Definitions and notations

Zhehao Wang

January 24, 2019

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 $\mathbf{R}^n$

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

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 $\mathbf{R}^n$ 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.

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

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 Inner product space / pre-Hilbert space, Hilbert space

An **inner product space** (over reals) is a vector space $\mathbb{V}$ and an **inner product**, which is a mapping

$$\langle ., . \rangle : \mathbb{V} \times \mathbb{V} \to \mathbf{R}$$

This space has the properties that $\forall x, y, z \in \mathbb{V}$ and $a, b \in \mathbf{R}$,

- symmetry: $\langle x, y \rangle = \langle y, x \rangle$
- linearity: $\langle ax + by, z \rangle = a\langle xz \rangle + b\langle y, z \rangle$
- positive-definitiveness: $\langle x, x \rangle \geq 0$. And $\langle x, x \rangle = 0 \iff x = 0$

A **norm** on the inner product space can then be defined as

$$\|x\| = \sqrt{\langle x, x \rangle}$$

Parallelogram law states that a norm can be written in terms of an inner product on $\mathbb{V}$ iff $\forall x, x' \in \mathbb{V}$,

$$2\|x\|^2 + 2\|x'\|^2 = \|x + x'\|^2 + \|x - x'\|^2$$

And if it can be, the inner product is given by the **polarization identity**

$$\langle x, x' \rangle = \frac{\|x\|^2 + \|x'\|^2 - \|x - x'\|^2}{2}$$

Pythagorean theorem states that two vectors are **orthogonal** if $\langle x, x' \rangle = 0$, denoted as $x \perp x'$. And x is orthogonal to a set S if for all $s \in S$, x is orthogonal to s.
If $x \perp x'$, then $\|x + x'\|^2 = \|x\|^2 + \|x'\|^2$

Roughly, **projection onto a plane** can then be defined as for $x \in \mathbb{V}$, let M be a subspace of inner product space $\mathbb{V}$, then m_0 is the projection of x onto M if $m_0 \in M$ and is the closest point to x in M. Meaning $\forall m \in M$, $\|x - m_0\| \leq \|x - m\|$.
A Hilbert space is a complete* inner product space. E.g. $\mathbf{R}^d$ and the standard inner product. Any finite dimensional inner product space is a Hilbert space.

*a space is complete if all Cauchy sequences in the space converge.

The **projection theorem** suggests, for a Hilbert space $\mathbf{H}$, M is a closed subspace of $\mathbf{H}$, for any $x \in \mathbf{H}$, there exists a unique $m_0 \in M$ for which

$$\|x - m_0\| \leq \|x - m\| \ \forall m \in M$$

This m_0 is called the orthogonal projection of x onto M. Furthermore, $m_0 \in M$ is the projection of x onto M iff $x - m_0 \perp M$.
Projection reduces norm: let M be a closed subspace of $\mathbf{H}$. For any $x \in \mathbf{H}$, let m_0 be the projection of x onto M, then $\|m_0\| \leq \|x\|$ with equality only when $m_0 = x$. *

1.8 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).

1.9 Jacobian matrix

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

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

1.10 Order of a matrix

1.11 Inversibility of a matrix

1.12 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

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

*think Pythagorean theorem on a $\mathbf{R}^2$ space
†Think square transformation matrix

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**

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

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).

1.13 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 $z \in \mathbf{R}^n$.
The identity matrix I is positive definite.
For any real invertible matrix A, the product $A^T A$ is a positive definite matrix.

1.14 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** $f : \mathbf{R}^n \to \mathbf{R}$ 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 $\mathbf{R}$ for all $a, b \in \mathbf{R}$.
- $f(x) = |x|^p$ for $p \geq 1$ is convex on $\mathbf{R}$.
- $f(x) = e^{ax}$ for all a is convex on $\mathbf{R}$.
- Every norm on $\mathbf{R}^n$ is convex.
- $f(x) = \max\{x\}$ is convex on $\mathbf{R}^n$.

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.

1.15 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 $x \in \mathbf{R}^n$ 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 $f(x^*) = p^*$.

1.16 Lagrangian duality

Given a general optimization problem:

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

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:

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

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

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: $f_i(x^*)$

*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

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

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.

1.17 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 $E = \mathbf{R}$
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

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

*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

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.

