problem:

$$p^* = \inf_x \sup_{\lambda \succeq 0} (f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x))$$

We get the **Lagrangian dual problem** by swapping the inf and sup.

$$d^* = \sup_{\lambda \succeq 0} \inf_x (f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x))$$

Weak duality $p^* \geq d^*$ holds for any optimization problem. *
Duality gap is $p^* - d^*$, for convex problems, we often have **strong duality**: $p^* = d^*$.
The **Lagrangian dual function** is

$$g(\lambda) = \inf_x L(x, \lambda) = \inf_x (f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x))$$

The dual function is always concave.†
Lagrangian dual function gives a lower bound on optimal solution:

$$p^* \geq d^* = \sup_{\lambda \geq 0} g(\lambda)$$
$$p^* \geq g(\lambda) \quad \forall \lambda \geq 0$$

The **Lagrangian dual problem** is a search for best lower bound on p^*: maximize $g(\lambda)$ subject to $\lambda \succeq 0$.
λ is **dual feasible** if $\lambda \succeq 0$ and $g(\lambda) > -\infty$, and λ^* is **dual optimal** or **optimal Lagrange multipliers** if they are optimal for the Lagrange dual problem.

For a general optimization problem, if we have **strong duality**, we get a relationship called **complementary slackness** between

- the optimal Lagrange multiplier λ_i^*, and
- the i-th constraint at optimumm: $f_i(x^*)$

$$\lambda_i^* f_i(x^*) = 0$$

Always have Lagrange multiplier being zero, or constraint is active at optimum, or both.
Lagrangian can be applied to illustrate the equivalence of Tikhanov and Ivanov forms of penalizing complexity measure.

*Hint, for any general $f : W \times Z \to R$, this can be proved given $\inf_{w \in W} f(w, z_0) \leq f(w_0, z_0) \leq \sup_{z \in Z} f(w_0, z)$, add another sup to the leftmost term and inf to the rightmost term.

†Pointwise min of affine functions

1.16 Convex optimization standard form

Standard form of convex optimization problem: minimize $f_0(x)$ subject to $f_i(x) \leq 0, i = 1, \dots m$. Where $f_0, \dots f_m$ are convex functions.
We usually have strong duality for convex optimization problems, but not always. The additional conditions needed are called **constraint qualifications**.

2 Probability

2.1 Random variable

A random variable $X : \Omega \to E$ is a measurable function from a set of possible outcomes Ω to a measurable space E. Often times $E = \mathbb{R}$
The probability that X takes on a value in a measurable set $S \subseteq E$ is written as

$$Pr(X \in S) = P(\omega \in \Omega | X(\omega) \in S)$$

Intuition: mapping outcomes of a random process to numbers, like this definition of X

$$X = \begin{cases} 0, & \text{if heads} \\ 1, & \text{if tails} \end{cases}$$

Instead of a traditional algebraic variable that can be solved for one value, a random variable can have different values (each with a probability) under different conditions.

2.2 Law of large numbers

$X_i, X_2, \dots$ is an infinite sequence of independent and identically distributed (iid) random variables with expected value $E(X_1) = E(X_2) = \dots = \mu$, and

$$\bar{X}_n = \frac{1}{n}(X_1 + \dots + X_n) \ ^*$$

The weak law states that for any positive number ϵ

$$\lim_{n \to \infty} Pr(|\bar{X}_n - \mu| > \epsilon) = 0$$

The strong law states that

$$Pr(\lim_{n \to \infty} \bar{X}_n = \mu) = 1$$

Intuition: law of large numbers is a theorem that describes the average of the results obtained from a large number of trials should be close to the expected value, and will tend to become closer as more trials are performed.

*What does the sum of random variables mean? Seems that each X_i here means 'sampled' values following the distribution defined by the random variable, like, the outcome of an experiment

3 Statistical learning

Let $\mathbf{X}$ denote the input space, $\mathbf{Y}$ denote the output space, and $\mathbf{A}$ denote the action space.
A **decision function / prediction function** $f : \mathbf{X} \to \mathbf{A}$ maps an input to an action.
A **loss function** $l : \mathbf{A} \times \mathbf{Y} \to \mathbb{R}$ evaluates an action in the context of an output, and maps the error to a real. In traditional problems we can usually assume action has no effect on the output (like stock market prediction).

3.1 Risk, Bayesian function, empirical risk

Assume there is a data generating distribution $P_{\mathbf{X}\times\mathbf{Y}}$.
Risk of a decision function can then be defined as

$$R(f) = \mathbb{E}l(f(x), y)$$

Where an input/output pair (x, y) is generated independently and identically distributed from $P_{\mathbf{X}\times\mathbf{Y}}$.
A **Bayesian decision function / target function** $f^* : \mathbf{X} \to \mathbf{A}$ is a function that achieves the minimal risk among all possible functions.

$$f^* = \arg\min_f R(f)$$

Risk cannot be calculated as we are not given $P_{\mathbf{X}\times\mathbf{Y}}$.
Let $\mathbf{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be drawn iid from $P_{\mathbf{X}\times\mathbf{Y}}$.
The **empirical risk** of $f : \mathbf{X} \to \mathbf{A}$ with respect to $\mathbf{D}_n$ is

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i), y_i)$$

By strong law of large numbers (2.2), when n is big enough, empirical risk can be used to approximate risk.

3.2 Empirical risk minimization, constrained empirical risk minimization

A function $\hat{f}$ is an **empirical risk minimizer** if

$$\hat{f} = \arg\min_f \hat{R}_n(f)$$

The prediction function that produces the smallest empiricall risk over set $\mathbf{D}_n$. This naturally leads to overfit.
So we minimize empirical risk, but only within a hypothesis space $\mathbf{F}$, being a set of prediction functions.

Constrained empirical risk minimizer can then be defined as

$$\hat{f}_n = \arg\min_{f \in \mathbf{F}} \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)$$

If we plug Risk in instead, a **constrained risk minimizer** becomes

$$f^*_{\mathbf{F}} = \arg\min_{f \in \mathbf{F}} \mathbb{E} l(f(x), y)$$

Approximation error is the risk difference between the constrained risk minimizer (in $\mathbf{F}$) and the target

$$R(f_{\mathbf{F}}) - R(f^*)$$

Estimation error is the risk difference between the constrained empirical risk minimizer and the constrained risk minimizer (both in $\mathbf{F}$)

$$R(\hat{f}_n) - R(f_{\mathbf{F}})$$

Optimization error is the risk difference between a function $\tilde{f}_n$ (which we find in practice) and the constrained empirical risk minimizer *

$$R(\tilde{f}_n) - R(\hat{f}_n)$$

Excess risk is the risk between a function and the target

$$\text{Excess Risk}(\hat{f}_n) = \text{Estimation Error} + \text{Approximation Error} = R(\hat{f}_n) - R(f^*)$$

$$\begin{aligned}\text{Excess Risk}(\tilde{f}_n) &= \text{Optimization Error} + \text{Estimation Error} + \text{Approximation Error} \\ &= R(\tilde{f}_n) - R(f^*)\end{aligned}$$

3.3 Complexity measure, l_1, l_2 regularization, ridge regression, lasso regression, coordinate gradient descent

Regularization is a mechanism to reduce overfitting. By narrowing down the hypothesis space to only those functions who satisfies being within a complexity measure (or penalize the empirical risk minimizer with another term representing complexity).
Complexity measure $\Omega : \mathbf{F} \to [0, \infty)$ for linear decision functions $f(x) = w^T x$ can be defined using the p-norm (to the p) of w.