2.3 Expectation, variance

Variance

$$\mathbb{V}[X] = \mathbb{E}[x^2] - \mathbb{E}[x]^2$$

Derivation:

$$\begin{aligned}
\mathbb{V}[X] &= \frac{1}{n}\sum_i (x_i - \mathbb{E}[X])^2 \\
&= \frac{1}{n}\sum_i x_i^2 - 2\mathbb{E}[X]\cdot x_i + \mathbb{E}[X]^2 \\
&= \mathbb{E}[X^2] - 2\cdot\mathbb{E}[X]\cdot\mathbb{E}[X] + \mathbb{E}[X]^2 \\
&= \mathbb{E}[X^2] - \mathbb{E}[X]^2
\end{aligned}$$

Immediate observations:

$$\mathbb{V}[X - a] = \mathbb{V}[X]$$

$$\mathbb{E}[X - a] = \mathbb{E}[X] - a$$

Say we are to use one value m to represent a random variable X, s.t. mean square error is minimized.

$$\begin{aligned}
\text{MSE}(m) &= \mathbb{E}[(X-m)^2] \\
&= (\mathbb{E}[X-m])^2 + \mathbb{V}[X-m] \\
&= (\mathbb{E}[X]-m)^2 + \mathbb{V}[X]
\end{aligned}$$

$$\frac{\text{dMSE}(m)}{\text{d}m} = 2m - 2\mathbb{E}[X]$$

Thus when $m = \mathbb{E}[X]$, $\frac{\text{dMSE}(m)}{\text{d}m} = 0$.
So the value μ we should use is the expectation.

The expectation can be estimated from the mean of samples $(y_1, \dots y_n)$.

$$\hat{\mu} \equiv \frac{1}{n} \sum_{i=1}^{n} y_i$$

*

If samples are i.i.d, the **law of large number** suggests that

$$\hat{\mu} \to \mathbb{E}(Y) = \mu$$

Law of total expectation: for any random variables U and V,

$$\mathbb{E}(U) = \mathbb{E}[\mathbb{E}[U|V]]$$

Now instead of predicting one value for a random variable, we are given input X and want to find a function $f : X \to Y$ s.t. the mean square error is minimal. This optimal function is the **regression function**, and

$$\begin{aligned}
\text{MSE}(f) &= \mathbb{E}[(Y - f(X))^2] \\
&= \mathbb{E}[\mathbb{E}[(Y - f(X))^2|X]] \\
&= \mathbb{E}[\mathbb{V}[Y - f(X)|X] + (E[Y - f(X)|X])^2] \\
&= \mathbb{E}[\mathbb{V}[Y|X] + (E[Y - f(X)|X])^2]
\end{aligned}$$

To minimize this, the first term in the expectation doesn't depend on our prediction, and the second term looks like previously when we are predicting the value of a random variable with one value, only with all expectations conditional on X.
So the optimal function $\mu(x) = \mathbb{E}[Y|X = x]$, and this is called **regression function**.

When Y depends on X (causal model), we can use a noise random variable ϵ (whose expectation is 0) and regression function $\mu(x)$ to estimate Y:

$$\mu(X) + \epsilon \to Y$$

When causal relationship $X \to Y$ cannot be assumed, the noise random variable can be dependent on X, thus the general form $Y|X = \mu(X) + \eta(X)$ (without loss of generality, $\eta(X)$'s expectation is 0).

2.4 Bias-variance trade-off

When X takes only a finite set of values and we have enough sample points, we can reliably approximate a regression function (law of large number).

*Where the hat indicates empirical, as seen later on in empirical risk minimizer

But our sample points is almost always undersampled (e.g. when X is continuous), in which case we need interpolation, extrapolation and smoothing, and there are different methods to do these (different regression methods).
Suppose the true regression function is $\mu(x)$ and we use $\hat{\mu}$ to make prediction.*
The MSE of $\hat{\mu}$ (at x) can be written as such:

$$\begin{aligned}
\text{MSE}(\hat{\mu}) &= \mathbb{E}[(Y - \hat{\mu}(x))^2] \\
&= \mathbb{E}[(Y - \mu(x) + \mu(x) - \hat{\mu}(x))^2] \\
&= \mathbb{E}[(Y - \mu(x))^2 + 2(Y - \mu(x))(\mu(x) - \hat{\mu}(x)) + (\mu(x) - \hat{\mu}(x))^2] \\
&= \mathbb{E}[\eta^2] + 2(\mu(x) - \hat{\mu}(x))\mathbb{E}[\eta] + \mathbb{E}[(\mu(x) - \hat{\mu}(x))^2] \\
&= \mathbb{V}[\eta] + (\mu(x) - \hat{\mu}(x))^2
\end{aligned}$$