- l_0 complexity is then the number of non-zero coefficients
- l_1 **lasso** complexity is $\sum_{i=1}^{d} |w_i|$

*Optimization can be negative, however, this $\hat{R}(\tilde{f}_n) - \hat{R}(\hat{f}_n)$ can't. In this hypothesis space, this function can fit the overall world better than the constrained empirical risk minimizer, but it can't on the training set, on which the constrained ERM performs best in this space

- l_2 **ridge** complexity is $w^T w = \sum_{i=1}^{d} w_i^2$

To factor complexity measure in constrained empirical risk minimization, we can use **Ivanov** or **Tikhonov** regularization. where

$$\arg\min_{f \in \mathbf{F}} \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i) \text{ s.t. } \Omega(f) \leq r$$

is Ivanov regularization, and

$$\arg\min_{f \in \mathbf{F}} \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i) + \lambda \Omega(f)$$

is Tikhonov regularization. The two can be shown to be equivalent for many performance measure of f, and Ω, using Lagrangian duality theory.
Plugging in the lasso / ridge definition of Ω to either form, we get lasso / ridge regression.
For example, ridge regression in Tikhonov form looks like

$$\hat{w} = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|_2^2$$

Lasso regression typically lead to feature sparsity (more coefficient equal to 0). This can be shown intuitively by comparing the two on the intersection of contour plot (of empirical risk minimizer) and the area constrained by lasso (diamond on a $f(x) = ax + b$ prediction function) / ridge (circle on the same prediction function) regression, as show in Figure 1*.

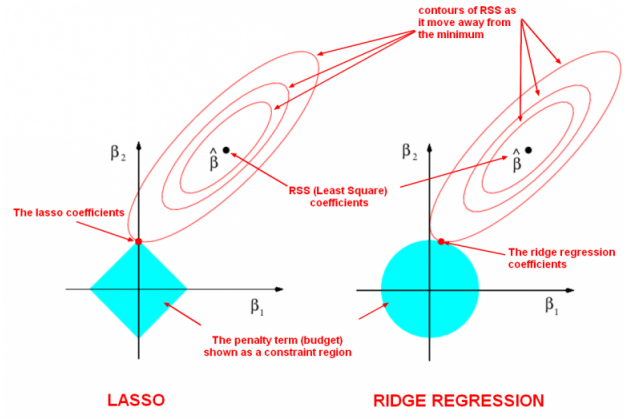

Figure 1: Why lasso regression tends to lead to feature sparsity

*https://niallmartin.wordpress.com/2016/05/12/shrinkage-methods-ridge-and-lasso-regression/

Lasso regression (l_1-norm) is not differentiable, thus we cannot apply gradient descent as-is. We split each coefficient into positive and negative parts: rewrite the vector $w = w^+ - w^-$ ($|w| = w^+ + w^-$), and we have this equivalent problem*:

$$\arg\min_{w^+, w^-} \frac{1}{n} \sum_{i=1}^{n} ((w^+ - w^-)^T x_i - y_i)^2 + \lambda(w^+ + w^-)$$

subject to $w_i^+ \geq 0 \; \forall i$, $w_i^- \leq 0 \; \forall i$
Now we have a constraint and two vectors, to find the minimum, we can use **projected SGD**† or **coordinate descent**.
Coordinate gradient descent on a vector w works by adjusting only a single w_i in each step, as opposed to possibly alter the entire w in one step as in regular gradient descent. We iteratively adjust each coordinate several times, where we can pick a random one to adjust, or do cyclic adjustment.
Coordinate gradient descent has a **closed form solution** for lasso (in the form below‡), where we can use a formula to solve each w_i, instead of having to loop over them.

$$\hat{w}_j = \arg\min_{w_j \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|_1$$

3.4 Elastic nets

Consider what happens in lasso and ridge regressions when features are linear: lasso would assign all the weight to the feature with larger scale, and ridge would assign the weights proportional to the features' scales.
It's similar when features are highly correlated (near-linear relationship between features): think Figure 1 instead of parallel lines, we get elongated ellipses, whose intersection with the hypothesis space would reflect the scales of correlated features.
Elastic net combines lasso and ridge penalties.

$$\hat{w} = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$$

Geometrically, we end with a hypothesis space like a diamond with edges bulging out (between a circle and a diamond).

*whose equivalence can be proved. And this can be plugged in to a quadratic solver to give us w^+ and w^-

†Normal gradient descent but project / reset each coordinate to the constrained space after each step

‡This form is not differentiable, but it can be shown that for coordinate gradient descent to work there is a weaker condition, which lasso satisfies. *This would indicate that the split into positive and negative shown above is not required, for solving using coordinate gradient descent?*

3.5 Regression loss functions

A **distance-based loss function** is a loss function ($l(\hat{y}, y) \in \mathbb{R}$) that only depends on the **residual** $r = \hat{y} - y$. Most regression losses are distance-based.
Distance-based losses are translation-invariant: $l(\hat{y} + a, y + a) = l(\hat{y}, y)$.
Square loss ($l(r) = r^2$) penalizes outlier points more heavily than absolute (Laplacian) loss ($l = |r|$) does, and is considered less robust.
Downside with absolute loss is that it's not differentiable.
We are able to construct **Huber loss function** that is robust and differentiable: quadratic for $|r| \leq \delta$ and linear for $|r| > \delta$.

3.6 Classification loss functions

If we have action space $\mathbf{A}$ and outcome space $\mathbf{Y}$ both being $\{-1, 1\}$.
0-1 loss for $f : \mathbf{X} \to \{-1, 1\}$:

$$l(f(x), y) = 1\ (f(x) \neq y)$$

Where $1\ (f(x) \neq y)$ denotes score 1 whenever $(f(x) \neq y)$.

If we allow **real-valued prediction function** $f : \mathbf{X} \to \mathbb{R}$, meaning action space $\mathbf{A} = \mathbb{R}$ where the value represents confidence of our prediction.
We can define **margin** m as $m = y\hat{y}$, where $\hat{y}$ stands for the predicted score. We want to maximize the margin. Most classification losses are **margin-based**.
Empirical risk for 0-1 loss is

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} 1\ (y_i f(x_i) \leq 0)$$

$\hat{R}_n(f)$ is non-convex, not differentiable, discontinuous, and its optimization is NP-hard.

SVM/Hinge loss is defined as

$$l_{Hinge} = max\{1 - m, 0\} = (1 - m)_+$$

It is a convex, upper bound on 0-1 loss, and not differentiable at $m = 1$. And we have **margin error** when $m < 1$.
Using hinge loss function, we define **soft margin linear support vector machine** (a constrained empirical risk minimizer) as

$$\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (1 - y_i f_w(x_i))_+ + \lambda \|w\|_2^2$$

With $l2$ regularization, and hypothesis space is linear $\{f(x) = w^T x | w \in \mathbb{R}^d\}$.

Logistic/log loss is defined as

$$l_{Logistic} = \log(1 + e^{-m})$$

It always wants more margin, and is differentiable.

0-1 loss, hinge and Logistic loss functions are illustrated in Figure 2 *.

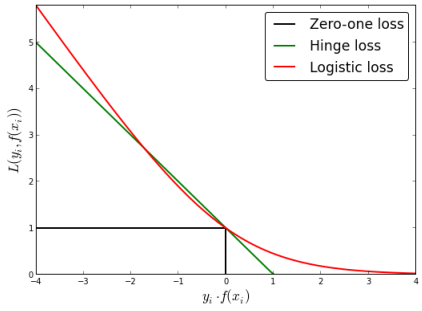

Figure 2: 0-1 loss, hinge loss and Logistic loss functions

3.7 SVM and Lagrangian duality

Adding a **bias** term b to the constrained empirical risk minimizer of **support vector machine**, we have

$$\underset{w \in \mathbb{R}^d,\ b \in \mathbb{R}}{\arg\min} \frac{c}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b)) + \frac{1}{2}\|w\|_2^2$$

whose solution gives us the SVM prediction function. c is a constant that can be tweaked to decide how much regularization matters.
This problem is not differentiable due to max. We can turn it into an equivalent problem.

$$\begin{aligned} \text{minimize} \quad & \frac{c}{n} \sum_{i=1}^{n} \xi_i + \frac{1}{2}\|w\|_2^2 & \\ \text{subject to} \quad & -\xi_i & \leq 0,\ \forall i \in \{1, \dots n\} \\ & (1 - y_i(w^T x_i + b)) - \xi_i & \leq 0,\ \forall i \in \{1, \dots n\} \end{aligned}$$

This has a differentiable objective function, $n+d+1$ unknowns and $2d$ affine constraints. This can be solved by off-the-shelf QP solver. $\sum_{i=1}^{n} \xi_i$ now represents the margin loss.