This is our first **bias-variance decomposition**, the second term is a bias by which our predictions are systematically off, and the first term is a variance that affect even the best prediction (which in general depends on x, let's denote it as σ_x^2).
In practice, $\hat{\mu}$ is not a single fixed function: it's something we estimate from sample data. If sample data are random, then exact regression function we get is random, too. Let's call this random function $\hat{M}_n$.†
Now if we are to analyze the prediction error of the *method*, averaging over all the possible training data sets

$$\begin{aligned}
\text{MSE}(\hat{M}_n(x)) &= \mathbb{E}[(Y - \hat{M}_n(X))^2 | X = x] \\
&= \sigma_x^2 + (\mu(x) - \mathbb{E}[\hat{M}_n(x)])^2 + \mathbb{V}[\hat{M}_n(x)]
\end{aligned}$$

This our second bias-variance decomposition. The first term is the same as before. The second term is in using $\hat{M}_n$ to estimate μ, the **approximation error / approximation error**. The third term is the variance in our estimate of the regression function, even if we have an unbiased method $\mu(x) = \mathbb{E}[\hat{M}_n(x)]$, if there is a lot of variance in our estimates, we can expect to make large errors.
The catch is, at least past a certain point, decreasing the approximation bias can only come through increasing the estimation variance. This is the **bias-variance trade-off**.
The trade off doesn't have to be one-for-one, sometimes we can lower the total error by introducing some bias as it gets rid of more variance than it adds approximation error.
In general, both approximation bias and estimation variance depend on n. A method is **consistent** when both of these go to 0 as $n \to \infty$. There can be multiple consistent methods for the same problem, and their biases and variances don't have to go to 0 at the same rate.

*As seen later, the true Bayesian form, and our prediction function which hopes to be as close as possible.

†The previous analysis really is $\text{MSE}(\hat{M_n}(x) | \hat{M_n}(x) = \hat{\mu})$

2.5 Ordinary least squares linear regression, linear smoothers

We choose to approximate $\mu(x)$ by $\alpha + \beta x$ and ask for the best a, b of the those constants.
For the sake of simplicity, we assume X is one dimensional and X and Y have mean 0.

$$\begin{aligned}\text{MSE}(\alpha, \beta) &= \mathbb{E}[(Y - \alpha - \beta X)^2] \\ &= \mathbb{E}[\mathbb{V}[Y|X]] + \mathbb{E}[(\mathbb{E}[Y - \alpha - \beta X|X])^2]\end{aligned}$$

The first term doesn't depend on α or β, so we can drop it. Taking the derivative of the second term, we have

$$\frac{\partial MSE}{\partial \alpha} = \mathbb{E}[-2 \cdot (Y - \alpha - \beta X)]$$

and $a = \mathbb{E}[Y] - b\mathbb{E}[X] = 0$. (we assumed X and Y both have mean 0)

$$\frac{\partial MSE}{\partial \beta} = \mathbb{E}[-2X \cdot (Y - \alpha - \beta X)]$$

and $b = \frac{\mathbb{E}[XY]}{\mathbb{E}[X^2]} = \frac{Cov[X,Y]}{\mathbb{V}[X]} = \frac{\sum_i y_i x_i}{\sum_i x_i^2}$. (we assumed X and Y both have mean 0)

And we are now in a position to see how the least square linear regression model is really a smoothing of the data:

$$\hat{\mu}(x) = \hat{b}x = \sum_i y_i \frac{x_i}{\sum_j x_j^2} x = \sum_i y_i \frac{x_i}{n\hat{\sigma}_X^2} x$$

Where $\hat{\sigma}_X^2$ is the sample variance of X.
The **intuition** is that our prediction is a weighted average of observed values y_i of the dependent variable, where the weights are proportional to how far x_i is from the center (relative to the variance), and proportional to the magnitude of x.
The linear regression line is a special case of **linear smoothers**, which are estimates of the regression function with the following form:

$$\hat{\mu}(x) = \sum_i y_i \hat{w}(x_i, x)$$

They are called linear as the predictions are linear in the responses y_i, as functions of x they are generally nonlinear. In linear regression's case, as shown earlier,

$$\hat{w}(x_i, x) = \frac{x_i}{n\hat{\sigma}_X^2} x$$

This ignores the distance between x_i and x.