*http://fa.bianp.net/blog/2013/loss-functions-for-ordinal-regression/

We apply **Lagrangian multiplier** to the constrained optimization problem to get the following

$$\begin{aligned} L(w,b,\xi,\alpha,\lambda) &= \frac{1}{2}\|w\|_2^2 + \frac{c}{n}\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1-y_i(w^T x_i+b)-\xi_i) - \sum_{i=1}^{n}\lambda_i\xi_i \\ &= \frac{1}{2}w^T w + \sum_{i=1}^{n}\xi_i(\frac{c}{n}-\alpha_i-\lambda_i) + \sum_{i=1}^{n}\alpha_i(1-y_i(w^T x_i+b)) \end{aligned}$$

Its **primal** and **dual problems** look like

$$\begin{aligned} p^* &= \inf_{w,\xi.b}\sup_{\alpha,\lambda\geq 0} L(w,b,\xi,\alpha,\lambda) \\ &\geq \sup_{\alpha,\lambda\geq 0}\inf_{w,\xi.b} L(w,b,\xi,\alpha,\lambda) \\ &= d^* \end{aligned}$$

This satisfies strong duality by **Slater's constraint qualification**. *
The **Lagrangian dual function** then becomes

$$\begin{aligned} g(\alpha,\lambda) &= \inf_{w,\xi.b} L(w,b,\xi,\alpha,\lambda) \\ &= \inf_{w,\xi.b}(\frac{1}{2}w^T w + \sum_{i=1}^{n}\xi_i(\frac{c}{n}-\alpha_i-\lambda_i) + \sum_{i=1}^{n}\alpha_i(1-y_i(w^T x_i+b))) \end{aligned}$$

This is convex and differentiable: quadratic in w, and linear in ξ_i and b. The global minima is at $\frac{\partial L}{\partial w}=0$, $\frac{\partial L}{\partial \xi}=0$, $\frac{\partial L}{\partial b}=0$.
Solving the three partial derivations (**first order conditions**), we have

$$\begin{aligned} \frac{\partial L}{\partial w}=0 &\implies w=\sum_{i=1}^{n}\alpha_i y_i x_i \\ \frac{\partial L}{\partial \xi}=0 &\implies \alpha_i+\lambda_i=\frac{c}{n} \\ \frac{\partial L}{\partial b}=0 &\implies \sum_{i=1}^{n}\alpha_i y_i=0 \end{aligned}$$

Replacing the w, ξ and bs from the **dual function**, the **dual problem** then becomes

*Where a convex problem + affine constraints $\implies$ strong duality iff problem is **feasible**. This problem is feasible if we simply let $w=b=0$ and $\xi_i=1\ \forall i\in\{1,\dots n\}$

$$\begin{aligned}\sup_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j x_j^T x_i \\ s.t. & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i \in \left[0, \tfrac{c}{n}\right] \quad \forall i \in \{1, \dots n\}\end{aligned}$$

Note

- Given a solution α^* to dual, primal solution is $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$.
- w^* is a linear combination of the data $x_1, \dots x_n$. The x_i's corresponding to $\alpha_i^* > 0$ are called **support vectors**.
- $\alpha^* \in \left[0, \frac{c}{n}\right]$, so c controls the maximum weight on each sample. This is **robust**.
- This is a quadratic objective with n unknowns and $n+1$ constraints.
- Efficient minimization algorithm, sequential minimal optimization, exists.