2.6 k-Nearest-Neighbor Regression

Consider **nearest-neighbor regression**, which is very sensitive to the distance between x_i and x:

$$\hat{w}(x_i, x) = \begin{cases} 1 & x_i \text{ nearest neighbor of } x \\ 0 & \text{otherwise} \end{cases}$$

This will include the noise into its prediction. We might instead do k-nearest neighbor regression (as noise tend to cancel each other out), where as we increase k we get smoother functions, until $k = n$ where we get a constant.

$$\hat{w}(x_i, x) = \begin{cases} \frac{1}{k} & x_i \text{ one of the } k \text{ nearest neighbor of } x \\ 0 & \text{otherwise} \end{cases}$$

Because k-nearest-neighbors averages over only a fixed number of neighbors, each of which is a noisy sample, it always has some noise in its prediction, and is generally not consistent.

2.7 Kernel smoothers

With kNN regression each testing point is predicted using information from only a few data points, say we want to use all the training data like ordinary linear regression, but in a location-sensitive way.
Kernel smoothing does this. We pick a kernel function $K(x, x)$ which satisfies

- $K(x_i, x) \geq 0$
- $K(x_i, x)$ depends only on the distance $x_i - x$, not the individual arguments. (Hence we can write K as a one-argument function, $K(x_i - x)$)
- $\int xK(0, x)dx = 0$ and
- $0 < \int x^2 K(0, x)dx < \infty$

These conditions together imply $|x_i - x| \to \infty$, $K(x_i, x) \to 0$.*
The Nadaraya-Watson estimate of the regression function is

$$\hat{\mu}(x) = \sum_i y_i \frac{K(x_i, x)}{\sum_j K(x_j, x)}$$

$K(x_i, x)$ is large if x_i is close to x, so this places a lot of weight on training data points close to the point where we are trying to predict.

*Example of such functions include the density of the uniform distribution $(-\frac{h}{2}, \frac{h}{2})$, and the density of the standard Gaussian $\mathbf{N}(0, \sqrt(h))$ distribution. h can be any positive number and is called **bandwidth**, which controls the degree of smoothing, $h \to \infty$ we revert to taking the global mean, and $h \to 0$ results in spikier functions.

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** $l : \mathbf{A} \times \mathbf{Y} \to \mathbf{R}$ 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

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

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*.

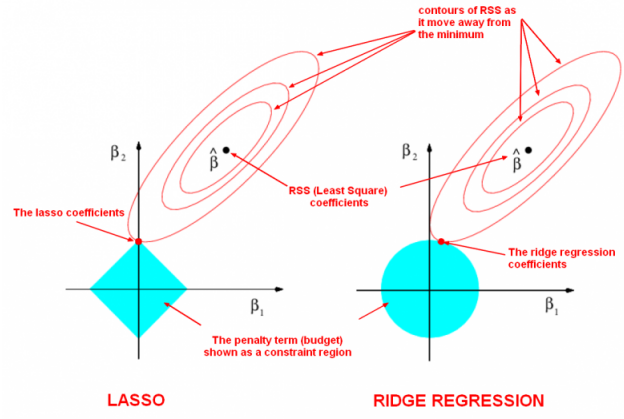

Figure 1: Why lasso regression tends to lead to feature sparsity

*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.

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

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.

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

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 ($l(\hat{y}, y) \in \mathbf{R}$) 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** $f : \mathbf{X} \to \mathbf{R}$, meaning action space $\mathbf{A} = \mathbf{R}$ 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

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

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

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 *.

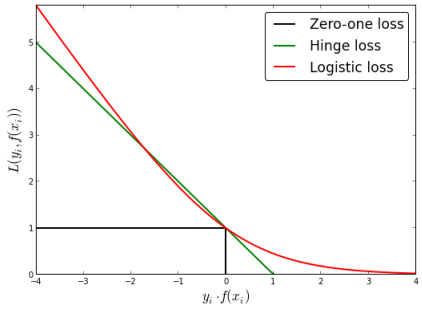

Figure 2: 0-1 loss, hinge loss and Logistic loss functions

3.7 SVM and Lagrangian duality

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

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

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

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

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

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

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 $f : \mathbf{R}^d \to \mathbf{R}$ be a function. A **level set** or **contour line** for the value c is the set of points $x \in \mathbf{R}^d$ for which $f(x) = c$.
A **sublevel set** for the value c is the set of points $x \in \mathbf{R}^d$ for which $f(x) \leq c$.
If $f : \mathbf{R}^d \to \mathbf{R}$ 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

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

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

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

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

3.10 Subgradients

A vector $g \in \mathbf{R}^d$ is a **subgradient** of $f : \mathbf{R}^d \to \mathbf{R}$ 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 $f : \mathbf{R}^d \to R$,

*Where local information gives global information!