Since we have **strong duality**, **complementary slackness** holds, where the Lagrangian multiplier $\times$ the constraint function is 0.

$$\begin{aligned}\alpha_i^*(1 - y_i f^*(x_i) - \xi_i) &= 0 \\ \lambda_i^* \xi_i^* = (\frac{c}{n} - \alpha_i^*)\xi_i^* &= 0\end{aligned}$$

This means

- $y_i f^*(x_i) > 1 \implies \alpha_i^* = 0,\ \xi_i^* = 0$
- $y_i f^*(x_i) < 1 \implies \alpha_i^* = \frac{c}{n},\ \xi_i^* > 0$
- $y_i f^*(x_i) = 1 \implies \alpha_i^* \in \left[0,\ \frac{c}{n}\right]$
- $\alpha_i^* = 0 \implies \xi_i^* = 0$ (margin loss is 0), so $y_i f^*(x_i) \geq 1$
- $\alpha_i^* = \frac{c}{n} \implies y_i f^*(x_i) \leq 1$
- $\alpha_i^* \in \left(0, \frac{c}{n}\right) \implies y_i f^*(x_i) = 1$

Suppose we have an i s.t. $\alpha_i^* \in \left(0, \frac{c}{n}\right)$, using the complementary slackness conditions we can solve the bias term b^*, whose value stays the same for any choice of i satisfying $\alpha_i^* \in \left(0, \frac{c}{n}\right)$. *

$$b^* = y_i - x_i^T w^*$$

If there are no such α_i^*s, then we have a degenerate SVM training problem.

*With numerical error, it's more rubust to average over all eligible i and use the mean of resulting b^*

3.8 Level sets / contours, sublevel sets and convex optimization

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a function. A **level set** or **contour line** for the value c is the set of points $x \in \mathbb{R}^d$ for which $f(x) = c$.
A **sublevel set** for the value c is the set of points $x \in \mathbb{R}^d$ for which $f(x) \leq c$.
If $f : \mathbb{R}^d \to \mathbb{R}$ is **convex**, then sublevel sets are convex. Level sets and superlevel sets of convex functions are not generally convex, hence, the **standard form of convex optimization problem** uses ≤ 0 to express constraints.
The **implicit form of convex optimization problem** goes as

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad x \in C \end{aligned}$$

where f is a convex function and C is a convex set.
Also, intersection of convex sets is convex.

3.9 First-order approximation

Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable.
We can use **linear / first-order approximation** to predict $f(y)$ given $f(x)$ and $\bigtriangledown f(x)$

$$f(y) \approx f(x) + \bigtriangledown f(x)^T (y - x)$$

Now suppose $f : \mathbb{R}^d \to \mathbb{R}$ is convex and differentiable.
The linear approximation to f at x is a **global underestimator** of f.

$$\forall\ x, y \in \mathbb{R}^d, \quad f(y) \geq f(x) + \bigtriangledown f(x)^T (y - x)$$

And if $\bigtriangledown f(x) = 0$ then x is a global minimizer of f *.

3.10 Subgradients

A vector $g \in \mathbb{R}^d$ is a **subgradient** of $f : \mathbb{R}^d \to \mathbb{R}$ at x if $\forall\ z$,

$$f(z) \geq f(x) + g^T (z - x)$$

f is **subdifferentiable** at x if $\exists$ at least one subgradient at x.
The set of all subgradients at x is called the **subdifferential**: $\partial f(x)$
f is convex and differentiable $\implies \partial f(x) = \{\bigtriangledown f(x)\}$. At any point x, there can be 0, 1, or infinite many subgradients. $\partial f(x) = \emptyset \implies f$ is not convex.
If $0 \in \partial f(x)$, then x is a **global minimizer** of f.
As a reminder, for function $f : \mathbb{R}^d \to R$,

*Where local information gives global information!

- graph of function lives in $\mathbb{R}^{d+1}$
- gradient, subgradient of f live in $\mathbb{R}^d$, and
- contours, level sets, sublevel sets are in $\mathbb{R}^d$

Gradient at x is orthogonal to level set at x. (assuming f continuously differentiable.)

Now to figure out the **direction** on which to do a subgradient descent.
Let $f : \mathbb{R}^d \to \mathbb{R}$ have a subgradient g at x_0, hyperplane H orthogonal to g at x_0 must **support** the level set $S = \{x \in \mathbb{R}^d | f(x) = f(x_0)\}$, meaning H contains x_0 and all of S lies on one side of H. The proof of which* suggests the following: Points on subgradient g side of H have larger f-values than $f(x_0)$, and points on $-g$ side may not have smaller f-values, meaning $-g$ may not be a descent direction.

In **subgradient descent**, suppose we repeatedly step in a negative subgradient direction $x = x_0 - tg$ where $t > 0$ is the step size and $g \in \partial f(x_0)$.
We can prove $-g$ gets us closer to minimizer. Meaning, suppose f is convex, let $x = x_0 - tg$ for $g \in \partial f(x_0)$, and let z be any point for which $f(z) < f(x_0)$, then for small enough $t > 0$, $\|x - z\|_2 < \|x_0 - z\|_2$.
Proof:

$$\begin{aligned}\|x - z\|_2^2 &= \|x_0 - tg - z\|_2^2 \\ &= \|x_0 - z\|_2^2 - 2tg^T(x_0 - z) + t^2 \|g\|_2^2 \\ &\leq \|x_0 - z\|_2^2 - 2t(f(x_0) - f(z)) + t^2 \|g\|_2^2\end{aligned}$$

Consider

$$g(t) = -2t(f(x_0) - f(z)) + t^2 \|g\|_2^2$$

It's a convex quadratic facing upwards, has zeros at $t = 0$ and $t = 2(f(x_0) - f(z))/\|g\|_2^2 > 0$, so

$$g(t) < 0 \ \forall \ t \in \left(0, \tfrac{2(f(x_0)-f(x))}{\|g\|_2^2}\right)$$

Thus we have $-g$ gets us closer to a smaller $f(z)$.

3.11 Features extraction

Mapping an input space $\mathbf{X}$ (sound wave, image, DNA sequence) to a vector $\mathbb{R}^d$ is called **feature extraction** or **featurization**.
A **feature template** is a group of features all computed in a similar way (e.g. $last\_three\_chars\_of(x) = \_\_\_$).†

*Using the definition of subgradient, and inner product $g^T \cdot (y - x_0) > 0$ if y is on the side g points in

†With regularization, our resulting prediction function won't be too complicated.

A **one-hot encoding** is a feature template that always has exactly one non-zero value.
Features can be encoded with an array (good for dense features), or a map (good for sparse features).
Some factors to consider when featurizing

- Non-monotonicity. E.g. temperature as a feature for health prediction, health is not monotonic with temperature. We can transform the input $\Phi x = [1, \{temperature(x) - 37\}^2]$*, but this would require domain knowledge, instead, we could do $\Phi(x) = [1, temperature(x), \{temperature(x)\}^2]$, where having one extra feature, we increase the flexibility and make it easier to use.†
- Saturation. E.g. find products relevant to user's query, where given a product x, we have a feature map $\Phi(x) = [1, N(x)]$ where $N(x) =$ number of people who bought x. However at some point x should saturate: a product bought 50000 times is not necessarily 10 times more relevant than a product bought 5000 times, as a linear model would suggest. We could do $log(1+N(x))$, or $arctan(x)$‡ to slow down the growth. Or we could do a discretization, like $\Phi(x) = [1(x < 10), 1(10 \leq x < 100), 1(100 \leq x)]$§.
- Interaction. E.g. predicting health with weight and height, it's not that they individually matter, but rather weight relative to height. You can add a cross term $h(x)w(x)$, making $\Phi(x) = [1, h(x), w(x), h^2(x), x^2(x), h(x)w(x)]$

A **predicate** of the input space is a function $P : \mathbf{X} \to \{\text{True, False}\}$.
What about given features $\Phi(x) = [x_1, x_2]$, we want to classify if a point will fall into a circle in the two-dimensional input space? We could add $x_1^2 + x_2^2$ as another feature.
Similaryly, output space can be transformed as well.
We essentially grow the (linear) hypothesis space by adding more features.