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

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 $f : \mathbf{R}^d \to \mathbf{R}$ have a subgradient g at x_0, hyperplane H orthogonal to g at x_0 must **support** the level set $S = \{x \in \mathbf{R}^d | f(x) = f(x_0)\}$, 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 $\mathbf{R}^d$ 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 Kernel methods

From a high level, goal is to allow access to huge feature spaces without suffering heavy computational cost.
With featurization, we can map input space $\mathbf{X} \subset \mathbf{R}^n$ to $\mathbf{R}^d$¶, and the hypothesis space is of affine functions on feature space looks like

$$\mathbf{H} = \{x \to w^T \Phi x + b | w \in \mathbf{R}^d, b \in \mathbf{R}\}$$

*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 $\Phi(x) = [1(x \geq 5), 1(x \geq 10), 1(x \geq 100)], whereas mall bucket would fall back to the previous feature.$

¶And as it would seem, this optimization makes sense when $n < d$

To get expressive hypothesis spaces using linear models, we need high-dimensional feature spaces (say, adding in all $x_i^{k_i}$ where $\sum k_i \leq C$). This complicates computation, and may cause overfitting. Overfitting can be handled with regularization, and kernel methods can help with memory and computational costs.
For example, recall a featurized SVM prediction function, which is the solution to the following

$$\underset{w\in\mathbf{R}^d, b\in\mathbf{R}}{\arg\min}\ (\frac{1}{2}w^Tw + \frac{c}{n}\sum_{i=1}^{n}(1 - y_i[w^T\Phi(x_i) + b])_+$$

whose dual is

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

Where Φx only shows up as inner products with other xs.
A method is **kernerlized** if inputs only appear inside inner products. $\langle\Phi(x), \Phi(x')\rangle$ for $x, x' \in \mathbf{X}$. The **kernel function** corresponding to Φ and inner product $\langle ., .\rangle$ is

$$k(x, x') = \langle\Phi(x), \Phi(x')\rangle$$

For example, consider quadratic feature map for $x = (x_1, \dots x_d) \in \mathbf{R}^d$

$$\Phi(x) = (x_1, \dots x_d, x_1^2, \dots x_d^2, \sqrt{2}x_1x_2, \dots, \sqrt{2}x_{d-1}x_d)^T$$

has dimension $O(d^2)$, but for any $x, x' \in \mathbf{R}^d$

$$k(x, x') = \langle\Phi(x), \Phi(x')\rangle = \langle x, x'\rangle + \langle x, x'\rangle^2$$

Bringing down the computation cost from $O(d^2)$ to $O(d)$.*
Continuing on with the SVM example, with linear kernel $k(x, x') = x^Tx'$, we define the **kernel matrix** (**Gram matrix**) as such:

$$K = (\langle x_i, x_j\rangle)_{i,j} = \begin{pmatrix}\langle x_1, x_1\rangle \dots \langle x_1, x_n\rangle\\ \dots\\ \langle x_n, x_1\rangle \dots \langle x_n, x_n\rangle\end{pmatrix}$$

Then for the standard Euclidean inner product $\langle x_i, x_j\rangle = x_i^Tx_j$, we have

$$K = XX^T$$

*Often useful to think of the kernel function as a **similarity score**

Where $X = (x_1, \dots x_n)^T$, an $n \times d$ vector, and K is all the $Phi(x)-$terms we are interested in.*
And the dual problem becomes

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

This also gives us the flexibility to change kernels to a different K of $n \times n$ dimension, which can, e.g., correspond to a high dimensional feature space. This is **kernel trick**.

And there are other non-linear kernels, our quadratic example earlier and **polynomial kernel** $k(x, x') = (1 + \langle x, x' \rangle)^M$, which corresponds to a feature map with all monomials up to degree M, the computational cost of the kernel does not grow with M, while plugging in the featurized terms directly the computational cost grows rapidly with M.
And **Radial Basis Function / Gaussian kernel**, $\forall x, x' \in \mathbf{R}^d$

$$k(w, x) = e^{-\frac{\|x - x'\|^2}{2\sigma^2}}$$

†

3.13 Representer theorem and its usage in kernelization

Consider SVM objective function in a more general term,

$$\arg\min_{w \in \mathbf{H}} J(w)$$

And let $J(w)$ be

$$J(w) = R(\|w\|) + L(\langle w, \Phi(x_1) \rangle, \dots, \langle w, \Phi(x_n) \rangle)$$

Where $\|.\|$ corresponds to the inner product on space $\mathbf{H}$. $R : [0, \infty) \to \mathbf{R}$ is the regularization term and $L : \mathbf{R}^n \to \mathbf{R}$ is the loss term.‡

*Recall with ridge regression, we worked with $X^T X$, which is $d \times d$, meaning we are interested in kernel methods when $n \leq d$

†Note how this acts like a similarity score, and with this kernel, your featurization dimension d may grow, but to compute the kernel is always $n \times n$ where n is the number of data points.

‡Each inner product represents a (linear) prediction. Ridge regression and SVM are both of this form. Lasso is not, as l_1 norm does not correspond to an inner product.

If $J(w)$ has a minimizer, then it has a minimizer of the form $w^* = \sum_{i=1}^{n} \alpha_i \Phi(x_i)$.* With this, we don't have to search over the feature space w^* is in, $\mathbf{R}^d$, but rather having computed the n $\Phi(x_i)$s, we search over that space of n dimensions. This lets you deal with infinite dimensional feature space.
As an example, a kernelized objective function for SVM looks like

$$\arg\min_{\alpha \in \mathbf{R}^n} R(\sqrt{\alpha^T K \alpha}) + L(K\alpha)$$

Where there is no direct access to $\Phi(x_i)$, and all references are via kernel matrix K.

3.14 Performance evaluation

Intuition,

- Always spend some time building a linear baseline model. (lasso / ridge / elastic net regression; l1 / l2 regularized logistic regression or svm)
- Prefer simpler models if performance is the same.
- Consider building an oracle model, which is helpful to get an upper bound on achievable performance. E.g. fit your validation data without regularization. This can give estimate of the approximation error of our hypothesis space.

Confusion matrix for binary classification problem, the 4-cell matrix of true / false positive / negatives.

- **True positive**: predicted positive, and it's right
- **True negative**: predicted negative, and it's right
- **False positive / type 1 error**: predicted positive, and it's wrong
- **False negative / type 2 error**: predicted negative, and it's wrong

Criteria based on the confusion matrix,

- **Accuracy** $(TP + TN)/(TP + TN + FP + FN)$
- **Error rate** $(FP + FN)/(TP + TN + FP + FN)$
- Accuracy + Error rate = 1.
- Accuracy by itself is often not enough, consider a no-information classifier (always produced 0, e.g.) on a distribution that's predominantly one-sided (e.g. 0).

*This is the same conclusion we reached at the end of SVM: w^* is a linear combination of features. This can be proved using Pythagorean theorem and projection theorem: do a projection from w^* to M, span of the data

- **Precision** $TP/(TP+FP)$: high precision means low false alarm rate.
- **True positive rate / recall / sensitivity** $TP/(TP+FN)$: high recall means you are not missing many positives.
- **False negative rate / miss rate** $FN/(FN+TP)$
- **False positive rate / fall-out / false-alarm rate** $FP/(FP+TN)$
- **True negative rate / specificity** $TN/(FP+TN)$
- F_1 **score**. $F_1 = 2\frac{precision \cdot recall}{precision+recall}$. F_β. Weigh towards precision or recall

In a classification problem, the real-valued prediction function (score function) does not have to be thresholded at 0*. Thresholding the score function differently causes the performance metrics to shift. We can threshold it at a value to meet our performance goals, e.g. $recall > 80\%$.
Measurement curves based on these rates.

- **Precision-recall curve**: as you change the threshold, how precision and recall change.
- **Receiver-operation characteristic (ROC) curve**: as you change threshold, how TP rate and FP rate change.
- **AUC / AUC ROC**: area under the ROC curve.

3.15 Probablistic modeling

Instead of producing a single value or classification (regression or classification), produce a probability distribution of values, with which we can decide how likely a test data point is. This is applicable in e.g. anonamaly detection.
The approach usually entails selecting a particular probability distribution, and fit the samples to derive the parameters of that distribution. (E.g. the mean value of Poisson distribution)
Stratification means partitioning our input data into groups, and treat each group separately. E.g. we stratify each symbol traded into time ranges like 9:30A to 10:30A on every weekday, and train a model for that particular time range.
Bucketing / binning is the opposite of stratification, where we combine natural groups of data into a single group, and train a model for that single group. E.g. we can bucket multiple tech stock symbols into one group, and train one model to predict their overall price movement.
Stratification brings out specificity (also more bias), bucketing smoothes bias out but lose specificity.

*> 0 predict positive, < 0 predict negative

3.16 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):

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

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)

3.17 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