3.12 Neural nets

Linear prediction functions, like SVM, ridge, lasso generate feature vector $\Phi(x)$ by hand, and learn parameter vector w from data. A neural net adds **hidden nodes** between the $\Phi(x)$ and score. Multi-layer perceptron, e.g. in this two-layer representation (two affine combinations):

$$h_i = \sigma(v_i^T \Phi(x))$$
$$score = w^T h + b$$

*Array representation of features, where 1 is a bias term
†Think less, make the computer do more
‡A sigmoid function. Note how this transformation does not need to be convex.
§Where if not bucketed carefully, buckets with few items will have coefficients 0, due to regularization. Instead, we could have $\Phi(x) = [1(x \geq 5), 1(x \geq 10), 1(x \geq 100)], whereas mall bucket would fall back to the previous feature.$

where σ is a nonlinear **activation function**, and we need to learn the vectors v_i (with a bias term), w, and scalar b. We can add hidden nodes, hidden layers, etc, where > 1 hidden layer is a deep network.
Hyperbolic tangent is a common activation function.

$$\sigma(x) = tanh(x)$$

Rectified linear function (ReLU) is another. ReLU in practice often works better.

$$\sigma(x) = max(0, x)$$

One way to think about neural nets is that it's learning the non-linear featurization functions we want to create. (From the hidden layer h_i to *score* looks just like learning in a linear context, and neural nets learn $\Phi(x)$ to h_i as well). Our objective function (with $tanh$) is then differentiable w.r.t. all parameters (we can use gradient descent), but not convex. In practice gradient descent seems sufficient.
Universal approximation theorem: a neural network with one (possibly huge) hidden layer can uniformly approximate any continuous function on a compact set iff the activation function is not a polynomial. (the bias term is necessary)

3.13 Multinomial logistic regression

4 Exercises

4.1 Deriving gradient affine form

1. Given $f(w) = c^t w$, $\bigtriangledown f(w)$?

$$f'(x; u) = \lim_{h \to 0} \frac{f(w + hu) - f(w)}{h} = \lim_{h \to 0} \frac{c^T hu}{h} = c^T u$$

This shows

$$\bigtriangledown f(x) = c$$

2. Given $f(w) = w^T Aw$, $\bigtriangledown f(w)$?

$$\begin{aligned} f'(w; u) &= \lim_{h \to 0} \frac{f(w + hu) - f(w)}{h} \\ &= \lim_{h \to 0} \frac{(w + hu)^T A(w + hu) - w^T Aw}{h} \\ &= \lim_{h \to 0} \frac{w^T Aw + hw^T u + hu^T Aw + h^2 u^T Au - w^T Aw}{h} \\ &= u^T Aw + w^T Au \\ &= w^T A^T u + w^T Au \end{aligned}$$

This shows

$$\bigtriangledown f(x) = (w^T A^T + w^T A)^T = (A + A^T)w$$

3. Given $f(w) = \|Aw - y\|_2^2$, $\bigtriangledown f(w)$?

$$f(w) = \|Aw - y\|_2^2 = (Aw - y)^T (Aw - y)$$

$$\begin{aligned} f'(w;u) &= \lim_{h \to 0} \frac{(A(w+hu) - y)^T (A(w+hu) - y) - (Aw - y)^T (Aw - y)}{h} \\ &= \lim_{h \to 0} (A(w+hu) - y)^T Au + (Au)^T (A(w+hu) - y) \\ &= (Aw - y)^T Au + (Au)^T (Aw - y) \\ &= 2(Aw - y)^T Au \end{aligned}$$

This shows

$$\bigtriangledown f(x) = 2((Aw - y)^T A)^T = 2A^T (Aw - y) = 2A^T Aw - 2A^T y$$

4. Given $f(w) = \|Aw - y\|_2^2 + \lambda \|w\|_2^2$, express $f(w) = \|Bw - z\|_2^2$?
The gradient of the two need to be the same, using previous result, we need to solve

$$\begin{cases} A^T A + \lambda & = B^T B \\ A^T y & = B^T z \end{cases}$$

Note the equivalence between extending a matrix and addition, let

$$B = \begin{pmatrix} A \\ \sqrt{\lambda} I_{n \times n} \end{pmatrix} \text{ and } z = \begin{pmatrix} y \\ 0_{n \times 1} \end{pmatrix}$$

written in block-matrix form.

4.2 Recap: linear regression with square loss